

Freiburg, den 4. Dezember 2000

Informatik 1, WiSe 2000/2001

FAQ zum Programmierprojekt 1

1. ABGABE

1.1. **Wie?** Die Abgabe erfolgt über das WWW. Dazu wird ab dem 4.12.2000 auf den Web-Seiten zur Vorlesung eine spezielle Seite zur Verfügung stehen.

1.2. **Was?** Das abgegebene Programm muß im ASCII-Format abgespeichert sein und sich unter der auf den Unix-Workstations im Rechnerpool installierten Version von DrScheme ausführen lassen. Falls Sie auf einem anderen Rechner (z.B. unter MS Windows) entwickeln, müssen Sie dafür Sorge tragen, daß Ihre Programmdatei diese Bedingungen erfüllt.

Es sollen nur die selbst geschriebenen Funktionen abgegeben werden. Die Funktionen aus `programm-1a.ss` sind weder nötig noch hilfreich.

Jedes Programm muß einen Kommentar mit dem Namen des Programmierers/der Programmiererin enthalten.

Falls Sie Aufgabe 11 lösen, soll die dort verlangte Funktion `picture-optimize` heißen.

Die abgegebenen Programme müssen zumindest von DrScheme eingelesen werden können. Programme, bei denen bereits bei der Betätigung des *Execute*-Buttons Fehler angezeigt werden, können nicht gewertet werden. (Gegebenfalls müssen die betreffenden Aufgabenteile auskommentiert werden.)

1.3. **Bis wann?** Die Abgabefrist wurde auf den 9.12.2000, 12 Uhr mittags, verlängert.

1.4. **Wie oft?** Falls man nach der Abgabe sein Programm verbessert, kann man es in der regulären Frist noch ein- oder mehrfach abgeben. Das zuletzt abgegebene Programm wird gewertet.

1.5. **Danach?** Nach der Abgabe werden die Programme den Tutoren gegeben, so daß jeder Tutor die Programme der Teilnehmer seiner (Theorie-)Übungsgruppe hat.

2. BEWERTUNG

2.1. **Vorführung.** In den (Theorie-)Übungsgruppen vom 15.12. bis zum 21.12. werden die Programme den Tutoren vorgeführt. Ohne Vorführung kann das Projekt nicht gewertet werden.

Dabei werden einige vorgegebene Test-Graphiken berechnet.

2.2. **Plagiate.** Während es erlaubt und erwünscht ist, mit anderen Ideen auszutauschen, muß der Code von jedem alleine geschrieben werden. Falls dennoch Plagiate vorkommen, erhalten alle Teilnehmer, die Code teilen, Null Punkte für das Programmierprojekt.

2.3. **Publizierte Algorithmen.** Publizierte Algorithmen, etwa aus einem der zahlreichen Bücher von Sedgewick oder aus Büchern über graphische Datenverarbeitung, dürfen bei Angabe der Quelle für das Programmierprojekt verwendet werden, ohne daß es dafür Punktabzug gibt.

2.4. **Dokumentation.** Punkte werden getrennt für Programmdokumentation und -code vergeben. Zu jeder Funktion gehört dabei eine Spezifikation im Stil der in der Vorlesung angegebenen Scheme-Funktionen.

Falls zwei Funktionen ähnlich funktionieren, kann man auch auf die Spezifikation der anderen Funktion verweisen. (Bei `pixel-color` muß man also kein Beispiel für der Fall eines `ellipse`-Objekts hinschreiben, wenn bei `ellipse-hits` ein Beispiel angegeben ist und in der Spezifikation von `pixel-color` darauf verwiesen wird.)

Aus den Kommentaren sollte auch hervorgehen, zu welchem Aufgabenteil der sich anschließende Scheme-Code gehört.

3. GRUNDIDEEN DES PROGRAMMIERPROJEKTS

3.1. **Linienobjekte.** Für jedes Pixel der Bildfläche läßt man einen Strahl auf die geometrischen Objekte fallen. Falls der Strahl auf einen Teil des Objekts trifft, wird das Pixel in der entsprechenden Farbe gezeichnet.

Bei Linienobjekten kann es vorkommen, daß Strahlen das Objekt verfehlen. (Z.B. wird das Quadrat mit den Eckpunkten $(1/2, 1/2)$, $(11/2, 1/2)$, $(11/2, 11/2)$, $(1/2, 11/2)$ von keinem Strahl getroffen.)

Um zu verhindern, daß es zu Darstellungsfehlern kommt, arbeiten wir deshalb statt mit einem Strahl mit einem Zylinder. Der Schnittkreis dieses „Strahlzylinders“ wird Strahlkreis genannt.

Das Pixel der Bildebene wird dann gefärbt, wenn der Strahlkreis, dessen Mittelpunkt den Pixelkoordinaten entspricht, das Objekt schneidet.

3.2. **Flächige Objekte.** Bei Flächen reicht es, zu schauen, ob der Mittelpunkt des Strahlkreises in der Fläche liegt.

3.3. **Die Funktion `pixel-color`.** Übergeben wird eine `picture`-Struktur, die das zu zeichnende geometrische Objekt darstellt, sowie der Strahlkreis.

Zurückgeliefert wird `#f`, falls der Strahlkreis das geometrische Objekt nicht trifft, `#t`, falls er das Objekt trifft und dieses nicht eingefärbt ist, und einen `color`-Wert, falls er es trifft und das Objekt gefärbt ist.

3.4. **Die above-Struktur.** Die Idee bei der **above-Struktur** ist, daß ein Bild zusammengesetzt wird, indem man nacheinander elementare **picture-Objekte** auf der Bildfläche ablegt. Falls dabei ein Objekt auf ein anderes gelegt wird, verdeckt es dieses ganz oder teilweise. Den verdeckten Teil des Objekts kann man dann auf der Graphik nicht sehen.

4. ELLIPSEN

4.1. **Entartete Ellipsen.** Falls bei einer Ellipse eine oder beide Halbachsen gleich Null sind, heißt die Ellipse (zu einer Strecke oder einem Punkt) entartet. In diesem Fall darf man die im Aufgabenblatt verwendete Formel nicht verwenden, da bei ihr die Halbachsen im Nenner vorkommen. Man müßte in diesem Fall die Ellipse wie eine einteilige Polyline oder einen Punkt behandeln. (Punkte sind in dem Projekt nicht vorgesehen.) Um diese Fallunterscheidung zu vermeiden, darf man im Projekt annehmen, daß keine **ellipse-Struktur** vorkommt, die eine entartete Ellipse darstellt. (Siehe aber den Abschnitt über nicht ausgefüllte Ellipsen.)

4.2. **Nicht ausgefüllt Ellipsen.** Eigentlich müßte man dabei überprüfen, ob der Strahlkreis die Ellipsenkurve schneidet. Eine einfachere Lösung, die fast genauso schöne Graphiken liefert, besteht darin, daß man stattdessen schaut, ob der Mittelpunkt des Strahlkreises innerhalb einer leicht vergrößerten Ellipse und außerhalb einer leicht verkleinerten Ellipse liegt. Die Ellipse wird vergrößert, indem man zu beiden Hauptachsen den Radius des Strahlkreises addiert. Sie wird verkleinert, indem man ihn subtrahiert. Bei letzterem kann der Radius kleiner oder gleich Null werden. In diesem Sonderfall beschränkt man sich einfach auf den Test, ob der Strahlkreismittelpunkt innerhalb der vergrößerten Ellipse liegt.

5. FARBEN

5.1. **Verschachtelte Farbstrukturen.** Die Idee ist, daß eine **colored-Struktur** der ganzen in ihr enthaltenen **picture-Struktur** eine Farbe zuweist mit Ausnahme derjenigen Teilstrukturen, die bereits eine Farbe haben. Wenn man eine **picture-Struktur** graphisch als Baum darstellt, bedeutet das, daß eine in Richtung zu den Zweigen liegende **colored-Struktur** höhere Priorität als eine in Richtung zu den Wurzeln liegende hat.

6. TRANSFORMATIONEN

6.1. **Grundidee.** Die Grundidee bei Transformationen besteht darin, anstatt des Objektes den Strahl umgekehrt zu transformieren. (Wenn man ein Objekt um 10 Pixel nach oben verschiebt, muß man den Strahlkreis um 10 Pixel nach unten verschieben.)

6.2. **Skalierung.** Bei der Streckung/Stauchung (mit der **scale-Struktur**) muß man zunächst den Mittelpunkt des Strahlkreises entsprechend transformieren. Der Radius des Strahlkreises muß auch transformiert werden. Würde er es nicht, würde sich die Strichbreite verändern, was zu unterbrochenen Linien führen kann. Eigentlich müßte man aus dem Strahlkreis eine Strahlellipse machen, was allerdings kompliziert würde. (Man kann ja noch weitere Transformationen anwenden!)

Deshalb setzt man den Radius des Strahlkreises einfach auf folgenden Wert:

$$R = \max(|r/(\text{scale-x})|, |r/\text{scale-y}|)$$

R ist dabei der Radius des transformierten Strahlkreises, r der Radius des ursprünglichen Strahlkreises, scale-x der Skalierungsfaktor in Richtung der x -Achse, scale-y der Skalierungsfaktor in Richtung der y -Achse. Falls sich scale-x und scale-y stark unterscheiden, kommt es dabei zu Artefakten, die sich darin äußern, daß senkrechte und waagrechte Linien unterschiedliche Stärke haben.

Dabei darf man annehmen, daß weder scale-x noch scale-y gleich Null sind.

6.3. Polylines und Polygone. Bei Polylines und Polygonen werden alle Punkte transformiert, einschließlich der `origin`-Komponente.

6.4. Rotation. Drehungen verstehen sich um den Ursprung des Koordinatensystems, im mathematisch positiven Drehsinn (entgegen der Uhr).

6.5. Darstellung von Rotationen. Eine Drehung um den Ursprung ist durch den Drehwinkel φ eindeutig beschrieben. Zur Berechnung der Koordinaten eines gedrehten Punktes wendet man folgende Gleichung an:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Dabei sind x und y die Koordinaten des ursprünglichen Punktes, x' und y' die Koordinaten des Bildpunktes unter der Drehung.

Da die Berechnung der trigonometrischen Funktionen aufwendig ist, kann man das Programm beschleunigen, indem man die Rotationsmatrix nur einmal ausrechnet und die Drehung dann durch sie beschreibt.

7. POLYLINES UND POLYGONE

7.1. Datenstruktur. Die in der Datenstruktur angegebenen Punkte sind die Eckpunkte der Polyline bzw. des Polygons. Die Koordinaten verstehen sich bezüglich des Koordinatensystems.

7.2. Wie allgemein dürfen die Polygone sein? Der Einfachheit halber beschränken wir uns auf konvexe (d.h. wenn zwei Punkte im Polygon liegen, dann liegt auch ihre Verbindungslinie darin), schnittfreie Polygone. Weiterhin dürfen Eckpunkte nicht zusammenfallen.

Wer allgemeinere Funktionen programmiert, hat die Aufgabe natürlich auch gelöst.

7.3. Ausgefüllte Polygone. Es gibt verschiedene Algorithmen, um herauszufinden, ob ein Punkt innerhalb eines Polygons liegt. Wenn man sich auf konvexe Polygone beschränkt, führen folgende Überlegungen zu einer Lösung:

Jede Seite des Polygons ist (durch die Reihenfolge der Eckpunkte des Polygons) gerichtet. Dadurch teilt jede Seite des Polygons die Ebene in eine linke und eine rechte Halbebene. Ein Punkt liegt innerhalb des Polygons, wenn er entweder für alle Seiten auf der jeweils linken Halbebene oder für alle Seiten auf der jeweils rechten Halbebene liegt.

Nun bleibt nur noch das Problem, herauszufinden, ob ein Punkt (x, y) in der linken oder rechten Halbebene einer Polygonseite $((x_1, y_1), (x_2, y_2))$ liegt. Dazu betrachtet man die Vektoren $v_1 = (x_1 - x, y_1 - y)$ und $v_2 = (x_2 - x, y_2 - y)$. Der Punkt liegt in der linken Halbebene, falls das Kreuzprodukt $v_1 \times v_2$ in die Blattebene hineinzeigt. Er liegt in der rechten Halbebene, falls es zum Betrachter zeigt.

Da das Kreuzprodukt durch

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 - b_2 a_3 \\ a_3 b_1 - b_3 a_1 \\ a_1 b_2 - b_1 a_2 \end{pmatrix}$$

definiert ist, muß man nur noch schauen, ob $a_1 b_2 - b_1 a_2$ größer oder kleiner als Null ist. (Man betrachtet dabei v_1 und v_2 als dreidimensionale Vektoren, deren dritte Komponente gleich Null ist.)