



Institut für Informatik  
Prof. Dr. Peter Thiemann  
Jochen Walter

Georges-Köhler-Allee 79  
D-79110 Freiburg i. Br.

Freiburg, den 23. November 2000

## Informatik 1, WiSe 2000/2001

### Übungsblatt 5 Programmierprojekt 1

Das erste Programmierprojekt läuft bis zum 8.12.2000.

In diesem Zeitraum werden in den Übungs- und Programmiergruppen Probleme im Zusammenhang mit dem Programmierprojekt besprochen.

*Die Aufgaben auf diesem Blatt werden alleine bearbeitet. Bevor drscheme gestartet werden kann, muß setup lang eingegeben werden. Der Sprachumfang muß auf „Full Scheme“ eingestellt werden. Die Aufgaben sind mit einem bis drei Sternen versehen, wobei die Aufgaben mit einem Stern am einfachsten, die mit dreien am schwierigsten sind.*

**Die Abgabe erfolgt bis zum 8.12.2000 über das WWW. Informationen dazu finden sich ab dem 4.12.2000 auf den Web-Seiten zu Vorlesung.**

**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.**

**Das von Ihnen abgegebene Programm sollte für jede Funktion eine vollständige Spezifikation im Stil der in der Vorlesung angegebenen Scheme-Funktionen haben. Punkte werden getrennt für Programmcode und Dokumentation vergeben.**

Das Thema dieses Programmierprojektes ist es, ein Programm zur Darstellung von Graphiken auf dem Bildschirm zu schreiben. Wir behandeln dabei die zweidimensionale Variante eines sogenannten Raytracers. Dabei wird eine Szenerie durch eine Anzahl von geometrischen Objekten dargestellt. Der Raytracer berechnet nun für jeden Punkt der Bildfläche die Farbe und Helligkeit des Lichtstrahles, der durch diesen Punkt in das Auge tritt.

In unserem Programmierprojekt beschränken wir uns auf zweidimensionale geometrische Objekte, die in einer Ebene beliebig angeordnet werden dürfen. Dabei darf ein Objekt ein anderes ganz oder teilweise verdecken, kein Objekt darf aber die zweidimensionale Ebene verlassen. Objekte können eine Farbe haben.

Die geometrischen Objekte, die von dem Programm verarbeitet werden, sind Ellipsen, Linienzüge und Polygone. Dazu gibt es entsprechende Strukturen (`ellipse`, `polyline`, `polygon`). Um farbige Objekte darstellen zu können, gibt es eine weitere Struktur (`colored`). Um transformierte Objekte zu behandeln, gibt es drei weitere Strukturen `translate`, `rotate`

und `scale`. Aus zwei geometrischen Objekten schließlich kann man mit Hilfe der Struktur `above` eine neues Objekt zusammensetzen, wobei das erste Teilobjekt (mit allen eventuell darin enthaltenen Teilobjekten) unterhalb des zweiten Teilobjektes (mit allen eventuell darin enthaltenen Teilobjekten) zu liegen kommt. Unter einem `picture` versteht man jede dieser Strukturen.

Die Bildfläche ist ein um den Koordinatenursprung zentrierter rechteckiger Ausschnitt der Ebene, in der die Objekte liegen. Wir berechnen die Graphik, indem wir einen Strahl senkrecht auf die Ebene fallen lassen und dem entsprechenden Punkt der Bildfläche diejenige Farbe zuordnen, die das Objekt hat, das von dem Strahl zuerst getroffen wird. Falls sich an dieser Stelle kein Objekt befindet, wird kein Punkt gezeichnet. Wir stellen den Strahl durch einen Kreis (eine `circle`-Struktur) mit kleinem Radius  $r_0$  dar.

### **Aufgabe 1 (0C+0D):**

Besorgen Sie sich von der Web-Seite der Vorlesung den erwähnten Scheme-Code und legen Sie eine Datei für Ihrem eigenen Schem-Code an. Sie können den vorgegebenen Code durch die Zeile (`load "program-1.ss"`) von Ihrem eigenen Code aus nachladen.

Informieren Sie sich anhand der Online-Hilfe von DrScheme (etwa unter dem Stichwort `rgb`) darüber, wie man in DrScheme Farben darstellt.

### **Aufgabe 2 (1C+1D):**

Schreiben Sie eine Funktion `pixel-color` mit der Signatur `picture circle -> color-bool`. (`color-bool` ist eine variante Struktur, die ein entweder `boolean` oder `color` sein kann.) Diese Funktion soll zunächst aus einem `cond`-Statement bestehen, das nur aus einer `else`-Klausel besteht. In dieser `else`-Klausel soll der Wert `#f` zurückgegeben werden. Rufen Sie nun die (vorgegebene) Funktion `render` auf. Es sollte ein Fenster mit weißem Inhalt angezeigt werden.

Die Funktion `pixel-color` wird im Laufe des Projekts für jede der möglichen `picture`-Arten (wie `ellipse` oder `colored`) um eine Klausel erweitert.

### **Aufgabe 3 (1C+1D):**

Für alle Punkte  $(x, y)$ , die innerhalb einer Ellipse, deren Mittelpunkt mit dem Koordinatenursprung zusammenfällt und deren Achsen auf den Koordinatenachsen liegen, gilt

$$\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} \leq 1.$$

Dabei ist  $r_x$  ( $r_y$ ) die Länge der Ellipsenhalbachse, die mit der  $x$ -Achse ( $y$ -Achse) zusammenfällt.

Interessiert man sich nur für die Punkte, die auf dem Rand der Ellipse liegen, lautet die entsprechende Formel

$$\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} = 1.$$

Wir dürfen diese Gleichung allerdings nicht so verwenden, da dabei die Wahrscheinlichkeit, daß der Strahl den Ellipsenrand trifft, verschwindend gering ist. Stattdessen testen wir, ob

der Punkte innerhalb der Ellipse mit den Halbachsen  $r_x + r_0$  und  $r_y + r_0$  und außerhalb der Ellipse mit den Halbachsen  $\max(r_x - r_0, 0)$  und  $\max(r_y - r_0, 0)$  liegt. Mit dem Parameter  $r_0$  kann man die Linienstärke der Ellipse steuern.

Schreiben Sie eine Scheme-Funktion `ellipse-hits` mit der Signatur `ellipse circle -> boolean`, die für eine Ellipse und einen Kreis angibt, ob Ellipse und Kreis sich überschneiden. Dabei muß die `filled`-Komponente der `ellipse`-Struktur beachtet werden!

Erweitern Sie die Funktion `pixel-color` um Code für die Behandlung von Ellipsen. Die von Ihnen geschriebene `cond`-Klausel soll `#t` zurückgeben, falls sich Ellipse und Kreis schneiden, `#f` sonst.

**Aufgabe 4 (1C+1D):**

Erweitern Sie die Funktion `pixel-color` um Code, der `translate`-Objekte behandelt.

**Aufgabe 5 (1C+1D):**

Erweitern Sie die Funktion `pixel-color` um Code, der `scale`-Objekte behandelt.

**Aufgabe 6 (1C+1D):**

Erweitern Sie die Funktion `pixel-color` um Code, der `colored`-Strukturen behandelt. (Dabei können Sie natürlich beliebig viele Hilfsfunktionen verwenden.) Der Code soll einen `color`-Wert zurückliefern, falls der Bildpunkt in dieser farbe gezeichnet werden soll, `#f` sonst.

Mit dem bisher geschriebenen Code läßt sich schon eine farbige Ellipse mit beliebigem Mittelpunkt zeichnen.

**Aufgabe 7 (1C+1D):**

Erweitern Sie die Funktion `pixel-color` um Code, der `above`-Strukturen behandelt. Dabei soll `#f` zurückgeliefert werden, falls der Kreis keines der beiden Unterobjekte (`lower` und `upper`) schneidet, einen `color`-Wert (oder im Falle von ungefärbten Objekten `#t`).

Nun können Graphiken aus mehreren Ellipsen gezeichnet werden.

**Aufgabe 8 (1C+1D):**

Schreiben Sie eine Funktion `polyline-hits` mit der Signatur `polyline circle -> boolean`, die für eine `polyline`-Struktur und einen Kreis `#t` zurückliefert, falls sich der Linienzug und der Kreis schneiden, `#f` sonst. Erweitern Sie `pixel-color` entsprechend.

**Aufgabe 9 (1C+1D):**

Erweitern Sie die Funktion `pixel-color` um Code, der `rotate`-Objekte behandelt.

**Aufgabe 10 (1C+1D):**

Erweitern Sie die Funktion `pixel-color` um Code, der `polygon`-Objekte behandelt.

**Aufgabe 11 (1C+1D):**

Falls Ihnen das alles zu langweilig war, können Sie das Programm noch etwas optimieren. Führen Sie dazu eine Struktur `rotate-mat` ein, in der die Rotationsmatrix (statt des Drehwinkels) abgespeichert ist. Erweitern Sie `pixel-color`, so daß es Ihre neue Struktur behandeln kann.

Schreiben Sie eine Funktion, die ein `picture` als Argument nimmt und alle `rotate`-Strukturen durch entsprechende `rotate-mat`-Strukturen ersetzt.