

Programmierpraktikum zur Informatik I

Testat am 4.12., CAP4, Raum 709

Weitere Informationen zum Praktikum finden Sie auf der Web-Seite zum Praktikum:

<http://www.informatik.uni-kiel.de/~jac/p1>

Bitte beachten sie, dass Sie sich bis zum 11.11. in der Studierenden Datenbank für dieses Praktikum (zu finden unter Übungen) angemeldet haben müssen! Nach diesem Termin ist eine Anmeldung nicht mehr möglich!

Einen genauen Termin für die Abnahme Ihrer Lösungen erhalten Sie in einer separaten Online-Anmeldung, welche Sie in der Woche vom 19. bis zum 23.11. vornehmen müssen.

Grundlagen

Ein beliebtes Beispiel der fraktalen Geometrie sind Apfelmännchen. In dieser Praktikumsaufgabe sollen Sie die zugrundeliegenden Algorithmen implementieren. Zur Berechnung des Apfelmännchens betrachtet man für komplexe Zahlen z_n und c die Folge

$$z_0 = 0 \qquad z_{n+1} = z_n^2 + c$$

Für $|z_n| > 2$ weiß man, dass diese Folge divergiert (genauer: $\lim_{n \rightarrow \infty} |z_n| = \infty$). Die *Mandelbrotmenge* ist als die Menge aller c definiert, für die diese Folge nicht (betragsmäßig gegen ∞) divergiert. Das Apfelmännchen stellt diese Mandelbrotmenge dar. Hierbei entspricht jeder Pixel des Bildschirms einer komplexen Zahl (x -Koordinate entspricht dem Realanteil, y -Koordinate entspricht dem Imaginäranteil), welche als Konstante c für die Berechnung der Folge $(z_n)_{n \in \mathbb{N}}$ verwendet wird. Bei der Berechnung approximiert man die Mandelbrotmenge durch die Betrachtung des Folgenglieds z_N mit einem festen $N \in \mathbb{N}$: Gilt $|z_N| > 2$, so bewertet man diesen Punkt als divergent.

Für eine hübsche Darstellung des Apfelmännchens färbt man zusätzlich auch die Bereiche, für die Divergenz gezeigt wurde. Hierfür bestimmt man das kleinste $n \leq N$ mit $|z_n| > 2$. Ab diesem Folgenglied weiß man sicher, dass die Folge divergiert. Man verwendet dieses kleinste $n \leq N$ zur Festlegung der Farbe. Alle Punkte, bei denen die Schwelle 2 mit dem Folgenglied z_n überschritten wird, werden mit der gleichen Farbe dargestellt. Um schöne Farbverläufe zu erhalten sollten Punkte mit nah beieinander liegenden n ähnlich eingefärbt werden.

Zur Implementierung des Apfelmännchens finden Sie auf der WWW-Seite zum Praktikum das Scheme-Teachpack `apfel.scm`, welches folgende Prozeduren und Konstanten zur Verfügung stellt:

- Die Konstanten `height` und `width` geben die Größe des Fensters in Pixeln an.
- `(farbe r g b)` erzeugt eine RGB-Farbe, wobei `r`, `g` und `b` Zahlen zwischen 0 und 1 sind und den entsprechenden Farbanteil für rot, grün und blau definieren.

- `(setpixel x y color)` färbt als Seiteneffekt den Pixel an der Position (x,y) mit der Farbe `color`, welche mittels `farbe` (s.o.) definiert wird; funktionales Ergebnis ist `true` und kann unberücksichtigt bleiben.
- `seq` führt zwei Berechnungen hintereinander aus und liefert das Ergebnis der zweiten Berechnung als Gesamtergebnis: `(seq (setpixel 10 20 (farbe 1 0 0)) (+ 35 7))` färbt den Pixel $(10,20)$ rot und liefert den Wert 42 als Ergebnis.

Aufgabe 1

Implementieren Sie unter Verwendung des *Teachpacks* folgende Funktionen.

Repräsentieren Sie komplexe Zahlen c als zwei Parameter `cR` und `cI`.

- Definieren Sie eine Scheme Funktion `mag`, welche den Betrag einer komplexen Zahl (repräsentiert als zwei Parameter) berechnet.
- Definieren Sie eine Funktion `calcPoint` zur Iteration der Folge z_n . Diese Funktion soll folgende Parameter erhalten:
 - die maximale Iterationstiefe $N \in \mathbb{N}$
 - die Konstante c , repräsentiert durch die Parameter `cR` und `cI`.

Als Ergebnis liefert `calcPoint` die Anzahl der Iterationen, welche benötigt wurden um die Divergenz der Folge zu zeigen. Konnte keine Divergenz gezeigt werden, liefert sie das Ergebnis 0.

Beachten Sie, dass DrScheme standardmäßig mit Brüchen maximaler Genauigkeit rechnet, was bei der Apfelmännchenberechnung sehr langsam werden kann. Verwenden Sie als Startwerte für Ihre Iterationen inexakte Zahlen (z.B. `#i0` statt 0).

- Definieren Sie drei verschiedene Funktionen zur (hübschen) Generierung der Farben. Als Parameter erhalten diese Funktionen die gewählte Tiefe $N \in \mathbb{N}$ und einen Wert $n \leq N$ und liefern eine RGB-Farbe als Ergebnis.
- Implementieren Sie Funktionen `pixelToCR` und `pixelToCI` zur Umwandlung der Bildschirmkoordinaten (übergeben als zwei Parameter `x` und `y`) in komplexe Zahlen (Real- und Imaginäranteil). Der zu berechnende Ausschnitt der komplexen Zahlenebene sei durch die Argumente `left`, `top` (für die linke obere Ecke) und `right`, `bottom` (für die rechte untere Ecke) gegeben. `left` und `right` sind also Realanteile und `top` und `bottom` Imaginäranteile der genannten Eckpunkte. Die Auflösung der Darstellung ist durch die globalen Konstanten `width` und `height` des *Teachpacks* gegeben.
- Die Berechnung des Apfelmännchenbildes erfolgt für jeden Pixel in folgenden Schritten:
 - Umwandlung der Pixelkoordinate in zugehörigen Wert der komplexen Zahlenebene
 - Bestimmung des Divergenzverhaltens dieses Punktes unter Berücksichtigung von N
 - Umrechnung des Ergebnisses in einen Farbwert

- Einfärbung des entsprechenden Pixels

Verwenden Sie die Prozedur `seq` um das Einfärben des Pixels in Ihre Iteration einzubauen.

- f) Definieren Sie eine Funktion (`apfelmaennchen left top right bottom N`), welche das Apfelmännchen mit übergebenen Parametern darstellt. Einen Überblick über das Apfelmännchen erhalten Sie mit dem Aufruf:

```
(apfelmaennchen -2 1.5 1 -1.5 50)
```

- g) Um das Apfelmännchen mit einer anderen Farbfunktion darzustellen, müssen Sie eine Veränderung tief in Ihrer Implementierung vornehmen. Schöner wäre es, wenn man die Farbfunktion über einen Parameter verändern könnte.

In Scheme ist es auch möglich Funktionen als Parameter zu übergeben. Erweitern Sie Ihre Funktion `apfelmaennchen` um einen Parameter `colorFun`, welcher im Aufruf an die konkrete Farbfunktion gebunden werden kann.

Beachten Sie bitte, dass Sie für die Verwendung von funktionalen Parametern den Sprachlevel auf Mittelstufe erhöhen müssen.

Aufgabe 2

In DrScheme sind komplexe Zahlen vordefiniert. Sie werden geschrieben als $r+ii$, wobei r der Realanteil und i der Imaginäranteil der komplexen Zahl darstellt. Alle Rechenoperationen sind auch für komplexe Zahlen definiert (`(+ 4+5i 1+2i)` ergibt `5+7i`). Außerdem kann mit der Funktion `magnitude` der Betrag einer komplexen Zahl berechnet werden.

- a) Überarbeiten Sie Ihre Implementierung so, dass Ihr Programm mit komplexen Zahlen rechnet. Wo können Sie nun geeignete Abstraktionen einfügen, die Ihren Code sinnvoll strukturieren? Warum war dies bei der vorherigen Implementierung nicht möglich?
- b) Erklären Sie, warum komplexe Zahlen einen abstrakten Datentypen bilden. Suchen Sie hierzu in der DrScheme Dokumentation nach den zugehörigen Konstruktoren, Selektoren und Operatoren.
- c) Durch die zusätzlich gewonnene Abstraktion, kann man auch eine weitere Funktion als möglichen Parameter von `calcPoint` (und den `calcPoint` aufrufenden Funktionen) identifizieren.

Überlegen Sie sich für diesen Parameter eine andere sinnvolle Funktion, mit dem Ihr Programm eine Variante des Apfelmännchens (bzgl. der definierenden Folge) berechnet.