Programmieren C: Fraktale - Apfelmännchen und Julia

Klaus Kusche

Zu Weihnachten widmen wir uns den schönen Seiten der Informatik: Den Fraktalen.
Heute stecken Fraktale in vielen Landschafts-Renderern und anderen Programmen,
die pseudo-natürliche Bilder generieren.
Wir befassen uns mit einer der Urformen der Fraktale: Den Mandelbrot-Mengen
(wegen ihres Aussehens auch "Apfelmännchen" genannt) und den Julia-Mengen.

Was sind Fraktale grundsätzlich? Fraktale sind mathematische Gebilde in der Ebene
oder im Raum, die unendlich detailliert bzw. strukturiert und "selbstähnlich" sind. Das
bedeutet, dass man an manchen Stellen (bei manchen Fraktalen sogar an allen Stellen)
beliebig tief in ein Fraktal hineinzoomen kann und dabei immer wieder auf neue
Strukturen und Gebilde trifft, die in ihrer Form alle demselben Grundmuster
ähnlich sind: Die Gebilde bestehen sozusagen aus unendlich vielen verkleinerten und
ev. leicht veränderten Kopien von sich selbst.

Wie werden Fraktale berechnet? Da gibt es viele verschiedene Varianten.
Die meisten beruhen auf Rekursion, andere sind Folgen, bei denen die Formel für
den nächsten Wert in jedem Schritt zufällig aus mehreren Möglichkeiten gewählt wird,
aber Mandelbrot- und Julia-Mengen lassen sich ganz einfach mit Schleifen berechnen
(wenn man *komplexe Zahlen* beherrscht).

Die mathematische Idee ist folgende (am Internet findet man anschauliche Erklärungen):

- Man betrachtet den Ausschnitt mit Radius 2 rund um den Nullpunkt
  der komplexen Zahlenebene.
- Für jede komplexe Zahl in diesem Bereich (in der Praxis: Für jede komplexe Zahl,
  die einem Pixel in der Anzeige entspricht) berechnet man die Zahlenfolge
  $z_{n+1} = z_n^2 + c$ (für eine komplexe Konstante c und einen Anfangswert z_0).
- Je nachdem, ob diese Folge z_n ewig innerhalb des Bereiches mit Radius 2 bleibt
  oder nicht, gehört die Ausgangszahl zur Mandelbrot- bzw. Julia-Menge oder nicht.
- Mandelbrot und Julia unterscheiden sich nur in der Initialisierung:
  - Bei den Mandelbrot-Mengen ist die Konstante c die komplexe Zahl
    (Position in der Zahlen-Ebene), die man testet, und der Anfangswert z_0
    der Folge ist ein global vorgegebener, fixer Parameter (meist 0).
  - Bei den Julia-Mengen ist es genau umgekehrt:
    Die zu testende Zahl selbst ist der Anfangswert z_0 ,
    und die Konstante c ist der global vorgegebene fixe Parameter.

In der Praxis werden die Bilder im einfachsten Fall wie folgt berechnet:

- Man berechnet der Reihe nach jeden Bildpunkt einzeln,
  d.h. man braucht *zwei geschachtelte Schleifen über alle Pixel* in x- und y-Richtung.
- Man verwandelt die x/y-Koordinaten jedes Bildpunktes in *Gleitkomma-x/y-Werte*,
  die dem entsprechenden *mathematischen Punkt des anzuzeigenden Ausschnittes*
  der Fläche [-2,2] * [-2,2] entsprechen.

- Für jeden dieser x/y-Werte berechnet man *immer wieder die oben genannte Formel*, und zwar mit 2 Abbruch-Kriterien:
    - Wenn der x/y-Wert aus dem Kreis mit Radius 2 *herausfällt* (d.h. $x^2 + y^2 >= 4$ wird), so gibt man dem Bildpunkt eine *Farbe*, die zyklisch von der *Anzahl* der bisher gemachten *Wiederholungen* der Formel für diesen Punkt abhängt (d.h. man hat eine fixe Anzahl verschiedener, vordefinierter Farben, die einen geschlossenen Farbkreis (d.h. Farbverlauf) bilden, setzt *f* auf "Rest von *'Anzahl der Schleifendurchläufe'* geteilt durch *'Anzahl der Farben'* ", und malt den Bildpunkt in Farbe Nummer *f*).
    - Wenn eine fix vorgegebene *Maximalzahl von Wiederholungen* erreicht ist (üblich sind 200 bis 1000, je mehr, umso schöner sind die Ränder zwischen schwarzen und bunten Bereichen, aber umso langsamer ist die Berechnung), so bricht man ab und lässt den Bildpunkt *schwarz*.

Zur Programmierung:

- Für die Grafik verwenden wir die SDL, die du auf meinen Webseiten samt Installationsanleitung zum Download findest. Lege dein Programm laut Anleitung im DevCpp als Projekt an, nicht als einzelnen C-File!

  Welche Funktionen wie aufgerufen werden müssen, um etwas graphisch anzuzeigen, entnimmst du der von mir bereitgestellten Beschreibung der Funktionen in **`sdlinterf.h`** und meinen beiden Beispiel-Programmen **`demo1.c`** und **`demo2.c`**. In meiner Sdlinterf ist die Berechnung des zyklischen Farbenkreises schon in meine Punkt-Zeichen-Funktion **`sdlDrawCyclicPoint`** eingebaut, es reicht, dieser Funktion eine *beliebige ganze Zahl >= 0 als Farbwert* zu übergeben.

  Du kannst die Fenstergröße **`SDL_X_SIZE`** und **`SDL_Y_SIZE`** in **`sdlinterf.h`** an deinen Computer anpassen.
- Dein Programm wird mit 9 Argumenten auf der Befehlszeile aufgerufen (die du prüfen solltest!):
    - Den x- und y-Koordinaten des *Bildmittelpunktes* (beides Gleitkomma-Zahlen zwischen -2.0 und 2.0).
    - Dem x- und y-Wert des *fixen Parameters* für die Mandelbrot- bzw. Julia-Formel (ebenfalls beides Gleitkomma-Zahlen zwischen -2.0 und 2.0).
    - Dem *Limit* für die Anzahl der *Wiederholungen* der Zahlenfolgen-Schleife (eine positive ganze Zahl, bei deren Überschreitung das Pixel schwarz ist).
    - Einem Parameter, der zwischen *Julia* (0) und *Mandelbrot* (1) umschaltet.
    - Dem *anfänglichen Zoom* (eine Gleitkommazahl >= 1.0), dem *Faktor*, um den der Zoom bei jedem Bild erhöht wird, und dem *Zoom*, bei dessen Erreichen das Programm enden soll (auch beides Gleitkommazahlen > 1.0).
- Die *äußerste Schleife* steigert den *Zoom*: Er wird vom Start- bis zum End-Zoomwert bei jedem Durchlauf mit dem angegebenen Faktor multipliziert.
  Für jeden Zoom-Wert wird ein komplett neues Bild gerechnet:
    - Zuerst wird der interne Bildspeicher gelöscht (dadurch erspart man sich das Zeichnen einzelner schwarzer Bildpunkte),

    - dann wird einzeln der Reihe nach für alle Bildpunkte die Farbe berechnet und gesetzt (zwei ineinander geschachtelte Schleifen über alle Pixel),
    - und schließlich wird die Anzeige aktualisiert.
- Zur Berechnung eines Bildpunktes innerhalb der drei soeben genannten Schleifen:
    - Wir rechnen nicht mit komplexen Zahlen, sondern mit zwei getrennten **`double`**-Koordinaten (x und y) pro komplexer Zahl. Wir brauchen also je ein x und y für die aktuelle Folgenzahl z_n und für die Konstante c.
    - Zuerst einmal muss man die Pixelkoordinaten der beiden Schleifen auf mathematische Gleitkomma-Koordinaten im Bildausschnitt umrechnen. Für einen x-Pixelwert zwischen 0 und (SDL_X_SIZE – 1) und einen Gleitkomma-Zoomfaktor *zoom* leistet folgende Formel das Gewünschte:

      $x_{math} = x_{bildmittelpunkt} + 4 * (x_{pixel} - SDL\_X\_SIZE / 2) / (maxpixel * zoom)$

      (analog für y)

      maxpixel muss dabei für x und y *gleich* sein, und zwar der *kleinere Wert* von SDL_X_SIZE und SDL_Y_SIZE, sonst wird das Fraktal verzerrt (in einer Richtung gestaucht oder gedehnt).

      zoom ist größergleich 1, die Formel ist so ausgelegt, dass das Fenster bei *zoom* gleich 1 eine Breite bzw. Höhe von 4 hat (d.h. bei Bildmittelpunkt im Ursprung von -2 bis +2, den für Mandelbrot / Julia "interessanten" Bereich der komplexen Zahlenebene), deshalb 4 * … .
    - Dann müssen der x- und y-Wert der Konstante c (x_c und y_c) und der Anfangszahl der Folge z_n (x_z und y_z) mit dem eben berechneten x_{math} und y_{math} sowie dem fixen x- und y-Parameter aus der Befehlszeile initialisiert werden, und zwar je nachdem, ob Mandelbrot- oder Julia-Menge berechnet werden soll, in dieser oder in der umgekehrten Reihenfolge (siehe oben).
    - Schließlich kommt eine Schleife, die in jedem Durchlauf das nächste x_z und y_z berechnet und dabei die Durchläufe mitzählt:
        - Zuerst wird geprüft, ob $x_z^2 + y_z^2 >= 4$ ist. Wenn ja: Den Bildpunkt zeichnen (der Durchlaufzähler ist der Farbwert) und die Schleife verlassen.
        - Sonst wird das nächste z_n berechnet. Aufgespalten in x und y ergibt die komplexe Formel von oben: $x_{pwr} = x_z^2 - y_z^2$; $y_{pwr} = 2*x_z y_z$; $x_z = x_c + x_{pwr}$; $y_z = y_c + y_{pwr}$;

          Achtung: Wie angegeben zuerst die x- und y-Werte von z_n^2 berechnen (das x_{pwr} und das y_{pwr}) und erst dann die neuen x_z und y_z, nicht zuerst x_z fertig rechnen und dann erst mit y_z beginnen, denn y_z muss noch auf Basis des *alten* x_z gerechnet werden und nicht des neuen!
        - Wenn die maximale Anzahl von Durchläufen (auf der Befehlszeile angegeben) erreicht wird: Nichts zeichnen, Schleife verlassen!

Tipps für schöne Bilder:

- Ich stelle einige Aufrufe, die ich selbst probiert habe, auf meine Webseite.

- Schön strukturierte Julia-Mengen erhält man dadurch, dass man sich zuerst einen Bildmittelpunkt im Apfelmännchen sucht, der bis in große Tiefe hohe Detail-Vielfalt zeigt, und dann die Koordinaten dieses Punktes als Konstante (nicht als Mittelpunkt) bei der Julia-Berechnung angibt.
- Der Farb-Effekt wird etwas lebendiger, wenn man beim Zeichnen eines Pixels für den Farbwert zum Schleifendurchlauf-Zähler noch eine Variable addiert, die anfangs 0 ist und die man bei jedem Durchlauf der Zoom-Schleife (also bei jedem neuen Bild) um eins (oder 2...5 für schnellere Farbwechsel) erhöht.

Zusatzaufgaben:

- **Winkel-Färbung:**

  Neben der Färbung der Bildpunkte (zyklische Farbnummer) abhängig von der Anzahl der Durchläufe der inneren Schleife gibt es noch eine zweite Variante: Sobald man die innere Schleife wegen $x_z^2 + y_z^2 >= 4$ verlässt, berechnet man den *Winkel*, den dieser Punkt x_z / y_z vom Ursprung 0 / 0 aus gesehen hat. Dieser Winkel legt die Farbe des Punktes fest. Derartige Bilder sind in den "schönen" Bereichen des normalen Apfelmännchens chaotisch bunt, aber in den eher eintönigen Bereichen wunderschön strukturiert gefärbt.

  Im Detail wird der zyklische Farbwert wie folgt berechnet:

  - Die Funktion **`atan2`** (aus **`math.h`**) liefert zu den Koordinaten x / y den entsprechenden Winkel, und zwar gemessen in Radiant von $-\pi$ bis π (Achtung: Erster Parameter von **`atan2`** ist y, zweiter Parameter ist x !):

    $w = \text{atan2}(y_z, x_z)$

  - Dann rechnet man w ($-\pi$ bis π) in die Farbe f (0 bis 192) um (denn der Farbkreis, der in meiner SDL-Funktion eingebaut ist, hat 192 verschiedene Farben), wobei man für π die Konstante **`M_PI`** aus **`math.h`** verwendet:

    $f = (w / \pi + 1) * 96$ (in einen **`int`** umgewandelt)

- **Andere Formeln:**

  Am Internet findet man Dutzende Formeln, die sich statt $z_{n+1} = z_n^2 + c$ verwenden lassen. Die Berechnung ist allerdings mühsam, wenn man keine komplexe Arithmetik verwendet und die Formel in getrennte Berechnungen für x und y aufspalten muss.

  Am einfachsten ist es, statt dem Quadrat *höhere Potenzen* zu implementieren. Das erhöht die Anzahl der Symmetrie-Achsen der berechneten Gebilde: Bei z_n^5 ist die Julia-Menge beispielsweise ein Gebilde mit 5 symmetrischen Armen, und das Apfelmännchen ist spiegelbildlich zur x- und y-Achse (außerdem werden beim Apfelmännchen die bunten Ränder dünner, aber stärker strukturiert).

  Für die 3. Potenz ergibt sich $x_{pwr} = x * (x^2 - 3*y^2)$ und $y_{pwr} = y * (3*x^2 - y^2)$, für die 5. Potenz ergibt sich $x_{pwr} = x * (x^4 - 10*x^2y^2 + 5*y^4)$ und $y_{pwr} = y * (5*x^4 - 10*x^2y^2 + y^4)$. Aus Performance-Gründen ist es sinnvoll, x^2 und x^4 sowie y^2 und y^4 nicht zwei Mal auszumultiplizieren, sondern einmal zu berechnen und in Hilfsvariablen zu speichern.

- Weiters könntest du natürlich die *Parameter* statt von der Befehlszeile *aus einer Datei lesen*, damit man eine "Fraktal-Show" automatisch ablaufen lassen kann.