Programmieren C: Rekursion, globale Variablen: Raumfüllende Kurven

Klaus Kusche

Raumfüllende Kurven sind eine mathematische Spielerei: Es handelt sich um rekursiv definierte geometrische Muster, die zwei Punkte mit einem aus mehreren Teilstrecken zusammengesetzten Linienzug verbinden (d.h. man kann solche Muster „in einem Zug“ ohne Absetzen des Bleistifts Linie für Linie zeichnen, die Linien überkreuzen sich nicht).

Erhöht man die Rekursionstiefe um 1, wird jede einzelne gerade Teilstrecke des Linienzuges wieder durch eine verkleinerte Kopie des ganzen Linienzuges ersetzt. Man nennt sie deshalb auch „selbstähnliche“ Gebilde, wissenschaftlich „Fraktale“.

Würde man die Rekursionstiefe beliebig wachsen lassen, hätte man zwischen Startpunkt und Endpunkt auf endlicher Fläche einen unendlich langen und aus unendlich vielen unendlich kurzen Teilstücken bestehenden Linienzug.

Es gibt viele solcher Kurven, wir beschränken uns auf Muster mit 90-Grad-Winkeln und Längen, die in ganzzahligen Verhältnissen zueinander stehen (viele solche Kurven enthalten 60-Grad Winkel und Längenverhältnisse mit Wurzeln).

Die diagonale Peano-Kurve

Die einfachste solche Kurve ist die diagonale Peano-Kurve, sie verbindet zwei diagonal gegenüberliegende Ecken eines Quadrates und kommt mit einer einzigen rekursiven Funktion aus. Ich gebe das Hauptprogramm vor, du brauchst nur mehr die rekursive Zeichen-Funktion dazuschreiben.

Um die Parameter-Liste der Funktion kurz zu halten,
verwenden wir *einige globale Variablen*:

- Die aktuelle *x- und y-Pixel-Koordinate* **`xPos`** und **`yPos`**, an der wir gerade zeichnen (0/0 ist links oben). Sie wird bei jedem Zeichnen eines Striches verändert.
- Die *Richtung* **`richtung`**, in die wir die nächste Linie zeichnen,
  bzw. wie oft wir uns seit dem Start um 90 Grad gedreht haben.

  Wir haben eigentlich nur 4 Richtungen, aber da wir uns je nach Rekursionstiefe auch öfter als drei Mal um 90 Grad drehen können, kann der Wert beliebig groß und beliebig klein werden.

  Wir betrachten daher immer den *Rest* des Richtungs-Wertes bei Division durch 4:
  0 = rechts runter (+x+y), 1 = rechts rauf (+x-y),
  2 = links rauf (-x-y), 3 = links runter (-x+y)

  Erhöht man den Richtungswert um 1, dreht man sich also 90 Grad gegen den Uhrzeigersinn, zieht man 1 ab, dreht man sich um 90 Grad im Uhrzeigersinn.
- Die *Länge (in Pixel)* **`schritt`** eines einzelnen Teilstriches in x- bzw. y-Richtung.
- Die *Verzögerung* **`verz`** (in Millisekunden), die wir nach jedem Teilstrich warten, damit man der Kurve beim Wachsen zusehen kann.

Nur die *Rekursionstiefe* übergeben wir der Funktion *als Parameter*.
Sie wird *bei jedem rekursiven Aufruf um 1 reduziert*.

Unsere Funktion **zeichne** besteht im Wesentlichen aus einem großen **if** mit **else**:

- Wenn unsere Rekursionstiefe *gleich 0* ist:
  Ende der Rekursion, zeichne einen *einfachen Strich*. Im Detail:
  - *Merk* dir die in den globalen Variablen gespeicherte *alte Position*.
  - *Berechne* abhängig von der aktuellen *Richtung* (ein **switch**-Befehl wäre sinnvoll!) die *neue Position* (Strichlänge zur x- und y-Koordinate dazu- oder wegrechnen!) und *speichere* sie in den globalen Variablen.
  - *Zeichne eine Linie* von der alten zur neuen Position.
    Dafür gibt es die Funktion **sdlDrawLine(***x1***,** *y1***,** *x2***,** *y2***,** *r***,** *g***,** *b***);**

    Unsere *Farbe* machen wir *abhängig von der aktuellen Richtung*:
    Ich habe die Rot-, Grün- und Blau-Werte von 9 Farben in globalen Arrays **R**, **G** und **B** gespeichert, verwende **richtung % 9** als Index in diese Arrays.
  - Aktualisiere die *Anzeige* (**sdlUpdate();**)
    und warte die *Verzögerungszeit* (**sdlMilliSleep(verz);**).
- Wenn unsere Rekursionstiefe *größer als 0* ist:
  Setze die Verbindung aus *Teilstücken mit Richtungsänderungen* dazwischen zusammen, jedes Teilstück ist ein *rekursiver Aufruf*.

  Die „Formel" der Peanokurve ist folgende: **S-S+S+S+S-S-S-S+S**
  Jedes S steht für ein *Teilstück* (*rekursiver Aufruf* mit um 1 kleinerer Tiefe),
  jedes + für eine *90-Grad-Drehung* gegen den Uhrzeigersinn (Richtung *erhöhen*),
  jedes - für eine *90-Grad-Drehung* im Uhrzeigersinn (Richtung *erniedrigen*).

Die Hilbert-Kurve

Hier ist die Zeichen- und Rekursionslogik komplizierter:

- Die Funktion **zeichne** ist *nicht rekursiv* und hat daher *keinen Tiefen-Parameter*:
  Sie macht immer das, was bei der Peano-Kurve bei Tiefe gleich 0 passiert ist:
  Sie zeichnet einen *einfachen Strich*, nur sind die Striche jetzt nicht mehr schräg, sondern je nach Richtungs-Wert *waagrecht oder senkrecht*:
  0 = rechts (+x), 1 = rauf (-y), 2 = links (-x), 3 = runter (+y)
- Getrennt davon gibt es jetzt *zwei rekursive Funktionen* **lr** und **rl**
  für eine Links-Rechts- und eine Rechts-Links-Strichkombination (*spiegelbildlich*).

  Sie haben folgende „Formeln" für eine *Tiefe größer 0*
  (- … dreh dich rechts, + … dreh dich links, S … **zeichne** einen einfachen Strich,
  R … rekursiver **rl**-Aufruf mit um 1 kleinerer Tiefe,
  L … rekursiver **lr**-Aufruf mit um 1 kleinerer Tiefe):

  lr: **-RS+LSL+SR-**
  rl: **+LS-RSR-SL+**

  Bei einem Aufruf mit *Tiefe 0* kehren beide sofort *zurück*, *ohne irgendetwas zu tun*.

Die normale Peano-Kurve

… funktioniert genauso, aber mit komplizierteren Formeln:

lr: **LSRSL+S+RSLSR-S-LSRSL** **rl**: **RSLSR-S-LSRSL+S+RSLSR**