Programmieren C: Grafik, Mathematik, `double`-Schleifen, ...: Lissajou'sche Kurven

Klaus Kusche

Die Lissajou'schen Kurven kommen eigentlich aus der Analog-Elektronik:
Das sind die kronenförmigen, geschwungenen Linien, die im Oszilloskop entstehen, wenn man sowohl an die x-Achse als auch an die y-Achse eine Sinusschwingung legt, und die Frequenzen der beiden Schwingungen verschiedene *ganzzahlige Vielfache derselben Grundfrequenz* sind (z.B. x-Achse 400 Hz = 4 * 100, y-Achse 300 Hz = 3 * 100).

Die *Form der Kurve* hängt nicht von den absoluten Frequenzen (der Grundfrequenz) ab, sondern nur von ihrem *Frequenzverhältnis* zueinander (den beiden Multiplikations-Faktoren, hier 4 und 3) und der relativen *Phasenlage* (Verschiebung) ihrer Schwingungen.

Das Programm enthält daher etwas Mathematik, ist aber sonst ganz kurz und einfach:
Es benötigt keine Arrays und keine komplizierten Programm-Strukturen.

Für die Grafik verwenden wir die SDL, die du auf meinen Webseiten
samt Installationsanleitung zum Download findest.

Du kannst die Fenstergröße **`SDL_X_SIZE`** und **`SDL_Y_SIZE`** in **`sdlinterf.h`** an deinen Computer anpassen, wenn das Fenster für deine Grafik-Auflösung zu klein / zu groß ist.

Wie wird das Programm aufgerufen?

Mit *zwei positiven ganzen Zahlen*, die angeben, *welches Vielfache der Grundfrequenz* die x- und y-Schwingung ist, und einer *Kommazahl*, die angibt, *wie schnell* die Kurve rotiert (das ist die Schrittweite der Phasen-Schleife, z.B. 0.001, siehe unten).

Wie zeichnet man so eine Kurve?

Ganz einfach: *Punkt für Punkt*, einen nach dem anderen. Wir brauchen also eine Schleife.

Um *einmal die ganze Kurve* zu zeichnen, muss die Grundschwingung *eine* komplette Sinus-Welle durchlaufen. Die Funktion **`sin`** hat als Argument einen Winkel, und für eine komplette Sinus-Welle muss der Winkel von 0 bis 360 Grad gehen.
Winkel werden aber am Computer immer in *Radiant* gemessen und nicht in Grad:
360 Grad entsprechen $2*\pi$ Radiant. Unsere Schleife braucht also
eine **`double`**-*Zählvariable*, und die muss von 0 bis $2*\pi$ zählen.

Hinweis: **`sin`** kommt aus **`math.h`**, **`M_PI`** für π auch.

Jetzt fragt sich noch mit welcher *Schrittweite* wir zählen. Da pro Schritt ein Bildpunkt gezeichnet wird, stellt man das am besten durch "scharfes Hinsehen" ein:

- Ist die Schrittweite *zu groß*, dann sind die Punkte zu weit auseinander, und es entstehen zwischen den einzelnen Punkten *Löcher in der Kurve*.
- Ist die Schrittweite *zu klein*, liegen die Punkte so dicht, dass viele Bildpunkte mehrfach übereinander gezeichnet werden. Das schadet zwar nichts, zu viele Punkte auszurechnen *verschwendet aber unnötig Rechenzeit*.

- Bei 800 * 600 Pixel hat sich für die einfache Lissajou-Kurve (x- und y-Frequenz gleich Grundfrequenz, daraus ergibt sich ein Kreis oder eine Ellipse) ein Wert von 1/500 als Schrittweite bewährt. Bei höherer Auflösung bzw. größerem Fenster braucht man entsprechend mehr Bildpunkte, d.h. kleinere Schrittweite (z.B. 1/800).
- Ist die x- oder y-Frequenz ein Vielfaches der Grundfrequenz, wird die Kurve länger, und es müssen entsprechend mehr Punkte gezeichnet werden: Die Schrittweite muss daher *kleiner* sein. Wenn *f* der *größere* der beiden Frequenz-Faktoren ist, so braucht man eine Schrittweite von 1/(*f* * 500)
  (achte darauf, dass das eine *Gleitkomma-Division* sein muss,
  denn bei einer ganzzahligen Division käme immer 0 heraus!).
  Am besten rechnest du die Schrittweite am Anfang des Programmes aus
  und speicherst sie in einer eigenen Variablen.

Wie zeichnet man einen einzelnen Punkt der Kurve?

Wir haben einen Winkel *winkel* der Grundschwingung (die Zählvariable der soeben besprochenen Schleife für den Winkel-Wert), einen Phasenwinkel *phase* (siehe unten) und (aus der Eingabe) die beiden ganzzahligen Faktoren *xfaktor* und *yfaktor*, die angeben, welche Vielfachen der Grundfrequenz die x- und die y-Frequenzen sind.

Der *Sinus-Wert* für die x- und y-Richtung berechnet sich daraus wie folgt:

x = sin(*winkel* * *xfaktor*) und y = sin(*winkel* * *yfaktor* + *phase*)

Das ergibt Werte zwischen -1 und 1, aus denen wir noch *Pixelkoordinaten* zwischen 0 und (SDL_X_SIZE – 1) bzw. (SDL_Y_SIZE – 1) machen müssen (oder besser zwischen 10 und (SDL_X_SIZE – 10), damit die Kurve nicht ganz am Fensterrand klebt). Das geht mit

x_{pixel} = SDL_X_SIZE / 2 + x * (SDL_X_SIZE / 2 – 10) (und in selber Art für y)

und an diese Stelle kommt der Bildpunkt für die aktuellen Winkelwerte.

Warum bewegt sich die Kurve noch nicht?

Am Oszilloskop entsteht die Bewegung dadurch, dass die Frequenzen eben *nicht exakt ganzzahlige Vielfache* sind:

Durch kleine Fehler in den Frequenzen *ändert sich die Phasenverschiebung*
(das ist der Unterschied zwischen den Winkeln der beiden Schwingungen:
An welcher Stelle der Sinuswelle ist y, wenn die Sinuswelle von x im Nulldurchgang ist?)
zwischen x- und y-Schwingung ganz langsam, und das lässt die Kurve rotieren.

Am Computer können wir die schon gezeichnete Kurve nicht mehr verschieben, sondern wir müssen sie mit leicht geänderter Phasenverschiebung immer wieder neu zeichnen.

Wir brauchen also ganz *außen noch eine Schleife* für die Phase. Das ist auch eine Schleife mit einer **double**-*Zählvariable*, eben dem Unterschied zwischen den beiden Winkeln. Dieser Unterschied fängt bei 0 an und wird bei jedem Durchlauf ganz wenig erhöht (die Schrittweite wird eingegeben, je nachdem, wie schnell die Kurve rotieren soll: Zwischen 0,0001 und 0,01 wirkt gut).

Diese Schleife läuft endlos bis wir das Programm abbrechen.
In dieser Schleife passiert jedesmal folgendes:

- Den Inhalt des Grafikfensters löschen.
- Die ganze Kurve wie oben beschrieben komplett neu zeichnen (mit der Schleife über den Winkel).
- Das Grafikfenster aktualisieren.