Programmieren C: Mathe, Grafik, `double`-schleifen: Schwingung, Ballflugbahn

Klaus Kusche

Wir wollen ein paar physikalische Effekte grafisch veranschaulichen.

Für die Grafik verwenden wir wieder die SDL, für die auf meinen Webseiten eine Installations- und Benutzungsanleitung liegt.

1.) Gedämpfte Schwingung

Wir wollen den Verlauf einer *gedämpften Sinusschwingung* zeichnen, wie sie z.B. in elektronischen Schaltkreisen mit Verlusten oder bei einem Pendel mit Reibung entsteht.

Unser Programm erwartet 3 *Eingaben* auf der Befehlszeile (alles Kommazahlen):

- *Wie viele Schwingungen* n soll unser Fenster in x-Richtung darstellen?
- Was ist die *Phasenlage* ω links am Beginn der Kurve,
  d.h. beginnt sie im Nulldurchgang oder in einem Maximum?

  Die Phase ω wird in *Winkelgrad* eingegeben, aber sofort
  auf *Bogenmaß* umgerechnet (Eingabewert * π / 180, π ist **`M_PI`** aus **`math.h`**).
- Wie stark ist die *Dämpfung* d (0 … ungedämpft)?

Unser Programm zeichnet die Kurve Pixel für Pixel in einer Schleife:

- Die Schleife geht über den *Winkel* α der Sinusschwingung
  (als Komma-Zahl, in Bogenmaß, d.h. 0 … 2π ist eine komplette Sinus-Welle),
  und zwar *in kleinen Schritten* (z.B. 0.002, schöne Konstante definieren!).
- Zu jedem Winkel α müssen wir den *Funktionswert* ausrechnen:

  $f(\alpha) = \exp(-\alpha * d) * \sin(\alpha+\omega)$

  (**`exp(`***x***`)`** ist e^x, die Funktion kommt so wie **`sin`** aus **`math.h`**)
- Und dann müssen wir das α in eine *x-Koordinate*
  und das $f(\alpha)$ in eine *y-Koordinate* umrechnen
  und das *Pixel* mit **`sdlDrawPoint`** *zeichnen*
  (überlege, welche Multiplikationen, Divisionen usw. du dafür brauchst):
  - Das α geht von 0 bis $2*\pi*n$ (n … Anzahl der Schwingungen),
    das x soll dabei von 0 bis **`SDL_X_SIZE`** gehen.
  - Das $f(\alpha)$ liegt zwischen -1 und 1,
    daraus muss ein y zwischen 0 und **`(SDL_Y_SIZE-1)`** werden
    (1 entspricht 0, -1 entspricht **`(SDL_Y_SIZE-1)`**).

  Sobald dabei ein x-Wert größergleich **`SDL_X_SIZE`** herauskommt,
  endet unsere Schleife (*ohne* Zeichnen!).

Vor der Schleife braucht man ein **`sdlInit`** und ein **`sdlSetBlack`**,
nach der Schleife ein **`sdlUpdate`**, ein **`sdlMilliSleep`** und ein **`sdlExit`**
(die kannst du dir aus einem der Demo-Beispiele abschauen).

2.) Aufspringender Ball

Wir wollen die Flugbahn eines *schräg nach oben geworfenen Gummiballes* aufzeichnen. Der Einfachheit halber beschränken wir uns darauf, die *Elastizität des Balles* vorgeben zu können (1 … theoretisch ideales Sprungverhalten, 0 … springt gar nicht), alle anderen Effekte wie Reibung, Luftwiderstand, Drall usw. vernachlässigen wir.

Die Grundstruktur ist genau dieselbe wie im vorigen Beispiel, nur die Berechnung ist etwas komplizierter. Außerdem machen wir das **`sdlUpdate`** diesmal gleich *in* der Schleife, damit man der Flugbahn beim Entstehen zusehen kann.

Wir brauchen wieder drei *Eingaben*: Die *Wurfgeschwindigkeit* v, den *Abwurfwinkel* α (zwischen 0 und 90, bei der Eingabe wieder umrechnen von Grad auf Bogenmaß!) und die *Elastizität* k des Balles (siehe oben).

Weiters brauchen wir die *Gravitationskonstante* g (9.81) und 5 *Komma-Variablen*:

- Die *Geschwindigkeit* v_x in *x-Richtung*. Sie ist konstant $v * \cos(\alpha)$.
- Den *Aufwärtsanteil* v_y (ohne Erdanziehung) der *Geschwindigkeit in y-Richtung*. Er ist am *Anfang* $v * \sin(\alpha)$ und *ändert sich* bei jedem Aufschlag (siehe unten).
- Die *Zeit* t_x *seit dem Wurf*. Das ist unsere *Schleifen-Variable*, wir erhöhen sie wieder in ganz *kleinen Schritten* (0,002) und zeichnen *jedesmal ein Pixel*.
- Die *Zeit* t_y *seit dem letzten Aufschlag*. Auch sie beginnt bei 0 und wird in der Schleife jedesmal um *denselben Schritt* wie t_x erhöht, aber zusätzlich *bei jedem Aufschlag* auf 0 gesetzt.
- Die *Höhe* h des Balles. Sie wird jedesmal neu berechnet aus linearem Aufwärtsanteil und freiem Fall: $h = v_y * t_y - (g/2) * t_y^2$.
- Ist die Höhe *kleinergleich* 0, haben wir einen *Aufschlag* (außer beim allerersten Schleifendurchlauf mit $t_x = 0$!):

  Wir setzen h und t_y auf 0 und berechnen die *neue Vertikalgeschwindigkeit* $v_y = k * v_y$

  (die Aufschlag-Geschwindigkeit ist gleich der alten Absprung-Geschwindigkeit, wenn man Luftreibung usw. vernachlässigt, und die neue Absprung-Geschwindigkeit ist um den Faktor k geringer).
- Die Berechnung der *Pixelkoordinaten* ist diesmal einfach: x ist $v_x * t_x$ und y ist **`(SDL_Y_SIZE - 1)`** - h .
- Die Schleife endet, wenn x größergleich **`SDL_X_SIZE`** oder y kleiner 0 wird.