Programmieren C++: Erste Klassen und Objekte: Grafik

Klaus Kusche

1. Installiere die SDL und versuche, das *Beispielprogramm* von meiner Webseite zum Laufen zu bringen.
2. Lies dir den Code des Beispielprogramms durch und versuche, ihn zu verstehen.
3. Mach *kleinere Änderungen* am Beispielprogramm, z.B. einen zweiten Punkt im Hauptprogramm anlegen, die Farbe des Punktes nachträglich ändern, …
4. Mein **`sdlinterf.h`** enthält auch eine Funktion zum Zeichnen eines Rechteckes.
   - Baue die Klasse für Punkte auf eine **Klasse für Rechtecke** um (und ändere ihren Namen auf **`Rect`**):
     - Die bisherigen Koordinaten des Punktes bleiben erhalten und definieren den *Mittelpunkt* des Rechteckes.

       Dazu kommen *zwei zusätzliche Member-Variablen* und *zwei zusätzlichen Parameter im Konstruktor* für die *Ausdehnung* des Rechteckes in Pixeln ab Mittelpunkt in x- und y-Richtung, also die halbe Breite und halbe Höhe (diese Darstellung ist im Hinblick auf die Weiterentwicklung des Beispiels günstiger als eine Darstellung mit linker oberer Ecke, Breite und Höhe).
     - Weiters muss die Implementierung von **`draw()`** und **`undraw()`** jetzt meine *SDL-Rechteck-Zeichen-Funktion* aufrufen.
     - Wenn dein Rechteck so wie der Punkt "Ping-Pong spielen" soll, muss man in der Methode **`fly`** die *Bedingungen* in den **`if`**'s für die Richtungsänderung *erweitern*: Das Rechteck muss nicht erst dann umdrehen, wenn sein Mittelpunkt den Fensterrand erreicht, sondern schon dann, wenn die *Rechteckseite* anschlägt. Man muss daher bei der x-Prüfung die *Rechteck-Breite dazu- bzw. wegzählen* und bei der y-Prüfung die Rechteck-Höhe.
   - Schreib ein paar zusätzliche *Methoden*:
     - Eine Methode **`setSize`** mit zwei Parametern: Sie soll die *Größe*, d.h. die x- und y-Ausdehnung des Rechteckes, *neu setzen*.
     - Eine Methode **`scale`**: Sie wird mit *zwei* **`int`**-Werten aufgerufen, die *Prozentwerte* darstellen, mit denen *die Breite und die Höhe skaliert* wird (d.h. **`100`** … bleibt gleich, **`200`** … doppelt so groß, **`50`** … halb so groß).

     Alle diese Methoden sollen als erstes das alte Rechteck weglöschen und am Ende das neue Rechteck zeichnen!
     - Auch *zwei* **`get`**-*Methoden* für die beiden neuen Member wären praktisch, damit man im **`main`** die *aktuelle Größe des Rechteckes abfragen* kann.
   - Ändere dein *Hauptprogramm*, um ein Rechteck statt einen Punkt anzulegen (der Konstruktor braucht *mehr Parameter*, und ich habe auch die Geschwindigkeit erhöht) und um die neuen Methoden zu testen!

Ich habe Folgendes gemacht:

- Jedesmal, nachdem das Rechteck *vom Rand abgeprallt* ist (das erkennt man am *Returnwert* von **`fly`**), habe ich es *um 10 % verkleinert*.
- Wenn dadurch entweder *die Breite oder die Höhe* **`0`** geworden sind, habe ich es wieder *in die Bildschirm-Mitte* gesetzt, ihm eine *neue, zufällige Breite und Höhe* zwischen **`1`** und **`30`** sowie in beiden Richtungen eine *neue, zufällige Geschwindigkeit* zwischen **`-10`** und **`10`** (außer **`0`**) gegeben.

  Für die *Geschwindigkeit* habe ich eine *Hilfsfunktion* **`int randPM(int n)`** geschrieben, die mir eine *Zufallszahl* zwischen **`-n`** und **`+n`** liefert, aber *nicht* **`0`** (denn eine solche Zufallszahl brauchen wir noch öfters).

  Dein Programm sollte natürlich *bei jedem Lauf andere Zufallszahlen* würfeln!

5. Unsere Konstruktoren mit dem Member-Zuweisungen sind noch nicht wirklich professionell. Bau die *Konstruktoren* von **`Color`** und **`Rechteck`** so um, dass sie eine **Initialisierungsliste** verwenden!

   Wenn der Umbau geglückt ist, solltest du die *Default-Werte* für die Parameter des **`Color`**-Konstruktors wieder *entfernen* können, denn dann sollte *kein Standard-Konstruktor* für **`Color`** mehr benötigt werden!

   Warum wurde er bisher benötigt?