Programmieren 1 Übung: Geometrische Objekte: Klassen-Member & -Methoden: Rechtecke in Z-Reihenfolge neu zeichnen

Klaus Kusche

In Fortsetzung des vorigen Beispiels (nimm die Musterlösung von Übung 15 / Punkt 1-4, wenn du keine eigene Ausgangsbasis hast):

Bisher hat unser Programm das Problem, dass Bewegungen eines Rechtecks schwarze Löcher in früher gezeichneten Rechtecken hinterlassen, und dass jede Operation auf einem Rechteck bewirkt, dass es in den Vordergrund ("obenauf") kommt und alle dahinter liegenden Rehtecke verdeckt.

Das wollen wir lösen, und zwar wie folgt:

- Man speichert in jedem Rechteck einen *Tiefenwert* ("Z-Wert", ein **`int`**).

  Dieser legt fest, wie weit vorne oder hinten das Rechteck im Gesamtbild liegt, d.h. welche anderen Rechtecke es verdeckt und von welchen es verdeckt wird.

  Für diesen Z-Wert gibt es eine Set- und eine Get-Methode: **`setZ`** und **`getZ`**.

- Alle Rechtecke mit Z-Wert werden nach Z-Wert geordnet in ein *für alle Objekte der Klasse gemeinsames* Array eingetragen.

  Eine *Methode* **`drawAll`** geht dieses Array durch (von den im Bild ganz hinten liegenden niedrigen Z-Werten zu den vordersten hohen) und zeichnet alle darin enthaltenen Rechtecke neu.

- Wurde zu einem Rechteck noch kein Z-Wert angegeben, oder wird der Z-Wert auf 0 gesetzt, so ist das Rechteck nicht im Z-Array enthalten (bzw. wird aus dem Array entfernt) und wird daher auch von **`drawAll`** nicht frisch gezeichnet.

Im Detail sind dazu folgende Schritte notwendig:

1.) Im *Header-File* der Rechteck-Klasse:

   - Zuerst einmal braucht man eine Konstante für die Array-Größe (diese darf fix festgelegt werden, mach sie Klassen-intern, nicht global!).
   - Weiters braucht man eine *normale Membervariable* für den *Z-Wert* eines jeden Rechteckes und zwei *Klassenvariablen* (*statische Membervariablen*) für das *Array* und die *Anzahl* der darin gerade enthaltenen Rechtecke.

     Welchen Typ müssen die Array-Elemente haben (sie zeigen auf Rechtecke)?

     Außer den Methoden dieser Klasse soll niemand direkt auf den Z-Mechanismus zugreifen können. Wie sind die drei Member daher zu deklarieren?

   - Dann braucht man Deklarationen für die Methoden **`getZ`** (Code gleich inline!) und **`setZ`** sowie für die *Klassen-Methode* (statische Methode) **`drawAll`**.
   - In der Initialisierung neuer Rechtecke muss der Z-Wert auf 0 gesetzt werden.
   - Wird ein Rechteck gelöscht, so muss es sich selbst aus dem Z-Array *entfernen* (Code nicht nochmals schreiben, sondern **`setZ`** mit Z-Wert 0 aufrufen!).

2.) Im *Implementierungs-File* der Rechteck-Klasse:

a) Zuerst einmal muss der Implementierungsfile die *Definitionen* der beiden Klassenvariablen enthalten.

b) Weiters sind die beiden Methoden **`setZ`** und **`drawAll`** zu implementieren.

`setZ` ist eine relativ lange Methode:

- Wenn das Rechteck im Array enthalten ist (schon bisher einen Z-Wert ungleich 0 hatte), muss es zuerst einmal daraus *entfernt* werden.

  Dabei wird das Array kürzer, alle Elemente hinter dem entfernten Rechteck *rutschen einen Platz nach vor*.

- Wenn der neue Z-Wert nicht 0 ist, muss an der richtigen Stelle im Array ein Pointer auf das Rechteck *eingefügt* werden.

  Dabei wird das Array größer (auf die Maximal-Größe achten!), und alle Elemente nach dem neuen Pointer *rutschen einen Platz nach hinten*.

  In welche Richtung durchwandert man das Array dafür sinnvollerweise?

- Und schließlich muss auch die *Z-Membervariable des Rechtecks* selbst auf den neuen Wert gesetzt werden.

Erinnere dich:

- Wie bekommt man einen Pointer auf das *eigene* Objekt?
- Wie ruft man die Methode eines Objektes auf (oder greift auf seine Member-Variablen zu), wenn man nicht das Objekt selbst, sondern einen Pointer darauf hat?

`drawAll` ist simpel:

- Einmal das Array durchwandern und für jedes Rechteck **`draw`** aufrufen.

(Eigentlich sollte statt diesem Array ein Container der STL verwendet werden, aber so weit sind wir noch nicht.)

3.) Im **`main`**:

- Du kannst dir ein **`main`** herunterladen, das ein Array von Rechtecken erzeugt und diese in zufälliger Reihenfolge bewegt.

  Dabei werden die Probleme der bisherigen Implementierung sichtbar.

- Daher sollen in dieses **`main`** die richtigen Aufrufe von **`setZ`** und **`drawAll`** eingefügt werden (Wie ruft man eine Klassenmethode auf?), und zwar so, dass die ursprüngliche Schichtung der Rechtecke erhalten bleibt (hellstes Rechteck obenauf, dunkelstes Rechteck zuunterst).

  Auch die schwarzen Ränder und Löcher sollten damit verschwinden.