

Programmieren C++: Objekte mit Pointern & dynamischen Daten: Zusammengesetzte grafische Objekte

Klaus Kusche

Dieses Beispiel ist eine Fortsetzung des Beispiels mit den geometrischen Objekten, nimm deine Lösung (oder meine Musterlösung) des Schrittes mit Vererbung und Kreisen als Ausgangsbasis (mit "normalem" Copy-Konstruktor ohne Farb- oder Größenänderung).

Wir leiten von unserer abstrakten Klasse **GraObj** eine dritte Klasse **Comp** ("complex", "compound" oder "composite") für zusammengesetzte Objekte ab, d.h. für Objekte, die selbst wieder aus mehreren Ellipsen und Rechtecken (und ev. wieder **Comp**'s) bestehen.

Grundidee ist folgende:

- Ein zusammengesetztes Objekt enthält einen Pointer auf ein dynamisch angelegtes Array von **GraObj-Pointern**.
- Die Elemente dieses Arrays zeigen auf die Subobjekte, aus denen sich das Comp-Objekt zusammensetzt.
- Operationen auf dem zusammengesetzten Objekt werden im Normalfall an alle Subobjekte durchgereicht.

Im Detail:

- Neben dem Pointer auf das Subobjekt-Pointer-Array (das ist typmäßig ein Pointer auf GraObj-Pointer!) bekommt **Comp** noch zwei Member-Variablen für die angelegte Größe dieses Arrays und für die Anzahl der derzeit darin enthaltenen Subobjekte. Alle Member sollen nur für die eigene und abgeleitete Klassen sichtbar sein.
- Der Konstruktor für **Comp** hat folgende Parameter:
 - Die x- und y-Koordinate für den Mittelpunkt des Objektes
 - Die Fluggeschwindigkeit in x- und y-Richtung
 - Die gewünschte Größe des Subobjekte-Arrays (maximal mögliche Anzahl von Subobjekten)

Die von **GraObj** geerbte Farbe des Objektes ist irrelevant (wird nie benötigt) und wird auf schwarz gesetzt. Die Größe des Objektes wird auf **0/0** gesetzt, da ein neues **Comp**-Objekt noch keine Subobjekte und daher keine Ausdehnung hat.

Das Pointer-Array für die Subobjekte wird im Konstruktor dynamisch angelegt, aber bleibt uninitialisiert: Die Subobjekte werden erst nachträglich dazugefügt (daher wird auch die aktuelle Anzahl von Objekten auf **0** gesetzt).

Ein **draw** ist nicht nötig: Es gibt noch nichts zum Zeichnen.

- Der Copy-Konstruktor für **Comp**-Objekte muss nach dem Aufruf des Vater-Copy-Konstruktors ein neues Pointer-Array allokalieren und die Subobjekte aus dem Array des Originals in dieses Array kopieren, und zwar mit **clone**, denn das neue **Comp**-Objekt soll ja eigene Subobjekte haben und nicht Pointer auf die Subobjekte des Original-Objektes.

- Da wir mit unserem Wissensstand keinen Zuweisungs-Operator schreiben können, der für alle Arten von **GraObj** funktioniert, wird der Zuweisungs-Operator in allen vier Klassen (**GraObj**, **Rect**, **Circ** und **Comp**) verboten.
- Der Destruktor muss zuerst alle Subobjekte freigeben und dann das Array selbst.
- Das **clone** wird wie üblich mittels Copy-Konstruktor implementiert.
- Für die folgenden Methoden brauchen wir eine neue Hilfsmethode recalcSize zur Berechnung der eigenen Größe. Sie wird immer dann aufgerufen, wenn Subobjekte hinzugefügt oder irgendwie verändert wurden, und setzt das eigene Breiten- und Höhen-Member neu.

Diese Methode soll nur innerhalb der Klasse und für abgeleitete Klassen nutzbar sein. Sie hat keine Parameter und keinen Returnwert.

Die Methode muss ermitteln, wie weit der äußerste Rand eines Subobjektes vom Mittelpunkt des **Comp**-Objektes entfernt ist. Dazu berechnet sie in einer Schleife über alle Subobjekte das Maximum in x- und in y-Richtung von "Breite (bzw. Höhe) des Subobjektes" + Absolutbetrag von "x-Abstand (bzw. y-Abstand) des Subobjekt-Mittelpunktes vom eigenen Mittelpunkt".

Diese Maxima werden dann im eigenen Breiten- und Höhen-Member gespeichert.

- In **Comp** kommt eine neue Methode add dazu, die mit einem **GraObj**-Objekt als by-Reference-Parameter aufgerufen wird und eine Kopie (clone !) des angegebenen Objektes zum Array der Subobjekte dazufügt (prüfen, ob noch Platz ist, hinten anhängen, Objekt-Zähler erhöhen).

Danach muss die eigene Größe frisch berechnet werden (**recalcSize**).

Der Returnwert soll **bool** sein (**true**: erfolgreich, **false**: Array voll).

- **setPos** berechnet den Unterschied zwischen der aktuellen eigenen Position und der Ziel-Position und ruft damit das eigene **move** auf.
- **move** verschiebt zuerst den eigenen Mittelpunkt und ruft dann in einer Schleife für alle Subobjekte move auf. Um die dabei entstehenden "Löcher" in der Anzeige zu vermeiden, wird am Ende nochmals **draw** aufgerufen.
- **setSize** berechnet, um wieviel Prozent sich die gewünschte neue Größe von der aktuellen Größe unterscheidet, und ruft damit das eigene **scale** auf..
- **scale** ist kompliziert, denn es muss für jedes Subobjekt zwei Dinge machen:
 - **scale** aufrufen.
 - Die x- und y-Abstände zwischen dem eigenen Mittelpunkt und dem Mittelpunkt des Subobjektes berechnen, diese Abstände skalieren und zum eigenen Mittelpunkt dazurechnen, und das Subobjekt mit **setPos** auf die resultierende neue Position setzen.

Danach muss **scale** noch **recalcSize** und **draw** aufrufen.

- **draw** und **undraw** rufen einfach in einer Schleife für alle Subobjekte draw und **undraw** auf.
- In der Basisklasse **GraObj** müssen alle Methoden, die in **Comp** überschrieben werden, als **virtual** deklariert werden.

- Ansonsten sollte die Implementierung von **GraObj**, **Rect**, **Circ** und **Color** unverändert bleiben.

Schreib dazu wieder ein kleines Hauptprogramm zum Testen.

Ich habe mir dazu folgendes Array angelegt:

```
GraObj *parts[] = {
    new Circ(Color(255, 255, 0), 400, 300, 80, 100),
    new Circ(Color(200, 100, 0), 400, 300, 10, 30),
    new Circ(Color(0, 0, 255), 360, 280, 25, 25),
    new Circ(Color(0, 0, 255), 440, 280, 25, 25),
    new Circ(Color(255, 0, 0), 400, 350, 40, 10),
    new Circ(Color(255, 200, 0), 310, 290, 10, 40),
    new Circ(Color(255, 200, 0), 490, 290, 10, 40)
};
```

Dann habe ich mir eine Hilfsfunktion zum Erzeugen eines **Comp**-Objektes geschrieben: Sie legt ein **Comp**-Objekt dynamisch an (in Fenster-Mitte), fügt mit **add** der Reihe nach die **parts** dazu, und returniert das entstehende Objekt.

Im Hauptprogramm habe ich wieder ein Array von **GraObj**-Pointern.

Das erste Element erzeuge ich mit meiner Hilfsfunktion.

Alle weiteren Elemente befülle ich, indem ich das vorige Objekt **clone** .

Als nächstes gehe ich mit einer Schleife nochmals alle Objekte durch und gebe jedem eine zufällige Flugrichtung (-7 ... +7).

Außerdem verforme ich jedes Objekt zufällig: Ich rufe **scale** auf,

wobei ich für jede Richtung einen zufälligen Prozentwert zwischen **33** und **132** angebe.

Dann lasse ich die Objekte wieder in einer Endlos-Schleife herumfliegen.