Moebelgruppe-Kompositum

Die Klasse Moebelgruppe
als Kompositum
modellieren

Moebelgruppe-Kompositum

Anlass ist die Erkenntnis, dass

- Schrankwand und Tischgruppe gleiche Methoden haben und daher die gemeinsamen Eigenschaften in einer Klasse Moebelgruppe ausmodelliert werden können,
- diese Lösung aber immer noch nicht die Möglichkeit bietet, die Moebel-Objekte aus der Gruppe verschiedenfarbig darzustellen.

Moebelgruppe-Kompositum

Notwendige Methoden

- *GibFigur(self)*
  gibt die Gesamtfigur zurück
- FuegeHinzu(self, moebel):
  Fügt zu einem Moebelgruppenobjekt ein (weiteres) Moebel-Objekt hinzu
- Entferne(self, moebel):
  Entfernt eines der Moebel-Objekte aus dem Moebelgruppenobjekt

Moebelgruppe-Kompositum

Es soll die Möglichkeit geben, die Moebel-Objekte aus der Gruppe verschiedenfarbig darzustellen.

- Das geht, wenn wir nicht die Gruppe als Ganzes darstellen, sondern die einzelnen Objekte getrennt von einander darstellen,
- sie aber trotzdem als Gruppe behandeln können, also verschieben (*geht einfach*) und drehen (*geht nicht einfach*) können.

Moebelgruppe-Kompositum

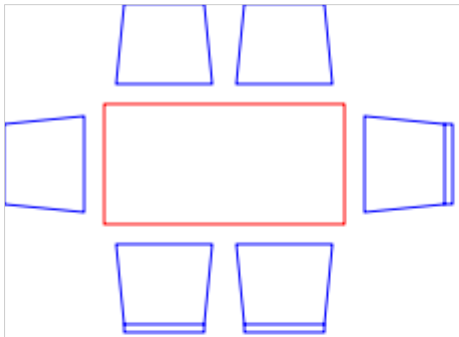
... wenn … die einzelnen Objekte getrennt von einander darstellen ...

- Die Klasse Moebelgruppe muss also die Methode ***Update()*** von Moebel überschreiben.
- Sie muss sie durch eine eigene Methode ersetzen, die von allen Moebel-Objekten der Moebelgruppe jeweils die Update-Methode aufruft.

Moebelgruppe-Kompositum

Ein erster Test mit einer so erzeugten (Schrankwand oder) Tischgruppe ist leider unbefriedigend:

- Die Teile stehen zwar richtig innerhalb der Tischgruppe, diese steht aber an der relativen Position der Teile, also in der linken oberen Ecke und
- Verschieben und Drehen haben keine Wirkung.



Moebelgruppe-Kompositum

Das wird besser, wenn die verändernden Methoden ebenfalls passend zum Kompositum bearbeitet in die Klasse Tischgruppe eingefügt werden:

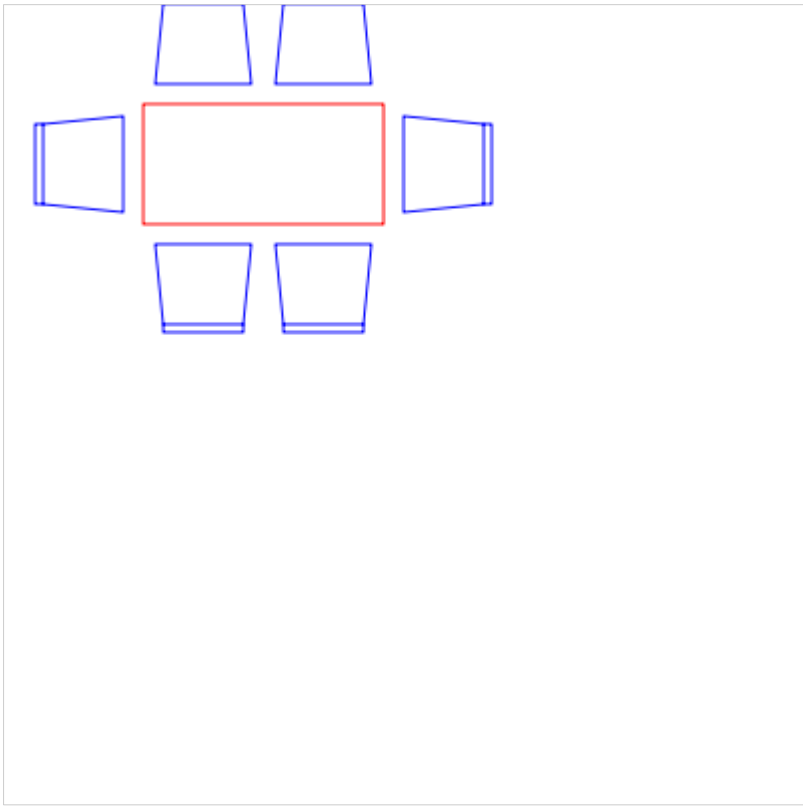
```
def BewegeHorizontal(self, weite):
    """Veraendernde Methode fuer die x-Position"""
    for moebel in self.__moebel:
        moebel.BewegeHorizontal(weite)

def BewegeVertikal(self, weite):
    """Veraendernde Methode fuer die y-Position"""
    for moebel in self.__moebel:
        moebel.BewegeVertikal(weite)

def Drehe(self, winkel):
    """Veraendernde Methode fuer die Orientierung [Winkel]"""
    for moebel in self.__moebel:
        moebel.Drehe(winkel)
```

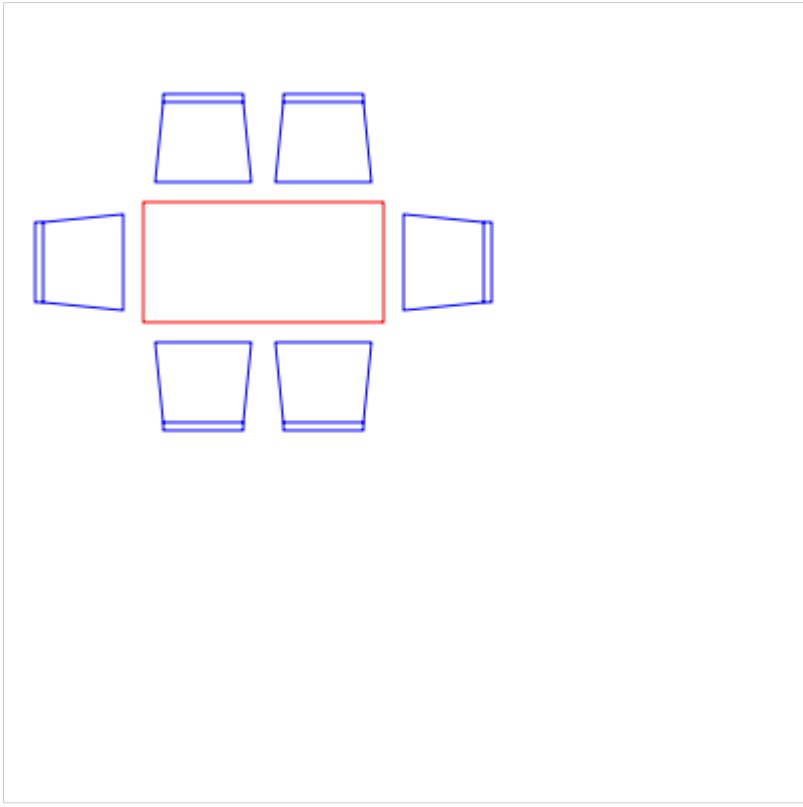
Moebelgruppe-Kompositum

- Nun gelingt das Verschieben in x-Richtung



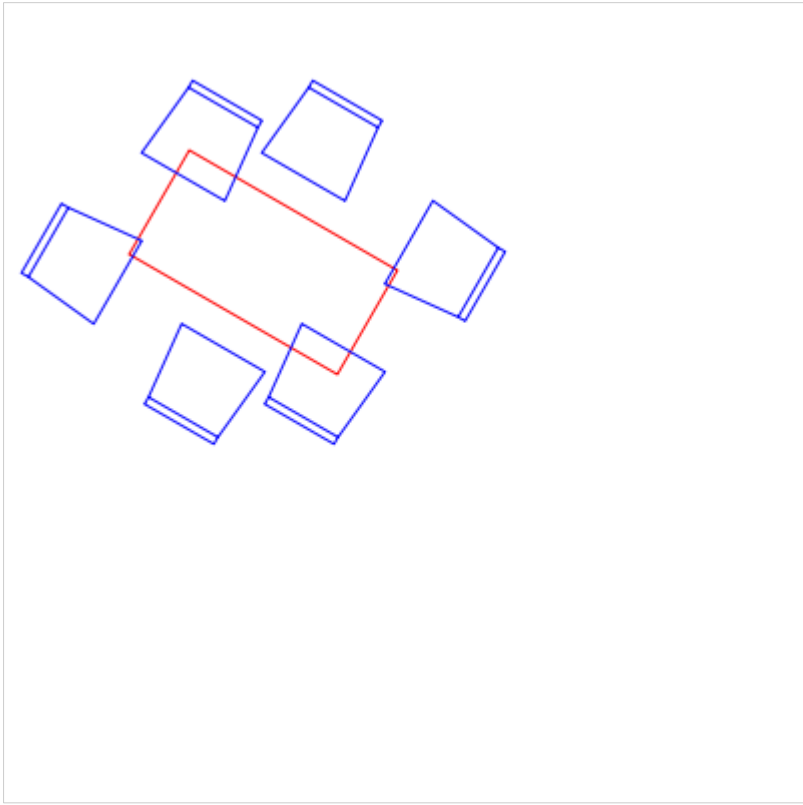
Moebelgruppe-Kompositum

- Nun gelingt das Verschieben in x-Richtung
- und in y-Richtung



Moebelgruppe-Kompositum

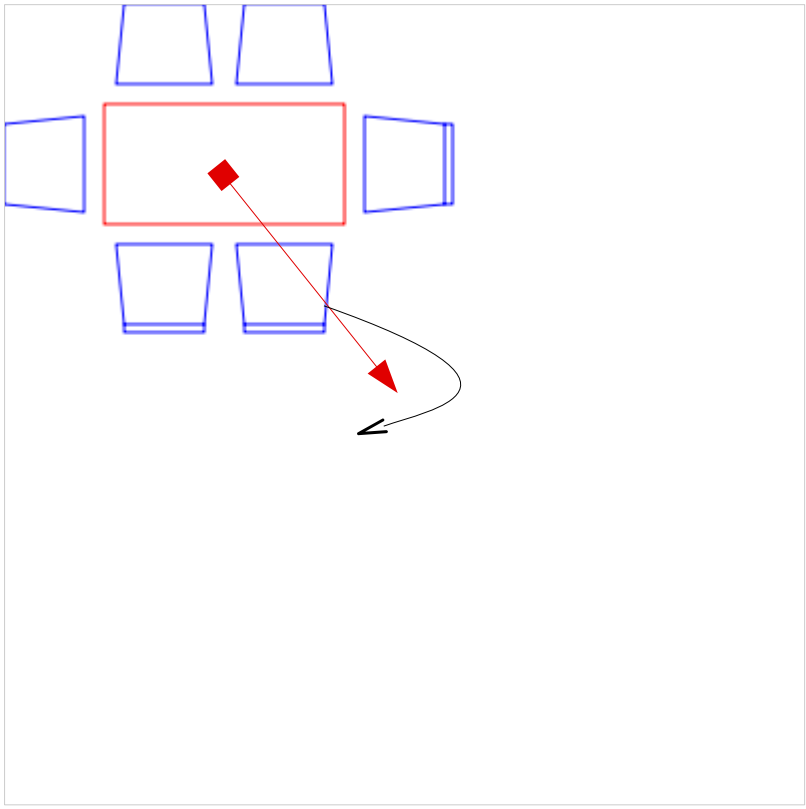
- Nun gelingt das Verschieben in x-Richtung
- und in y-Richtung.
- Dafür misslingt aber die Drehung völlig:



Moebelgruppe-Kompositum

Zurück zum Anfangszustand:

- Wie sollen die Objekte denn auch wissen, dass sie zu einer Gruppe gehören, die an anderer Stelle positioniert werden und gegebenenfalls auch noch gedreht werden soll.



Moebelgruppe-Kompositum

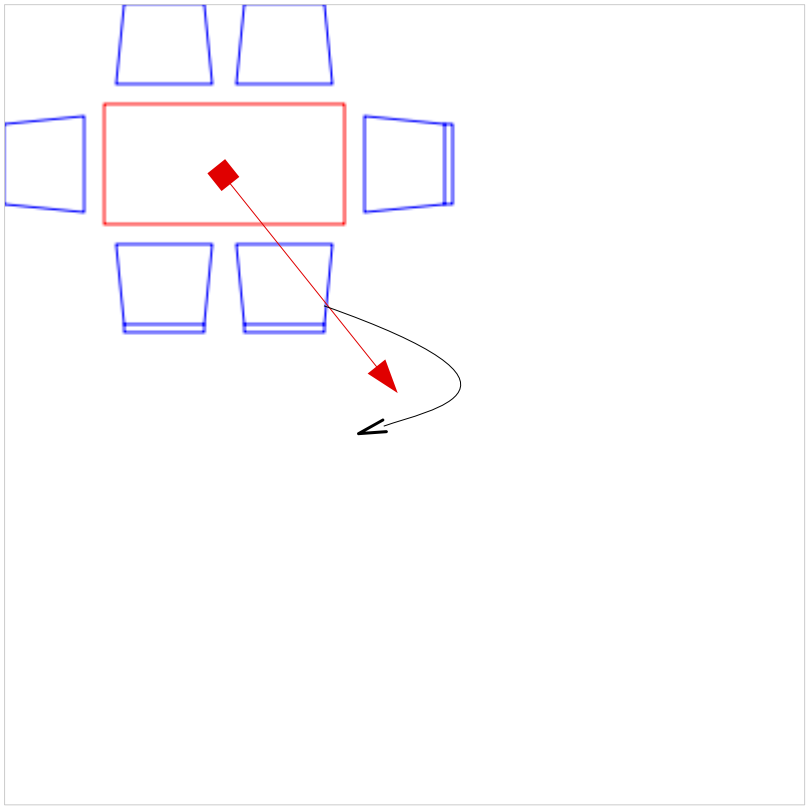
Die Lösung ist prinzipiell sehr einfach:

Die einzelnen Moebelobjekte

- müssen wissen, ob sie zu einer Gruppe gehören

  und

- dann deren Transformation nutzen.



Moebelgruppe-Kompositum

Da die Möglichkeit einer Zugehörigkeit zu einer Gruppe eine Eigenschaft aller einzelnen Moebelobjekte ist, werden der Klasse Moebel

- ein Attribut **`self.__gruppe`** mit dem Standardwert **`None`**
  und
- eine Set-Methode
  `setzeGruppe(self, gruppe)`
  hinzugefügt.

Moebelgruppe-Kompositum

- Nun muss in der Klasse Moebel noch die Transformation der Gruppe aufgerufen werden, falls das Moebel-Objekt zu einer Gruppe gehört.
- Das geschieht am Ende der Methode ***Transformiere(self, path)*** .

```
gc.PopState()
path.Transform(transformation)
if self.__gruppe!=None:
    return self.__gruppe.Transformiere(path)
return path
```