SFML #7.2 - Methoden mit SFML

Wir können Methoden nutzen, um uns das Arbeiten mit Objekten zu vereinfachen. Will man zum Beispiel, dass ein Objekt sich nicht hinter den Bildschirmrand bewegt, könnte man das so Programmieren:

**if (BoxA.Position.X > 800) {**

**BoxA.Position = new Vector2f(800,BoxA.Position.Y);**

**}**

**if (BoxA.Position.X < 0)**

**{**

**BoxA.Position = new Vector2f(0, BoxA.Position.Y);**

**}**

**if (BoxA.Position.Y > 600)**

**{**

**BoxA.Position = new Vector2f(BoxA.Position.X, 600);**

**}**

**if (BoxA.Position.Y < 0)**

**{**

**BoxA.Position = new Vector2f(BoxA.Position.X, 0);**

**}**

Will man das jetzt für viele Objekte ausführen, wird es recht schnell sehr viel Code der sehr ähnlich ist. In solchen Situation kann man das ganze auch sehr einfach auslagern. Dazu schreibt man den Code verallgemeinert einmal, und kann für jedes Objekt immer einmal die Methode aufrufen:

**void checkBorders(RectangleShape shape)**

**{**

**if (shape.Position.X > 800)**

**{**

**shape.Position = new Vector2f(800, shape.Position.Y);**

**}**

**if (shape.Position.X < 0)**

**{**

**shape.Position = new Vector2f(0, shape.Position.Y);**

**}**

**if (shape.Position.Y > 600)**

**{**

**shape.Position = new Vector2f(shape.Position.X, 600);**

**}**

**if (shape.Position.Y < 0)**

**{**

**shape.Position = new Vector2f(shape.Position.X, 0);**

**}**

**}**

Und jetzt kann man in "loop()" einfach die CheckBorders Methode für jedes Objekt ausführen und spart sich damit viel überflüssigen Code:

**public override void loop()**

**{**

**checkBorders(BoxA);**

**checkBorders(BoxB);**

**checkBorders(BoxC);**

**checkBorders(BoxD);**

**checkBorders(BoxE);**

**draw(BoxA);**

**draw(BoxB);**

**draw(BoxC);**

**draw(BoxD);**

**draw(BoxE);**

**}**  
Hierbei übergeben wir ein beliebiges RectangleShape als Parameter. Diese Art der Nutzung von Methoden nennt man **Auslagern**(Methoden die Parameter nehmen aber keinen Rückgabetyp haben).

Wenn man jetzt die Distanz von zwei Shapes wissen, könnte man das über den Satz von Pythagoras so machen:

**double dX = BoxA.Position.X - BoxB.Position.X;**

**double dY = BoxA.Position.Y - BoxB.Position.Y;**

**double distance = Math.Sqrt(dX \* dX + dY \* dY);**

Wenn man das öfters tut, kann man das auch über eine Methode machen. Diesmal ist es aber kein Auslagern, da die Distanz als Ergebnis zurückgegeben werden muss. Also setzen wir den Rückgabetyp auf double. Mit "return" können wir dann den Errechneten Wert zurückgeben.

**double distance(RectangleShape shapeA, RectangleShape shapeB)**

**{**

**double dX = shapeA.Position.X - shapeB.Position.X;**

**double dY = shapeA.Position.Y - shapeB.Position.Y;**

**return Math.Sqrt(dX \* dX + dY \* dY);**

**}**

Die Distanz zweier Objekte können wir dann ganz einfach per Aufruf berechnen:  
 **double d = distance(BoxA, BoxB);**