Hinzufügen & Entfernen von Objekten

Wenn wir Objekte unserem Spiel hinzufügen, machen wir das bisher indem wir sie im Code direkt als Variable deklarieren. Sobald es aber nötig ist, ein Element während der "Laufzeit" unseres Programms zu hinzuzufügen oder zu entfernen, klappt das so nichtmehr. Deswegen nutzen wir an dieser Stelle eine Liste.

// Liste mit Rectangle Shapes

List<RectangleShape> shapes = new List<RectangleShape>();

Diese Liste kann nun eine beliebige Anzahl an Shapes speichern und sie während der Laufzeit hinzufügen oder entfernen.

Um also auf "knopfdruck" ein neues Rectangle hinzuzufügen, schreiben wir:

void OnKeyPress(object sender, KeyEventArgs e)

{

// Wenn C gedrückt wird:

if (e.Code == Keyboard.Key.C)

{

// Füge ein neues RectangleShape hinzu

shapes.Add(new RectangleShape());

}

}

(Hier wurden Events verwendet, siehe 4.2)

Allerdings haben wir das Problem, dass wir so nicht die Eigenschaften des RectangleShapes verändern können. Deswegen kann man eine kleine Hilfsvariable benutzen, um das Shape zwischenzuspeichern.

// Wenn C gedrückt wird:

if (e.Code == Keyboard.Key.C)

{

// Erstelle ein neues Shape und setze die Eigenschaften

RectangleShape nShape = new RectangleShape();

nShape.Size = new Vector2f(50, 50);

nShape.FillColor = Color.Green;

nShape.Position = new Vector2f(400, 300);

// Füge nShape hinzu

shapes.Add(nShape);

}

Natürlich darf man nicht vergessen, die Shapes mit der draw() Methode zu zeichnen. Wenn wir eine Liste benutzen, kann man das über eine for-Schleife tun:

for (int i = shapes.Count - 1; i >= 0; i--)

{

draw(shapes[i]);

}

Jetzt werden alle Elemente der Liste nacheinander gezeichnet. Wir sehen allerdings maximal 1, weil sie sich alle überdecken. Deswegen lassen wir

nach unten bewegen. Um also eine Bewegung für alle Listen-Elemente zu machen, benutzen wir wieder eine For-Schleife:

for (int i = shapes.Count - 1; i >= 0; i--)

{

shapes[i].Position += new Vector2f(0,5);

}

Jetzt können wir noch programmieren, dass man das neuste Element löscht, wenn man X drückt:

// Wenn X gedrückt wird:

if (e.Code == Keyboard.Key.X)

{

if (shapes.Count > 0)

{

shapes.RemoveAt(shapes.Count - 1);

}

}

Aber Vorsicht! Wenn man löscht, sollte man immer prüfen, ob es überhaupt ein Element gibt, sonst crasht das Programm!