Kisten stapeln

In dieser Übung werden wir ein sehr einfaches Design paramtrisieren. Damit sollen Variationen des Originalkonzepts erzeugt werden. Das Skript wird dabei immer weiter ausgebaut, mit dem Ziel eines immer komplexeren Designs, mit einer steigenden Zahl von Parametern.

Die Aufgabe: Kisten stapeln

Das Konzept ist denkbar einfach: Mehrere Kisten, sagen wir fünf, sollen aufeinander gestaplet werden. Von dem Ausgangsdesign von gleichen, regelmäßig aufeinander gestapelten Boxen werden wir uns schrittweise zu einem komplexeren, unregelmäßigeren Stapel aus Kisten verschiedener Größen und Orientierungen weiterentwickeln.
Dabei sollen aber immer die grundlegenden Charakteristiken des Originals beibehalten werden. D.h. Alle Kisten sollen so aufeinander stehen, daß sie sich nicht durchdringen, oder daß ein Zwischenraum zwischen Ihnen entsteht.

Schritt 1:

Das Ausgangsskript: Ein Polycube wird generiert. Dieser wird dann kopiert und um zwei Einhheiten nach oben verschoben, so daß er auf dem vorhergehenden steht.
Dieser zweite Teil des Duplizierens und Verscheibens wird einfach kopiert und insgesamt viermal am Ende des Skripts eingesetzt. (Elegantere Arten Programmschritte wiederholt auszuführen werden wir später kennenlernen).

```
polyCube -w 2 -h 2 -d 2 -ax 0 0 1;

duplicate -rr;
move -r 0 0 2;

duplicate -rr;
move -r 0 0 2;

duplicate -rr;
move -r 0 0 2;

duplicate -rr;
move -r 0 0 2;
```

Schritt 2:

Die expliziten Maße des Stapels wollen wir nun paramtrisieren: Wir ersetzen Höhe Breite und Tiefe des Polycubes durch Variablen vom typ float.

```
float $width=2.0;
float $height=2.0;
float $depth=2.0;

polyCube -w $width -h $height -d $depth -ax 0 0 1;

duplicate -rr;
move -r 0 0 $height;

duplicate -rr;
move -r 0 0 $height;

duplicate -rr;
move -r 0 0 $height;

duplicate -rr;
move -r 0 0 $height;
```

Richtig: Die Höhe der Cubes entspricht ja genau dem Wert, um den diese jeweils nach oben verschoben werden, also setzen wir die Variable $height auch im move Befehl ein.

Durch verändern der drei Variablen können nun Stapel aus verschiedenen großen Kisten erzeugt werden.

Schritt 3:

Der Stapel muß nicht zwangsläufig so regelmäßig sein. Vielleicht sind die Kisten ja zueinander verdreht. Wir führen ein Rotation durch, die wiederum von einer Variablen kontrolliert wird.

Jetzt wird es knifflig:
Die Kisten sollen nach oben hin jeweils um 10% kleiner werden.

```
float $width=2.0;
float $height=2.0;
float $depth=2.0;
float $rotation=30;

polyCube -w 2 -h 2 -d 2 -sx 1 -sy 1 -sz 1 -ax 0 0 1 -tx 1 -ch 1;

duplicate -rr;
move -r 0 0 $height;
rotate -r 0 0 $rotation;

duplicate -rr;
move -r 0 0 $height;
rotate -r 0 0 $rotation;


duplicate -rr;
move -r 0 0 $height;
rotate -r 0 0 $rotation;

duplicate -rr;
move -r 0 0 $height;
rotate -r 0 0 $rotation;
```

Schritt 4:

Es ergibt sich nun ein spiralartig nach oben schraubende Stapel. Um diese Drehung frei zu variieren, wollen wir nun einen Zufallswert einführen.

Die rand – erzeugt einen Zufallswert im float Format, der zwischen den angegeben Zahlen liegt.
rand -90 90 erzeugt also eine Zahl zwischen -90 und 90.
Um den Wert in eine Variable zu speichern, ihn also später wieder verwenden zu können, benutzt man folgende Schreibweise:

float $rotation = `rand -90 90`;

Im Script sieht das Ganze so aus.

```
float $width=2.0;
float $height=2.0;
float $depth=2.0;
float $rotation;

polyCube -w 2 -h 2 -d 2 -sx 1 -sy 1 -sz 1 -ax 0 0 1 -tx 1 -ch 1;

duplicate -rr;
move -r 0 0 $height;
$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;

duplicate -rr;
move -r 0 0 $height;
$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;
```

```
duplicate -rr;
move -r 0 0 $height;
$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;

duplicate -rr;
move -r 0 0 $height;
$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;
```

Der Wert für die Rotation wird dabei jedesmal neu ausgewürfelt, so daß sich unterschiedliche Drehungen ergeben.

Schritt 5:

Nun sind also auch ungleichmäßig gedrehte Stapel möglich, noch mehr Variationen können wir durch unterschiedliche Boxgrößen erhalten.
Zunächst ändern wir nur die Breite und Tiefe durch jeweiliges skalieren um einen Zufallswert.

```
float $width=2.0;
float $height=2.0;
float $depth=2.0;
float $rotation;

float $scaleX;
float $scaleY;

polyCube -w 2 -h 2 -d 2 -sx 1 -sy 1 -sz 1 -ax 0 0 1 -tx 1 -ch 1;


duplicate -rr;
move -r 0 0 $height;
$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;
$scaleX=`rand 0.5 2`;
$scaleY=`rand 0.5 2`;
scale -a $scaleX $scaleY 1;

duplicate -rr;
move -r 0 0 $height;
$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;
$scaleX=`rand 0.5 2`;
$scaleY=`rand 0.5 2`;
scale -a $scaleX $scaleY 1;


duplicate -rr;
move -r 0 0 $height;
$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;
$scaleX=`rand 0.5 2`;
$scaleY=`rand 0.5 2`;
scale -a $scaleX $scaleY 1;


duplicate -rr;
move -r 0 0 $height;
$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;
$scaleX=`rand 0.5 2`;
$scaleY=`rand 0.5 2`;
scale -a $scaleX $scaleY 1;
```

Schritt 6:

Nun wird es etwas komplexer: Die Höhe der Kisten soll ebenfalls einem Zufallswert folgen. Dieser Parameter bestimmt ja ebenso die vertikale Verschiebung der Kisten, deswegen genügt es nicht, einfach nur den Z Skalierung zu erwürfeln und einzusetzen.

```
float $width=2.0;
float $height=2.0;
```

```
float $depth=2.0;
float $rotation;

float $scaleX;
float $scaleY;
float $scaleZ;

polyCube -w 2 -h 2 -d 2 -sx 1 -sy 1 -sz 1 -ax 0 0 1 -tx 1 -ch 1;


duplicate -rr;
move -r 0 0 $height;
$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;
$scaleX=`rand 0.5 2`;
$scaleY=`rand 0.5 2`;
$scaleZ=`rand 0.5 2`;
scale -a $scaleX $scaleY $scaleZ;

duplicate -rr;
move -r 0 0 $height;
$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;
$scaleX=`rand 0.5 2`;
$scaleY=`rand 0.5 2`;
$scaleZ=`rand 0.5 2`;
scale -a $scaleX $scaleY $scaleZ;

duplicate -rr;
move -r 0 0 $height;
$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;
$scaleX=`rand 0.5 2`;
$scaleY=`rand 0.5 2`;
$scaleZ=`rand 0.5 2`;
scale -a $scaleX $scaleY $scaleZ;

duplicate -rr;
move -r 0 0 $height;
$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;
$scaleX=`rand 0.5 2`;
$scaleY=`rand 0.5 2`;
$scaleZ=`rand 0.5 2`;
scale -a $scaleX $scaleY $scaleZ;
```

Schritt 7:

Die Polycubes werden kopiert, skaliert und nach oben verschoben. Der Wert der Höhenverschiebung wird nun aber nicht nur durch die Höhe der aktuellen Kiste bestimmt, sondern auch durch die der Kiste davor, aus der sie kopiert wurde. Der Verschiebungswert enspricht vielmehr der Summe aus der jeweils der halben Höhe beider Kisten.

Dazu führen wir zwei neue Variablen ein:

```
float $width=2.0;
float $height=2.0;
float $depth=2.0;
float $rotation;

float $scaleX;
float $scaleY;
float $scaleZ;

float $heightOld;
float $heightNew;
float $moveZ;

polyCube -w 2 -h 2 -d 2 -sx 1 -sy 1 -sz 1 -ax 0 0 1 -tx 1 -ch 1;

$heightOld=$height;
```

```
// Dieser Bereich wiederholt sich 4 mal !!!

duplicate -rr;

$scaleX=`rand 0.5 2`;
$scaleY=`rand 0.5 2`;
$scaleZ=`rand 0.5 2`;

$heightNew=$heightOld*$scaleZ;
$moveZ=($heightOld+$heightNew)/2.0;
move -r 0 0 $moveZ;

$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;
scale -r $scaleX $scaleY $scaleZ;
$heightOld=$heightNew;

// Dieser Bereich wiederholt sich 4 mal !!!

duplicate -rr;

$scaleX=`rand 0.5 2`;
$scaleY=`rand 0.5 2`;
$scaleZ=`rand 0.5 2`;

$heightNew=$heightOld*$scaleZ;
$moveZ=($heightOld+$heightNew)/2.0;
move -r 0 0 $moveZ;

$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;
scale -r $scaleX $scaleY $scaleZ;
$heightOld=$heightNew;


// Dieser Bereich wiederholt sich 4 mal !!!

duplicate -rr;

$scaleX=`rand 0.5 2`;
$scaleY=`rand 0.5 2`;
$scaleZ=`rand 0.5 2`;

$heightNew=$heightOld*$scaleZ;
$moveZ=($heightOld+$heightNew)/2.0;
move -r 0 0 $moveZ;

$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;
scale -r $scaleX $scaleY $scaleZ;
$heightOld=$heightNew;

// Dieser Bereich wiederholt sich 4 mal !!!

duplicate -rr;

$scaleX=`rand 0.5 2`;
$scaleY=`rand 0.5 2`;
$scaleZ=`rand 0.5 2`;

$heightNew=$heightOld*$scaleZ;
$moveZ=($heightOld+$heightNew)/2.0;
move -r 0 0 $moveZ;

$rotation=`rand -90 90`;
rotate -r 0 0 $rotation;
scale -r $scaleX $scaleY $scaleZ;
$heightOld=$heightNew;
```