Exercise: Sorting Algorithms

In dieser Übung wird ein einfacher Sortieralgorithmus anhand eines kleinen Beispiels vorgestellt:
Eine Reihe von zufällig skalierten Spheres soll der Größe nach sortiert werden.

Schritt 1: Random Ellipsoiden erzeugen

Diese kleine Script erzeugt eine Reihe von 8 Spheres, die mit einem Zufallswert in Z skaliert werden.
Der Code kann als shelfbutton abgespeichert werden um immer wieder ein neues Setup zu erzeugen.

```
for ($i=0; $i<8; $i++)
{
	sphere -r 2 -ax 0 0 1;
	float $scaleZ=`rand 0.5 4`;
	scale -a 1 1 $scaleZ;
	move -a ($i*5) 0 ($scaleZ*2.0);
}
```

Schritt 2: Random Ellipsoiden erzeugen

Das zweite Skript enthält allein den Sortieralgorithmus. Zunächst wollen wir nur nacheinander die Höhe jeder Sphere abfragen und ausgeben.

```
string $OBJ[]=`ls -sl`;
int $numOBJ=size($OBJ);

for ($i=0; $i<$numOBJ; $i++)
{

	string $qHeight=($OBJ[$i]+".scaleZ");
	float $Height=`getAttr $qHeight`;

	print ($OBJ[$i]+" "+$Height+"\n");
}
```

Schritt 3: Größtes Element finden

Der maximale ScaleZ-Wert der Spheres sollgefunden werden. Dazu wird eine neue float-Variable $maxHeight eingeführt. Der Ausgangswert ist 0. In jedem Durchgang der Schleife wird nun überprüft, ob die Höhe der gerade betrachteten Sphere nun größer ist als der gegenwärtige Maximalwert. Ist dies der Fall, wird dieser Wert zum neuen Maximalwert.

```
string $OBJ[]=`ls -sl`;
int $numOBJ=size($OBJ);

float $maxHeight=0;

for ($i=0; $i<$numOBJ; $i++)
{

	string $qHeight=($OBJ[$i]+".scaleZ");
	float $Height=`getAttr $qHeight`;
	print ($OBJ[$i]+" "+$Height+"\n");

	if ($Height>$maxHeight) $maxHeight=$Height;
}

print ("Highest Sphere: "+$maxHeight+"\n");
```

Schritt 4: Größtes Element selektieren

Zusätzlich zum maximalen ScaleZ-Wert wird nun auch der Name der dazugehörigen Sphere mit angegeben.

```
string $OBJ[]=`ls -sl`;
int $numOBJ=size($OBJ);

float $maxHeight=0;
string $Highest;

for ($i=0; $i<$numOBJ; $i++)
{

	string $qHeight=($OBJ[$i]+".scaleZ");
	float $Height=`getAttr $qHeight`;
	print ($OBJ[$i]+" "+$Height+"\n");

	if ($Height>$maxHeight)
	{
		$maxHeight=$Height;
		$Highest=$OBJ[$i];
	}
}

print ("Highest Sphere: "+$Highest+" "+$maxHeight+"\n");
select $Highest;
```

Schritt 4: Sortierungsschleife einrichten

Um nun alle Spheren der Größe nach zu sortieren, genügt es nicht, nur die größte zu ermitteln, alle Elemente müssen ihrer Größe nach analysiert werden.
Dafür werden wieder zwei Schleifen ineinander geschachtelt. Das string[] Array $sorted[] soll alle Spheres der Größe nach aufnehmen.

```
string $OBJ[]=`ls -sl`;
int $numOBJ=size($OBJ);

string $sorted[];
clear $sorted;

float $maxHeight=0;
string $Highest;

for ($i1=0; $i1<$numOBJ; $i1++)
{


	for ($i=0; $i<$numOBJ; $i++)
	{

		string $qHeight=($OBJ[$i]+".scaleZ");
		float $Height=`getAttr $qHeight`;
		print ($OBJ[$i]+" "+$Height+"\n");

		if ($Height>$maxHeight)
		{
			$maxHeight=$Height;
			$Highest=$OBJ[$i];
		}
	}

	$sorted[$i1]=$Highest;
	print ("Highest Sphere: "+$Highest+" "+$maxHeight+"\n");
	select $Highest;
}

print "Sorted: \n";
print $sorted;
print "\n";
```

Schritt 5: Sortieren

Um die zweitgrößte und alle darauf folgenden Spheres zu ermittlen, müssen sie zwei Voraussetzungen erfüllen: Sie müssen größer sein als die anderen, aber immer noch kleiner als die Nächstgrößere.
Deshalb wird eine neue Variable eingeführt: $prevMaxHeight speichert die Größe der zuletzt einsortierten Sphere. Ihr Wert wird zu Beginn auf 10 gesetzt, ist als mit Sicherheit größer als alle anderen.
Am Ende jedes Durchgangs der äußeren Schleife, wenn also ein Element neu eingeordnet wurde, wird der Wert überschrieben, so daß nun die Höhe der letzten Sphere als Referenzwert verglichen wird.

Am Ende fügen wir noch eine Schleife ein, die die sortierten Spheres der Größe nach anordnet.

```
string $OBJ[]=`ls -sl`;
int $numOBJ=size($OBJ);

string $sorted[];
clear $sorted;

float $maxHeight=0;
float $PrevMaxHeight=10;


string $Highest;

for ($i1=0; $i1<$numOBJ; $i1++)
{
	$maxHeight=0;

	for ($i=0; $i<$numOBJ; $i++)
	{

		string $qHeight=($OBJ[$i]+".scaleZ");
		float $Height=`getAttr $qHeight`;
		print ($OBJ[$i]+" "+$Height+"\n");

		if ($Height>$maxHeight && $Height<$PrevMaxHeight)
		{
			$maxHeight=$Height;
			$Highest=$OBJ[$i];
		}
	}

	$sorted[$i1]=$Highest;
	print ("Highest Sphere: "+$Highest+" "+$maxHeight+"\n");
	select $Highest;
	$PrevMaxHeight=$maxHeight;
}

print "Sorted: \n";
print $sorted;
print "\n";

for ($i2=0; $i2<($numOBJ); $i2++)
{
	select $sorted[$i2];
	move -a ($i2*5) 0 0;
}
```

Schritt 6: Fallunterscheidung

Den Wert für $PrevMaxHeight anfangs hoch zu setzen ist ein nicht sehr eleganter Trick, der nicht immer funktionert. Besser ist eine allgemeinere Fallunterscheidung zwischen dem ersten Durchgang und allen darauffolgenden.

```
string $OBJ[]=`ls -sl`;
int $numOBJ=size($OBJ);

string $sorted[];
clear $sorted;

float $maxHeight=0;
float $PrevMaxHeight=0;


string $Highest;

for ($i1=0; $i1<$numOBJ; $i1++)
{
	$maxHeight=0;

	for ($i=0; $i<$numOBJ; $i++)
	{

		string $qHeight=($OBJ[$i]+".scaleZ");
		float $Height=`getAttr $qHeight`;
		print ($OBJ[$i]+" "+$Height+"\n");

		if ($i1==0)
		{
			if ($Height>$maxHeight)
			{
				$maxHeight=$Height;
				$Highest=$OBJ[$i];
			}
		}

		else
		{
			if ($Height>$maxHeight && $Height<$PrevMaxHeight)
			{
				$maxHeight=$Height;
				$Highest=$OBJ[$i];
			}
		}

	}

	$sorted[$i1]=$Highest;
	print ("Highest Sphere: "+$Highest+" "+$maxHeight+"\n");
	select $Highest;
	$PrevMaxHeight=$maxHeight;

}

print "Sorted: \n";
print $sorted;
print "\n";


for ($i2=0; $i2<($numOBJ); $i2++)
{
	select $sorted[$i2];
	move -a ($i2*5) 0 0;
}
```