# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Was ist es?

"selftree.inc" ist ein in der PoVRay-Scriptsprache geschriebener Include-File-Generator. Die erzeugten Includefiles enthalten ein oder mehrere Objekte, die bei guter Wahl der Parameter recht brauchbare Bäume ergeben.

Form, Aussehen und Komplexizität der erzeugten Objekte sind durch eine Vielzahl von Parametern beeinflußbar.

Die folgende Anleitung muß nicht komplett gelesen werden, um mit der Datei umgehen zu können. Zur Anwendung reicht der Abschnitt über die Benutzung und die Erlärung der einzelnen Variablen und der Blätter. Die restlichen Abschnitte enthalten allgemeines Gelaber und ein paar für andere POVRay-Programmierer interessante Beschreibungen der Funktionsweise des Files.

## 1.2 Motivation

Die Idee, ein Bäume erzeugendes Include-File zu schreiben hatte ursprünglich ein Mathematikstudent (Hallo, Holger!), dessen Files aber keine Rekursion enthielten, sondern verschiedene Schachtelungen von verschiedenen Include-Files. Zehn Stufen Rekursion = zehn verschiedene Include-Files... Nach langer

Diskussion, wie man dieses Problem lösen könnte und dem definitiven Beweis, daß Rekursion in POVRay derzeit nicht möglich ist, packte mich der Ehrgeiz, eben doch ein rekursives Include-File zu schreiben. Heraus kam selftree.inc.

# 2 Benutzung

## 2.1 Allgemein

Um die von "selftree.inc" erzeugten Bäume benutzen zu können, ist ein kleiner Trick notwendig. Das Programm schreibt den Source-Code der erzeugten Objekte per #debug-Anweisung in den Debug-Stream. Um das Objekt im Scene-File benutzen zu können, muß man den Debug-Stream also in eine Datei umleiten. Das geht mit der Anweisung -GD_FILENAME_ (bzw. +GD_FILENAME_, wenn man die Ausgabe von "selftree.inc" auch gleichzeitig auf der Console betrachten möchte, was aber nicht nötig ist, da der erzeugte Code selber nicht sonderlich interessant ist und eine Statistik über die erzeugten Objekte wird nach dem Parsen derselben auf dem Statistik-Stream ausgegeben). Diese Anweisung leitet den Debug-Stream in die Datei _FILENAME_ um. Diese Datei wiederum muß **nach** "selftree.inc" in das Scene-File eingebunden werden. Und noch vor alledem sollte man einen Haufen Variablen setzen, um das Aussehen des Baumes zu beeinflussen (wobei auch die Default-Werte schon einen brauchbaren Baum erzeugen).

Folgende Schritte sind also nötig, um mit "selftree.inc" einen Baum zu erzeugen:

- Einen Haufen Variablen setzen (oder mit dem Standardbaum zufrieden sein).
- "selftree.inc" in das Scene-File einbinden.
- Evtl. mehrfach die ersten beiden Schritte wiederholen, um mehrere Bäume zu erzeugen.
- _FILENAME_ in das Scene-File einbinden. Wobei _FILENAME_ der Name der mit dem POVRay-Aufruf erzeugten Datei ist.
- object-Statement mit dem erzeugten Baum-Object deklarieren:
  object { TREE texture {...} rotate {...} translate {...}}
- POVRay aufrufen, etwa so:
  **povray +Imyscene.pov -GD_FILENAME_**... weitere POVRay-Argumente

## 2.2 Variablen

"selftree.inc" kann durch eine große Anzahl von Variablen beeinflußt werden. Es gibt im Grunde genommen drei verschiedene Arten von Variablen:

- Einfache Zahlen: Davon gibt es nur zwei namens DEPTH und NRLEAFINCS.
- Stringvariablen, die bei der Deklaration in Anführungszeichen eingebunden werden müssen und einen normalen String am besten ohne Leer- und Sonderzeichen enthalten sollten. Beispiel:
  **#declare OBJECTNAME = "mytree"**
- Funktionen: Der größte Teil aller Variablen sind Funktionen, deren Wert in Abhängigkeit von der aktuellen Rekursionsstufe berechnet wird. Die Funktionen können konstante Werte haben (**#declare NRBRANCHES = "3"**) oder als Funktionen in Abhängigkeit von X (großes X) angegeben werden (**#declare NRBRANCHES = "int(sqrt(X)+(X/2))"**). Diese Art Werte zu setzen hat den Vorteil, das erzeugte Objekt weitaus genauer beeinflussen zu können. So kann man z.B. mit einfachen Funktionen schon den Baum in den Spitzen mehr verzweigt sein lassen, als weiter unten bei den noch dickeren Ästen.

Bis auf DEPTH, NRLEAFINCS und die Textur-Modifier erwartet "selftree.inc" alle Variablen in Anführungszeichen definiert.

"selftree.inc" nimmt keine Überprüfung der Variablen vor. Es wird (sicherlich fälschlicherweise) davon ausgegangen, daß nie irgendjemand einen Fehler macht, und wenn doch, er es schon selber merken wird...

Die Werte vieler Variablen werden bei der Erzeugung des Baums noch durch zufallsabhängige Funktionen beeinflußt, so daß die Ergebnisse nicht immer genau vorhersagbar sind (man kann bei ansonsten gleichen Einstellungen und nur verändertem Startwert einer der Zufallsgeneratoren sehr verschiedene Bäume erhalten), ein bißchen Übung und Glück helfen, gute Werte zu finden.

Es folgt nun eine Beschreibung aller Variablen, die gesetzt werden können, einschließlich ihrer Default-Werte und Vorschläge für sinnvolle Wertebereiche. (Eine kurze Referenz der Variablen findet man im Include-file selbst.) Und dann ran ans rendern...

### 2.2.1 OBJECTNAME

1. Beschreibung: Um mehrere verschiedene erzeugte Bäume unterscheiden zu können, muß man ihnen Namen geben. Die in dieser Variablen

gesetzten Namen sind dann die Objekt-Bezeichner, unter denen man die Bäume im Scene-File benutzen kann.

2. Default: “TREE” (Stringvariable)

3. Anmerkungen: Erzeugt man durch mehrfaches Einfügen von “selftree.inc” mehrere Bäume, ohne OBJECTNAME jedesmal neu zu setzen, haben alle erzeugen Objekte den gleichen Namen, was zu unerwünschten Effekten führen kann.

### 2.2.2 USESTDTRUNK

1. Beschreibung: Die erzeugten Bäume bestehen aus cone-Objekten mit Kugeln an den Enden, um die Übergänge weicher zu gestalten. Um einem Baum realitätsnäher erscheinen zu lassen, kann man einen eigenen Stamm definieren, den “selftree.inc” dann statt des Standardstamms benutzt. Diese Variable ist ein Bool, der anzeigt, ob der Standardstamm oder ein anderer benutzt werden soll.

2. Default: “YES”

### 2.2.3 TRUNKLEN

1. Beschreibung: Wie lang soll der Stamm sein? Die Angabe gilt sowohl für den Standardstamm wie auch für selbstdefinierte.

2. Default: “5”

### 2.2.4 TRUNKBASER

1. Beschreibung: Welchen Radius soll der Standardstamm am Boden haben?

2. Default: “0.7”

### 2.2.5 TRUNKCAPR

1. Beschreibung: Welchen Radius soll der Standardstamm an der Spitze haben?

2. Default: “0.5”

### 2.2.6 DEPTH

1. Beschreibung: DEPTH gibt die Rekursionstiefe an, d.h. wie oft der Baum vom Stamm bis zu den Astspitzen verzweigt.
2. Default: 6
3. Anmerkungen: Diese Variable muß nicht in Anführungszeichen definiert werden.

### 2.2.7 BLEN

1. Beschreibung: Die Länge der Äste in der ersten Rekursionsstufe (die Äste, die direkt vom Stamm abzweigen).
2. Default: "5"
3. Anmerkungen: Es wird nur die Länge dieser Äste benötigt, da die weiteren Äste skalierte und zufällig leicht variierte Versionen dieser Äste sind.

### 2.2.8 LDIFF

1. Beschreibung: Der Bereich, in dem die Länge der Äste zufällig variieren darf.
2. Default: "0"
3. Anmerkungen: Der Wert aus diesem Bereich wird zu BLEN addiert. Dieser Wert wird ebenfalls mit skaliert in den Rekursionsstufen.

### 2.2.9 RADIUS

1. Beschreibung: Der Startradius der Äste der ersten Rekursionsstufe.
2. Default: "0.5"
3. Anmerkungen: Für einen weichen Übergang sollte man hier am besten den gleichen Wert wie für TRUNKCAPR nehmen.

### 2.2.10 RADSCALE

1. Beschreibung: Der Skalierungswert für den Astradius zwischen den Rekursionsstufen.
2. Default: "0.5"

#### 2.2.11 RDIFF

1. Beschreibung: Der Bereich, in dem der RADIUS zufällig variieren darf.
2. Default: “0”
3. Anmerkungen: Wird von RADIUS abgezogen und ebenfalls mitskaliert.

#### 2.2.12 RECSCALE

1. Beschreibung: Der allgemeine Skalierungswert für die Rekursion. Skalierungswert haben.
2. Default: “0.5”
3. Anmerkungen: Dieser Faktor gilt für alle Werte, die keinen eigenen Skalierungswert haben.

#### 2.2.13 SLOPEANGLE

1. Beschreibung: Der Winkel, in dem die Äste von ihren Vorgängern abzweigen.
2. Default: “40”

#### 2.2.14 SANGLEDIFF

1. Beschreibung: Der Bereich, in dem SLOPEANGLE variieren darf.
2. Default: “0”
3. Anmerkungen: Wird zu SLOPEANGLE addiert.

#### 2.2.15 ANGLESCALE

1. Beschreibung: Der Skalierungswert für SLOPEANGLE.
2. Default: “1.1”

#### 2.2.16 NRBRANCHES

1. Beschreibung: Die Anzahl der Kind-Äste die von einem Ast, bzw. dem Stamm abzweigen.
2. Default: “4”

3. Anmerkungen: Hier machen kleine Funktionen in Abhängigkeit von X Sinn und geben sehr interessante Bäume (VORSICHT: Man sollte darauf achten, daß die Funktionswerte nicht zu groß wachsen, die Anzahl Objekte aus denen der Baum besteht, nimmt extrem schnell astronomische Werte an!).

#### 2.2.17 NRBDIFF

1. Beschreibung: Der Bereich, in dem die Anzahl der Äste (NRBRANCHES) variieren darf.

2. Default: "0"

3. Anmerkungen: Wird zu NRBRANCHES addiert.

#### 2.2.18 NRBSCALEADD

1. Beschreibung: Einer von zwei Skalierungswerten für die Anzahl der Äste.

2. Default: "0"

3. Anmerkungen: Dieser Wert ist ein Überbleibsel aus alten Versionen, als "selftree.inc" noch keine Funktionen als Variablen kannte, und kann besser durch eine Funktion in NRBRANCHES ersetzt werden. Wenn man allerdings keine eigenen Funktionen schreiben mag, sind dieser und der folgende Wert gute Möglichkeiten, NRBRANCHES zu variieren.

   Der Wert wird in jeder Rekursionsstufe zu NRBRANCHES addiert.

#### 2.2.19 NRBSCALEMUL

1. Beschreibung: Der zweite Skalierungswert für NRBRANCHES.

2. Default: "1"

3. Anmerkungen: NRBRANCHES wird in jeder Rekursionsstufe mit NRBSCALEMUL multipliziert. Siehe auch Anmerkung zu NRBSCALEADD.

#### 2.2.20 BRANCHTOP

1. Beschreibung: Ein Schwellenwert, der die Wahrscheinlichkeit angibt, mit der ein Ast von der Spitze seines Vorgänger abzweigt.

2. Default: “0.6”

3. Anmerkungen: Ist ein zufällig zwischen 0 und 1 gewürfelter Wert größer als BRANCHTOP, verzweigt der Ast an einer beliebigen Stelle seines Vorgängers, sonst an dessen Spitze.

#### 2.2.21 LEAFINC

1. Beschreibung: Hat der Baum Blätter?

2. Default: “NO”

#### 2.2.22 NRLEAFINCS

1. Beschreibung: Wie viele verschiedene Blätter gibt es?

2. Default: 0

3. Anmerkungen: Dieser Wert wird **ohne** Anführungszeichen angegeben.

   Es müssen bestimmte Blattobjekte definiert werden, die “selftree.inc” benutzt. Siehe hierzu Kapitel 3 über Blätter.

#### 2.2.23 NRLEAFS

1. Beschreibung: Wieviele Blätter soll ein Ast haben?

2. Default: “0”

3. Anmerkungen: Es werden nur an den äußersten Äste Blätter angebracht.

#### 2.2.24 NRLDIFF

1. Beschreibung: Der Bereich, in dem NRLEAFS variieren darf.

2. Default: “0”

3. Anmerkungen: Wird zu NRLEAFS addiert.

### 2.2.25 ROTRANGE

1. Beschreibung: Der Bereich, in dem Äste um ihren Vorgänger rotiert und verteilt werden.
2. Default: "360"
3. Anmerkungen: Mit ein bißchen Geschick kann man mit einer Einschränkung dieses Wertes Phänomene wie Wind simulieren.

### 2.2.26 RANDINIT_ROT

1. Beschreibung: Startwert für den Zufallsgenerator für ROTRANGE
2. Default: "1.1234"
3. Anmerkungen: Die Änderung eines dieser Wert kann das Aussehen des Baumes sehr verändern. Ist ein Baum z.B. strukturell gut gelungen, aber seine Form befriedig nicht, kann das Ändern eines dieser Werte den Baum deutlich verbessern (oder verschlechtern...).

### 2.2.27 RANDINIT_LDIFF

1. Beschreibung: Startwert für den Zufallsgenerator für LDIFF.
2. Default: "0.1234"

### 2.2.28 RANDINIT_RDIFF

1. Beschreibung: Startwert für den Zufallsgenerator für RDIFF.
2. Default: "0.1234"

### 2.2.29 RANDINIT_SDIFF

1. Beschreibung: Startwert für den Zufallsgenerator für SANGLEDIFF.
2. Default: "0.1234"

### 2.2.30 RANDINIT_NRBDIFF

1. Beschreibung: Startwert für den Zufallsgenerator für NRBDIFF.
2. Default: "0.1234"

#### 2.2.31 RANDINIT_NRLDIFF

1. Beschreibung: Startwert für den Zufallsgenerator für NRLDIFF.
2. Default: “0.1234”

#### 2.2.32 RANDINIT_LEAFSELECT

1. Beschreibung: Startwert für den Zufallsgenerator für die zufällige Auswahl der verschiedenen Blätter.
2. Default: “0.1234”

#### 2.2.33 RANDINIT_LEAFTRANS

1. Beschreibung: Startwert für den Zufallsgenerator für die Plazierung der Blätter an den Ästen.
2. Default: “0.1234”

#### 2.2.34 RANDINIT_LEAFROT

1. Beschreibung: Startwert für den Zufallsgenerator für die zufällige Rotation der Blätter um die Äste.
2. Default: “0.1234”

#### 2.2.35 RANDINIT_BT

1. Beschreibung: Startwert für den Zufallsgenerator zum Auswürfeln der Schwellenwerte für BRANCHTOP.
2. Default: “0.1234”

#### 2.2.36 MOD_TR

1. Beschreibung: Textur für den Stamm.
2. Default: es wird die #default-Textur verwendet.
3. Anmerkungen: Es sollte eine vollständige Textur-Definition angegeben werden. Beispiel:

```
texture {
   pigment {
      color DarkBrown
   }
   normal {
      bozo 0.2
      scale <0.05, 0.2, 0.08>
   }
}
```

#### 2.2.37 MOD_X

1. Beschreibung: Die Texturen für die Äste der verschiedenen Rekursionsstufen.
2. Default: es wird die #default-Textur verwendet.
3. Anmerkungen: Das X im Namen dieser Variable ist ein Platzhalter. X ist eine Nummer zwischen 0 und DEPTH-1 (0 sind die Äste, die direkt vom Stamm abzweigen, DEPTH-1 die obersten Astspitzen). Auf diese Weise kann für die Äste jeder Rekursionsstufe eine eigene Textur angegeben werden (es macht z.B. Sinn, die Textur bei den äußeren dünneren Ästen weniger grob und rauh sein zu lassen).

#### 2.2.38 #default

1. Beschreibung: Eine Default-Textur für den Baum.
2. Default: { pigment { color rgb $< 0.35, 0.16, 0.14 >$ } }
3. Anmerkungen: Hat man eine eigene Default-Textur definiert, sollte man diese vor Verwendung von "selftree.inc" sichern und anschließend restaurieren.

# 3 Blätter

Um nicht nur winterliche Bäume erzeugen zu können, kann man mit "selftree.inc" natürlich auch Blätter hinzufügen. Dafür müssen Blattobjekte definiert werden. "selftree.inc" erlaubt, verschiedene Blattformen an einem Baum zu verteilen. Wieviele verschiedene Formen es gibt, wird durch die Variable NRLEAFINCS festgelegt.

Die Namen der Blattobjekte folgen einem festen Schema, damit "selftree.inc" die Namen der Blätter kennt. Blätternamen müssen folgendermaßen aufgebaut sein: LEAFINC_X. X steht dabei für eine Zahl zwischen 0 und NRLEAFINCS-1. Hat NRLEAFINCS also den Wert 11, erwartet "selftree.inc" die Namen LEAFINC_0, LEAFINC_1, ..., LEAFINC_10.

Es wird davon ausgegangen, daß der Blattstiel am Punkt $< 0, 0, 0 >$ seinen Ursprung hat und das Blatt in Y-Richtung ausgerichtet ist.

Eine Blattdefinition kann z.B. so aussehen:

```
#declare LEAFINC_0 =
sphere
{
  <0, 0, 0>, 0.3
  scale <1, 0.1, 0.5>
  translate <0.3, 0, 0>
  pigment { color Green }
}
```

## 3.1 leafshapes.inc

Das in der Distribution mitgelieferte "leafshapes.inc" bietet eine kleine Auswahl an vordefinierten Blättern. Das Scenefile leafs.pov erzeugt ein Bild aller vordefinierten Blätter mit ihren Namen.

# 4 Funktionsweise

Um echte Rekursion zu erzeugen, muß man in POVRay leider auf einen Trick zurückgreifen: Über die ausreichend mächtigen Stringfunktionen bastelt man sich POVRay-Anweisungen und schreibt diese per Debug-Stream (oder eines beliebigen für Benutzer vorgesehenen Streams) und der der entsprechenden Komanndozeilenanweisung (für den Debug-Stream -GD_FILENAME_) in eine Datei, die ebenfalls in das Scenefile eingebunden wird.

Um dabei dann Rekursion zu erhalten, muß der erzeugte Sourcecode einfach endrekursiv, also aus verschachtelten Schleifenanweisungen bestehen. Das ist der ganze Trick.

Möchte man die Rekursionsstiefe noch variabel halten, kann man statt direkter Verschachtelungen auch einfach zwei hintereinander gestellte while-Schleifen nehmen, die erste öffnet alle Rekursionsstufen, die zweite schließt sie wieder.

# 5 Allgemeines

## 5.1 Historie

1. Version 1.0: Dies ist die erste richtige Zusammenstellung der Include-Files. Vorige Versionen wurden zwar auch schon herumgereicht, aber immer mit unbearbeiteten Source-Codes und ohne richtige (und vor allem unvollständiger) Anleitung.

## 5.2 Todo

1. Frei definierbare Objekte für die Astformen, um die Äste interessanter zu formen.

2. Ein "Translation-Tracker", um die Äste und Blätter nach bestimmten angegebenen Richtungen zu orientieren (Schwerkraft, Wind, etc.)

3. Ausgabekosmetic mit korrekt eingerückten erzeugten Sourcecodes.

4. "Joker-Variablen", die bestimmte vordefinierte Klassen von Bäumen erzeugen (klein und verwuchert, hoch und schlank oder aber Eichen-ähnlich, Pinien-ähnlich etc. — ein sehr utopisches Ziel...)

5. Erweitertes und verbessertes leafshapes.inc

6. Auswahl an Rinden-Texturen in bark.inc

7. Erzeugen anderer Pflanzen durch Angabe einer Pflanzenklasse (es ist auch zur Zeit schon möglich, durch geschickte Auswahl der Parameter z.B. einen Kaktus zu erzeugen, aber nicht viel mehr...).

8. Überprüfen der Variablen-Werte vor dem Erzeugen der Bäume.

9. Die default-texture restaurieren.

## 5.3 Bugs

Vermutlich schon, sind mir aber noch nicht aufgefallen. Wer welche findet oder Fragen und Vorschläge hat, kann mit mailen: darüber bitte an:
stk@informatik.uni-bremen.de
(http://www.informatik.uni-bremen.de/~stk)

## 5.4 Dankeschön

Dank für Motivation und viele gute Ideen bei der Realisierung an die beiden anderen Mitglieder meiner POVRay-Spielgruppe Holger Bettag (http://www.informatik.uni-bremen.de/ ˜ hobold) und Harald Wagener (http://www.informatik.uni-bremen.de/˜hollow).

## 5.5 Copyright

Obwohl POVRay nicht direkt aus dem Bereich der GNU-Ecke kommt, stelle ich das Include unter dessen Lizenzbedingungen (Version 2 oder nach belieben neuer), siehe dazu die Datei COPYRIGHT.