

# **Erstellung eines Programms zur dreidimensionalen Darstellung ausgewählter Fraktale**

Max Seelemann

Sascha Grehl

Freitag, 14. Januar 2005

# 1 Bibliographisches Verzeichnis

## Bibliographisches Verzeichnis

Max Seelemann und Sascha Grehl  
September 2003 bis Januar 2005, Leipzig

Besondere Lernleistung  
Erstellung eines Programms zur dreidimensionalen Darstellung ausgewählter Fraktale

Umfang: 48 Seiten + 18 Seiten Anhang; CD-R mit Programmdateien

### Referat

Die Chaosforschung des ausgehenden 19. und des 20. Jahrhunderts wurde 1975 schlagartig durch die Erkenntnisse Benoit Mandelbrots revolutioniert, denn dieser hatte erkannt, dass hinter dem scheinbaren Chaos und den Unregelmäßigkeiten der Natur ein geradezu einfaches Prinzip steht - das Prinzip der Fraktalen Geometrie. Damit war es erstmals möglich, komplexe Muster wie Berge, Wolken oder Pflanzen einfach zu beschreiben und auch Vorgänge wie das Wetter oder Erdbeben wahrheitsgetreu zu simulieren. Aus diesem Grund haben wir uns den Körpern gewidmet, die unser Leben so entscheidend beeinflussen - den Fraktalen. Die Zielstellung dieser *Besonderen Lernleistung* war es, sich mit ausgewählten Fraktalen zu beschäftigen und sie mittels eines Computerprogrammes dreidimensional darzustellen. Speziell haben wir uns für Fraktale von platonischen Körpern entschieden, da diese Körper eine genaue Eingrenzung des Themengebietes zulassen.

### Abstract

The research of chaos in the end of the 19th and begin of the 20th century has been revolutionized by the knowledges of Benoit Mandelbrot. This man had recognized that there was a astonishing simple principle behind the apparent chaos and irregularities of the nature - the principle of fractal geometry. Using this principle it was possible for the first time to describe complex pattern like mountains, clouds or plants in a simple way and to simulate processes like the weather or earthquakes truthfully. Using this base we devoted ourself to the solids that influence our live in a that important meaning - the fractals. The goal of this *Besondere Lernleistung* was to study selected fractals and to display them three-dimensional using an computer application. We decided to study especially platonic solids, as these solids allow to further narrow down the topic.

# Inhaltsverzeichnis

<b>1</b>	<b>Bibliographisches Verzeichnis</b>	<b>2</b>
<b>2</b>	<b>Platonische Körper</b>	<b>6</b>
2.1	Übersicht . . . . .	6
2.1.1	Definition . . . . .	6
2.1.2	Platonische Körper . . . . .	6
2.1.3	Allgemeine Eigenschaften . . . . .	6
2.1.4	Dualität . . . . .	7
2.2	Tetraeder . . . . .	7
2.3	Hexaeder . . . . .	8
2.4	Oktaeder . . . . .	8
2.5	Dodekaeder . . . . .	9
2.6	Ikosaeder . . . . .	9
<b>3</b>	<b>Fraktale von Platonischen Körpern</b>	<b>10</b>
3.1	Definition . . . . .	10
3.2	Geschichtliches . . . . .	10
3.3	Mathematischer Hintergrund . . . . .	10
3.4	Die Selbstähnlichkeits-Dimension . . . . .	11
3.4.1	Selbstähnlichkeit . . . . .	11
3.4.2	Skalierungsfaktoren . . . . .	11
3.5	Die Box-Dimension . . . . .	12
<b>4</b>	<b>Rekursive Algorithmen zur Erzeugung von Fraktalen</b>	<b>14</b>
4.1	Algorithmen . . . . .	14
4.2	Rekursive Algorithmen . . . . .	14
4.3	Programmierung . . . . .	14
4.4	Begriffe am Beispiel der Koch Schneeflocke . . . . .	15
4.4.1	Iteration und Generator . . . . .	15
4.4.2	Algorithmen für Fraktale . . . . .	15
4.4.3	L-Systeme (Turtlegraphik) . . . . .	16
4.5	Die Bildungsvorschriften der Fraktale . . . . .	16
4.5.1	Sierpinski-Schwamm . . . . .	16
4.5.2	Menger-Schwamm . . . . .	17

4.5.3	Oktaeder-Schwamm . . . . .	17
4.5.4	Dodekaeder-Schwamm . . . . .	17
4.5.5	Ikosaeder-Schwamm . . . . .	17
<b>5</b>	<b>Das Programm</b>	<b>18</b>
5.1	Allgemein . . . . .	18
5.2	Oberfläche . . . . .	18
5.3	Funktionsweise . . . . .	19
5.3.1	Prinzip . . . . .	19
5.3.2	Speicherung von Matrizen und Objekten . . . . .	19
5.3.3	Multiplikation zweier Matrizen . . . . .	20
<b>6</b>	<b>Bilder der Fraktale</b>	<b>22</b>
6.1	Sierpinski-Schwamm . . . . .	22
6.2	Menger-Schwamm . . . . .	25
6.3	Oktaeder-Schwamm . . . . .	28
6.4	Dodekaeder-Schwamm . . . . .	32
<b>7</b>	<b>Matrizen und Vektoren</b>	<b>38</b>
7.1	Der Begriff des Vektors . . . . .	38
7.1.1	Der Vektorraum . . . . .	38
7.1.2	Der Vektor . . . . .	38
7.1.3	Addition von Vektoren . . . . .	38
7.1.4	Multiplikation von Vektoren (Kreuzprodukt) . . . . .	39
7.1.5	Normalisierung eines Vektors . . . . .	39
7.2	Der Begriff der Matrix . . . . .	39
7.2.1	Einheitsmatrix . . . . .	40
7.2.2	Addition von Matrizen . . . . .	40
7.2.3	Multiplikation von Matrizen . . . . .	40
<b>8</b>	<b>Geometrische Transformation</b>	<b>42</b>
8.1	Einführung . . . . .	42
8.2	2D-Transformationen . . . . .	42
8.2.1	Translation . . . . .	42
8.2.2	Skalierung . . . . .	42
8.2.3	Rotation . . . . .	43
8.3	Homogene Koordinaten . . . . .	43
8.3.1	Translation . . . . .	43
8.3.2	Skalierung . . . . .	44
8.3.3	Rotation . . . . .	44
8.3.4	Transformationsmatrix . . . . .	44
8.4	3D-Transformationen . . . . .	44
8.4.1	Translation . . . . .	44
8.4.2	Skalierung . . . . .	45

## *Inhaltsverzeichnis*

8.4.3	Rotation . . . . .	45
8.4.4	Transformationsmatrix . . . . .	45
<b>9</b>	<b>OpenGL</b>	<b>46</b>
9.1	Einführung . . . . .	46
9.2	Programmierung . . . . .	46
9.2.1	Erzeugung der Körper . . . . .	46
9.2.2	Drawlist . . . . .	47
9.2.3	Zeichnen der Szene . . . . .	47
<b>10</b>	<b>Quellen</b>	<b>48</b>
10.1	Bücher . . . . .	48
10.2	Internet . . . . .	48
<b>11</b>	<b>Selbstständigkeitserklärung</b>	<b>49</b>
<b>12</b>	<b>Anhang</b>	<b>50</b>

# 2 Platonische Körper

## 2.1 Übersicht

### 2.1.1 Definition

Platonische Körper sind reguläre, konvexe Polyeder.

Ein Polyeder ist ein dreidimensionaler Körper, der durch ebene Polygone (Vielecke) begrenzt wird. Daher müssen in jeder Ecke mindestens drei Vielecke zusammenstoßen, um eine räumliche Ecke zu bilden.

Ein Polyeder heißt regulär, wenn alle seine Oberflächen aus demselben regelmäßigen Vieleck bestehen, und konvex, wenn in jeder Körperecke die Summe der Winkel der anliegenden Ecken echt kleiner als  $360^\circ$  ist.

Für jeden Polyeder muss der Eulersche Polyedersatz gelten:

$$\text{Anzahl der Ecken} + \text{Anzahl der Flächen} - \text{Anzahl der Kanten} = 2$$

### 2.1.2 Platonische Körper

Aus den Definitionen ergibt sich, dass dafür nur fünf reguläre konvexe Polyeder, die auch platonische Körper genannt werden, in Frage kommen:

- Ein gleichseitiges Dreieck hat einen Innenwinkel von  $60^\circ$ , daher sind nur die Winkelsummen von Polyedern mit drei, vier und fünf Seitenflächen kleiner als  $360^\circ$ .  
⇒ **Tetraeder, Oktaeder, Ikosaeder**
- Ein Quadrat hat einen Innenwinkel von  $90^\circ$ , daher sind nur die Winkelsummen von Polyedern mit drei Seitenflächen kleiner als  $360^\circ$ .  
⇒ **Hexaeder**
- Ein regelmäßiges Fünfeck hat einen Innenwinkel von  $108^\circ$ , daher sind nur die Winkelsummen von Polyedern mit drei Seitenflächen kleiner als  $360^\circ$ .  
⇒ **Dodekaeder**

### 2.1.3 Allgemeine Eigenschaften

Jeder Platonische Körper besitzt eine Innenkugel (auch: *einbeschriebene Kugel*), auf der die Mittelpunkte sämtlicher Flächen des Körpers liegen, und eine Außenkugel (auch: *umbeschriebene Kugel*), auf der sämtliche Körperecken liegen.

## 2 Platonische Körper

regelmäßiger Körper	n	m	e	f	k	Oberflächeninhalt $A_O$	Volumen V
Tetraeder	3	3	4	4	6	$a^2 \cdot \sqrt{3}$	$\frac{\sqrt{2}}{12} \cdot a^3$
Hexaeder	4	3	8	6	12	$6 \cdot a^2$	$a^3$
Oktaeder	3	4	6	8	12	$2 \cdot a^2 \cdot \sqrt{3}$	$\frac{\sqrt{2}}{3} \cdot a^3$
Dodekaeder	5	3	20	12	30	$3 \cdot a^2 \cdot \sqrt{25 + 10 \cdot \sqrt{5}}$	$\frac{15+7\sqrt{5}}{4} \cdot a^3$
Ikosaeder	3	5	12	20	30	$5 \cdot a^2 \cdot \sqrt{3}$	$\frac{15+5\sqrt{5}}{12} \cdot a^3$

$n$  = Anzahl der Kanten einer Fläche ( $n$ -Eck),  $m$  = Anzahl der Flächen an einer Ecke,  
 $e$  = Anzahl der Ecken,  $f$  = Anzahl der Flächen,  $k$  = Anzahl der Kanten

### 2.1.4 Dualität

Dualität ist eine weitere Eigenschaft der Platonischen Körper, welche bedeutet, dass die Mittelpunkte der Flächen des Körpers gleichzeitig die Eckpunkte des dualen Körpers sind. So ist der Oktaeder dem Hexaeder, der Ikosaeder dem Dodekaeder und der Tetraeder sich selbst dual.

Man kann in der Tabelle oben erkennen, dass sich die Anzahl der Kanten einer Fläche ( $n$ ) und die Anzahl der Flächen an einer Ecke ( $m$ ) beim Hexaeder und Oktaeder (sowie beim Dodekaeder und beim Ikosaeder) über Kreuz vertauschen lassen, ohne dass sich die Aussage ändert. Beim Tetraeder (der zu sich selbst dual ist) kann man sie direkt vertauschen. Das Selbe gilt auch für Anzahl der Ecken ( $e$ ) und die Anzahl der Flächen ( $f$ ).

## 2.2 Tetraeder

Der Tetraeder ist ein Polyeder, der aus 4 (grch.: tetra) gleichen Dreiecken besteht. Der Tetraeder hat 4 Ecken, an denen jeweils 3 Dreiecke zusammenstoßen, und 6 gleich lange Kanten.

Der Tetraeder gehört zu den Pyramiden, mit der Besonderheit, dass er, egal auf welche Seite gestellt, immer die gleiche Pyramide ergibt. Außerdem lässt er sich so in zwei Teile schneiden, dass die Schnittfläche ein Quadrat ergibt.

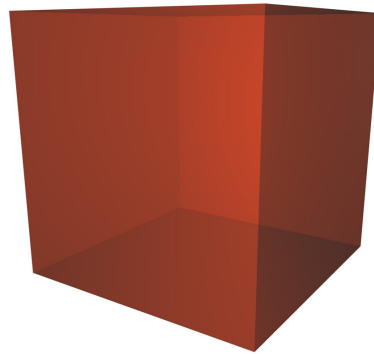


regelmäßiger Körper	n	m	e	f	k	Oberflächeninhalt $A_O$	Volumen V
Tetraeder	3	3	4	4	6	$a^2 \cdot \sqrt{3}$	$\frac{\sqrt{2}}{12} \cdot a^3$

## 2.3 Hexaeder

Der Hexaeder ist ein Polyeder, der aus 6 (grch.: hexa) gleichen Quadraten besteht. Der Hexaeder hat 8 Ecken, an denen jeweils 3 Vierecke zusammenstoßen, und 12 gleich lange Kanten.

Der Hexaeder gehört auch zu den Prismen, mit der Besonderheit, dass er, egal auf welche Seite gestellt, immer das gleiche Prisma ergibt. Außerdem lässt er sich so in zwei Teile zerschneiden, dass die Schnittfläche ein gleichseitiges Sechseck ergibt.



regelmäßiger Körper	n	m	e	f	k	Oberflächeninhalt $A_O$	Volumen $V$
Hexaeder	4	3	8	6	12	$6 \cdot a^2$	$a^3$

## 2.4 Oktaeder

Der Oktaeder ist ein Polyeder, der aus 8 (grch.: okta) gleichen gleichseitigen Dreiecken besteht. Der Oktaeder hat 6 Ecken, an denen jeweils 4 Dreiecke zusammenstoßen, und 12 gleich lange Kanten.

Den Oktaeder kann man auch durch zwei gleichmäßige vierseitige Pyramiden, die an den Grundflächen aneinander liegen, bilden. Damit ist er eine Bipyramide. Auch kann man jeweils 4 Kanten so wählen, dass sie ein Quadrat ergeben. Von diesen Verbindungen gibt es 3.

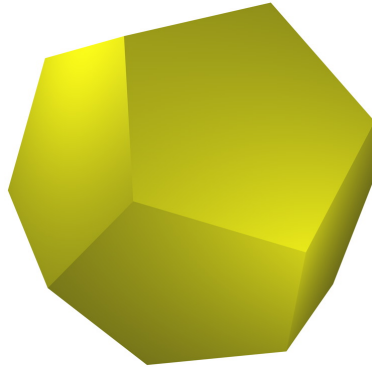


regelmäßiger Körper	n	m	e	f	k	Oberflächeninhalt $A_O$	Volumen $V$
Oktaeder	3	4	6	8	12	$2 \cdot a^2 \cdot \sqrt{3}$	$\frac{\sqrt{2}}{3} \cdot a^3$



## 2.5 Dodekaeder

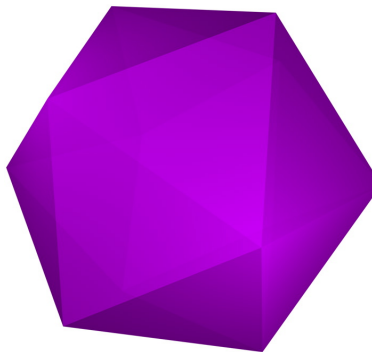
Der Dodekaeder (auch: Pentagondodekaeder) ist ein Polyeder, der aus 12 (grch.: dodeka) gleich großen, gleichseitigen, gleichwinkligen Fünfecken besteht. Der Dodekaeder hat 20 Ecken, an denen jeweils 3 Fünfecke zusammenstoßen, und 30 gleich lange Kanten.



regelmäßiger Körper	n	m	e	f	k	Oberflächeninhalt $A_O$	Volumen $V$
Dodekaeder	5	3	20	12	30	$3 \cdot a^2 \cdot \sqrt{25 + 10 \cdot \sqrt{5}}$	$\frac{15+7 \cdot \sqrt{5}}{4} \cdot a^3$

## 2.6 Ikosaeder

Der Ikosaeder ist ein Polyeder, der aus 20 (grch.: ikosa) gleich großen gleichseitigen Dreiecken besteht. Der Ikosaeder hat 12 Ecken, an denen jeweils 5 Dreiecke zusammenstoßen, und 30 gleich lange Kanten.



regelmäßiger Körper	n	m	e	f	k	Oberflächeninhalt $A_O$	Volumen $V$
Ikosaeder	3	5	12	20	30	$5 \cdot a^2 \cdot \sqrt{3}$	$\frac{15+5 \cdot \sqrt{5}}{12} \cdot a^3$

# 3 Fraktale von Platonischen Körpern

## 3.1 Definition

„Fraktal“ ist ein 1977 von Benoit Mandelbrot geprägter Begriff (lat. fractus, frangere: brechen, in Stücke zerbrechen, irregulär), der alle natürlichen oder künstlichen Gebilde oder Muster bezeichnet, die einen hohen Grad von Selbstähnlichkeit aufweisen.

Dabei wird zwischen deterministischen und stochastischen Fraktalen unterschieden. Die Deterministischen weisen eine sehr reguläre Struktur auf und besitzen eine feste Bildungsvorschrift (z.B. das Sierpinski-Dreieck), während bei den Stochastischen der Generierungsprozess weiterhin durch einen Zufallsfaktor mitbestimmt wird. Die stochastischen Fraktale werden meist zur virtuellen Berechnung von Bergen, Wolken oder ähnlichen Dingen verwendet. Da sie diese Arbeit hauptsächlich auf platonische Körper beziehen, werden nur deterministische Fraktale betrachtet.

## 3.2 Geschichtliches

1918 veröffentlichte der französische Mathematiker Gaston Julia (1893 - 1978) nach Lazarett-aufenthalt im ersten Weltkrieg seine Arbeit über rationale Iterationen. Erst später erkannte man, welche faszinierenden Sachverhalte sich hinter seinem nüchternen Formelwerk verbargen. Eng damit verknüpft sind die Begriffe „Fraktale“ und „Julia-Mengen“, deren Theorien der modernen Mathematik mit dem Zweig der fraktalen Geometrie und der Chaos-Forschung neue Impulse gaben.

Dem französisch-amerikanischen Mathematiker Benoit Mandelbrot (geb. 1924 in Warschau) gelang es, die fast in Vergessenheit geratene Arbeit von Julia wieder aufzugreifen, die mathematischen Zusammenhänge mit dem Computer zu veranschaulichen und auf dem Bildschirm sichtbar zu machen. Aus dem Jahr 1980 stammen die ersten Bilder, die als „Mandelbrot-Menge“ oder „Apfelmännchen“ um die Welt gingen.

## 3.3 Mathematischer Hintergrund

Eine Linie hat die Dimension 1.  
Eine Fläche hat die Dimension 2.  
Ein Körper hat die Dimension 3.  
Ein Fraktal hat die Dimension  $dim$ .

### 3 Fraktale von Platonischen Körpern

- Wenn man eine Strecke um den Faktor  $\frac{1}{2}$  skaliert dann brauchen man 2 Strecken, um wieder auf die alte Strecke zu kommen.
- Skaliert man allerdings ein Quadrat um den Faktor  $\frac{1}{2}$  braucht man 4 Quadrate, um das ursprüngliche Quadrat zu erhalten.
- Wird ein Würfel um den Faktor  $\frac{1}{2}$  skaliert, braucht man 8 dieser Würfel um wieder den original Würfel zu erhalten.

Daraus ergibt sich auch die Gleichung für die Dimension  $D$ :

$$(\text{Reziproker Skalierungsfaktor})^D = (\text{Anzahl der Teile})$$

Fraktale haben die besondere Eigenschaft, dass sie sich beim Skalieren weder wie Linien, Flächen oder Körper verhalten. Man muss bei ihnen zulassen, dass die Dimension nicht notwendigerweise eine ganze Zahl ist. Sie kann also eine Bruchzahl oder eine beliebige Kommazahl sein. Weil „fractum“ (lat.) gebrochen, bzw. „fraction“ (engl.) Bruch heißt, nennt man diese geometrischen Gebilde **Fraktale**.

Nicht genug, dass die Fraktale eine Erweiterung des Dimensionsbegriffes erfordern, es sind, je nach Art des oben genannten Maßes, auch noch verschiedene „Dimensionen“ für ein Fraktal sinnvoll. Zum Beispiel: Selbstähnlichkeits-Dimension, Zirkel-Dimension und Box-Dimension. Dies alles sind spezielle Formen der fraktalen Dimension.

## 3.4 Die Selbstähnlichkeits-Dimension

### 3.4.1 Selbstähnlichkeit

Strukturen sind sich selbst ähnlich, wenn bei ihrer der Zerlegung in beliebig kleine Teile jedes dieser Teile eine Kopie der ganzen Struktur ist. Hierbei ist es wichtig, dass diese kleinen Teile tatsächlich aus der ganzen Struktur durch eine Ähnlichkeitstransformation hervorgehen. (siehe Kapitel 7, Geometrische Transformationen)

Doch ist nicht jede sich selbst ähnliche Struktur ein Fraktal. So können z.B. eine Strecke, ein Quadrat oder ein Würfel mit Hilfe von Selbstähnlichkeitstransformation in kleine Kopien zerlegt werden. Diese Strukturen sind jedoch keine Fraktale.

### 3.4.2 Skalierungsfaktoren

Zwischen der Anzahl der Teile  $a$ , die bei einer Zerlegung aus einem Objekt entstehen, der Dimension  $D$  und dem Verkleinerungsfaktor  $s$  gilt folgendes Gesetz:

$$a = \frac{1}{s^D}$$

Durch equivalentes Umformen ergibt sich folgende wichtige Gleichung zur Berechnung der fraktalen Dimension:

$$D = -\frac{\log a}{\log s}$$

Genauer gesagt lässt sich auf diesem Wege Selbstähnlichkeits-Dimension  $D$  berechnen, welche bei Strecken 1, bei Flächen 2 und bei Körpern 3 ist. So ergibt sich bei der Zerlegung eines Würfels mit dem Verkleinerungsfaktor  $s = \frac{1}{3}$ , dass die Anzahl der Teile  $a = 27$  ist. Im Gegensatz hierzu gilt für die Koch-Kurve bei einem Verkleinerungsfaktor von  $s = \frac{1}{3}$  und einer Anzahl der neuen Teile von  $a = 4$ , dass die Dimension  $D \approx 1,2619$  ist. (siehe Anhang 1)

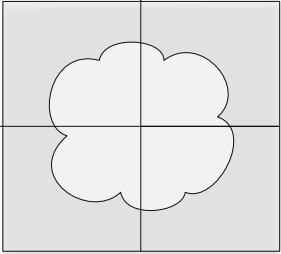
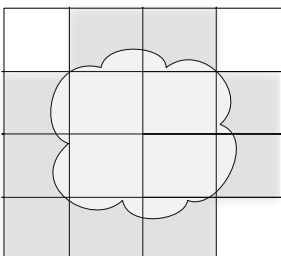
### 3.5 Die Box-Dimension

Bei nicht selbstähnlichen Fraktalen ist es nicht möglich die Selbstähnlichkeitsdimension  $D$  zu ermitteln, allerdings kann man die Box-Dimension  $D_{Box}$  bestimmen.  $D_{Box}$  bestimmt man durch Messungen von Fraktalen, die mit verschiedenen Box-Gittern gerastert werden. Bei jeder geometrischen Figur, d.h. auch bei jedem Fraktal, kann man  $D_{Box}$  wenigstens näherungsweise bestimmen.

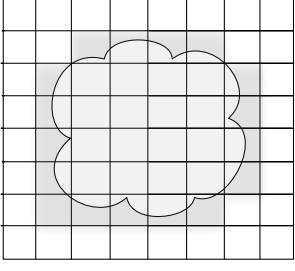
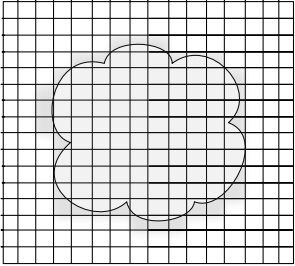
Dazu legt man auf die Struktur ein regelmäßiges Gitternetz mit der Maschenweite  $s$ . Nun werden Gittermaschen gezählt, die von der Struktur berührt werden. Die Anzahl dieser ist  $N(s)$ . Halbiert man nun die Maschenweite, bestimmt abermals  $N(s)$  und trägt die Werte in ein doppellogarithmisches Diagramm ( $\log(N(s))-\log(\frac{1}{s})$ -Diagramm) ein, so wird die Box-Dimension annähernd durch den Anstieg der Geraden durch die zwei Punkte bestimmt. Zur genaueren Berechnung der Dimension kann die Maschenweite weiter verkleinert werden. Der Anstieg der Ausgleichsgeraden durch diese Punkte entspricht dann der Box-Dimension. Um genaue Größe der Boxdimension herauszufinden, kann man einen Limes bilden:

$$D_{Box} = - \lim_{s \rightarrow 0} \frac{\log N(s)}{\log \frac{1}{s}}$$

Die Boxdimension ist vor allem für die Anwendung des Fraktalbegriffs in den Naturwissenschaften und in der Medizin wichtig und sinnvoll. Am einfachen Beispiel der Wolke lässt es sich die Box-Dimension anschaulich erklären:

	<p>Im ersten Bild werden bei einfacher Teilung noch alle 4 Kästchen berührt, deswegen berechnet sich die Box-Dimension wie folgt:</p>	$D_{Box} = - \frac{\log 4}{\log \frac{1}{2}} = 2$
	<p>Im zweiten Bild werden nach Halbierung nur noch 13 der 16 Kästchen berührt, was dazu führt das sie die Box-Dimension wie folgt verändert:</p>	$D_{Box} = - \frac{\log 13}{\log \frac{1}{4}} \approx 1,850$

### 3 Fraktale von Platonischen Körpern

	<p>Bild drei ist schon genauer, hier berühren auch nur noch 33 der 64 Kästchen die Wolke.</p>	$D_{Box} = -\frac{\log 33}{\log \frac{1}{8}} \approx 1,681$
	<p>Im letzten Bild sind es nach viermaliger Teilung sogar nur noch 106 der 256 Kästchen, die die Wolke berühren. Die Box-Dimension lässt sich damit schon recht genau berechnen:</p>	$D_{Box} = -\frac{\log 106}{\log \frac{1}{16}} \approx 1,682$

# 4 Rekursive Algorithmen zur Erzeugung von Fraktalen

## 4.1 Algorithmen

Algorithmen treten im Rahmen der Mathematik als allgemeine Verfahren zur Lösung von allen Aufgaben einer bestimmten Aufgabenklasse in Erscheinung. Durch sie sollen Prozesse so beschrieben werden, dass sie danach von einer Maschine nachgebildet oder gesteuert werden können. Ein Algorithmus hat die Eigenschaft, gegebene Größen auf Grund eines Systems von Regeln in Ausgabegrößen umzuformen. Ein Algorithmus weist drei wesentliche Merkmale auf:

1. Das System der Größen, die ineinander umgearbeitet werden, muß effektiv gegeben sein.
2. Der Algorithmus muß durch endlich viele Regeln beschreiben werden können.
3. Das Abarbeiten des Algorithmus geht in Form von Arbeitstakten und Prüftakten vor sich. Ein Arbeitstakt besteht darin, daß eine der gegebenen Regeln angewendet wird; ein Prüftakt besteht darin, daß das Zutreffen einer Bedingung nachgeprüft wird.

*Ein Beispiel:*

$$f_M(n) = \begin{cases} 0, & \text{falls } n \notin M \\ 1, & \text{falls } n \in M \end{cases}$$

## 4.2 Rekursive Algorithmen

Ein rekursiver Algorithmus hat, neben den Eigenschaften eines normalen Algorithmus, die Besonderheit, dass mindestens eine seiner Regeln sich wieder selbst aufruft.

*Ein Beispiel:* Die Summe aller ganzen Zahlen von 1 bis zu einer ganzen Zahl  $n$  soll ermittelt werden. Zum Beispiel für 5 lautet die Formel wie folgt:  $f_s(5) = 1 + 2 + 3 + 4 + 5 = 15$ . Diese Zahlensumme kann mathematisch induktiv als Funktion von  $n$  definiert werden :

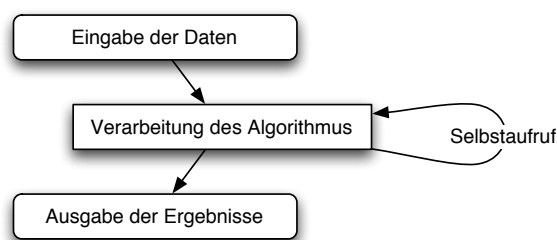
$$f_s(n) = \begin{cases} 0, & \text{falls } n = 0 \\ n + f_s(n - 1), & \text{falls } n > 0 \end{cases}$$

## 4.3 Programmierung

Es gibt in den meisten Programmiersprachen die Möglichkeit, dass sich Prozeduren und Funktionen selbst aufrufen. Diese Programmierweise bezeichnet man als rekursives Programmieren.

**Definition:** Eine Prozedur (Funktion oder Algorithmus) heißt rekursiv, wenn in ihrem Anwendungsteil ein Aufruf von ihr selbst steht (direkte Rekursion). Damit die Rekursion terminiert, muss ein Rekursionsende (bzw. eine Rekursionsbedingung) gegeben sein. (lat. recurrere = zurücklaufen)

Mit der rekursiven Programmieretechnik kann man sehr komplexe Aufgabenstellungen durch erstaunlich kurze und einfache Algorithmen programm-technisch sauber lösen. Ein Algorithmus beschreibt präzise, wie der Computer aus Eingabedaten schrittweise nach der ihm übergebenen Verarbeitungsvorschrift die gewünschten Ausgabedaten zu erzeugen hat (Input-Output). Rekursive Algorithmen sind also eine Variante von Algorithmen, die von sich selber beschrieben werden. Rekursion ist daher nicht nur eine Programmieretechnik, sondern ein wichtiges algorithmisches Prinzip zur Problemlösung:



### 4.4 Begriffe am Beispiel der Koch Schneeflocke

Die auch Schneeflocken-Kurve genannte iterative Folge zur Erzeugung eines, einer Schneeflocke ähnelnden, Fraktals soll hier als Beispiel zur Begriffserklärung dienen. (siehe Anhang 1)

#### 4.4.1 Iteration und Generator

Die Iteration bezeichnet eine Methode, sich der Lösung eines Problems schrittweise, aber zielgerichtet anzunähern. Der Generator ist dabei die Vorschrift (bzw. die Abfolge von Befehlen), die bei einem Schritt ausgeführt wird. Bei einem Iterationsschritt (häufig auch Iteration genannt) werden die Ergebnisse des vorherigen Schrittes weiterverarbeitet. In der Programmieretechnik ist eine Iteration normalerweise durch eine Schleife realisiert.

Bei einem Iterationsschritt bei der Schneeflocke wird jede Strecke in drei Teile zerlegt und wie folgt verändert:

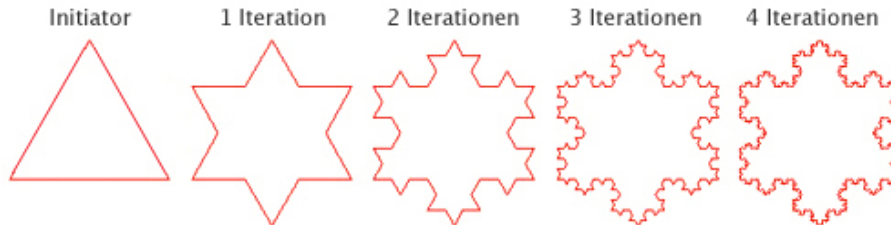


#### 4.4.2 Algorithmen für Fraktale

Der Begriff „Algorithmus“ schließt die Startfigur, die Erzeugungsweise (hier: Methode der Iteration) und den Generator ein. Die Startfigur, also das Objekt, auf das der erste Schritt der

Iteration angewendet wird, heißt auch Initiator.

Bei der Schneeflocke ist der Initiator ein Dreieck mit gleichen Seiten. Wenn man den Algorithmus mit bestimmter Iterationszahl (also der Anzahl der durchgeführten Iterationsschritte) anwendet erhält man folgende Figuren:



### 4.4.3 L-Systeme (Turtlegraphik)

Eine häufig benutztes System, Fraktale mit Programmen zu erzeugen, beruht auf der Verwendung des Lindenmayer-Systems (auch Turtlegraphik genannt). Dieses beinhaltet einfache geometrische Darstellungen und Operationen, die mit nur wenigen Zeilen Code und schrittweise nacheinander ausgeführten Vorschriften Fraktale zeichnen können.

Auch hier soll die Schneeflocke als Beispiel dienen. Zuerst müssen die Operationen vereinbart werden:

- F = um Strecke  $\frac{l}{n}$  vorwärts für festes n
- + = um den festen Winkel  $\alpha$  in positiver Richtung drehen
- = um den festen Winkel  $\alpha$  in negativer Richtung drehen

Für die Koch Schneeflocke lässt sich ein einfaches System von Vorschriften finden:

$n$	$\alpha$	Initiator	Generator
3	$60^\circ$	$F - F - F$	$F + F - -F + F$

Dieses System ist für Programme zu empfehlen, die eine dynamische Erzeugung von Fraktalen beabsichtigen und dies möglichst einfach umsetzen wollen. Da sich diese Arbeit allerdings nur auf Platonische Körper mit festen Bildungsvorschriften bezieht, würde die Verwendung von Turtlegraphik nur zu unnötigem Speicherverbrauch führen und somit das Programm langsamer machen. Des weiteren sind die Lindenmayer-Systeme eher für den zweidimensionalen Anwendungsfall geeignet.

## 4.5 Die Bildungsvorschriften der Fraktale

### 4.5.1 Sierpinski-Schwamm

Der Initiator des Sierpinski-Schwammes ist ein Tetraeder. Dieser wird auf halbe Kantenlänge skaliert und 3 mal kopiert (so dass man 4 kleine Tetraeder hat). Diese werden anschließend durch Translation so bewegt, dass sie in den Ecken des ursprünglichen Körpers liegen. Bei jeder weiteren Iteration wird diese Vorschrift auf jeden der neuen Tetraeder angewandt.



### 4.5.2 Menger-Schwamm

Der Initiator des Menger-Schwammes ist ein Hexaeder (Würfel). Dieser wird auf  $\frac{1}{3}$  der Originalgröße skaliert und 19 mal kopiert. Die 20 neuen Körper werden so verschoben, dass sie jeweils an den Kanten des ursprünglichen Körpers anliegen. Eine andere Darstellung wäre, dass man den Hexaeder in Höhe, Breite und Tiefe jeweils in drei Teile teilt und alle Hexaeder entfernt, die nicht an einer Außenkante liegen.

### 4.5.3 Oktaeder-Schwamm

Beim Oktaeder-Schwamm ist der Initiator ein Oktaeder. Dieser wird auf halbe Größe skaliert und 5 mal kopiert. Die 6 neuen Körper werden so verschoben, dass sie jeweils an den Ecken des ursprünglichen Körpers liegen und sich an den Kanten berühren.

### 4.5.4 Dodekaeder-Schwamm

Beim Dodekaeder-Schwamm ist der Initiator ein Dodekaeder, der beim ersten Iterationsschritt in verschiedene Körper zerlegt wird. Das sind die S-, C-, H-, W-Shapes und ein Dodekaeder. Nach dem ersten Schritt beginnt die eigentliche Iteration, wobei nur noch die Shapes in mehrere Körper zerlegt werden und die Dodekaeder unberührt bleiben. Genauere soll an dieser Stelle nicht darauf eingegangen werden, da alleine nach dem ersten Schritt über 150 neue Körper entstehen. Veranschaulicht wird dies im Kapitel 5.

### 4.5.5 Ikosaeder-Schwamm

Derzeit wird noch daran geforscht, eine Zerlegungsregel für den Ikosaeder-Schwamm zu finden, die dem Prinzip der Zerlegungsregel des Dodekaeders entspricht.

# 5 Das Programm

Der eigentliche Inhalt dieser Arbeit war es, ein Programm zu erstellen, welches bestimmte Fraktale mit einer bestimmten Iterationstiefe errechnen und darstellen kann. Dass dabei keine „echten“ Fraktale entstehen, dürfte klar sein, denn ein Fraktal hat eine unendliche Iterationstiefe und um dieses zu errechnen bräuchte, man unendlich viel Zeit.

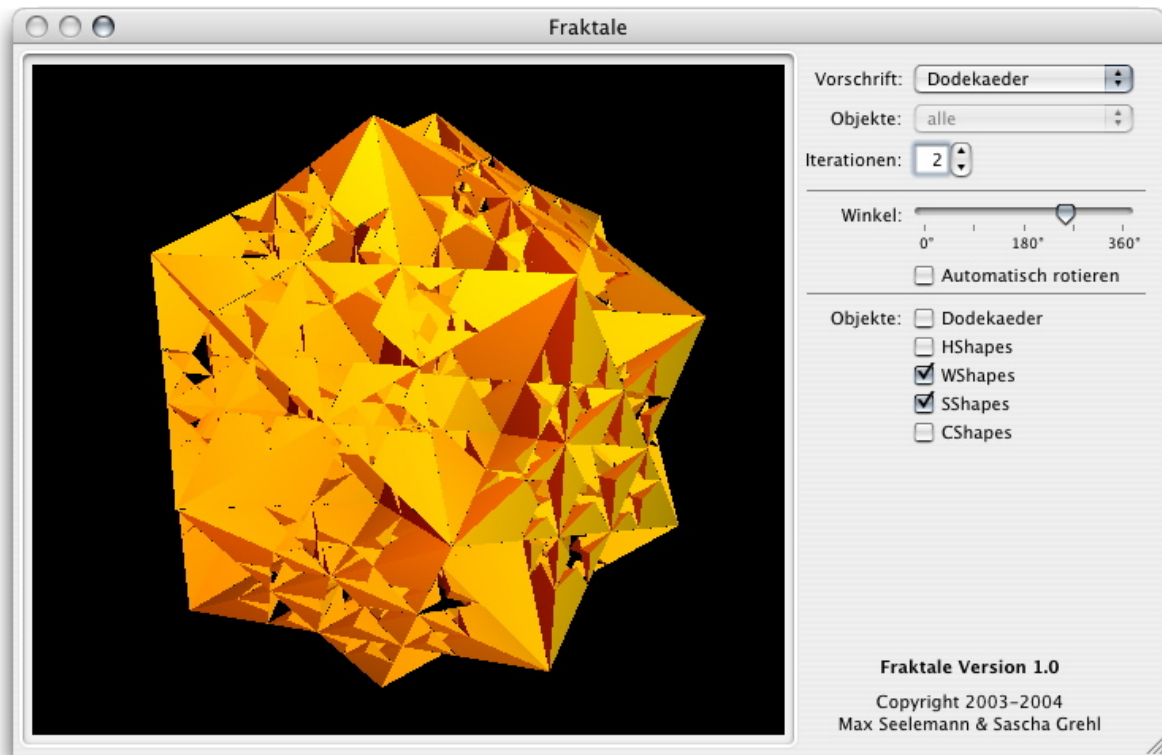
## 5.1 Allgemein

Das Programm wurde in „Objective C++“ geschrieben und stellt die verschiedenen Fraktale der platonischen Körper in frei wählbarer Iterationstiefe dar. Des weiteren ist es möglich, sich beim Dodekaeder-Schwamm die einzelnen Objekt-Typen anzeigen bzw. ausblenden zu lassen (wodurch auch wieder sehr interessante Figuren entstehen).

Auf viel Quellcode in dieser Dokumentation soll verzichtet werden, da dieser zum einen komplett auf der beigelegten CD enthalten ist und zum anderen im ausgedruckten Zustand über 100 Seiten veranschlagen würde. Im Gegenzug werden einige wenige OpenGL-Befehle im Kapitel 8 erklärt.

## 5.2 Oberfläche

Das Programm hat nur ein Fenster. Im linken, größeren Bereich befindet sich das Anzeigefenster, worauf die Fraktale gezeichnet werden. Im rechten Teil des Fensters ist die Benutzeroberfläche zu finden, mit der der Nutzer interagiert. Mit dem PopUp-Button kann man die fraktale Vorschrift auswählen, die betrachtet werden soll. Das Eingabefeld mit der Iterationstiefe gibt an, für die wievielte Iteration das Fraktal berechnet werden soll. Mit dem Winkel kann man einstellen, unter welchem Winkel das Fraktal betrachtet werden soll. Dabei kann die Funktion „Automatisch rotieren“ ausgewählt werden, die das automatische Drehen des Körpers aktiviert. Ist der Dodekaeder-Schwamm aktiviert, dann erscheinen außerdem noch mehrere Auswahlboxen, mit denen man die verschiedenen Objekte anzeigen, bzw. ausblenden kann.



## 5.3 Funktionsweise

### 5.3.1 Prinzip

Die Ausgangskörper werden als erstes mittels OpenGL Drawlists zur späteren Verwendung gespeichert. Dann werden rekursiv die Matrizen der einzelnen Iterationen miteinander multipliziert, so dass man am Ende nur noch eine einzige eindimensionale Liste aller Objekte und ihrer Positionen/Größen/Ausrichtungen hat. Diese Objekte werden (nachdem sie nach den vorgenommenen Einstellungen gefiltert wurden) wiederum mittels OpenGL in einer neuen Drawlist gespeichert. Diese Drawlist wird (wieder den Einstellungen entsprechend) in der Zeichen-Routine auf dem Bildschirm gezeichnet.

### 5.3.2 Speicherung von Matrizen und Objekten

Die Typdefinition einer Matrix sieht wie folgt aus:

```
typedef struct HomogeneMatrix {
    float m[4][4];
} HomogeneMatrix;
```

Ein einfaches Beispiel hierfür wäre eine Normalmatrix:

```
HomogeneMatrix normalMatrix = {
```

```

    {
    {1.0f, 0.0f, 0.0f, 0.0f},
    {0.0f, 1.0f, 0.0f, 0.0f},
    {0.0f, 0.0f, 1.0f, 0.0f},
    {0.0f, 0.0f, 0.0f, 1.0f}
    }
};

```

Die Typdefinition eines Objektes sieht wie folgt aus:

```

typedef struct AffineObject {
    HomogeneMatrix m;
    int type;
} AffineObject;

```

Dabei kann `type` hier den Wert einer der folgenden Konstanten annehmen:

```

enum {
    ObjectTypeSirpinski = 1,
    ObjectTypeMenger = 2,
    ObjectTypeOktaeder = 3,
    ObjectTypeDodekaeder = 4,
    ObjectTypeHShape = 5,
    ObjectTypeWShape = 6,
    ObjectTypeSShape = 7,
    ObjectTypeCShape = 8
};

```

Ein Beispiel für einen einfachen, untransformierten Dodekaeder sieht wie folgt aus:

```

AffineObject Dodekaeder = {
    {{
    {1.0f, 0.0f, 0.0f, 0.0f},
    {0.0f, 1.0f, 0.0f, 0.0f},
    {0.0f, 0.0f, 1.0f, 0.0f},
    {0.0f, 0.0f, 0.0f, 1.0f}
    }}, ObjectTypeDodekaeder
};

```

### 5.3.3 Multiplikation zweier Matrizen

Der wohl wichtigste Abschnitt des Programmes ist die Multiplikation zweier homogener Matrizen. Diese Funktion wird sehr oft aufgerufen - nämlich bei jeder Zerlegung jedes Objektes, aus diesem Grund muss sie auch sehr schnell sein.

```

- (HomogeneMatrix)multiplyMatrix:(HomogeneMatrix)m1 with:(HomogeneMatrix)m2
{

```

## 5 Das Programm

```
HomogeneMatrix *n;
unsigned i,j,k;

n = (HomogeneMatrix *) malloc(sizeof(HomogeneMatrix));

for (i=0; i<4; i++)
{
    for (j=0; j<4;j++)
    {
        (*n).m[i][j] = 0;

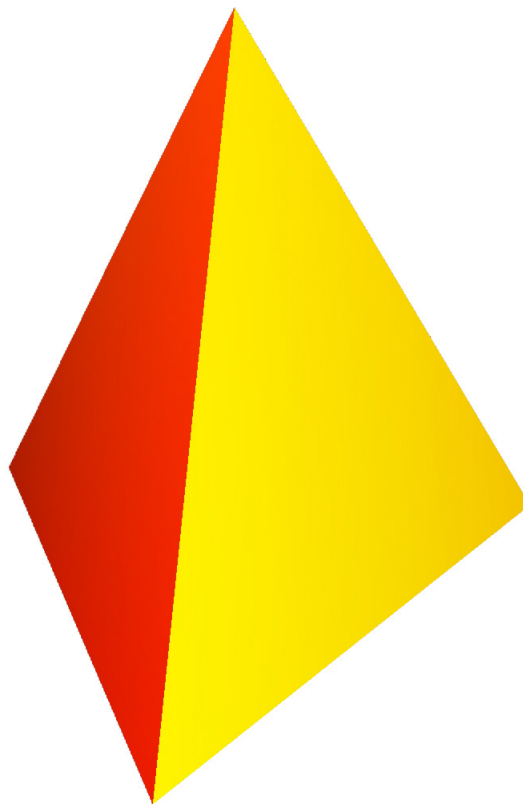
        for (k=0; k<4; k++)
        {
            (*n).m[i][j] += m1.m[i][k] * m2.m[k][j];
        }
    }
}

return (* n);
}
```

# 6 Bilder der Fraktale

## 6.1 Sierpinski-Schwamm

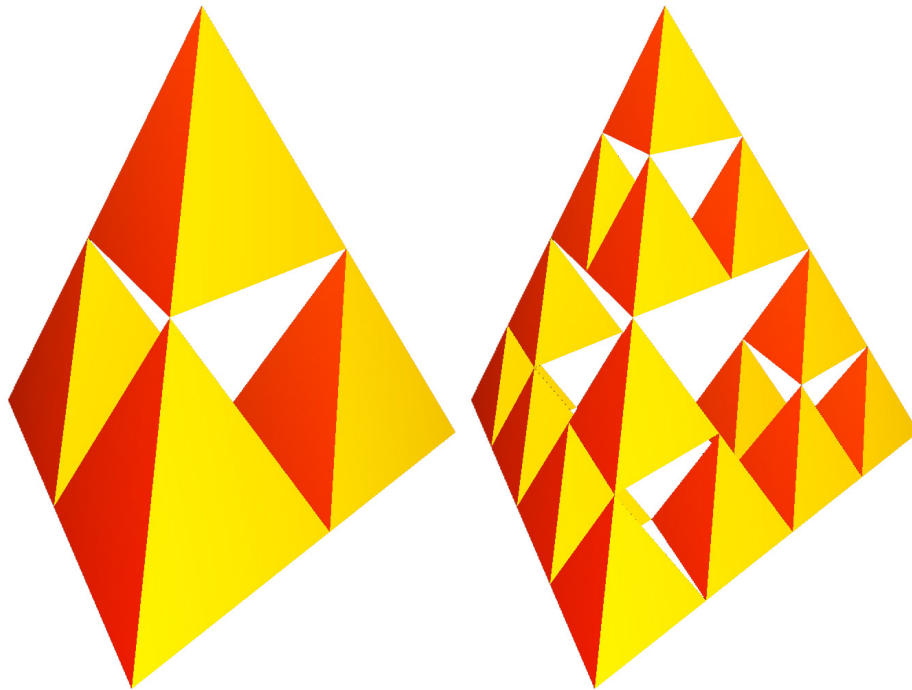
Der Initiator des Sierpinski-Schwammes ist (wie bereits im Kapitel 3.5.1 erwähnt) ein Tetraeder:



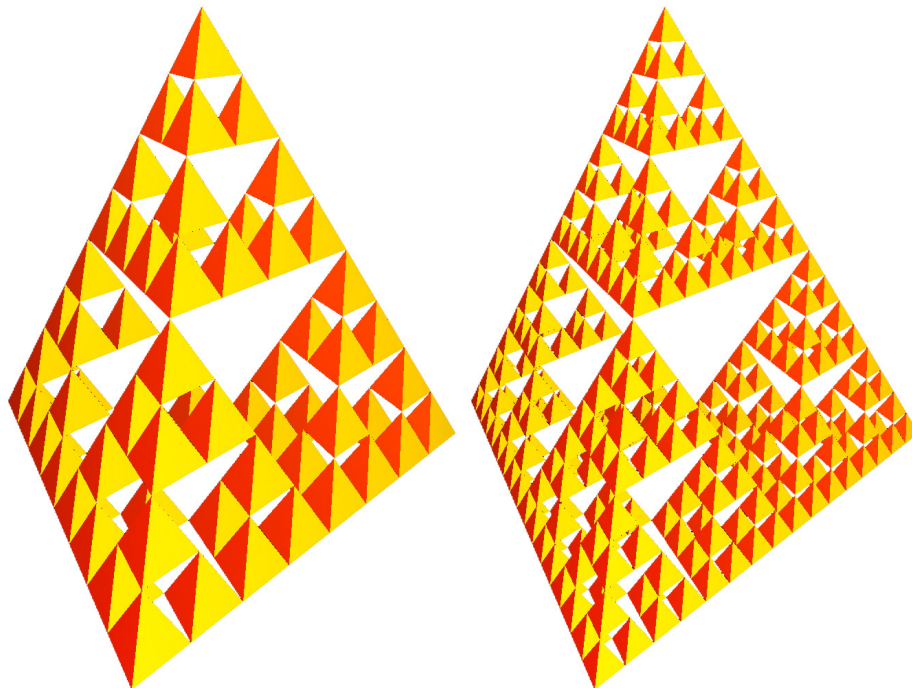
## 6 Bilder der Fraktale

Die weiteren Iterationen sehen wie folgt aus:

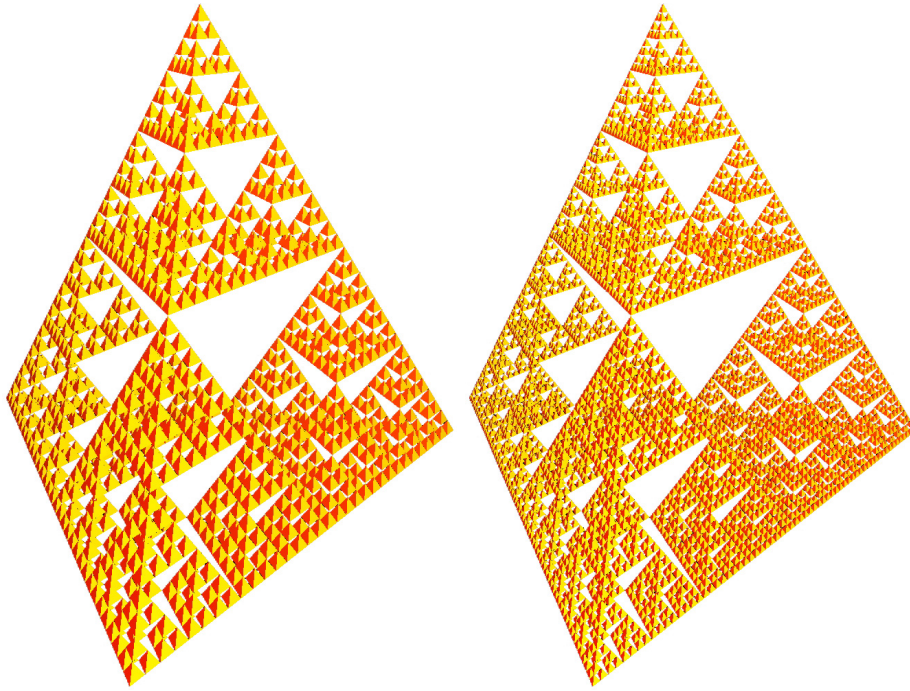
### 1. und 2. Iteration



### 3. und 4. Iteration



5. und 6. Iteration



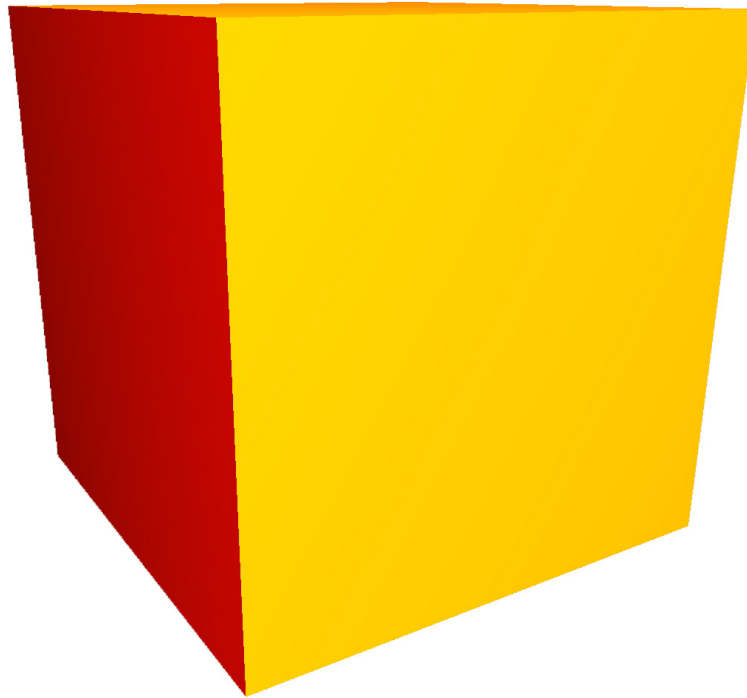
7. und 8. Iteration





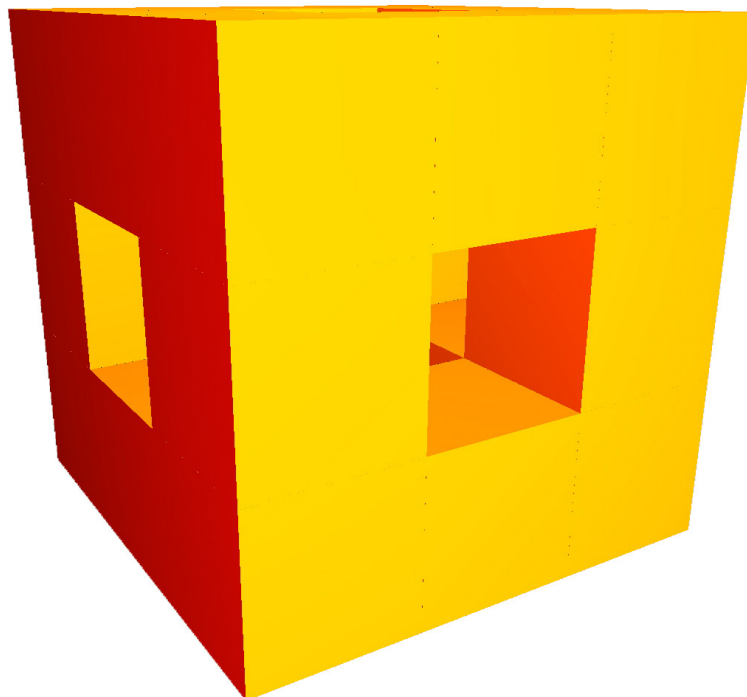
## 6.2 Menger-Schwamm

Der Initiator des Menger-Schwammes ist (wie bereits im Kapitel 3.5.2 erwähnt) ein Würfel:

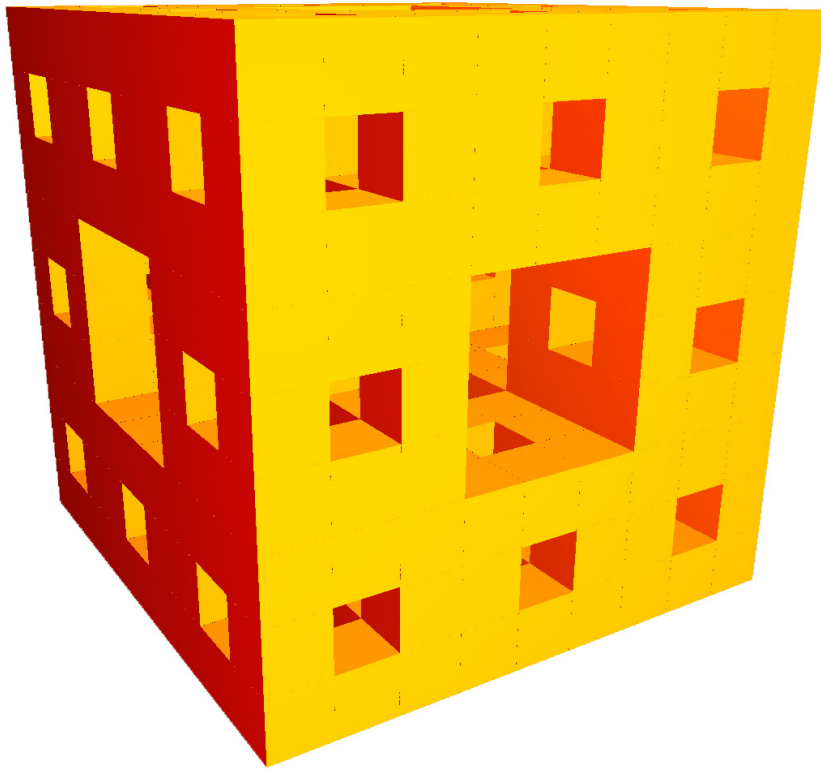


Die weiteren Iterationen sehen wie folgt aus:

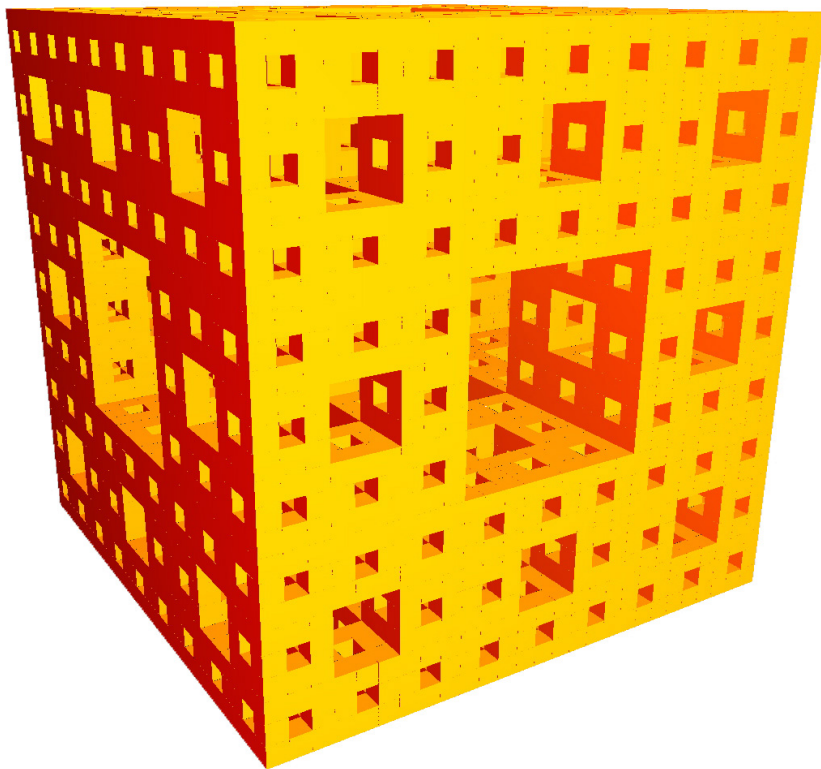
### 1. Iteration



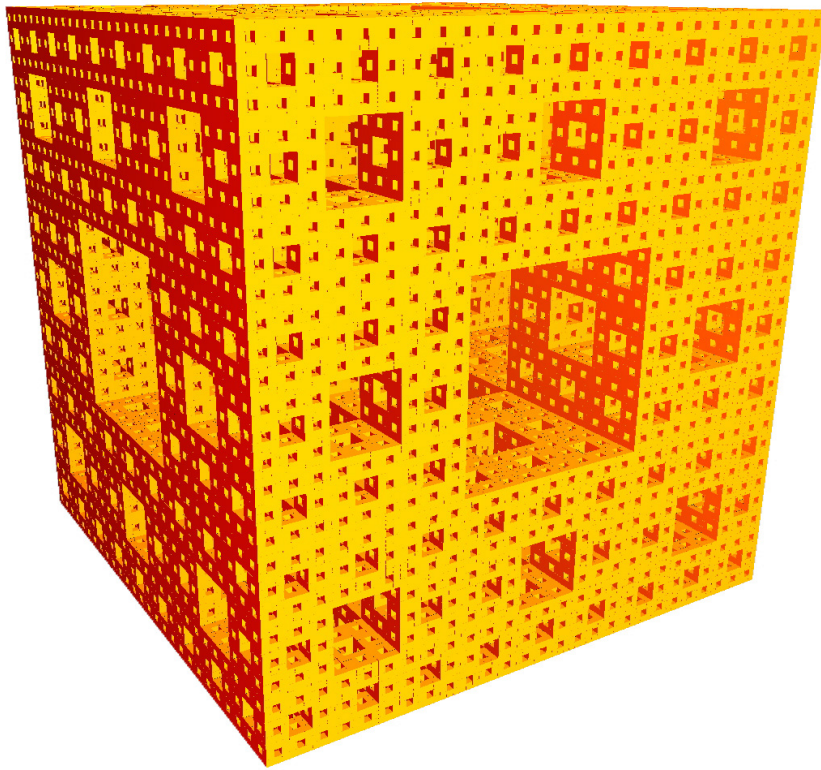
2. Iteration



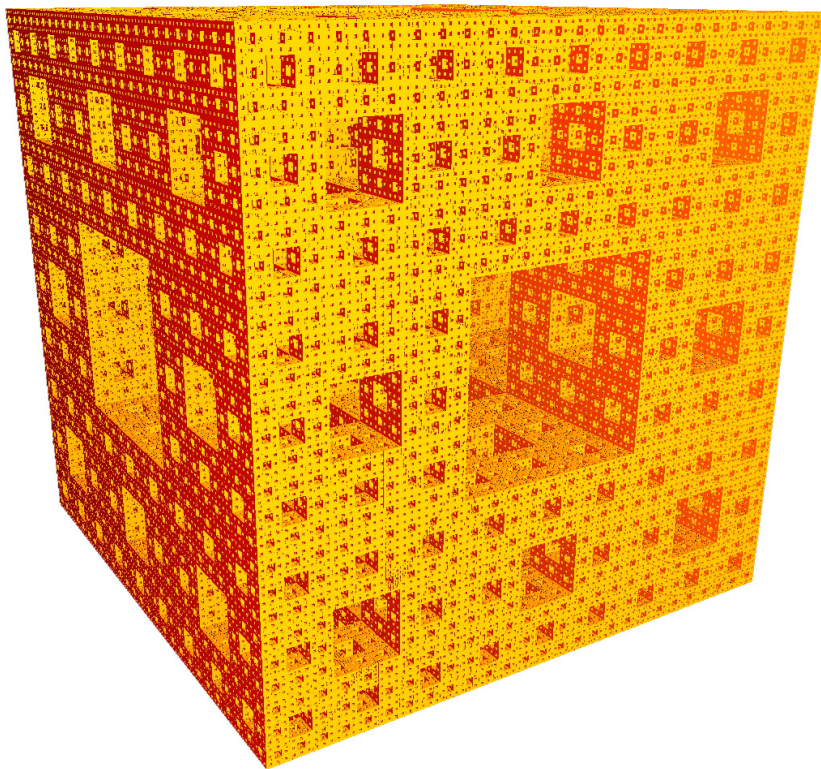
3. Iteration



4. Iteration

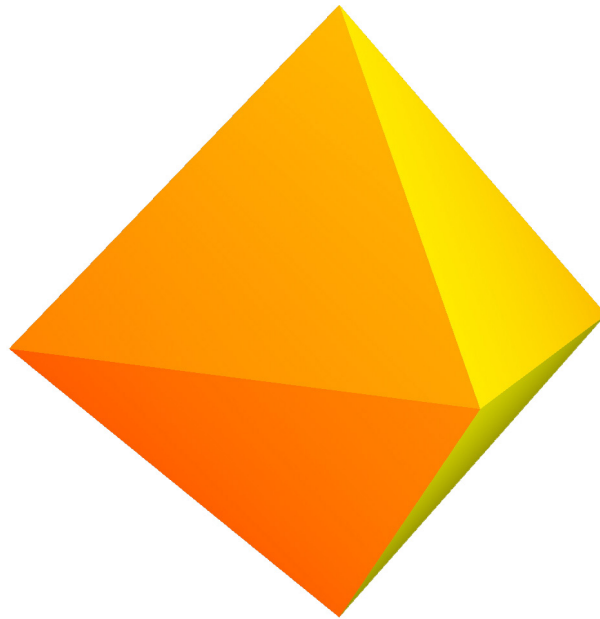


5. Iteration



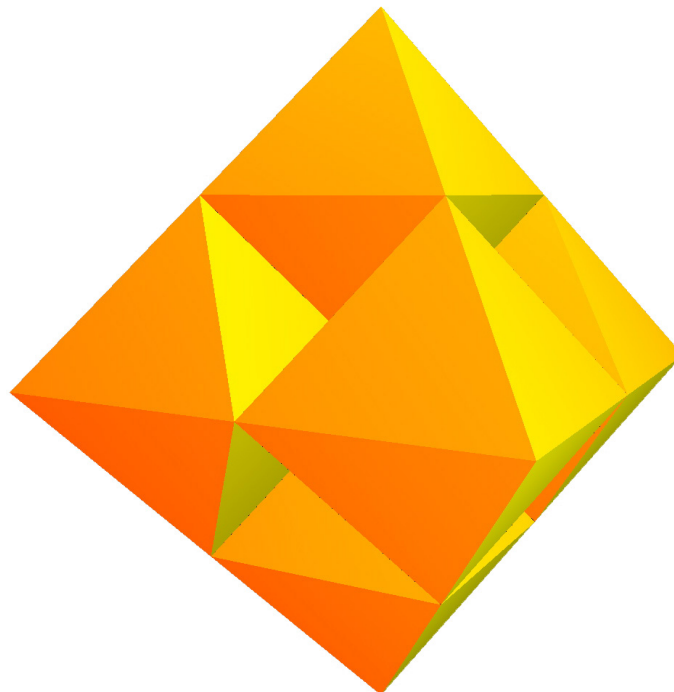
## 6.3 Oktaeder-Schwamm

Der Initiator des Oktaeder-Schwammes ist (wie bereits im Kapitel 3.5.3 erwähnt) ein Oktaeder:

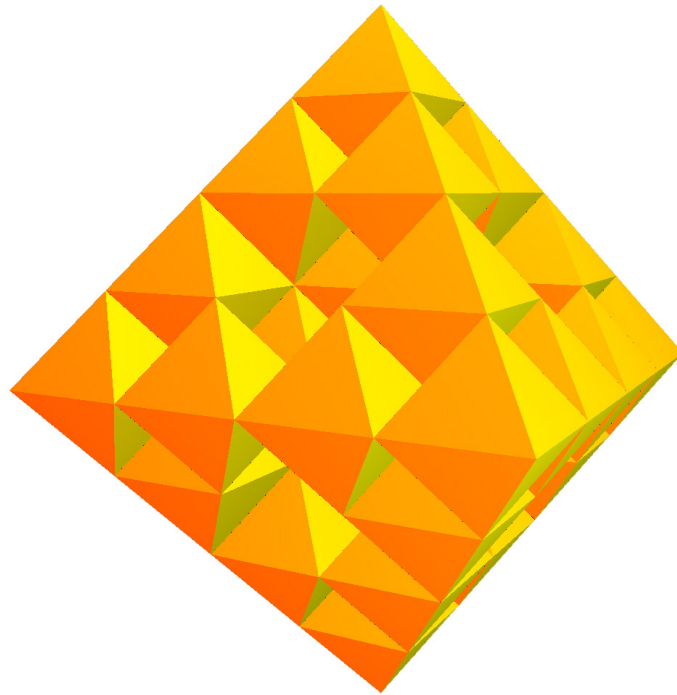


Die weiteren Iterationen sehen wie folgt aus:

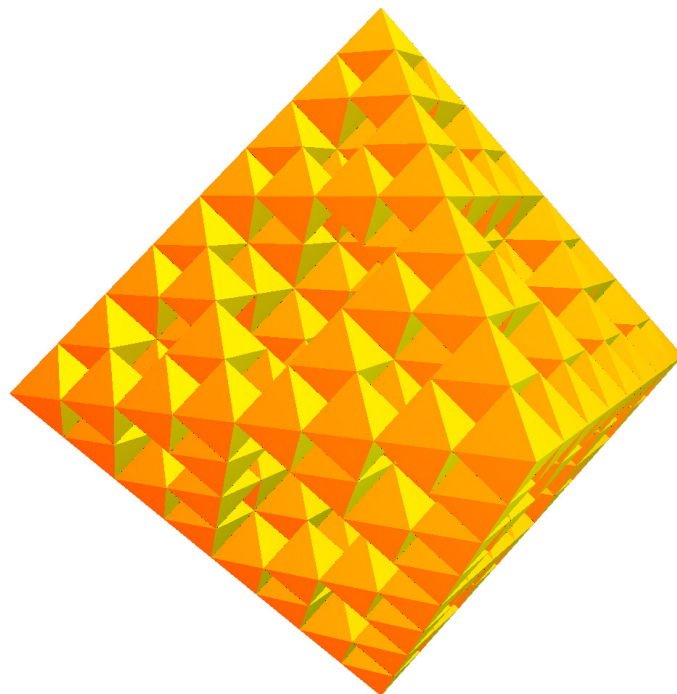
### 1. Iteration



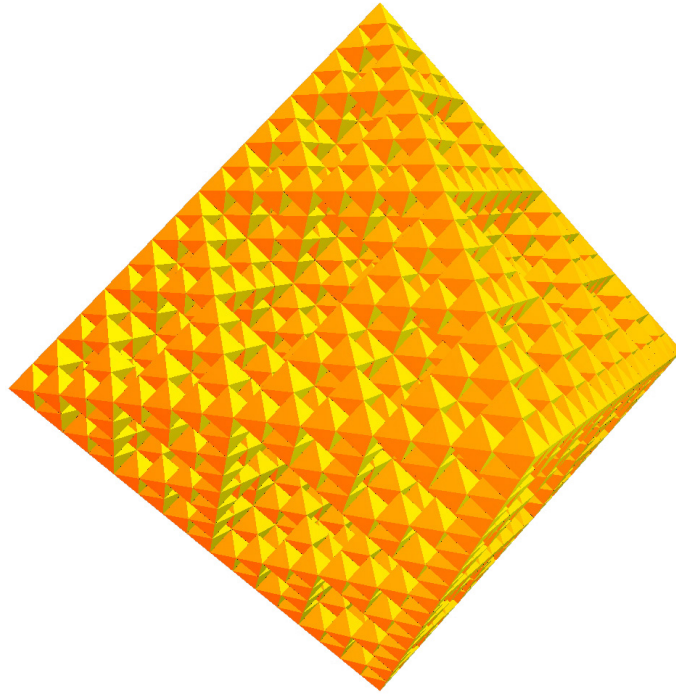
**2. Iteration**



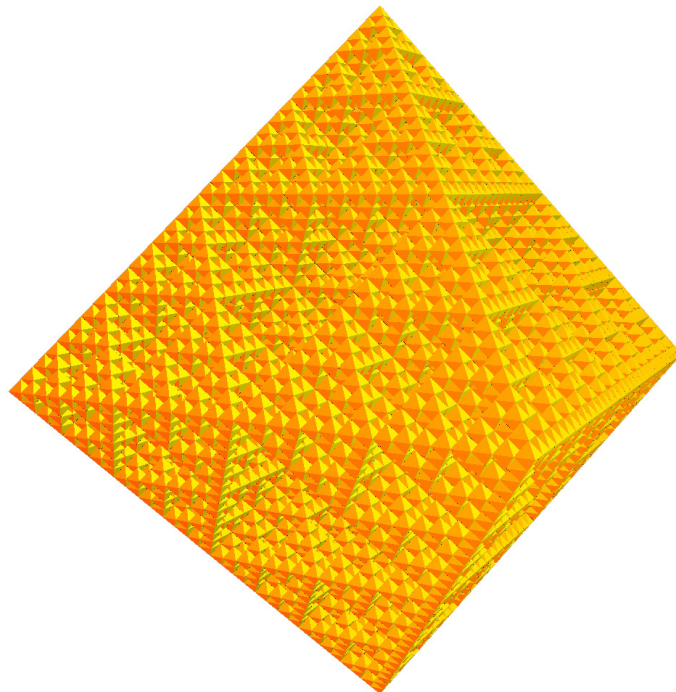
**3. Iteration**



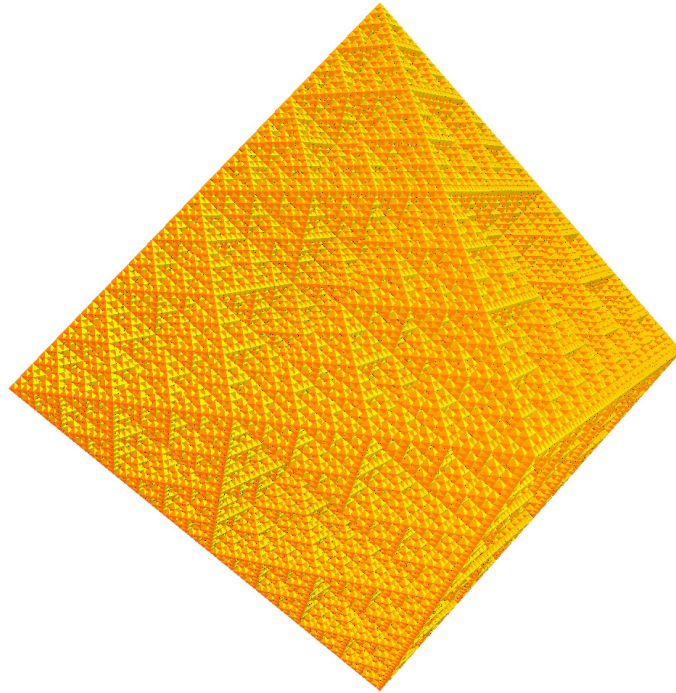
**4. Iteration**



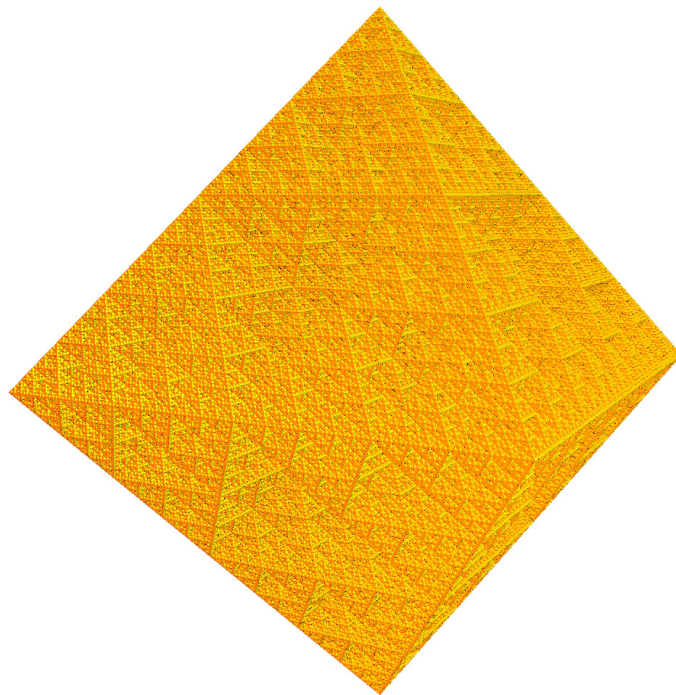
**5. Iteration**



**6. Iteration**

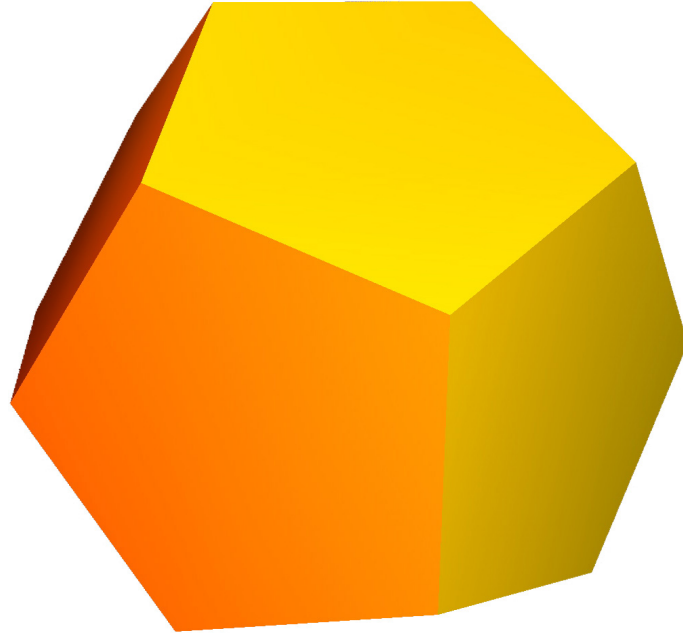


**7. Iteration**

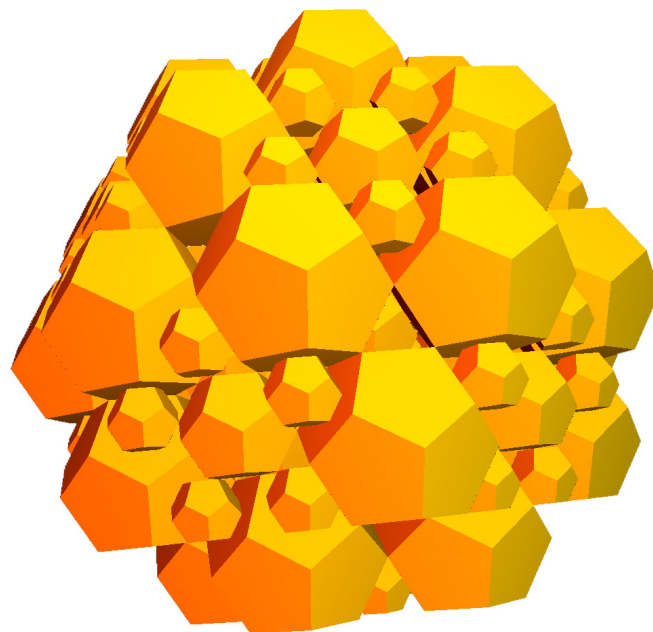


## 6.4 Dodekaeder-Schwamm

Da der Dodekaeder-Schwamm eine gänzlich andere Zerlegungsvorschrift als alle anderen, bisher behandelten Fraktale hat, sollen zuerst ein paar Bilder zum Verständnis der Zerlegung dienen. Der Initiator ist ein Dodekaeder:

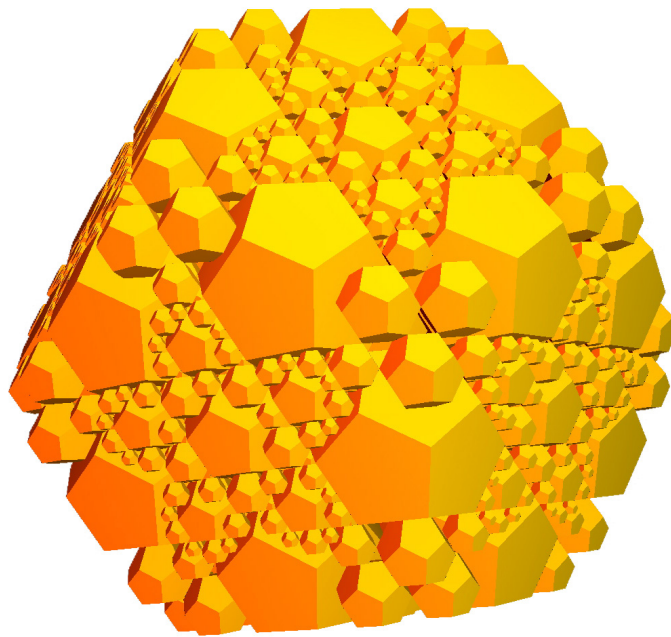


Nach der zweiten Iteration (die erste wird später erläutert) sieht der Körper wie folgt aus:

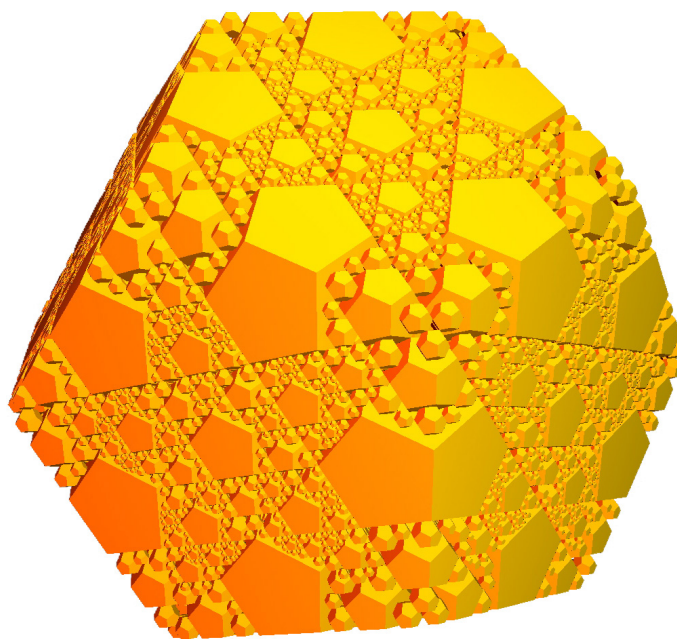




**3. Iteration:**



**4. Iteration:**



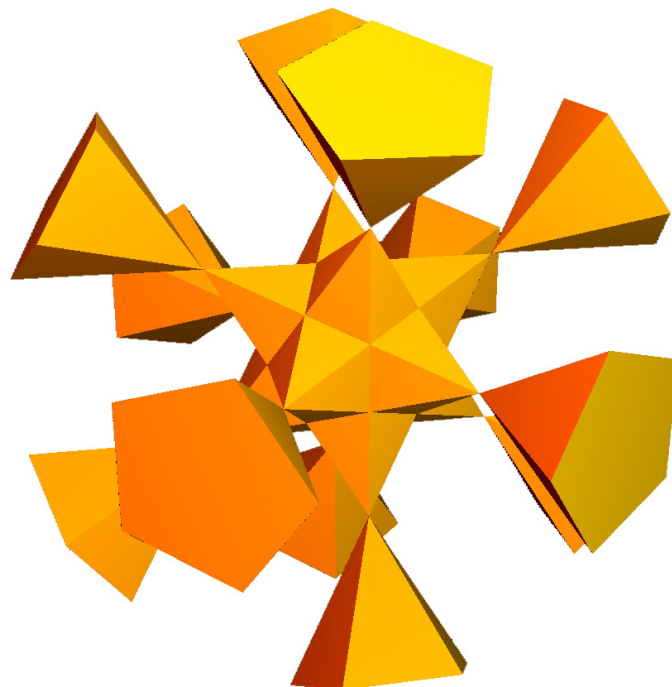
Hierbei fällt auf, dass bei jeder Iteration Objekte zum Dodekaeder-Schwamm hinzukommen und er somit immer dichter wird. Dazu muss man wissen, dass bei allen Darstellungen ausschließlich die Dodekaeder dargestellt werden, denn der eigentliche Körper besteht aus mehreren Objekttypen. Deshalb besitzt der Dodekaeder-Schwamm mehrere Zerlegungsvorschriften. So wird er bei der ersten Iteration mit einer Art „Einleitungsiteration“ in über 150 Objekte zerlegt. Das sind ein Dodekaeder, H-Shapes, W-Shapes, S-Shapes und C-Shapes. Nach dieser

## 6 Bilder der Fraktale

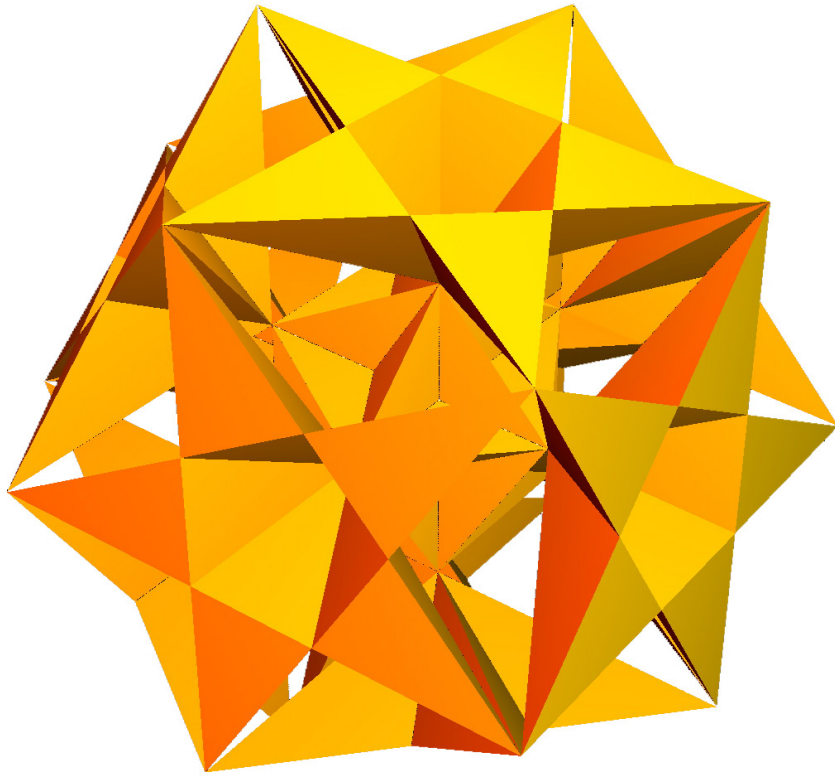
Zerlegung hat jedes Objekt eine eigene Zerlegungsvorschrift und wird wiederum in Dodekaeder, H-Shapes, W-Shapes, S-Shapes und C-Shapes zerlegt. Zu beachten ist, dass Dodekaeder nicht weiter iteriert werden, sondern einfach in ihrem Zustand der vorigen Iteration belassen werden. Nach der ersten Iteration sieht der Körper so aus:



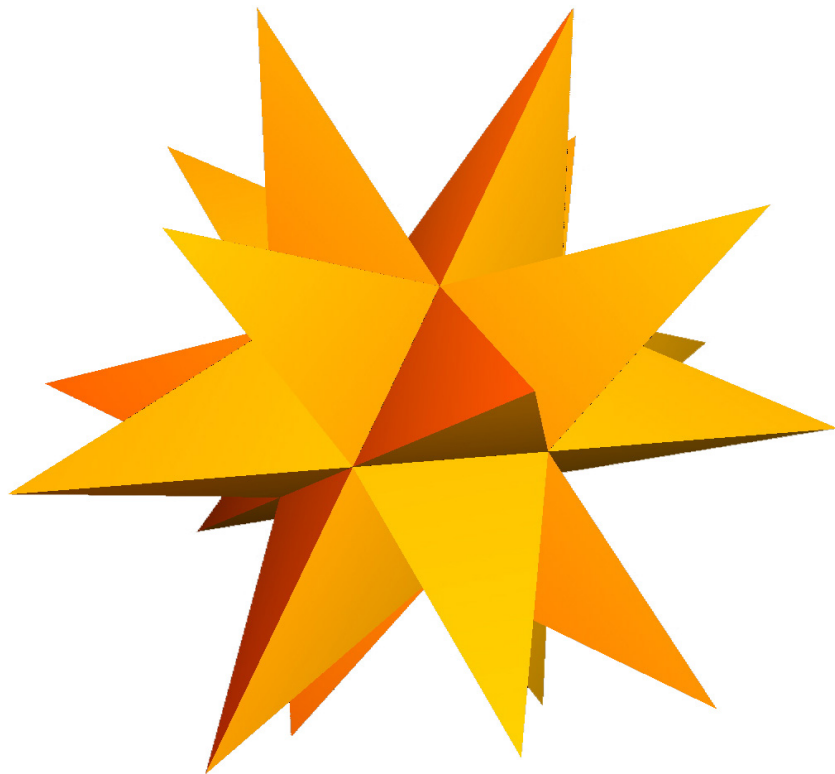
In dieser Darstellung sind nicht zu sehen **die H-Shapes**:



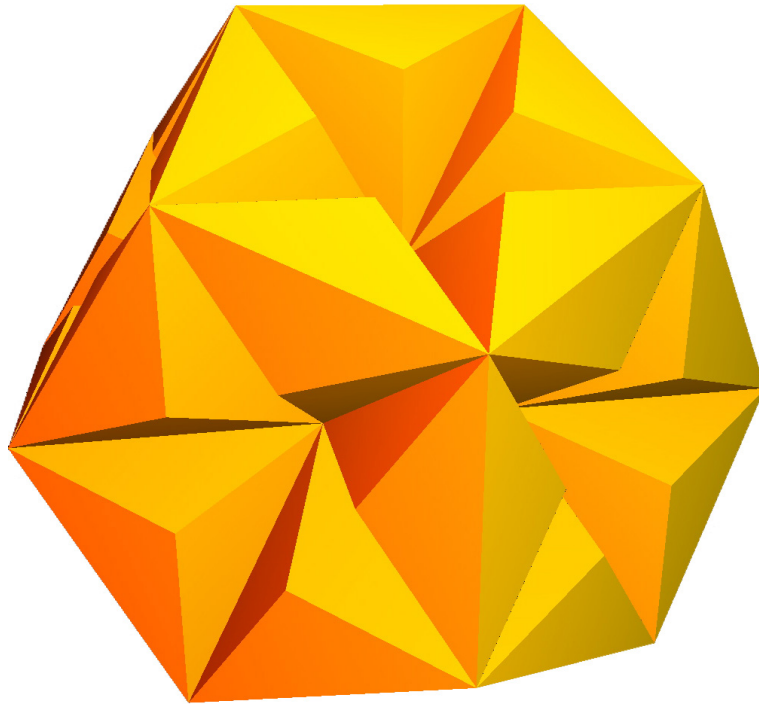
Die W-Shapes:



Die S-Shapes:

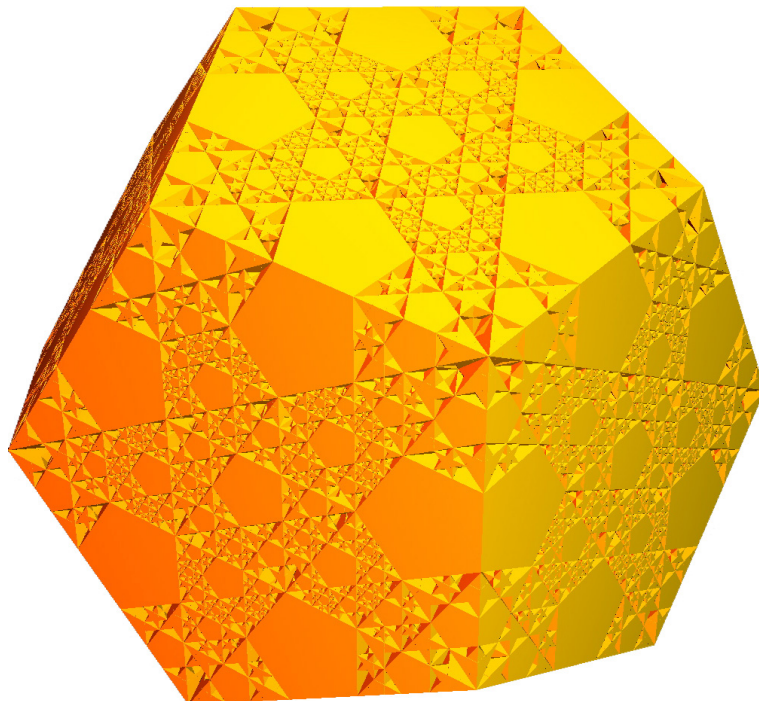


Und die C-Shapes:

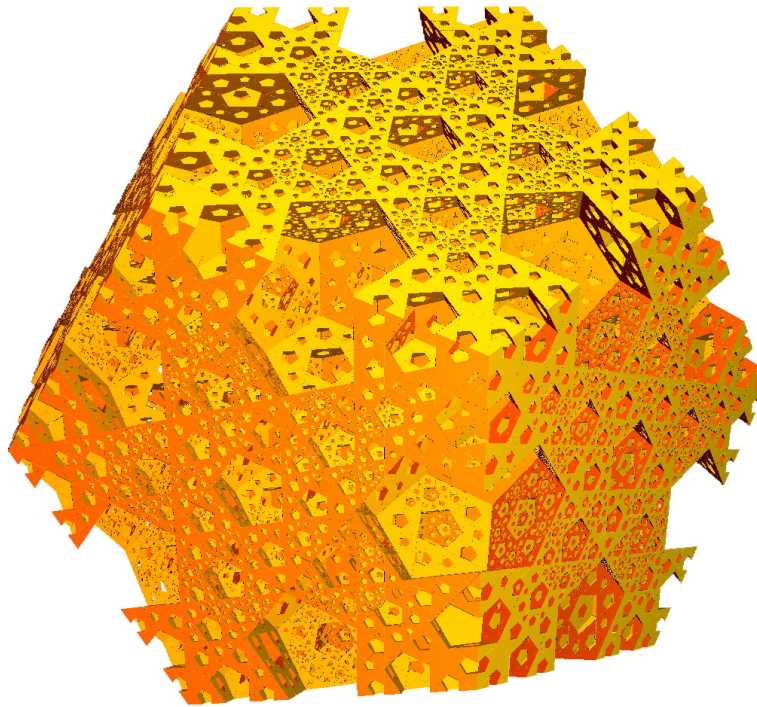


Das Programm bietet des weiteren die Möglichkeit sich verschiedene Objekttypen aus- bzw. einblenden zu lassen. Einige der interessantesten Objekte sind auf den nächsten Seiten zu sehen.

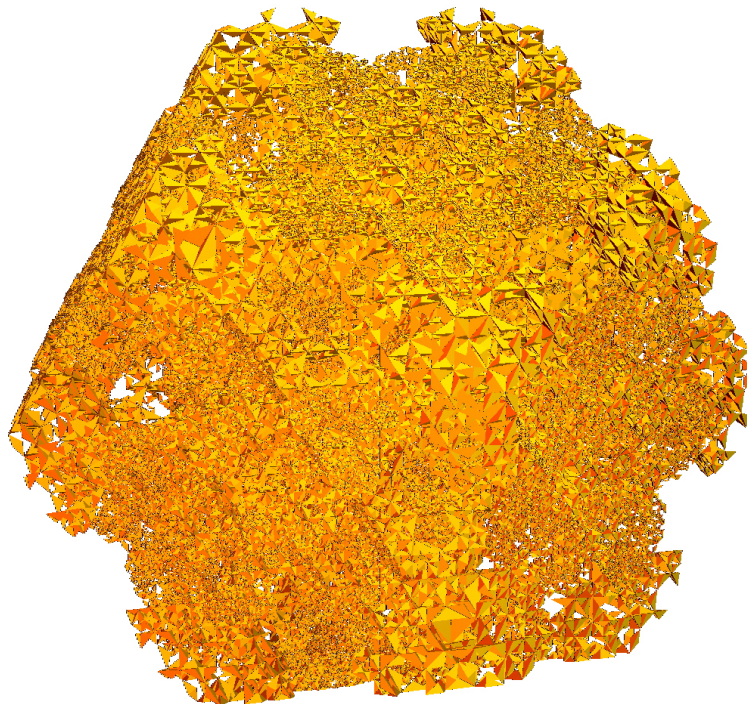
#### 4. Iteration, Anzeige Dodekaeder und H-Shapes



4. Iteration, Anzeige H-Shapes, W-Shapes und S-Shapes



4. Iteration, Anzeige W-Shapes



# 7 Matrizen und Vektoren

## 7.1 Der Begriff des Vektors

### 7.1.1 Der Vektorraum

Der Vektorraum  $V$  sei eine nichtleere Menge, für deren Elemente die Operationen „Addition“ und „S-Multiplikation mit reellen Zahlen“ definiert sind und diese jeweils nicht aus  $V$  herausführen.  $V$  heißt zusammen mit diesen Operationen linearer Vektorraum genau dann, wenn für beliebige Elemente  $\vec{a}, \vec{b}, \vec{c} \in V$  und  $r, s \in \mathbf{R}$  gilt:

- |  |  |
|--|--|
| 1) $\vec{a} + \vec{b} = \vec{b} + \vec{a}$                         | 5) $1 \cdot \vec{a} = \vec{a}$   |
| 2) $(\vec{a} + \vec{b}) + \vec{c} = \vec{a} + (\vec{b} + \vec{c})$ | 6) $(r \cdot s) \cdot \vec{a} = r \cdot (s \cdot \vec{a})$               |
| 3) $\exists \vec{o} : \vec{o} + \vec{a} = \vec{a}$                 | 7) $(r + s) \cdot \vec{a} = (r \cdot \vec{a}) + (s \cdot \vec{a})$       |
| 4) $\exists (-\vec{a}) : \vec{a} + (-\vec{a}) = \vec{o}$           | 8) $r \cdot (\vec{a} + \vec{b}) = (r \cdot \vec{a}) + (r \cdot \vec{b})$ |

**Anmerkung:** Im Folgenden werden hauptsächlich Vektoren des dreidimensionalen Vektorraums  $V^3$  betrachtet.

### 7.1.2 Der Vektor

Ein Vektor ist ein Element eines Vektorraums. Ein Vektor besitzt keinen Ausgangspunkt, sondern nur eine Richtung und eine Länge. Die Länge des Vektors wird auch Betrag des Vektors  $|\vec{a}|$  genannt. Einheitsvektoren sind Vektoren der Länge 1. Der Nullvektor  $\vec{o}$  ist ein Vektor bei dem Anfangspunkt und Endpunkt zusammen fallen; sein Betrag ist gleich 0, seine Richtung ist unbestimmt. Zwei Vektoren werden als gleich angesehen, wenn sie die gleiche Länge und gleiche Richtung haben. Unter einem Vektor  $\vec{a}$  mit  $n$  Koordinaten ( $n \in \mathbf{N}$ ) versteht man eine  $n \times 1$ -Matrix (siehe unten), auch genannt Zahlen- $n$ -Tupel. Man schreibt:

$$\vec{a} = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix} \quad (\text{mit } a_1, a_2, \dots, a_n \in \mathbf{R})$$

Der Betrag eines Vektors errechnet sich wie folgt:

$$|\vec{a}| = \sqrt{a_1^2 + a_2^2 + \dots + a_n^2}$$

### 7.1.3 Addition von Vektoren

Zwei Vektoren des Vektorraums  $V^3$  werden addiert, indem man ihre Komponenten addiert:

$$\vec{a} + \vec{b} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} + \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = \begin{pmatrix} a_x + b_x \\ a_y + b_y \\ a_z + b_z \end{pmatrix}$$

Diese Operation ist sowohl kommutativ als auch assoziativ. Es gilt:

$$\vec{a} + \vec{b} = \vec{b} + \vec{a}$$

und

$$\vec{a} + (\vec{b} + \vec{c}) = (\vec{a} + \vec{b}) + \vec{c}$$

### 7.1.4 Multiplikation von Vektoren (Kreuzprodukt)

Unter dem Vektorprodukt zweier Vektoren  $\vec{a}, \vec{b} \in V^3$  versteht man den Vektor:

$$\vec{a} \times \vec{b} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \times \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = \begin{pmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{pmatrix}$$

### 7.1.5 Normalisierung eines Vektors

Unter der Normalisierung eines Vektors versteht man, dass man zu einem Vektor  $\vec{a}$  einen gleichgerichteten Vektor mit dem Betrag 1 findet. Ein Vektor  $\vec{a}$  ( $\vec{a} \in V^3$ ) wird normalisiert, indem man jede Koordinaten durch die Wurzel der Summe der Quadrate der einzelnen Koordinaten dividiert. Der resultierende Vektor  $\vec{a}'$  hat folglich die Länge 1. Man nennt normalisierte Vektoren auch „Einheitsvektoren“.

$$\vec{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$$

$$n = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

$$\vec{a}' = \begin{pmatrix} \frac{a_x}{n} \\ \frac{a_y}{n} \\ \frac{a_z}{n} \end{pmatrix}$$

## 7.2 Der Begriff der Matrix

Sind  $m \cdot n$  Ausdrücke in einem rechteckigen Schema von  $m$  Zeilen und  $n$  Spalten angeordnet, so spricht man von einer  $(m,n)$ -Matrix. Die  $m \cdot n$  Ausdrücke  $a_{ik}$  heißen Elemente der Matrix. Die Stellung eines Elements innerhalb des Schemas wird durch einen Doppelindex gekennzeichnet, wobei der erste Index die Zeilennummer und der zweite die Spaltennummer angibt, in der das Element steht. Dabei verläuft die Nummerierung der Zeilen von oben nach unten, die der Spalten von links nach rechts.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

Eine Matrix vom Typ  $(n,n)$  heißt  $n$ -reihige quadratische Matrix oder quadratische Matrix der Ordnung  $n$ .

### 7.2.1 Einheitsmatrix

Eine Einheitsmatrix ist eine quadratische Matrix  $(a_{ik})$  der Ordnung  $n$ , für die gilt:

$$a_{ik} = \begin{cases} 0 & \text{für } i \neq k \\ 1 & \text{für } i = k \end{cases}$$

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix}$$

### 7.2.2 Addition von Matrizen

Zwei  $(m,n)$ -Matrizen  $A = (a_{ik})$  und  $B = (b_{ik})$  werden elementarweise addiert. Für  $C = A + B$  mit  $C = (c_{ik})$  gilt:

$$c_{ik} = a_{ik} + b_{ik} \quad (\text{für } i=1, 2, \dots, m; k=1, 2, \dots, n)$$

Ein Beispiel:

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{pmatrix}$$

Diese Operation ist sowohl kommutativ als auch assoziativ. Es gilt:

$$C = A + B = B + A \quad \text{und} \quad A + (B + C) = (A + B) + C$$

### 7.2.3 Multiplikation von Matrizen

Für eine  $(m,n)$ -Matrix  $A = (a_{ik})$ , eine  $(n,p)$ -Matrix  $B = (b_{ik})$  und eine  $(m,p)$ -Matrix  $C = (c_{ik})$  gilt  $C = A \cdot B$ , genau dann, wenn für alle  $c_{ik}$  gilt:

$$c_{ik} = a_{i1} \cdot b_{1k} + a_{i2} \cdot b_{2k} + \dots + a_{in} \cdot b_{nk}$$

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1p} \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & \dots & b_{np} \end{pmatrix}$$



## 7 Matrizen und Vektoren

$$= \begin{pmatrix} \sum_{j=1}^n a_{1j}b_{j1} & \sum_{j=1}^n a_{1j}b_{j2} & \cdots & \sum_{j=1}^n a_{1j}b_{jp} \\ \cdots & \cdots & \cdots & \cdots \\ \sum_{j=1}^n a_{mj}b_{j1} & \sum_{j=1}^n a_{mj}b_{j2} & \cdots & \sum_{j=1}^n a_{mj}b_{jp} \end{pmatrix}$$

Ein Beispiel:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \\ a_{31}b_{11} + a_{32}b_{21} & a_{31}b_{12} + a_{32}b_{22} \end{pmatrix}$$

# 8 Geometrische Transformation

## 8.1 Einführung

Bei der Darstellung von Flächen und Körpern im zwei- und dreidimensionalen Raum gibt es verschiedene Transformationen: Translation, Skalierung und Rotation. Diese Transformationen werden für verschiedene Anwendungen genutzt.

## 8.2 2D-Transformationen

### 8.2.1 Translation

Ein Punkt  $P(x, y)$  wird auf einen Punkt  $P'(x', y')$  abgebildet, indem man zu  $x$  einen Wert  $d_x$  und zu  $y$  einen Wert  $d_y$  addiert.

$$x' = x + d_x \quad (8.1)$$

$$y' = y + d_y \quad (8.2)$$

Alternativ kann man dies auch mit Spaltenvektoren darstellen:

$$P' = T + P \quad (8.3)$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} d_x \\ d_y \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix} \quad (8.4)$$

### 8.2.2 Skalierung

Ein Punkt  $P(x, y)$  wird skaliert, indem man alle Koordinaten mit einem konstanten Wert multipliziert:

$$x' = x \cdot s_x \quad s_x \neq 0 \quad (8.5)$$

$$y' = y \cdot s_y \quad s_y \neq 0 \quad (8.6)$$

Oder in der alternativen Darstellung mit Spaltenvektoren:

$$P' = S \cdot P \quad (8.7)$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \quad (8.8)$$

### 8.2.3 Rotation

Ein Punkt  $P(x, y)$  wird mit einem Winkel  $\alpha$  um den Ursprung  $(0,0)$  folgendermaßen gedreht (Herleitung siehe Anhang 3):

$$x' = x \cdot \cos \alpha - y \cdot \sin \alpha \quad (8.9)$$

$$y' = x \cdot \sin \alpha + y \cdot \cos \alpha \quad (8.10)$$

Oder mit Spaltenvektoren:

$$P' = R \cdot P \quad (8.11)$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \quad (8.12)$$

## 8.3 Homogene Koordinaten

Um die Translation nicht unterschiedlich (als Addition) zu den anderen Transformationen behandeln zu müssen, führt man Homogene Koordinaten ein. Das heißt, jedes Koordinatenpaar  $(x, y)$  wird künftig durch ein Tripel  $(x, y, W)$  repräsentiert:

$$\begin{pmatrix} x \\ y \end{pmatrix} \cong \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (8.13)$$

Zugleich wird festgelegt, dass zwei Tripel Mengen homogener Koordinaten,  $(x, y, W)$  und  $(x', y', W')$ , den selben Punkt repräsentieren, wenn das Tripel ein Vielfaches des anderen Tripels ist. Es gilt, dass mindestens eine der Koordinaten nicht Null ist. Falls  $W \neq 0$  gilt, kann durch  $W$  dividiert werden:

$$\begin{pmatrix} x \\ y \\ W \end{pmatrix} = \begin{pmatrix} \frac{x}{W} \\ \frac{y}{W} \\ 1 \end{pmatrix} \quad (8.14)$$

Punkte mit  $W = 0$  werden als Punkte im Unendlichen bezeichnet.

### 8.3.1 Translation

Mit Hilfe dieses neuen Vektors mit drei Komponenten kann man nun auch die Translation als Multiplikation beschreiben:

$$P' = T(d_x, d_y) \cdot P \quad (8.15)$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (8.16)$$

### 8.3.2 Skalierung

Die Skalierung mit Homogenen Koordinaten sieht folgendermaßen aus:

$$P' = S(s_x, s_y) \cdot P \quad (8.17)$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (8.18)$$

### 8.3.3 Rotation

Die Rotation mit Homogenen Koordinaten sieht folgendermaßen aus:

$$P' = R(\alpha) \cdot P \quad (8.19)$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (8.20)$$

### 8.3.4 Transformationsmatrix

Wenn man die 2D-Transformationsmatrizen für die Translation, Skalierung und Rotation mit dem Homogenen Punkt  $(x, y, 1)$  multipliziert erhält man eine komplette Transformationsmatrix:

$$P' = T(d_x, d_y) \cdot S(s_x, s_y) \cdot R(\alpha) \cdot P \quad (8.21)$$

$$P' = \begin{pmatrix} d_x + s_x \cdot y \cdot \cos \alpha - s_x \cdot x \cdot \sin \alpha \\ d_y + s_y \cdot x \cdot \cos \alpha + s_y \cdot y \cdot \sin \alpha \\ 1 \end{pmatrix} \quad (8.22)$$

## 8.4 3D-Transformationen

Analog zur zweidimensionalen Transformation gibt es auch im dreidimensionalen Raum Transformationen, wie Translation, Skalierung und Rotation. (Für den Beweis, dass die Homogenen Koordinaten ein „echter“ Vektorraum im mathematischen Sinne sind, siehe Anhang 2)

### 8.4.1 Translation

Die Translation im dreidimensionalen Raum ähnelt der im zweidimensionalen:

$$P' = T(d_x, d_y, d_z) \cdot P \quad (8.23)$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (8.24)$$

### 8.4.2 Skalierung

Auch die Skalierung im dreidimensionalen Raum ähnelt der im zweidimensionalen:

$$P' = S(s_x, s_y, s_z) \cdot P \quad (8.25)$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (8.26)$$

### 8.4.3 Rotation

Die Rotation im dreidimensionalen Raum ist wesentlich komplexer als im zweidimensionalen. Zu beachten ist, dass eine Rotation um die x-Achse, die y-Achse und die z-Achse möglich ist:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha & 0 \\ 0 & \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8.27)$$

$$R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \beta & 0 & \cos \beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8.28)$$

$$R_z(\gamma) = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 & 0 \\ \sin \gamma & \cos \gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8.29)$$

Die vollständige Rotationsmatrix im dreidimensionalen Raum lautet dann wie folgt:

$$R(\alpha, \beta, \gamma) = R_x(\alpha) \cdot R_y(\beta) \cdot R_z(\gamma) = \quad (8.30)$$

$$\begin{bmatrix} \cos \beta \cos \gamma & -\cos \beta \sin \gamma & \sin \beta & 0 \\ \cos \gamma \sin \alpha \sin \beta + \cos \alpha \sin \gamma & \cos \alpha \cos \gamma - \sin \alpha \sin \beta \sin \gamma & -\cos \beta \sin \alpha & 0 \\ -\cos \alpha \cos \gamma \sin \beta + \sin \alpha \sin \gamma & \cos \gamma \sin \alpha - \cos \alpha \sin \beta \sin \gamma & \cos \alpha \cos \beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8.31)$$

### 8.4.4 Transformationsmatrix

Wenn man jetzt die 3D-Transformationsmatrizen für die Translation, Skalierung und Rotation mit dem Homogenen Punkt  $(x,y,z,1)$  multipliziert erhält man eine komplette Transformationsmatrix (allerdings nur für eine feste Reihenfolge):

$$P' = T(d_x, d_y, d_z) \cdot S(s_x, s_y, s_z) \cdot R(\alpha, \beta, \gamma) \cdot P = \quad (8.32)$$

$$\begin{pmatrix} x \cdot s_x \cdot \cos \beta \cos \gamma + d_x + z \cdot s_x \cdot \sin \beta + y \cdot s_x \cdot \cos \beta \sin \gamma \\ d_y - z \cdot s_y \cdot \cos \beta \sin \alpha + x \cdot (s_y \cdot \cos \gamma \sin \alpha \sin \beta + s_y \cdot \cos \alpha \sin \gamma) + y \cdot (s_y \cdot \cos \alpha \cos \gamma - s_y \cdot \sin \alpha \sin \beta \sin \gamma) \\ z \cdot s_z \cdot \cos \alpha \cos \beta + d_z + x \cdot (-s_z \cdot \cos \alpha \cos \gamma \sin \beta + s_z \cdot \sin \alpha \sin \gamma) + y \cdot (s_z \cdot \cos \gamma \sin \alpha + s_z \cdot \cos \alpha \sin \beta \sin \gamma) \\ 1 \end{pmatrix} \quad (8.33)$$

# 9 OpenGL

## 9.1 Einführung

OpenGL ist die Abkürzung für „Open Graphics Language“ und ist eine plattform- und programmiersprachenunabhängige API (Application Programming Interface) zur Entwicklung von 3D-Computergrafiken. Der OpenGL-Standard beschreibt etwa 250 Befehle, die die Berechnung komplexer 3D-Szenen in Echtzeit erlauben.

Da OpenGL nur ein Standard ist, und keine Implementierung, werden die Befehle vom Betriebssystem verarbeitet - wie das geschieht ist Sache des Grafikkarten-Treibers, der die Befehle an die Grafikkarte weitergibt (Hardware- oder Direct Rendering) oder auf der CPU ausführt, wenn die Grafikkarte den entsprechenden Befehl nicht bearbeiten kann (Software Rendering). Aufgrund seiner Plattformunabhängigkeit ist OpenGL im professionellen Bereich als 3D-Standard nach wie vor führend.

## 9.2 Programmierung

Um den Quellcode besser zu verstehen, führen wir hier die wichtigsten Befehle von OpenGL auf, die wir in unserem Programm verwendet haben.

### 9.2.1 Erzeugung der Körper

Jeder dreidimensionale Körper in OpenGL besteht aus endlich vielen Flächen. Um diese zu erzeugen muss man erst einen Flächentyp festlegen, der angibt, um was für ein n-eck es sich handelt. Danach werden die n Punkte angegeben, welche OpenGL dann automatisch verbindet.

```
glBegin(GL_TRIANGLES);  
Initialisierung eines Dreiecks
```

```
glColor3f(r, g, b);  
Auswahl der aktuellen Zeichenfarbe (mit Rot-, Grün-, Blauanteilen)
```

```
glVertex3f(x1, y1, z1);  
glVertex3f(x2, y2, z3);  
glVertex3f(x3, y2, z3);  
Festlegung des drei Punkte
```

`glEnd();`  
Abschließen der Fläche

## 9.2.2 Drawlist

Eine wichtige Funktion unter OpenGL ist die Drawlist. Diese Funktion ermöglicht es, mehrere Zeichenschritte im RAM zu speichern und sie bei Aufruf nacheinander auszuführen. Da die Körper nicht jedes mal neu berechnet werden müssen, spart dies erheblich an Rechenzeit. Des weiteren erlaubt eine Drawlist, alle in ihr enthaltenen Zeichenoperationen mit ein und derselben Transformationsmatrix zu multiplizieren.

`glNewList(ListNr, GL_COMPILE);`  
Eine Drawlist wird vereinbart

`glPushMatrix();`  
Sichern der aktuell definierten Transformationen

`glMultTransposeMatrixf(Einheitsmatrix);`  
Die Drawlist wird komplett mit einer Transformationsmatrix multipliziert

`glCallList(ListNr);`  
Eine Drawlist wird komplett gezeichnet

`glPopMatrix();`  
Wiederherstellen der gesicherten Transformationen

`glEndList();`  
Schließt eine Liste, wenn alles gezeichnet worden ist

## 9.2.3 Zeichnen der Szene

`glClear();`  
Löscht den Zeichenbereich

`glTranslatef( $d_x$ ,  $d_y$ ,  $d_z$ );`  
Verschiebt die Szene um  $d_x$ ,  $d_y$  und  $d_z$

`glRotatef( $\omega$ ,  $x$ ,  $y$ ,  $z$ );`  
Dreht die Szene um  $\omega \cdot x$  um die x-Achse, um  $\omega \cdot y$  um die y-Achse und um  $\omega \cdot z$  um die z-Achse

# 10 Quellen

## 10.1 Bücher

- [1] Kleine Enzyklopedie Mathematik, VEB Bibliographisches Institut Leipzig, 1986
- [2] Heinz-Otto Peitgen, Hartmund Jürgens, Dietmar Saupe: Bausteine des Chaos - Fraktale, Klett-Cotta/Springer-Verlag 1992
- [3] Benoit B. Mandelbrot: Die fraktale Geometrie der Natur, 1987

## 10.2 Internet

*(Anmerkung: Alle Zeitangaben sind in Mitteleuropäischer Zeit)*

- [1] <http://www.mathe.tu-freiberg.de/hebisch/cafe/platonische.html>, 11.05.2004, 12:44
- [2] <http://de.wikipedia.org/wiki/Polyeder>, 11.05.2004, 14:24
- [3] <http://de.wikipedia.org/wiki/Fraktal>, 01.06.2004, 13:43
- [4] <http://www.saar.de/awa/rekursio.htm>, 02.06.2004, 19:20
- [5] <http://www.cs.fhm.edu/schieder/programmieren-2-ss97/recursion.html>, 02.06.2004, 19:20
- [6] <http://de.wikipedia.org/wiki/OpenGL>, 05.06.2004, 13:17
- [7] <http://home.t-online.de/home/Siegfried.Beyer/>, 15.06.2004, 12:29



# 11 Selbstständigkeitserklärung

Wir erklären hiermit, dass wir die WPA-Arbeit ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benutzt haben.

Ort, Datum, Unterschrift des Schülers Max Seelemann

Ort, Datum, Unterschrift des Schülers Sascha Grehl

# 12 Anhang