Bau eines dreidimensionalen Fraktals

1 Einleitung

Im Herbst 2003 war ich mit meinem Mathematik-Leistungskurs zu den Schülertagen an der Universität Duisburg eingeladen, wo wir unter anderem vielfältige Einblicke in die ingenieurwissenschaftliche Fakultät bekamen. Man zeigte uns dort eine so genannte Rapid-Prototyping-Maschine. Diese Maschine verfügt über einen Computer, über den man die Daten eines dreidimensionalen Objektes eingeben kann. Der Computer zerlegt dieses dann in viele hauchdünne Schichten. Die Daten der ersten Schicht werden anschließend an die Maschine geschickt. Innerhalb der Maschine befindet sich eine Schicht aus Kunststoffstaub. Die Stellen, die das Objekt in dieser Schicht ausfüllt, werden dann mit einem Laser bestrahlt. Das führt dazu, dass der Kunststoffstaub an diesen Stellen verklebt und fest wird. Ist die erste Schicht vollständig in den Staub gelasert, legt die Maschine eine neue Schicht Staub über die alte und lasert die nächste Schicht darüber, die dadurch auch mit der darunter liegenden verklebt wird. Wenn alle Schichten abgearbeitet sind, kann man das dreidimensionale Objekt, dessen Daten man zu Beginn in den Computer eingegeben hat, aus der Maschine nehmen.
Normalerweise wird diese Technologie in den Ingenieurwissenschaften benutzt, um Modelle mechanischer Bauteile anzufertigen, bevor sie in die Serienfertigung gehen. Schon seit einiger Zeit suchte ich ein konkretes Thema für eine neue „Jugend forscht" - Arbeit. Da kam mir der Gedanke, die beschriebene Technologie auch für die Mathematik nutzbar zu machen. Ich dachte in diesem Zusammenhang über eine Arbeit im Bereich der dreidimensionalen fraktalen Geometrie nach.
Nach meinem Wissen ist es noch nie versucht worden, ein Fraktal nicht nur mathematisch zu entwerfen und am Computer zu modellieren, sondern auch konkret zu bauen. Das Problem meiner Arbeit besteht also darin, ein interessantes dreidimensionales Fraktal mathematisch zu entwerfen, dieses dann am Computer zu modellieren und schließlich zu zeigen, dass es mit Hilfe des Rapid-Prototyping-Verfahrens tatsächlich technisch möglich ist, solche Figuren konkret zu bauen. Aus der Vielzahl möglicher Konstruktionsverfahren für derartige Fraktale habe ich einen Typ ausgewählt, dessen gesamter Aufbau nur aus einer einzigen Kurve besteht, ähnlich wie etwa bei der Peano-Kurve, der Hilbert-Kurve oder dem Harter-Heighway-Drachen (sog. Lindenmayer-Systeme). [6, S.503]

2 Der mathematische Entwurf

2.1 Notwendiges Vorwissen

Bevor ich nun beschreibe, wie ich zu meinem Fraktal gekommen bin, möchte ich kurz einige andere mathematische Objekte vorstellen, da diese zum Verständnis meines Fraktals notwendig sind.

2.1.1 Hamiltonsche Rundreise

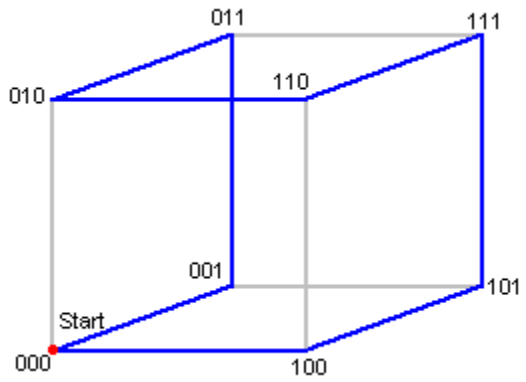
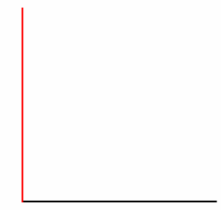


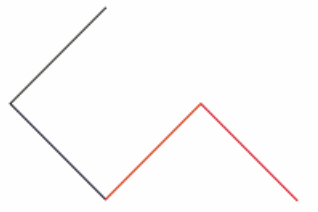
Abb. 1: Hamiltonsche Rundreise

Betrachten wir einmal den links dargestellten Würfel. Angenommen, wir befinden uns am rot markierten Startpunkt. Wie können wir, indem wir an den Kanten entlanglaufen, jeden Punkt des Würfels genau einmal besuchen und am Ende wieder am Startpunkt auskommen? Eine Lösung ist der blau eingezeichnete Pfad, der nach dem Mathematiker William Rowan Hamilton „Hamiltonsche Rundreise" benannt wurde. Diese Rundreise ist auf allen platonischen Körpern durchführbar. [3, S.3]

2.1.2 Konstruktionsprinzip

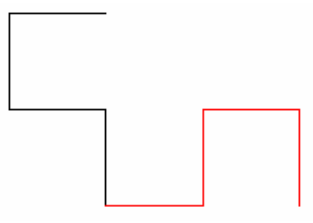


D_0



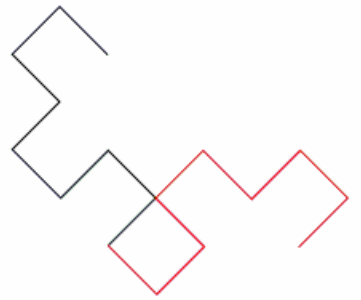
D_1

Fraktale der hier interessierenden Art werden immer nach einem typischen Prinzip konstruiert, welches nun am Beispiel der *Drachen-Kurve* erläutert werden soll. Wir beginnen mit einem Ausgangsobjekt, welches auch als *Initiator* bezeichnet wird. Da Fraktale in mehreren Generationen wachsen, ist dieser Anfangszustand quasi die nullte Generation. Wir bezeichnen die Drachen-Kurve zur n-ten Generation mit D_n. Der Initiator der Drachenkurve besteht aus zwei gleich langen Strecken, die in einem rechten Winkel zueinander stehen.
Um nun die nächste Generation zu erhalten, gehen wir hier folgendermaßen vor: Wir ersetzen den Initiator durch zwei Kopien seiner selbst und verkleinern diese um den Faktor $1/\sqrt{2}$. Die eine Kopie drehen wir um 45° nach links um den Scheitelpunkt des Winkels. Die andere drehen wir um 135° nach links und fügen sie an die andere an.
So erhalten wir die Drachenkurve der ersten Generation.
Wir können uns die Konstruktion von Fraktalen durch eine gedankliche Maschine veranschaulichen. In diese Maschine geben wir am Anfang den Initiator, der dann in den nächsten Teil der Maschine weitergeleitet wird. Diesen Teil, der aus der n-ten Generation die $(n+1)$-te erzeugt, bezeichnen wir als *Generator*.

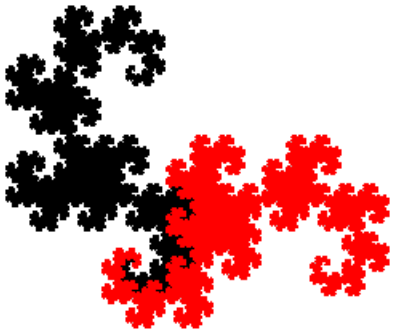


D_2

Der springende Punkt ist, dass dieses Verfahren nun beliebig oft wiederholt werden kann, indem wir das, was der Generator ausgibt, nun wieder in ihn eingeben können. Der Generator wird dann nicht mehr auf ein Ausgangsobjekt angewendet, sondern auf zwei, da wir ja im ersten Schritt zwei verkleinerte Kopien des Originals erzeugt haben. Diese Wiederholung des Konstruktionsverfahrens bezeichnet man auch als *Iteration* und eine solche Maschine als *Iterator*.



D_3



D_{17}

Abb. 2: Drachenkurve *[A1]*

Wenn man das Verfahren mehrfach anwendet, ergibt sich eine bizarre Struktur. Die Frage ist, was passiert, wenn wir es unendlich oft anwenden. Das Grenzgebilde, gegen das die Kurve konvergiert, bezeichnen wir als *Attraktor*. Der Attraktor dieses Fraktals ähnelt mit etwas Fantasie einem Drachen, den man nach seinen Entdeckern Harter-Heighway-Drachen nennt. Die Drachen-Kurve hat die bemerkenswerte Eigenschaft, dass sie eine Fläche mit fraktalem Rand lückenlos ausfüllt. [2]
Iteration ist das Grundprinzip bei der Konstruktion fraktaler Objekte. Es ist für das weitere Vorgehen hilfreich, sie als Regel etwas allgemeiner zu formulieren:

***Iterationsregel**:*

1. Suche im Objekt, welches in den Generator eingegeben wurde, alle Initiatoren.
2. Ersetze alle Initiatoren durch n Kopien, die mit dem Faktor $1/m$ verkleinert werden. ($n \in \mathbb{N}, m \in \Re^+$)
3. Gib das Ergebnis erneut in den Generator ein.

Wir halten an dieser Stelle fest, dass der Attraktor eines zweidimensionalen Fraktals durch Initiator und Generator eindeutig definiert ist.

2.1.3 Selbstähnlichkeit

Eine für alle Fraktale charakteristische Eigenschaft ist die Selbstähnlichkeit. *Eine Figur heißt **selbstähnlich**, wenn Teile von ihr verkleinerte Kopien der ganzen Figur sind.* [4, S.21] Der oben beschriebene Konstruktionsprozess verleiht den Attraktoren solcher Fraktale diese Eigenschaft. In dem Bestreben diese Selbstähnlichkeit näher zu charakterisieren, wurde der Begriff der Selbstähnlichkeitsdimension eingeführt. Es würde hier den Rahmen sprengen, die Diskussion um den Dimensionsbegriff noch einmal aufzurollen. Daher beschränke ich mich an dieser Stelle darauf, diesen Begriff einzuführen und wie folgt zu definieren:
Selbstähnlichkeitsdimension:
Wird ein Gebilde im Maßstab $1:n$ verkleinert und passen nun k Kopien von den kleineren Gebilden in das ursprüngliche Gebilde, so ist die Selbstähnlichkeitsdimension des Gebildes die Zahl d, für die gilt: $n^d = k$ [1, S.51]

2.2 Vorüberlegungen

Nach diesem kleinen Ausflug in bekannte Gebiete der fraktalen Geometrie, beschäftigen wir uns nun mit der Konstruktion eines neuen Fraktals. Aus der Tatsache, dass das Fraktal letztendlich im Rapid-Prototyping-Verfahren gebaut werden soll, folgen einige **Bedingungen**:

1. Der Attraktor des Fraktals sollte nicht „langweilig" sein. Es macht wenig Sinn, beispielsweise ein dreidimensionales Analogon zur zweidimensionalen Hilbert-Kurve [6, S.544] zu konstruieren, die dann anstatt gegen ein Quadrat gegen einen Würfel konvergiert; denn einen Würfel kann man sehr einfach herstellen, indem man auf eine fraktale Konstruktion verzichtet. Des Weiteren ist dafür wohl kaum ein so aufwendiges Verfahren wie Rapid-Prototyping erforderlich. Da dieses Verfahren hier einmal ausgereizt werden soll, ist eine unendlich zerklüftete Oberfläche wünschenswert.
2. Der Attraktor sollte nicht zu fragil sein. Da noch nie versucht wurde, ein Kunststoffmodell eines Fraktals auf diese Weise herzustellen und die Herstellung nicht ganz billig ist, sollte der Prototyp so kompakt sein, dass er die Entnahme aus der Maschine sicher übersteht. Ein zu fragiles Objekt fiele vielleicht schon bei leichter Berührung unter seinem eigenen Gewicht zusammen. Am besten wäre es also, wenn der Attraktor zumindest partiell raumfüllend ist. Eine absolut notwendige Voraussetzung ist natürlich auch, dass die Kurve überhaupt gegen eine Grenzfigur konvergiert.
3. Die Selbstähnlichkeitsdimension des Fraktals soll 3 sein. Marco Stolpe, ein ehemaliger Schüler meiner Schule und Teilnehmer bei „Jugend forscht", konstruierte in seiner Arbeit [7] ein Analogon zur Drachenkurve im dreidimensionalen Raum. Die Drachenkurve füllt eine zweidimensionale Fläche und hat auch die Selbstähnlichkeitsdimension 2. Sein Analogon füllt vermutlich einen dreidimensionalen Raum und hat die Selbstähnlichkeitsdimension 3. Der Beweis dafür steht zwar bisher noch aus, aber es erschien mir sinnvoll, für ein Objekt, welches den dreidimensionalen Raum füllen soll, auch die Selbstähnlichkeitsdimension 3 zu wählen.
4. Die Konstruktion des Fraktals im Raum soll möglichst analog zum Konstruktionsverfahren in der Ebene sein.

2.3 Konstruktion

Im folgenden Kapitel werden wir nun diskutieren, wie ein Fraktal konstruiert werden kann, dass alle oben geforderten Bedingungen erfüllt.

2.3.1 Grundüberlegung

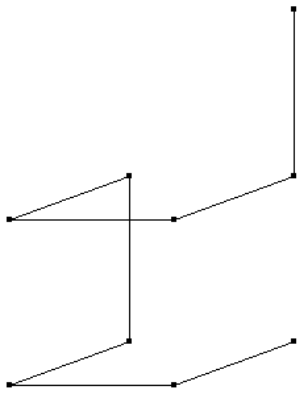


Abb. 3: Modifizierte Rundreise

Ich hatte bereits erwähnt, dass die Hamiltonsche Rundreise auch auf Objekten mit mehr oder weniger Dimensionen funktioniert, beispielsweise auf einem Quadrat. Betrachten wir nun nochmals die ersten Iterationen der Drachenkurve, so fällt auf, dass man diese auch aus Hamiltonsche Rundreisen auf einem Rechteck zusammenbauen kann, bei denen allerdings die letzte Strecke, die wieder zum Ursprungspunkt zurückführt, in die Gegenrichtung zeigt. Dieser Zusammenhang veranlasste mich, ein Fraktal auf Basis der Hamiltonschen Rundreise auf einem Würfel zu konstruieren, bei der die letzte Strecke ebenfalls in Gegenrichtung verläuft. Links sehen wir ein solches Objekt. Welche Selbstähnlichkeitsdimension hätte der Attraktor, den wir erhalten, wenn wir nun unendlich oft jede Strecke dieser Figur durch eine verkleinerte Kopie der gesamten Figur ersetzen würden? Für jede der $k = 8$ Strecken benötigen wir eine im Maßstab 1:2 ($n = 2$) verkleinerte Kopie der Figur. Daraus folgt:

$$n^d = k \Leftrightarrow d = \log_n k \Leftrightarrow d = \log_2 8 = 3$$

Diese Figur hat also die angestrebte Selbstähnlichkeitsdimension $d = 3$, womit die dritte Bedingung erfüllt ist.
Die diesem Fraktal zu Grunde liegende Figur, eine Hamiltonsche Rundreise auf einem Würfel, deren letzter Schritt jedoch in die entgegen gesetzte Richtung geht, bezeichne ich im Folgenden als ***erweiterte Hamilton-Figur, kurz eHF.***

2.3.2 Neue Freiheit und Uneindeutigkeit

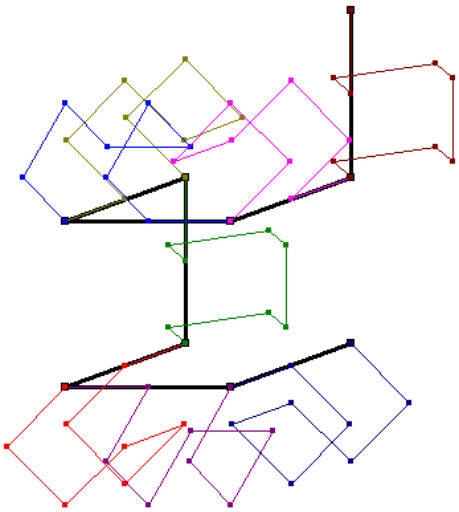


Abb. 4: Wie drehen wir die Lotvektoren?

Links sehen wir ein Ergebnis des oben beschriebenen Vorgangs, das durch ein von mir geschriebenes Programm im Konstruktionsmodus erstellt wurde: Die ursprüngliche Figur ist fett schwarz gezeichnet und zur Verdeutlichung nicht durch die 8 kleineren Figuren ersetzt, sondern stehen gelassen worden. Die 8 eHFs wurden zur besseren optischen Unterscheidung in verschiedenen Farben gezeichnet.
Bei der Formulierung der Iterationsregel haben wir festgehalten, dass der Attraktor eines zweidimensionalen Fraktals durch Initiator und Generator eindeutig definiert ist. Übertragen wir das Konstruktionsverfahren in den Raum, so stellen wir fest, dass dies nicht mehr der Fall ist! Wir haben jede Strecke durch eine erweiterte Hamilton-Figur ersetzt und somit die Iterationsregel erfüllt. Aber wir können die eHFs beliebig um die Achse drehen, die durch die Strecke geht, die sie ersetzen. Damit haben wir gleich achtmal unendlich viele Möglichkeiten die Iterationsregel zu erfüllen. Damit genügen wir zwar der vierten Bedingung unserer Vorüberlegung (Analogie des Konstruktionsverfahrens), erhalten aber nicht genau ein Fraktal, sondern gleich eine ganze Klasse von Fraktalen. Wir haben jetzt 8 Parameter, die wir frei verändern können, was uns bei der Erfüllung der anderen Bedinungen hilfreich sein wird.

2.3.3 Definition der Hamilton-Fraktalklasse

Bevor wir nun die Methoden diskutieren, die uns zur Untersuchung dieser Klasse von Fraktalen zur Verfügung stehen, werden wir zunächst einmal diese Fraktalklasse genau definieren.

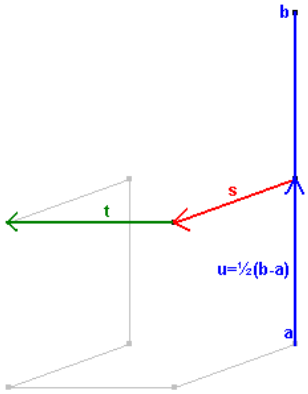


Abb. 5: Hamilton-Initiator

Befassen wir uns zunächst mit dem Initiator. Ein Initiator muss einerseits alle Informationen enthalten, die notwendig sind, um mit dem Bau des Fraktals zu beginnen. Andererseits stellt er die nullte Generation des Fraktals dar.
*Der **Hamilton-Initiator** ist durch eine Strecke, die durch zwei Ortsvektoren $\vec{a}$ und $\vec{b}$ definiert ist, sowie durch einen Startlotvektor $\vec{s}$ eindeutig definiert.*
Dass Anfangs- und Endpunkt eine Initiatorstrecke definieren, ist bei zweidimensionalen Fraktalen auch der Fall. Ersetzen wir jedoch diese Strecke durch eine eHF und erzeugen aus der nullten die erste Generation, so wird das Ergebnis erst dann eindeutig, wenn auch festgelegt ist, wie die neue Figur gedreht werden muss. Dies geschieht hier durch den rot gekennzeichneten Startlotvektor, der senkrecht auf dem Verbindungsvektor zwischen $\vec{a}$ und $\vec{b}$ steht. Dieser ist in der nullten Generation zwar nicht sichtbar, muss aber trotzdem zum Initiator gezählt werden, wenn man fordert, dass dieser alle Informationen enthalten soll, die zum Baubeginn notwendig sind. Diesen

Vektor können wir beliebig rotieren lassen, solange er auf der Verbindungsstrecke senkrecht steht, was wir in der Gleichung $(\vec{b}-\vec{a})\cdot\vec{s}=0$ formalisieren können.
Bei genauer Betrachtung stellt man fest, dass die eHF aus nur drei linear unabhängigen Vektoren konstruiert werden kann. Der Startlotvektor $\vec{s}$ ist bereits gegeben. Der Vektor $\vec{u}$ ist der halbe Verbindungsvektor zwischen $\vec{a}$ und $\vec{b}$. Der letzte noch fehlende Vektor $\vec{t}$ steht senkrecht auf den beiden anderen und senkrecht zu der von ihnen aufgespannten Fläche, womit wir ihn mit Hilfe des Kreuzprodukts berechnen können: $\vec{t}=\vec{s}\times\vec{u}$. Die Ortsvektoren zu allen anderen Punkten können aus diesen drei Vektoren berechnet werden.
Der ***Hamilton-Generator*** *ist durch den Hamilton-Initiator und 8 Lotvektoren eindeutig definiert.*
Vom Generator fordern wir analog zum Initiator, dass er alle notwendigen Informationen zur Erzeugung der nächsten Generation enthält. Der Initiator enthält bereits alle Informationen, die zur Zeichnung der eHF erforderlich sind. Als Konsequenz aus dem Freiheitsgrad des Raums müssen auch hier nicht sichtbare Vektoren hinzugefügt werden. In diesem Falle sind 8 Lotvektoren erforderlich, um zu beschreiben, wie die eHFs der nächsten Generation, die in dieser Figur alle Strecken ersetzen, gedreht werden müssen. Auch für diese Lotvektoren gilt prinzipiell lediglich die Einschränkung $(\vec{h}_n-\vec{h}_{n-1})\cdot\vec{l}_n=0$, wenn man die Ortsvektoren der eHF mit $\vec{h}_n$ und den zugehörigen Lotvektor mit $\vec{l}_n$ bezeichnet ($0<n<9$).
Um diese Einschränkung zu beachten, ist es hilfreich, die Lotvektoren nicht mit absoluten Werten anzugeben. Denn ein Lotvektor, der in der ersten Generation noch senkrecht auf einer Strecke stand, muss nicht unbedingt auch in der zweiten senkrecht auf ihr stehen. Daher ist es sinnvoll, die 8 Lotvektoren des Generators durch die drei Vektoren des Initiators anzugeben. Dadurch ist gewährleistet, dass in allen Generationen die eHFs richtig gedreht sind. Da die drei Vektoren des Initiators linear unabhängig sind, somit eine Basis des $\mathbb{R}^3$ bilden und man somit jeden beliebigen Vektor dieses Vektorraums darstellen kann, nimmt man sich dadurch keine Freiheiten.
Des Weiteren habe ich bei den von mir untersuchten Fraktalen die zusätzliche Einschränkung gemacht, dass der Freiheitsgrad nur einmal ausgenutzt wird, dass also einmal für alle Generationen festgelegt wird, wie die eHFs gedreht werden. Man könnte sie nämlich auch von Generation zu Generation unterschiedlich rotieren lassen.

3 Modellierung am Computer

Nachdem der mathematische Entwurf abgeschlossen war, musste ein Computerprogramm entwickelt werden, mit dem ich diese Fraktale untersuchen konnte. Ich habe seit der 9. Klasse Informatik und arbeite seit der 11. mit der Programmiersprache Borland Delphi 6, die wir auch im Informatik-Leistungskurs benutzen. Daher lag es nahe, diese Programmiersprache zu verwenden, um eine geeignete Untersuchungsumgebung für die Fraktale zu programmieren.

3.1 Der Hamilton-Iterator als MVKM

In der nächsten Phase des Projektes arbeitete ich also vorwiegend mit diesem Delphi-Programm, das ich immer wieder umschrieb und mit immer mehr Funktionen versah. Inzwischen ist es ca. 1000 Zeilen lang. Es würde den Rahmen sprengen, seine Funktionsweise hier detailliert zu erläutern.
Eine anschauliche Darstellung von Iteratoren ist die so genannte Mehrfachverkleinerungskopiermaschine, kurz MVKM. Diese Veranschaulichung des Konstruktionsprozesses eines Fraktals als gedankliche Maschine ist in der Literatur durchaus üblich [5, S.30] und für uns hier hilfreich, weil sie zwar sehr genau beschreibt, wie Fraktale konstruiert werden, zum anderen aber die Kenntnis einer speziellen Programmiersprache zum Verständnis nicht erforderlich ist. Das bedeutet auch, dass diese Darstellung unabhängig von der Programmiersprache ist, in der man die Maschine letztendlich implementiert. Dies geschieht durchaus

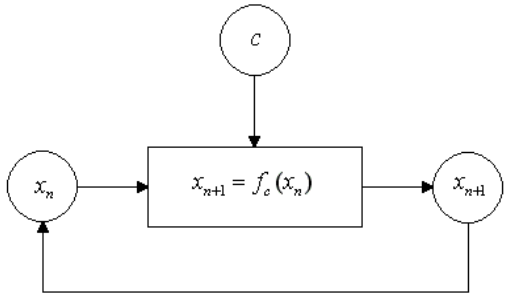


Abb. 6: Quadratischer Iterator

nicht nur aus ästhetischen Gründen. Wie in den folgenden Kapiteln dargelegt werden wird, war es notwendig, das Fraktal mit einer ganzen Reihe von Sprachen zu beschreiben.

Links ist das Grundschema einer solchen Rückkopplungsmaschine abgedruckt. Wir geben ein Element x_n in die Maschine. Diese erzeugt das nächste Element als Funktion von ihm, die noch von einer Konstante c abhängig ist. Das Ergebnis x_{n+1} wird erneut in die Maschine eingegeben. Ein einfaches Beispiel ist der bekannte quadratische Iterator ($x_{n+1} = x_n^2 + c$), der genau so funktioniert.

Eine Rückkopplungsmaschine, die die im vorigen Teil beschriebenen Fraktale erzeugt – wir wollen sie als Hamilton-Iterator bezeichnen –, unterscheidet sich vom quadratischen Iterator dadurch, dass die zu Grunde liegende Funktion und die zugeführte Konstante viel komplexer sind.

Wie kann man sich diese Rückkopplungsmaschine möglichst plastisch veranschaulichen? Den Initiator kann man sich als Chip vorstellen, auf dem sich drei Speicher für das Vektortripel befinden, das ihn definiert. Die Maschine hat einen Schlitz, durch den dieser Chip eingezogen wird. Die erste Einheit liest intern nun das Vektortripel aus und gibt es an den Generator weiter; der Chip wird entsorgt. Der Generator berechnet nun aus dem Vektortripel die 8 Anfangs- und Endpunkte der Strecken der (erweiterten) Hamilton-Figur. Außen am Generator befinden sich 8 Eingabeeinheiten, mit denen die Lotvektoren festgelegt werden können. Aus diesen Daten, den 8 Lotvektoren und den 8 Anfangs- und Endpunkten werden nun 8 neue Vektortripel erstellt und an eine Ausgabeeinheit übergeben, die diese auf 8 neue Chips schreibt und ausgibt. Dies ist die nächste Generation des Fraktals. Wenn man diese Zwischenstufe betrachten möchte, dann kann man die Maschine jetzt anhalten. Ansonsten werden diese Chips nun nacheinander auf einem Förderband wieder zum Eingabeschlitz transportiert und die nächste Generation wird erzeugt. Den Attraktor des Fraktals würde man theoretisch erhalten, wenn man die Maschine unendlich lange arbeiten lässt.

Das Herzstück meines Delphi-Programms bildet die Implementation dieser Maschine. Dabei habe ich – der sachkundige Leser wird es erahnen – das Prinzip der Rekursion verwendet. Vektoren an sich kann man aber noch nicht sehen. Daher musste noch ein Algorithmus geschrieben werden, der diese auf den Bildschirm projiziert. Dieses ist ein Grundproblem der Grafikprogrammierung, weswegen ich an dieser Stelle nicht näher darauf eingehen möchte.

Üblicherweise erhält man durch eine Projektion eine grafische Darstellung, die einen guten Eindruck von der zu untersuchenden Figur liefert. Es ist allerdings ein Trugschluss zu glauben, man könne dies problemlos auf die dreidimensionale fraktale Geometrie übertragen. Die bisherigen Abbildungen in dieser Arbeit stellen ja lediglich die erste oder zweite Generation der Kurven dar. Die Anzahl A der Vektortripel, die man benötigt, um ein Hamilton-Fraktal H in der n-ten Generation darzustellen, beträgt $A(H_n) = 8^n$. Die linke Abbildung stellt eine Projektion eines Hamilton-Fraktals der 8. Generation dar, welches also aus $8^8 = 16.777.216$ Eckpunkten besteht. Abgesehen davon, dass dieses stark exponentielle Anwachsen der Eckpunkte katastrophale Auswirkungen auf das Laufzeitverhalten des Algorithmus hat – die Berechnungszeit für dieses Bild beträgt ca. 300s bei 1,73Ghz-, ist es kaum möglich, aus einer solchen Projektion noch einen dreidimensionalen Eindruck von der Figur zu bekommen. Was sich eindeutig erkennen lässt, ist, dass es sich offenbar um eine sehr komplexe Figur handelt. Wenn wir uns noch einmal die Vorüberlegungen zum mathematischen Entwurf dieser Kurve in Erinnerung rufen, dann können wir also die erste Bedingung – der Attraktor des Fraktals sollte nicht „langweilig" sein – als erfüllt betrachten.

Abb. 7: Perspektivische Zeichnung

Nun müssen wir nur noch die zweite Bedingung erfüllen und sicherstellen, dass ein solches Modell nicht zu fragil ist. Dazu ist es aber notwendig, einen besseren Eindruck von der Struktur dieses Objekts zu bekommen.
Natürlich könnte man es mit einer verbesserten grafischen Darstellung versuchen. Aus einer dreh- und zoombaren (am besten auch noch beleuchteten) Darstellung gewinnt man, wie sich zu einem späteren Zeitpunkt herausstellte, tatsächlich einen besseren Eindruck.

3.2 Schnitte

Zu diesem Zeitpunkt allerdings entschied ich mich aus zwei Gründen gegen diese Lösung. Zum einen verfügte ich über zu wenig Erfahrung auf dem Gebiet der Grafikprogrammierung, um ein solch verbessertes Programm zu schreiben. Dies wäre ohnehin sehr aufwendig geworden und ich hätte mir das notwendige Wissen erst aneignen müssen. Zum anderen lag ein anderer Ansatz auf der Hand: Eine Rapid-Prototyping-Maschine baut das Modell Schicht für Schicht und somit wäre die Frage interessant, wie eigentlich Querschnitte durch meine Hamilton-Fraktale aussehen.
Die notwendigen Modifikationen am Programm sind verhältnismäßig einfach, wenn man einige Kompromisse macht. Selbstverständlich kann man vektoriell eine Schnittebene definieren, dann aus allen Strecken eine Geradengleichung aufstellen, diese mit der Ebene schneiden und dann alle Schnittpunkte darstellen. Es reicht aber völlig aus, einfach nur abzufragen, ob eine Strecke überhaupt die Schnittebene durchläuft, und im Erfolgsfall diese Strecke ganz normal zu zeichnen. So wird die „Ebene", mit der man das Fraktal dann durchschneidet, zwar etwas dicker, aber bei hohen Iterationsstufen fällt dies kaum noch auf, weil die Strecken extrem kurz sind. Die Ungenauigkeit, die dadurch entsteht, beschränkt sich also auf wenige Pixel. Eine exakte Berechnung durch die oben beschriebene Methode würde erheblich mehr Rechenaufwand erfordern. Dieser ist bei hohen Iterationsstufen aber ohnehin schon immens. Es gibt noch andere Möglichkeiten, Rechenzeit zu sparen: Wenn man nur eine Schnittebene betrachtet, ist es nicht mehr notwendig, auf die Dreidimensionalität des ganzen Objekts Rücksicht zu nehmen. Daher habe ich das Programm so abgeändert, dass es die Schnittebenen nicht nur perspektivisch, sondern auch aus der Frontansicht darstellen kann. Da die Strecken, die dann noch sichtbar sind, bei hohen Iterationsstufen so kurz sind, dass sie nur noch aus sehr wenigen Pixeln, manchmal sogar nur aus einem einzigen bestehen, kann man einen weiteren Kompromiss rechtfertigen: Die Anfangs- und Endpunkte der Linien werden durch ein kleines Quadrat mit einer Kantenlänge von zwei Pixeln dargestellt. Dadurch erhält man zwar eine etwas „verwaschene" Darstellung, aber man kann die Iterationsstufe von 8 auf 7 reduzieren und erhält trotzdem noch vernünftige Ergebnisse. Außerdem ist so auch das Problem beseitigt, dass sich Delphi weigert, Strecken mit einer Länge von genau einem Pixel zu zeichnen.
Um nun eine bessere Einsicht in die Struktur des Fraktals zu bekommen, wollen wir natürlich nicht nur einen Querschnitt betrachten, sondern eine ganze Serie, die das Fraktal von hinten nach vorne in Scheiben schneidet, sodass wir quasi durch das Fraktal hindurch reisen können. Auch auf Iterationsstufe 7 dauert die Berechnung für einen Querschnitt trotz der oben beschriebenen Optimierungen noch relativ lange. Daher habe ich in das Programm eine Screenshot-Funktion implementiert. Diese speichert die Schnitte als 700x700 Pixel große Bitmaps ab. Nun müssen wir noch herausfinden, wie viele Schnitte man in vertretbarer Zeit erzeugen kann und wie viele notwendig sind, um einen hinreichend tiefen Eindruck in die Struktur des Fraktals zu bekommen.
Es hat sich bei meinen Computerexperimenten nach vielen Versuchen folgender Standard etabliert: Das Fraktal wird auf Iterationsstufe 7 erzeugt, es werden von hinten nach vorne 160 Querschnitte berechnet, die denselben Abstand voneinander haben, und jeder Linien-Eckpunkt wird als Quadrat mit einer Kantenlänge von 2 Pixeln gezeichnet. Nach einigen Stunden erhält man dann 160 Bitmaps, die insgesamt gut 300MB Speicherplatz verbrauchen, weil Delphi die Bitmaps mit einer Farbtiefe von 32bit abspeichert. Vor dem Hintergrund der Tatsache, dass ich nicht nur eine solche Schnittmenge erzeugen wollte, sondern mehrere, war klar, dass ich entweder mehrere Gigabyte Speicherplatz oder eine effizientere Speichermethode benötigen würde. Ich entschied mich für Letzteres und ließ alle Bitmaps von einem Grafikprogramm in Bitmaps mit einer Farbtiefe von 1bit konvertieren. Da die Schnitte sowieso nur in schwarz-weiß berechnet und gespeichert sind, verliert man bei die-

sem Verfahren nichts an Information. Weil ich immer auf demselben Intervall die 160 Schnitte durchführte, die Fraktale aber unterschiedliche Größen haben, kam es häufig vor, dass auf vielen Schnitten gar nichts zu sehen war. Diese wurden dann entfernt und die verbleibenden Schnitte neu durchnummeriert. Die Datenmenge, die dann noch übrig bleibt, beläuft sich auf angenehme 3-5MB.
Als nächstes brauchte ich eine Möglichkeit, die Ergebnisse dieses Verfahrens ansehen zu können. Natürlich gibt es bereits Programme, die Bilder wiedergeben können. Mit relativ wenig Aufwand lässt sich ein solches Programm aber auch selbst schreiben, was ich dann auch getan habe. Darin implementierte ich alle Funktionen, die ich benötigte, wie etwa das oben beschriebene Löschen von leeren Bildern.

Abb. 8: Schnittzeichnung

Links können wir einen solchen Schnitt sehen. Er stammt aus demselben Fraktal, welches im vorherigen Abschnitt perspektivisch dargestellt wurde. In dieser Kurve wurden die Lotvektoren schräg nach innen geklappt, weil man vermuten wird, dass das Fraktal dann sehr kompakt wird. Aufgrund der hohen Iterationsstufe sind in der perspektivischen Darstellung weite Teile der Grafik vollständig schwarz ausgefüllt. Da die perspektivische Projektion also ziemlich flächenfüllend ist, neigt man zu der Annahme, das Fraktal selbst sei raumfüllend. Wie der Schnitt zeigt, ist das Fraktal löchrig wie ein Schweizer Käse. Es würde hier den Rahmen sprengen, sämtliche 160 Schnitte zu diskutieren, aber ich kann versichern, dass auch die anderen Schnitte sehr löchrig sind, was mich daran zweifeln ließ, ob ein Kunststoffmodell eines solchen Fraktals eine lange Lebensdauer haben würde.
Also klappte ich die Lotvektoren schräg nach außen und untersuchte erneut die Schnitte. Auch diese Figur war löchrig und spaltete sich in einigen Schnitten ebenfalls in zwei Bereiche, die gar nicht mehr miteinander verbunden waren. Danach klappte ich die Lotvektoren gerade nach außen, was aber auch kein zufrieden stellendes Ergebnis lieferte.
Die Schnitte waren zwar eigentlich nur als Hilfsmittel gedacht, aber sie werfen interessante Fragen auf. Einige Schnitte ähneln mit etwas Phantasie der Drachenkurve. Der Gedanke, dass es vielleicht eine Anordnung von Lotvektoren geben könnte, die eine Kurve liefert, in deren Schnitten die Drachenkurve vorkommt, ist verlockend. Außerdem fällt auf, dass die Schnitte zwar hochkomplexe Figuren darstellen, dass aber viele von ihnen symmetrisch zu sein scheinen. Hier stellt sich die Frage: Sind alle Schnitte durch ein Hamilton-Fraktal symmetrisch, wenn die Lotvektoren entsprechend angeordnet sind, oder ist dies nur eine naive Vermutung?

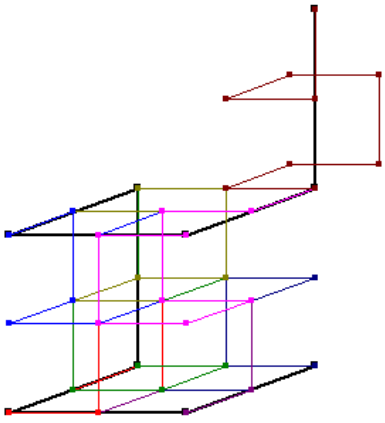


Abb 9: Gerade nach innen (Konstruktionszeichnung)

Abb. 10: Gerade nach innen (Schnittzeichnung)

Obwohl mich auch diese Fragen in Atem hielten, habe ich sie erst einmal nicht weiter verfolgen können, weil mich eine andere Hamilton-Kurve zu einer neuen Idee brachte. Die einfachste Wahl von Lotvektoren, wenn man ein möglichst kompaktes Objekt erhalten möchte, ist doch die, dass man sie einfach gerade nach innen klappt. Dies ist links im Konstruktionsmodus dargestellt. Vergleicht man jetzt dieses Bild mit dem Schnitt rechts, so wird klar, warum die Figuren so instabil werden: Die letzte Strecke der Hamiltonschen Rundreise wurde ja absichtlich in die entgegen gesetzte Richtung festgelegt. Dies war notwendig, um überhaupt eine durchgehende Kurve zu erhalten. Doch wie wir jetzt sehen, führt dies dazu, dass dort oben ein Teil des Fraktals „wegdriftet" und somit die Stabilität gefährdet. Die Iteration der ursprünglichen Hamiltonsche Rundreise führt jedoch zu einer divergenten Kurve.

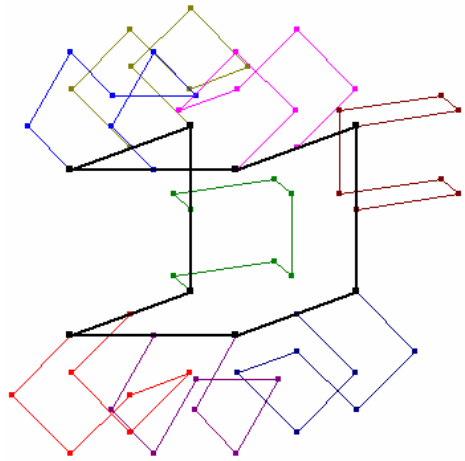


Abb. 11: Hamilton Komplett

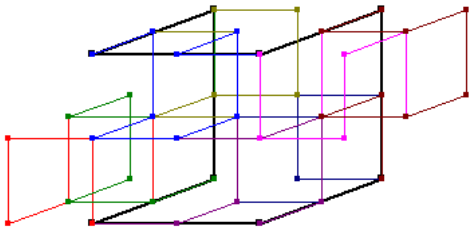


Abb. 12: Lotvektoren "folgen" Hamilton

Abb. 13: Schnitt

Aber es gibt einen „Kompromiss": Wir wählen als Startobjekt die komplette Hamiltonsche Rundreise; bei allen weiteren Streckenersetzungen benutzen wir die herkömmliche erweiterte Variante. Dies können wir links auf der 2. Iterationsstufe in der üblichen Weise abgebildet sehen.
Mit dieser Modifikation wiederholte ich alle vier Experimente; Lotvektoren schräg nach außen, schräg nach innen, gerade nach außen, gerade nach innen.
Kurz gesagt: Auch diese Ergebnisse waren ernüchternd. Diese Computerexperimente beschäftigten mich von den Osterferien bis zu den Sommerferien 2004 und noch immer hatte ich kein Ergebnis, mit dem ich wirklich zufrieden war.
In den Sommerferien schließlich kam dann die rettende Idee: Offenbar war kein zufrieden stellendes Ergebnis zu erhalten, wenn man es sich bei der Wahl der Lotvektoren so einfach machte. Also überlegte ich mir dafür eine andere Regel. Fassen wir die Hamilton-Figur als Vektorzug auf, bezeichnen wir die Strecken mit $\vec{h}_0$ bis $\vec{h}_8$ und die zugehörigen Lotvektoren mit $\vec{l}_1$ bis $\vec{l}_8$. Die Regel lautet nun folgendermaßen: Man wähle den Lotvektor $\vec{l}_n$ für den Vektor $\vec{h}_n$ $(0 < n > 8)$ so, dass er in dieselbe Richtung zeigt wie der Vektor $\vec{h}_{n+1}$. Der letzte Lotvektor zeigt in dieselbe Richtung wie die erste Strecke. Diese Wahl der Lotvektoren erschien mir etwas weniger willkürlich zu sein.
Und tatsächlich liefert diese Wahl, wie der Schnitt links zeigt, eine viel kompaktere Figur. Ich war der Meinung, dass von allen untersuchten Figuren diese hier mit Abstand die besten Chancen hat, dass aus ihr ein interessantes und zugleich hinreichend stabiles Modell gebaut werden kann.
Am Stand wird die Möglichkeit bestehen, einen genaueren Einblick in die Ergebnisse der Computerexperimente und das Delphi-Programm zu erhalten.

4 Bauvorbereitungen

Nachdem nun die genauen mathematischen Spezifikationen des zu bauenden Fraktals durch die Computermodellierungen gefunden waren, konnte nun endlich mit den Vorbereitungen des Baus begonnen werden. Daher nahmen Herr Hülsbusch und ich nach den Sommerferien 2004 Kontakt mit der Universität Duisburg-Essen (Campus Duisburg) auf, und zwar mit dem Dekan der Ingenieurwissenschaften, Professor Dr.-Ing. Andrés Kecskeméthy, der uns bereits während der Schülertage durch das Institut für Mechatronik und Systemdynamik führte, in dem wir die Rapid-Prototyping-Maschine zum ersten Mal zu Gesicht bekamen.
Bei unserem ersten Besuch am 02.09.04 stellte ich das Fraktal Professor Kecskeméthy und seinem wissenschaftlichen Mitarbeiter Martin Tändl vor, mit denen wir das weitere Vorgehen diskutierten.

4.1 Datenformate

In der klassischen Geometrie können Objekte wie Kugeln oder Quader mit einer relativ geringen Datenmenge beschrieben werden. Daher kann man die Daten solcher Objekte direkt in die Rapid-Prototyping-Maschine eingeben. Da das Fraktal aber je nach Iterationsstufe sehr viele Eckpunkte hat, kam eine manuelle Eingabe nicht in Frage. Ich hatte gehofft, dass die Bitmap-Schnitte, die ich während der Computermodellierung benutzt hatte, vielleicht als Datengrundlage benutzt werden könnten. Dies ist aber nicht der Fall. Die Daten der zu bauenden Objekte werden im so genannten STL-Format abgespeichert. Wenn man also nicht alle Eckpunkte selbst eingeben möchte, dann braucht man eine solche Datei, die alle nötigen Informationen enthält.
Wir diskutierten zwei Möglichkeiten, eine solche Datei zu erzeugen: Am einfachsten wäre es, mein Delphi-Programm so zu modifizieren, dass es das Fraktal nicht mehr grafisch auf dem Bildschirm darstellt, sondern die Koordinaten aller Eckpunkte als Textdatei abspeichert. Diese Datei könnte dann in mehreren Konvertierungsverfahren in das STL-Format gebracht werden. Das wäre zwar mit relativ wenig Programmieraufwand für mich verbunden gewesen, hätte aber verschiedene Nachteile gehabt. Zum einen ist die Konvertierung dann sehr aufwendig und zum anderen müsste dieser Aufwand dann jedes Mal betrieben werden, wenn ein solches Objekt gebaut werden soll. Sinn dieser Arbeit ist es aber, nach Lösungen zu suchen, mit denen man auch künftig möglichst schnell und einfach Modelle von Fraktalen herstellen kann.
Daher entschieden wir uns für einen anderen Weg: Im Bereich der Biomechanik werden im Institut für Mechanik dreidimensionale Computermodelle des menschlichen Beins hergestellt. Für C++ gibt es ein Paket namens Mobile, das auf mechanische Simulationen spezialisiert ist. Damit kann man Dateien im „Open Inventor"-Format erzeugen. Mit einer solchen Datei kann man dreidimensionale Objekte am Bildschirm frei dreh- und zoombar darstellen. Außerdem können solche Dateien in einem Schritt in das für das Rapid-Prototyping benötigte STL-Format konvertiert werden.

4.2 Aus der Mathematik in die Realität

Neben dieser Problematik der Datenformate stand ich nun vor der Herausforderung, ein Objekt, das normalerweise nur in der abstrakten Welt der Mathematik existiert, über die Zwischenstufen der Informatik und der Technik in die Realität zu holen.

4.2.1 Iterationsstufe

Iterationsstufe	Eckpunkte	Dateigröße
1	8	104 B
2	64	628 B
3	512	6,82 KB
4	4.096	68,4 KB
5	32.768	640 KB
6	262.144	5,74 MB
7	2.097.152	51,9 MB
8	16.777.216	(~0,5 GB)

Tab. 1: Exponentielles Wachstum

Der Attraktor des Fraktals kann offenbar nicht exakt nachgebaut werden, da er aus unendlich vielen Eckpunkten besteht. Diese Tatsache muss man bereits hinnehmen, wenn man das Fraktal nicht rein mathematisch betrachtet, sondern programmiert. Die Frage ist also, auf welcher Iterationsstufe der Bau des Fraktals technisch realisierbar ist. Dabei sollte man sich den folgenden Zusammenhang vor Augen führen: Die Anzahl der Eckpunkte wächst stark exponentiell mit der Iterationsstufe an. Um das Wachstum zu veranschaulichen habe ich die ersten 8 Iterationen berechnet und nur die grafische Darstellung als Bitmap abgespeichert. Das Abspeichern der Eckpunkte als Textdatei wird sehr schnell sehr speicherintensiv. Dies habe ich nur bis zur 7. Iteration durchgeführt, die 8. würde voraussichtlich etwa einen halben Gigabyte Speicherplatz erfordern. Daher entschieden wir uns, Iterationsstufe 7 anzustreben.

4.2.2 Initiatorlänge

Außerdem musste die Länge der Initiatorstrecke festgelegt werden, die die physikalische Größe des Fraktals bestimmt. In der Rapid-Prototyping-Maschine befindet sich ein Begrenzungskörper in Form eines Zylinders mit einer Höhe von 380mm und einem Durchmesser von 300mm. Ich veränderte das Koordinatensystem meines Programms so, dass die Koordinaten den echten Werten in Millimetern entsprachen und verschob den Ursprung des Koordinatensystems in den Mittelpunkt des Würfels, auf dem zu Beginn die Hamiltonsche Rundreise durchgeführt wird.
Ich erweiterte das Programm dann um einen Algorithmus, der anhand dieser Koordinaten prüft, ob sämtliche Eckpunkte innerhalb des Begrenzungskörpers der Rapid-Prototyping-Maschine liegen. Nach einigen Computersimulationen kam ich zu dem Wert $l = 180mm$.

4.2.3 Kurvendicke

Mathematisch betrachtet besteht das Fraktal bei einer endlichen Iterationsstufe aus einer langen Kette von Strecken. Eine Strecke ist ein eindimensionales Objekt; sie hat zwar eine Länge, aber keine Breite und keine Höhe. Bei der Computermodellierung muss man dies nicht selbst berücksichtigen. Zeichnet man eine Strecke auf dem Bildschirm, so ist diese automatisch mindestens einen Pixel dick. Nun ist es in der Realität natürlich nicht möglich, ein Kunststoffmodell aus eindimensionalen Objekten zu bauen. Daher müssen wir der Kurve nun eine endliche Dicke verleihen, sprich die eindimensionalen Strecken durch dreidimensionale Objekte ersetzen. Da läge natürlich der Zylinder nahe. Allerdings ist ein Zylinder rund und alles, was rund ist, schluckt noch mehr Rechenaufwand. Da dieser ohnehin schon gigantisch ist, habe ich mich für quadratische Säulen entschieden. Die absolute Minimaldicke, die ein Objekt haben muss, welches im Rapid-Prototyping-Verfahren hergestellt werden soll, beträgt aus technischen Gründen 0,5mm.
Die Endlichkeit der Dicke der Kurve hat Auswirkungen auf die zu ihrer Beschreibung notwendige Datenmenge, da sich die Anzahl der Eckpunkte erhöht. Die Strecke zwischen zwei Eckpunkten ist durch diese beiden Punkte bereits eindeutig beschrieben. Eine quadratische Säule hat 8 Eckpunkte. Es gibt nun unterschiedliche Möglichkeiten, diese zu beschreiben. An der Uni erklärte uns Herr Tändl, dass für Mobile folgende Methode am günstigsten ist: Man definiert für jeden Eckpunkt ein Relativkoordinatensystem, dessen Ursprung im Eckpunkt selbst liegt und dessen z-Achse zum nächsten Eckpunkt zeigt. Die quadratische Säule wird dann in diesem Koordinatensystem beschrieben.
Ein Koordinatensystem kann durch drei Einheitsvektoren beschrieben werden. Diese seien mit $\vec{e}_x$, $\vec{e}_y$ und $\vec{e}_z$ bezeichnet. Der Ortsvektor zum aktuellen Eckpunkt sei $\vec{p}_1$, der Ortsvektor zum folgenden Eckpunkt sei $\vec{p}_2$. Wir benötigen nun einen Algorithmus, der uns ein eindeutiges Koordinatensystem liefert. Das oben genannte Verfahren liefert uns zunächst nur die z-Achse:

$$\vec{e}_z = (\vec{p}_2 - \vec{p}_1)^\circ$$

Ähnlich wie bei der Konstruktion der Hamilton-Kurve haben wir es hier mit einem Uneindeutigkeitsproblem zu tun, welches aus der Dreidimensionalität des Raums resultiert. Zwar ist nun eine Achse eindeutig definiert, aber das ganze Koordinatensystem kann nun beliebig um diese Achse rotiert werden. Während dieser Phase des Projekts war ich mit dem Oberstufenstoff der Vektorrechnung endlich vertraut und so nutze ich das Kreuzprodukt, um diese Uneindeutigkeit zu beseitigen: Wenn $\vec{p}_0$ der Vorgängereckpunkt von $\vec{p}_1$ ist, so können wir diesen benutzen, um die y-Achse so erzeugen:

$$\vec{e}_y = \left((\vec{p}_0 - \vec{p}_1) \times \vec{e}_z\right)^\circ$$

Der Vektor $\vec{e}_y$ steht senkrecht auf $(\vec{p}_0 - \vec{p}_1)$, senkrecht auf $\vec{e}_z$ und senkrecht auf der von diesen beiden Vektoren aufgespannten Ebene. Die Berechnung der x-Achse ist nun einfach:

$$\vec{e}_x = \vec{e}_z \times \vec{e}_y$$

Eine Normierung der x-Achse ist nicht notwendig. Da die beiden anderen Vektoren bereits die Länge 1 haben, ist die x-Achse automatisch normiert.
Ich implementierte einen Algorithmus – zu Testzwecken zunächst auch in meinem Delphi-Programm -, der aus den zuvor berechneten Absolutkoordinaten des Fraktals für jeden Eck-

punkt nach der soeben beschriebenen Methode ein Relativkoordinatensystem erzeugt und die Einheitsvektoren, die dieses System beschreiben, ausgibt. Dabei ergeben sich zwei weitere Probleme: Zum einen hat der erste Eckpunkt keinen Vorgänger. Dies lösen wir, indem wir dafür einfach den Ursprung des Koordinatensystems definieren. Dem zweiten Problem bin ich leider erst nach längerer Fehleranalyse auf die Schliche gekommen: Das Herzstück des Algorithmus basiert darauf, dass die y-Achse eindeutig mit Hilfe des Kreuzproduktes festgelegt wird. Es kann aber passieren, dass $\left(\vec{p}_0 - \vec{p}_1\right)$ und $\vec{e}_z$ zufällig linear abhängig sind.

In diesem Fall spannen sie keine Ebene auf, ihr Kreuzprodukt ist $\vec{0}$ und beim Versuch, den Nullvektor durch seine Länge zu teilen, erhalten wir die Fehlermeldung „Division durch 0". Dieses Problem habe ich folgendermaßen gelöst: Falls das Kreuzprodukt der Nullvektor sein sollte, so wird $\left(\vec{p}_0 - \vec{p}_1\right)$ willkürlich um 90° um die x-Achse des Absolutkoordinatensystems rotiert und das Kreuzprodukt erneut berechnet. Es muss sichergestellt sein, dass der Algorithmus nicht versucht, ein Koordinatensystem aus Nullvektoren zu erzeugen.
Nach diesen Modifikationen verliefen alle Tests erfolgreich und ich berechnete die Relativkoordinatensysteme bis zur 6. Iterationsstufe. Da jetzt nicht mehr nur ein Vektor pro Eckpunkt gespeichert werden muss, sondern zusätzlich auch noch die drei Einheitsvektoren des zugehörigen Relativkoordinatensystems, vervierfacht sich die Datenmenge.

4.3 Portierung nach C++

Nachdem diese Algorithmen nun konzipiert und in das Delphi-Programm implementiert waren, machte ich mich an die Arbeit, dieses Programm nach C++ zu portieren. Ich wusste zwar, was C++ ist, hatte aber noch nie damit gearbeitet. Ich nahm dies zum Anlass, mir die erforderlichen Programmierkenntnisse anzueignen. Damit die Portierung nicht zu lange dauerte, entschied ich, nur die Algorithmen zu portieren, die für das weitere Vorgehen absolut notwendig waren. Da nur die Koordinaten des Fraktals relevant waren, verzichtete ich auf sämtliche grafischen Darstellungen. Außerdem stellt dieses Programm nur eine Zwischenstufe dar, sodass ich auch auf eine Benutzeroberfläche verzichten konnte. In dieser abgespeckten Version werden nur die Startparameter des Fraktals festgelegt und dann der rekursive Erzeugungsalgorithmus aufgerufen, welcher alle Eckpunktkoordinaten abspeichert. Danach wird ein zweiter Algorithmus gestartet, der daraus die Relativkoordinatensysteme berechnet und ebenfalls abspeichert. Dies findet alles im Hintergrund statt, das Programm selbst erscheint als DOS-Box, in der irgendwann „Berechnungen komplett!" erscheint. Die C++-Kenntnisse, die ein solches Programm erfordert, sind wesentlich weniger umfangreich, sodass ich innerhalb weniger Wochen das Programm erstellen konnte. Als Compiler verwendete ich das kostenlose DevC++.
Da ich diese Algorithmen zuvor schon in Delphi getestet hatte, konnte ich sicher sein, dass sie keine konzeptionellen Fehler enthalten. Trotzdem ließ ich das Delphi-Programm seine Ergebnisse mit denen aus C++ vergleichen und stellte fest, dass sie voneinander abwichen. Dies lag allerdings nur an der unterschiedlichen Rechengenauigkeit, was sich leicht beheben ließ.

4.4 Rapid-Prototyping

Die genaue Bestimmung der physikalischen Parameter und die Implementierung des C++-Programms beschäftigten mich bis zu den Herbstferien 2004. Ich mailte das C++-Programm Herrn Tändl und wir vereinbarten einen zweiten Termin am 12.11.04. Herr Tändl hatte das Programm leicht modifiziert, sodass es zu „Mobile" kompatibel war, und es auch schon einmal auf Iterationsstufe 6 gestartet. Im Open Inventor konnten wir eine dreidimensionale Darstellung des Fraktals bewundern, deren Qualität erheblich besser war als die perspektivische Zeichnung des Delphi-Programms. Durch diese Darstellung bekommt man bereits einen viel besseren Eindruck von der Struktur des Fraktals, weil die Berechnungen so schnell sind, dass die Figur in Echtzeit gedreht und gezoomt werden kann. Die dazu erforderlichen Programme laufen leider nur unter Linux, womit ich mich nur sehr wenig auskenne. Ich

werde aber trotzdem versuchen, diese eindrucksvolle Darstellung am Stand verfügbar zu machen und werde außerdem einige Screenshots mitbringen.
Es folgte nun die letzte Phase, die Bauphase. Das Fraktal musste aus dem Open-Inventor-Format in das Dateiformat der Rapid-Prototyping-Maschine (STL) konvertiert werden. Dies stellte sich als unerwartet schwierig heraus, weil die Software, die dazu normalerweise verwendet wird, mit der Datenmenge nicht fertig wurde und abstürzte. Letztendlich musste die Iterationsstufe doch auf 5 reduziert werden und als Ausgleich die Kurvendicke auf 2mm erhöht werden. Eine Spezialsoftware des Lehrstuhls für Fertigungstechnik schaffte es dann schließlich nach zwei Stunden Berechnungszeit, die Konvertierung durchzuführen. Dann wurden die Daten eingegeben und nach 40 Stunden Rapid-Prototyping hielt ich das Fraktal mit 32.768 Eckpunkten am letzten Tag vor den Weihnachtsferien endlich in meinen Händen. Hier ein paar Fotos:

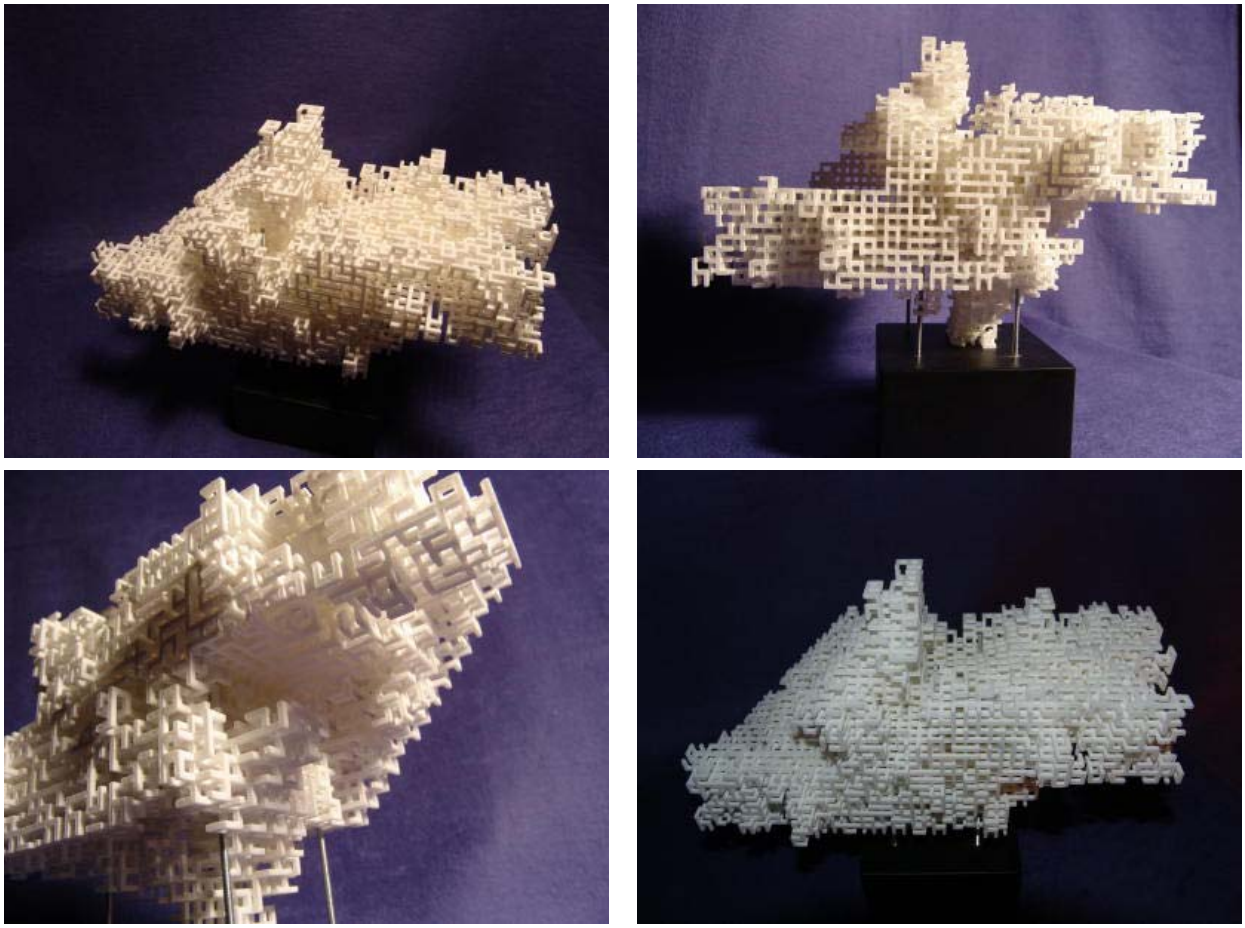


Abb. 14: Fotos des Kunststoffmodells

5 Schlussbemerkungen

Die hier vorgestellte Arbeit liefert keine neue mathematische Erkenntnis im üblichen Sinn. Es wurde kein einziger Lehrsatz entdeckt und kein wirklich neuer Begriff definiert. Vielmehr schildern die Ausführungen die - im wahrsten Sinne des Wortes – **Realisierung** einer **Idee**, nämlich der, ein Stück höchst komplizierte Mathematik **begreifbar** zu machen. Dabei mussten notwendig Kompromisse geschlossen werden. Diese sind oben beschrieben worden. Es ist aber zu erwarten, dass mit der erstmaligen Umsetzung der Idee, weitere Untersuchungen in dieser Richtung angestoßen werden könnten. So könnten beispielsweise auch andere Fraktaltypen für einen konkreten Nachbau von Interesse sein. Ein wichtiger Bereich der fraktalen Geometrie zielt beispielsweise darauf ab, komplizierte innere Organe wie etwa Leber oder Lunge eines Menschen durch Fraktale zu modellieren. Man kennt etwa Modelle, die durch Ausgießen der Blutgefäße in den Bronchien mit Kunststoff das Bronchialsystem eines Menschen darstellen. Derartige Abgüsse zeigen ein hohes Maß an Komplexität. Es wäre sicher höchst interessant, die entsprechenden Modellierungen solcher Systeme durch

Objekte der fraktalen Geometrie daneben zu halten, um sie mit dem Original vergleichen zu können. Hier könnte meine Idee vermutlich hilfreich sein. Ein Modell der gerade beschriebenen Art wäre vermutlich ein komplex verzweigter dreidimensionaler Baum. Mir ging es in meiner Arbeit darum, mehr wie ein Pionier tätig zu sein. Welche konkreten Möglichkeiten sich bei Fortschreiten der grafischen Möglichkeiten der Computer einerseits und der Technik der Rapid-Prototyping-Anlagen andererseits in Zukunft noch ergeben können, vermag ich derzeit noch nicht abzuschätzen.
„In der Mathematik ist die Kunst des Fragens oft von größerer Bedeutung als die Kunst der Problemlösung", sagte einst Georg Cantor. [5, S.81] Kunststoffmodelle können auf dem noch recht unerforschten Gebiet der dreidimensionalen fraktalen Geometrie neue Möglichkeiten eröffnen. Denn um zu interessanten Fragestellungen über ein Untersuchungsobjekt zu kommen, muss sich der Mathematiker dieses erst einmal begreifbar machen. In der Regel erfolgt dies durch eine simple Bleistiftskizze. Doch so hochgradig komplexe Objekte wie dreidimensionale Fraktale lassen sich prinzipiell nicht auf ein zweidimensionales Blatt Papier oder einen Bildschirm zeichnen. Selbst Computer generierte Projektionszeichnungen liefern nur dann einen zufrieden stellenden Eindruck von der Struktur des Objekts, wenn diese sehr aufwendig sind. Am besten kann man das Objekt jedoch verstehen, wenn man es anfassen kann. Dies bestätigte man mir auch an der Universität. Inzwischen machen sich eine ganze Reihe von Wissenschaften dieses Verfahren zu Nutze. Die Chemiker der Uni Duisburg stellten beispielsweise ein Modell eines größeren Moleküls her. Das bloße Vorhandensein dieses Modells hat ihre Problemlösung enorm beschleunigt. In der Biologie wurde Rapid-Prototyping eingesetzt, um die Doppelhelixstruktur eines DNA-Moleküls näher betrachten zu können. Auch die Mathematik sollte sich dieses Verfahren zu Nutze machen. Zum einen wird von Laien immer wieder die Forderung an die Mathematik gestellt, ihre Untersuchungsobjekte begreifbarer werden zu lassen. Zum anderen können wir am Beispiel der Chemiker sehen, dass dies aber auch im ureigenen Interesse der Mathematiker liegt, weil so neben Formelsprache und Computerzeichnungen eine zusätzliche Möglichkeit zur Verfügung steht, um zu neuen Ideen und Fragestellungen zu gelangen.
Abgesehen von dieser faszinierenden Aufgabe, abstrakte Objekte real werden zu lassen, bestand ein anderer Reiz meiner Arbeit aus dem engen Zusammenspiel von Mathematik, Informatik und Technik und der Kooperation zwischen Schule und Universität. Das Projekt hätte ohne die technischen Möglichkeiten und die Sachkenntnis der ingenieurwissenschaftlichen Fakultät nicht durchgeführt werden können. Daher möchte ich mich an dieser Stelle noch einmal bei der Abteilung für Maschinenbau für ihre Unterstützung bedanken, insbesondere bei Professor Kecskeméthy, Martin Tändl und Professor Witt.

6 Literaturverzeichnis

[1] Behr, Reinhart: *„Ein Weg zur fraktalen Geometrie"*, Klett Schulbuchverlag 1989

[2] Davis, Chandler / **Knuth**, Donald J: *„Number Representations and Dragon Curves",* Journal of Recreational Mathematics 3 (1970) 66-81 und 133-149

[3] Hülsbusch, Mathias: *„Interpretationen von Codes, die durch vollständige Binärbäume erzeugt werden",* Jugend forscht Regionalwettbewerb Duisburg 1997

[4] Peitgen, Heinz-Otto: *„Chaos: Iteration, Sensitivität, Mandelbrot-Menge; ein Arbeitsbuch"*, Berlin; Heidelberg; New York: Springer Stuttgard: Ernst Klett Schulbuchverlag 1992

[5] Peitgen, Heinz-Otto / **Jürgens**, Hartmut / **Saupe**, Dietmar: *„Fraktale – Bausteine des Chaos,* Klett-Cotta/Springer Verlag 1992

[6] Peitgen, Heinz-Otto / **Jürgens**, Hartmut / **Saupe**, Dietmar: *„Chaos – Bausteine der Ordnung",* Klett-Cotta/Springer Verlag 1992

[7] Stolpe, Marco: *„Konstruktion einer dreidimensionalen Drachenkurve",* Jugend forscht Regionalwettbewerb Duisburg 1995

[A1] Datengrundlage für die Bilder der Drachenkurve bildet ein JavaApplet von *http://did.mat.uni-bayreuth.de/~alfred/Dragon/d1.html*, 04.01.2005