Fraktale Terrainerzeugung

Jürgen Platzer[1], Mario Ruthmair[2]
Technische Universität Wien, Institut für Computergraphik und Algorithmen

Zusammenfassung

B. Mandelbrot entdeckte mit den stochastischen Fraktalen ein effizientes Mittel Gebirge und Oberflächenmodelle künstlich zu generieren. In dieser Ausarbeitung werden die grundlegenden Methoden beschrieben, die es der Computergraphik ermöglichen, ein Landschaftsrelief nachzuempfinden. In den letzten Jahren wurden diese Oberflächengeneratoren erweitert, um Effekte wie Erosionen oder Taleinschnitte in Gebirgszügen besser zu simulieren. Es werden daher einige Möglichkeiten besprochen, ein noch realistischeres Modell einer Landschaft zu erstellen.

Schlüsselwörter: midpoint displacement, fraktale Brownsche Bewegung, Multifraktale, spectral synthesis, hydraulische Erosion, temperaturbedingte Witterung

1 Einleitung

Für synthetisch generierte Szenen ist es immens wichtig Details aufzuweisen, um einer natürlichen Szenerie zu ähneln. Es ist unmöglich, Phänomene wie Wolken oder Berge durch einfache mathematische Modelle wie Kugeln oder Zylinder zu beschreiben. Fraktale ermöglichen es nun mittels einfacher Algorithmen Unregelmäßigkeiten auf Oberflächen zu erzeugen und natürliche Objekte zu simulieren. Dies und die Tatsache, dass Fraktale im Moment wohl die effizientesten Heuristiken zur Nachbildung von Komplexität zur Verfügung stellen, sind der Grund, warum hier eine kurze Einführung in die fraktale Geometrie gegeben wird. [9]

Laut K. Musgrave ist ein Fraktal „ein geometrisch komplexes Objekt, dessen Komplexität durch die Wiederholung einer gegebenen Form in verschiedenen Größen entsteht.“[9] Dies ist eine einfache Definition, die am Beispiel der Kochkurve veranschaulicht werden kann. In Abbildung 1 ist zunächst eine Strecke von 0 bis 1 zu sehen. Diese Strecke wird gedrittelt. Das zweite Drittel der Strecke wird durch die zwei oberen Kanten eines gleichseitigen Dreiecks ersetzt. Diese Ersetzung wird nun für jede der 4 neu entstandenen Strecken wiederholt. Um die fraktale Kochkurve exakt nachzuzeichnen muss diese Prozedur unendlich oft mit allen Streckenabschnitten durchgeführt werden. Die fraktale Komplexität kann somit beschrieben werden, als das wiederholte Ersetzen unter veränderter Größe. [9]

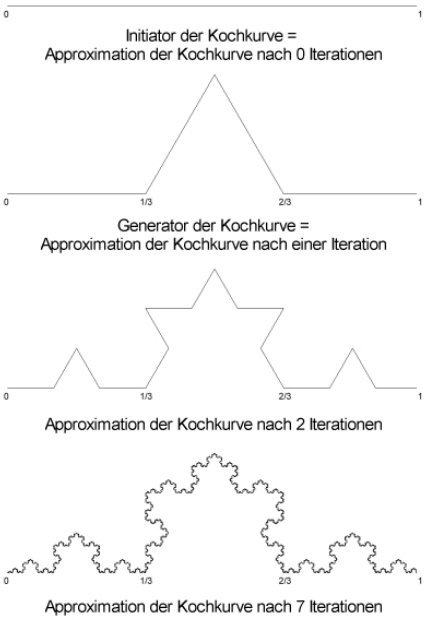

Abbildung 1: Erstellung der Kochkurve

Da bei jedem Iterationsschritt Strecken entstehen, die nur mehr ein Drittel der Länge der ursprünglichen Strecke aufweisen, besitzt die Kochkurve einen Skalierungsfaktor vom Wert 1/3. Da aber bei jeder Iteration drei Streckenteile durch 4 Abschnitte derselben Größe ersetzt werden weist die Kochkurve auf dem Intervall [0,1] eine unendliche Länge auf, ohne sich selbst zu schneiden. Weiters kann man feststellen, dass es keine algebraische Formel gibt, welche die Punkte, die auf der Kochkurve liegen, spezifiziert, obwohl der Erstellungsalgorithmus kurz und einfach zu beschreiben ist. Fraktale weisen also Eigenschaften auf, die man von Funktionen, die durch Gleichungen charakterisiert werden, nicht kennt. Deshalb wollen wir kurz auf die wichtigsten Eigenheiten von fraktalen Objekten eingehen. [12]

[1] bergfried@gmx.at
[2] ruthmair@utanet.at

Ein Merkmal, das die meisten Fraktale besitzen, und auch jene, die in dieser Ausarbeitung besprochen werden, ist die Selbstähnlichkeit. Wenn man einen beliebigen Teil eines selbstähnlichen Fraktals betrachtet, so beinhaltet dieser wieder das komplette Fraktal. Das hat zur Folge, dass diese Fraktale invariant bezüglich ihrer Skalierung sind. Egal wie stark man ein fraktales Objekt vergrößert, in dem Ausschnitt, den man betrachtet, lässt sich das gesamte Fraktal wieder finden. Im optimalen Fall ist der betrachtete Bereich äquivalent zum gesamten Fraktal. Dies lässt sich auch anhand der Kochkurve zeigen, die exakt selbstähnlich ist. Das heißt, dass jeder Abschnitt der Kurve die gesamte Kochkurve mit derselben geometrischen Form (aber mit anderer Skalierung) enthält. Um aus dem betrachteten Ausschnitt der Kurve wieder die Kochkurve in ihrer ursprünglichen Größe zu erhalten, muss dieser mindestens einmal um den Faktor 3 vergrößert werden. Für unsere Betrachtungen werden allerdings nicht optisch selbstähnliche Fraktale herangezogen, sondern jene mit statistischer Selbstähnlichkeit. Für diese Objekte gilt, dass jeder Ausschnitt die statistische Verteilung des Ganzen aufweist. Diese so genannten stochastischen Fraktale können daher auch nicht wie optisch selbstähnliche Fraktale aufskaliert werden. [13] [3] [8]

Eine Eigenschaft, die alle Fraktale aufweisen, ist die fraktale Dimension oder auch Hausdorff-Besicovitch Dimension, die sich von der geläufigen euklidischen Dimension, welche nur Werte von natürlichen Zahlen annimmt, unterscheidet. Die traditionelle euklidische Dimension definiert beispielsweise einen Punkt als null-, eine Line als ein-, ein Quadrat als zwei- und einen Würfel als dreidimensional. Die fraktale Dimension hingegen ist eine Erweiterung der euklidischen und erlaubt Werte von positiven reellen Zahlen. Man kann somit Fraktale erstellen, die jede beliebige Dimension zwischen 2 und 3 annehmen können, was für die Terrainmodellierung äußerst wichtig ist. Hierzu ist zu erklären, dass ein Fraktal mit der Dimension 2 stellenweise einer Fläche entspricht, mit steigender Dimension bis hinauf zu 3 immer unebener wird und letztlich so starke Deformationen aufweist, so dass es an bestimmten Stellen ein Volumen ausfüllt. [9]

Fraktale werden im Allgemeinen durch wiederholte Anwendungen von Prozeduren oder Berechnungsschritten erstellt, weshalb die fraktale Geometrie erst in der zweiten Hälfte des 20. Jahrhunderts zu boomen begann. Mit der Unterstützung von Computern war es möglich die umfangreichen Berechnungen durchzuführen, die notwendig waren, um fraktale Objekte näherungsweise darstellen zu können. Der Begründer der fraktalen Geometrie heißt Benoit B. Mandelbrot. Er ordnete Phänomenen wie der Kochkurve, die bisher nur mathematische Monster genannt wurden, den Namen Fraktal zu, weshalb Mandelbrots Definition von Fraktalen an dieser Stelle erwähnt wird. [12] Er bezeichnet ein Fraktal als „eine Menge, deren Hausdorff-Besicovitch Dimension echt deren topologische Dimension übersteigt.“ [8] Die topologische Dimension gibt den Grad der Verzweigungen pro Punkt in einem Objekt an. So ist ein Quadrat topologisch zweidimensional, ein Würfel hingegen dreidimensional. Somit stimmt die topologische Dimension dieser Objekte mit deren Hausdorff-Besicovitch Dimensionen überein. Es handelt sich also um keine Fraktale. Im Gegensatz dazu besitzen die Kochkurve und die Peanokurve eine Hausdorff-Besicovitch Dimension von 1.2618... respektive 2. Da Kurven topologisch gesehen eindimensional sind, handelt es sich hier um Fraktale. Anhand der Peanokurve sieht man, dass fraktale Objekte auch ganzzahlige Dimensionswerte aufweisen können. [8]

Ein Beispiel für eine Peanokurve mit Dimension 2 ist in Abbildung 2 zu sehen. Daraus ersieht man, dass die Kurve durch die unendlichfache Anwendung einer Vorschrift eine Fläche vollkommen ausfüllt. [14]

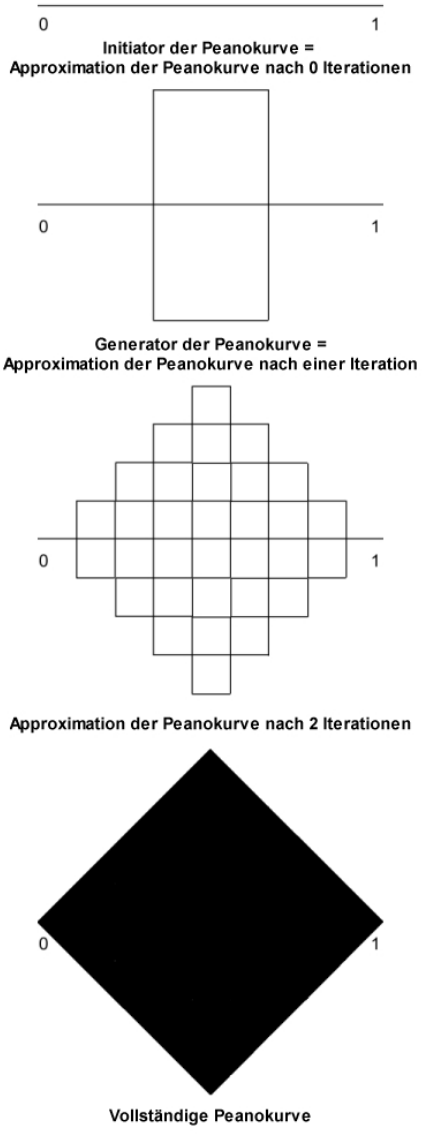

Abbildung 2: Peanokurve

2 Fraktale Brownsche Bewegung

Für die weiteren Untersuchungen ist eine Beobachtung von Robert Brown zu besprechen. Er entdeckte, dass kleinste Teilchen, die in einer Flüssigkeit schweben, ungeordnete Zickzackbewegungen vollführen. Diese Bewegung wird daher auch Brownsche Bewegung (Brownian Motion) genannt. Wenn man die Bewegung eines Teilchens im Raum aufzeichnet und nur eine Dimension betrachtet, so erhält man eine Funktion über die Zeit, welche in Abbildung 3 ersichtlich ist. Dabei stellt die horizontale Achse die Zeit, die vertikale den Zustand der Bewegung dar. Die Analyse dieser Funktion ergab, dass die Brownsche Bewegung statis tisch selbstähnlich ist und eine fraktale Dimension von 1.5 aufweist. [13]

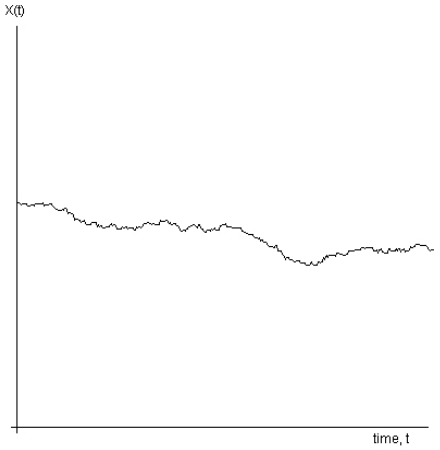

Abbildung 3: Brownian Motion

B. Mandelbrot und J. van Ness verallgemeinerten die Brownsche Molekular Bewegung, in dem sie die fraktale Brownsche Bewegung (fractional Brownian Motion, fBM) definierten. Damit charakterisierten sie eine Familie von stochastischen Prozessen, die auf der Normalverteilung basieren. Es war nun möglich die 1.5-dimensionale Brownsche Bewegung mit jeder beliebigen Dimension zwischen 1 und 2 zu simulieren. Bei diesen Untersuchungen entdeckte B. Mandelbrot die Ähnlichkeit zwischen der generierten Funktion mit fraktaler Dimension 1.2 und der Skyline eines Berges. Es lag somit nahe, dass man die fBM um eine zweite Dimension erweitert, um Terrains zu generieren. Bevor nun aber explizit auf Algorithmen zur Terrainerstellung eingegangen wird, wollen wir an dieser Stelle den eindimensionalen Fall besprechen. [6] [10]

Um die Eigenschaften der fBM zu erörtern, wird die fraktale Brownsche Funktion X(t) herangezogen. Da die Variable t die Zeit repräsentiert, kann man X(t) als Funktion auffassen, die jedem Zeitpunkt einen Wert zuordnet. Die Differenzen X(t2) – X(t1) sind normalverteilt und besitzen eine Varianz VAR, die direkt proportional zu der Zeitdifferenz zur Potenz 2H ist. Formal ausgedrückt bedeutet dies:

$$(1) \quad VAR = <| X(t_2) - X(t_1) |^2> \propto | t_2 - t_1 |^{2H}$$

wobei < und > den geschätzten Wert über viele Stichproben von X(t) kennzeichnet und H ein Parameter ist, dessen Wert im Intervall [0,1] liegt. Die Funktion X(t) ist sowohl stationär als auch isotropisch, was bedeutet, dass die Funktion in alle Richtungen dieselben Eigenschaften besitzt. Auch der Mittelwert der quadratischen Differenzen ist nur von der Zeitdifferenz $|t_2-t_1|$ abhängig, wobei alle Werte von t statistisch äquivalent, also gleichwahrscheinlich sind. Die Parameterfestlegung H = 0.5 beschreibt die herkömmliche Brownsche Bewegung mit der Varianz

$$VAR \propto | \Delta t | .$$

Mit H kann man also die Streuung der Funktionswerte in einem bestimmten Intervall steuern. Damit geht aber auch die Kontrolle der fraktalen Dimension D einher. Im Allgemeinen berechnet man D aus der Formel D = E + 1 – H, wobei E die euklidische Dimension eines Objektes repräsentiert. Für eine Kurve wie der eindimensionalen fBM, deren euklidische Dimension 1 ist, ergibt dies die Formel D = 2 – H. Die Auswirkungen der verschiedenen Werte von H werden im Anschluss durch ein Beispiel demonstriert.

Die zufälligen Werteverschiebungen von X(t) mit Zeitdifferenz $|t_2-t_1| = 1$ werden ermittelt, in dem man die durchschnittliche quadratische Inkrement auf σ^2 setzt.

$$(2) \quad VAR = <| X(t_2) - X(t_1) |^2> = | \Delta t |^{2H} \sigma^2$$

Setzt man σ auf 1 so wird Gleichung (1) erfüllt. Höhere Werte für σ haben rauere Funktionen zur Folge. Nun kann mit der Generierung der Funktion begonnen werden. Sei X(0) = 0, so sind die Punkte der Funktion an den Stellen t = ±1 Stichproben einer normalverteilten Zufallsvariable mit Varianz σ^2 und Mittelwert 0, da die Zeitdifferenz in diesem Fall genau 1 beträgt. Unter dieser Voraussetzung können die Funktionswerte für t = ±0.5 wie folgt ermittelt werden:

$$X(\pm 1/2) = 0.5(X(0) + X(\pm 1)) + \Delta_1,$$

wobei Δ_1 eine gaußsche Zufallsvariable mit Mittelwert 0 und einer Varianz von

$$\mu_1^2 = \frac{\sigma^2}{2^{2H}} - \frac{\sigma^2}{4} = \frac{\sigma^2}{2^{2H}}\left(1 - 2^{2H-2}\right)$$

ist. Diese Varianz errechnet sich wie folgt: Δt beträgt nun 0.5. Durch Einsetzen in Formel (2) ergibt sich mit

$$\left(\frac{1}{2}\right)^{2H} * \sigma^2$$

der 1. Teil der Gleichung. $\sigma^2/4$ wird abgezogen, da die Varianz entsprechend der Anwendung des Berechnungsschrittes auf die halbe Streckenlänge reduziert werden muss[3]. An dieser Stelle kann man auch die Kontrollfunktion von H bezüglich der fraktalen Dimension ersehen. Wenn H gegen 1 strebt, so wird die Varianz gleich 0, es werden immer exakt die Mittelwerte der gegebenen Funktionswerte für die Zwischenstellen berechnet und die Funktion hat eine fraktale Dimension von 1, da sie aus Geradenstücken besteht. Wird H sehr klein – nahe bei Null – gewählt, wird die Varianz beim ersten Iterationsschritt zwar um ein Viertel verringert, sie bleibt aber anschließend bei den weiteren Iterationen konstant bei diesem Wert, was wilde Schwankungen in den Funktionswerten erzeugt. Man erhält so eine sehr raue, an manchen Stellen flächenfüllende Funktion mit einer fraktalen Dimension knapp bei 2.

Wenn man mit dem Algorithmus fortfahrt und die Werte der Funktion für t = ±0.25 ermittelt, so wird wieder über die angrenzenden schon bestehenden Funktionswerte gemittelt und eine Zufallsvariable Δ_2 mit einbezogen.

$$X(\pm 1/4) = 0.5(X(0) + X(\pm 1/2)) + \Delta_2$$

$\Delta 2$ besitzt einen Mittelwert von 0 und eine Varianz von

$$\mu_2^2 = \frac{\sigma^2}{4^{2H}}\left(1 - 2^{2H-2}\right),$$

wobei nun für ein Viertel der Strecke der Wert $\sigma^2/8$ abgezogen werden muss. Wenn man bei der n-ten Iteration des Algorithmus angekommen ist, so haben sich die Streckenstücke zu einer Länge von 0.5^n verkürzt. Die normalverteilte Zufallsvariable Δ_n, die bei der n-1-ten Berechnungsstufe benötigt wird, besitzt eine Varianz von

$$(3) \quad \mu_n^2 = \frac{\sigma^2}{2^{2nH}}\left(1 - 2^{2H-2}\right).$$

[3] Wenn die Strecke halbiert wird, so muss auch die Streuung bei der Berechnung der Varianz halbiert werden. Dies führt dazu, dass die Varianz durch 2^2 dividiert werden muss. Dies zeigt folgende Gleichung mit dem allgemeinen Skalierungsfaktor 1/s.

$$\frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{x_i - \overline{x}}{s}\right)^2 =$$

$$= \frac{1}{N-1}\sum_{i=1}^{N}\frac{1}{s^2}(x_i - \overline{x})^2 = \frac{VAR}{s^2}$$

Um mehr Details zu erzeugen muss lediglich die Anzahl der Iterationen erhöht werden. [13] [5] Das bedeutet wiederum, dass jeder beliebig kleine Ausschnitt der Kurve genau so viele Details beinhalten kann wie die gesamte Funktion. Somit unterscheidet sich die fraktale Brownsche Bewegung von algebraisch beschreibbaren Funktionen, deren Ausschnitte nicht so viele Unregelmäßigkeiten wie ihr ganzes aufweisen. Doch um sagen zu können, dass fBM invariant bezüglich der Skalierung ist, muss sie eine Form von Selbstähnlichkeit besitzen. Im Gegensatz zu den exakt oder optisch selbstähnlichen Fraktalen spiegelt jeder Ausschnitt der fBM-Kurve die statistischen Eigenschaften der gesamten Kurve wider. Optisch unterscheiden sich die einzelnen Abschnitte allerdings vom Ganzen. fBM besitzt also eine statistische Selbstähnlichkeit, die in Abbildung 4 in einem Vergleich mit der Kochkurve erörtert wird.

Im ersten Teil von Abbildung 4 wird die erste Hälfte der Kochkurve ausgewählt und um den Faktor 3 (Reziprokwert des Skalierungsfaktors 1/3 der Kochkurve) vertikal und horizontal vergrößert. Man kann erkennen, dass das erste Teil des vergrößerten Ausschnitts deckungsgleich mit der Kochkurve ist. Würde man nun das erste Viertel der fBM-Kurve im zweiten Teil von Abbildung 4 auswählen und auf die Länge der ursprünglichen Kurve analog zur Kochkurve aufskalieren, in dem man die Werte der horizontalen und vertikalen Achse mit 4 multipliziert, so würde dieser Ausschnitt nicht mehr dieselben statistischen Eigenschaften wie jene der gesamten Kurve aufweisen. Diese falsche Skalierung ist im dritten Teil von Abbildung 4 dargestellt. Bei einer richtigen Skalierung eines Ausschnittes muss die vertikale Achse mit einem anderen Faktor vergrößert werden. Dieser ist zum einen vom Vergrößerungsfaktor S der horizontalen Achse und zum anderen vom Parameter H der fBM-Kurve und somit indirekt von deren fraktalen Dimension abhängig. Er errechnet sich gemäß der Formel S^H. In unserem Beispiel wäre der vertikale Vergrößerungsfaktor $4^{0.5}$, was in der letzten Grafik in Abbildung 4 zu sehen ist. Die unterschiedliche Skalierung der Achsen wird Selbstaffinität genannt und erhält die statistische Selbstähnlichkeit der Kurve. [13] Die Begründung dafür liegt in Gleichung (1), die das Verhältnis zwischen dem Zeitintervall und der Varianz der fBM-Kurve definiert.

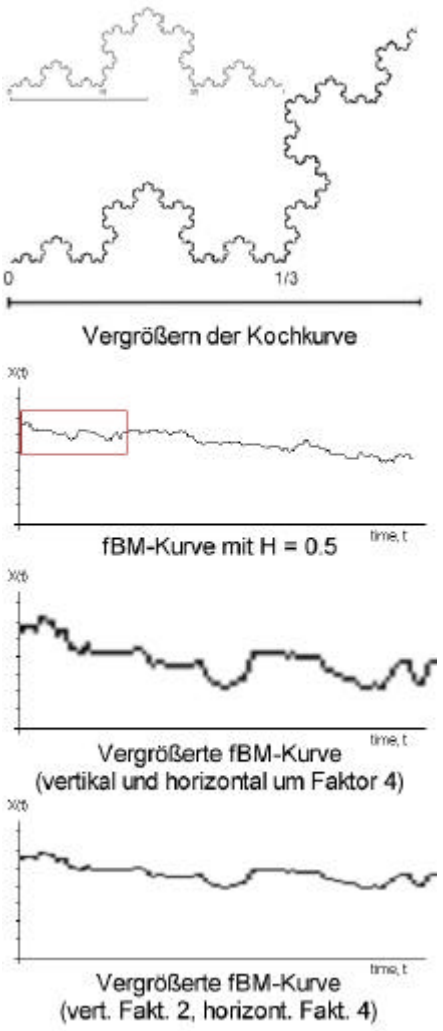

Abbildung 4: Selbstaffinität von fBM

Zum Abschluss der Ausführungen über fBM sollen mit Hilfe zweier Bilder einige Eigenschaften dieser Funktion demonstriert werden.

In Abbildung 5 sind 3 fBM-Kurven mit gleichem Verlauf aber mit verschiedenen Werten von H ersichtlich. Bei all diesen Funktionsverläufen ergänzen sich H und D zu 2. Dabei ist zu erkennen, dass je kleiner man H wählt, die fraktale Dimension D umso größer wird, was sich in der Rauheit der Kurve widerspiegelt. Jene Kurve mit D = 1.8 füllt schon stellenweise Flächen aus. [13] Die Begründung dafür lässt sich aus Formel (3) ableiten. Je kleiner der Parameter H ist, umso invarianter ist die Varianz bezüglich der Verkürzung der Strecken. In dieser Abbildung ist allerdings nicht zu sehen, dass bei größerem H der Bereich in dem die gesamte Funktion schwankt wesentlich größer ist als bei niedrigem H, was sich aus der Formel (2) ersehen lässt. Dies ist in Abbildung 6 dargestellt. (Für die Erstellung der Grafik wurde eine einheitliche Skalierung der vertikalen Achse verwendet. Diese ist allerdings intuitiv gewählt worden und erfüllt Formel (2) nicht. Es werden nur die verschiedenen Bandbreiten der Funktionen veranschaulicht.)

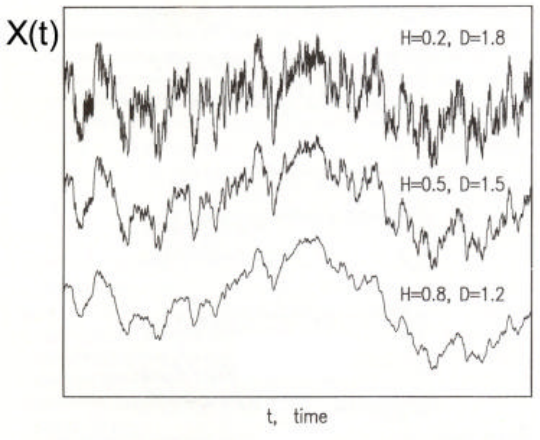

Abbildung 5: Fraktale Dimensionen von fBM

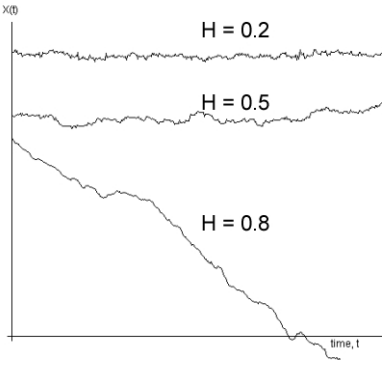

Abbildung 6: Bandbreiten von fBM

3 Terrain-Generierungs-Algorithmen

Um ein Terrain zu generieren, muss zunächst festgelegt werden, wie ein Gelände im Computer repräsentiert wird. In der Computergraphik verwendet man im Allgemeinen so genannte Höhenfelder. Diese Höhenfelder sind zweidimensionale Arrays, in deren Einträgen Höhenwerte gespeichert sind. Es wird also für jeden Punkt in einer Ebene eine Höhenangabe gespeichert. Das kann auch als zweidimensionale Funktion f(x,y) aufgefasst werden, wobei x, y die Koordinaten in der Ebene beschreiben und f(x,y) diesen Koordinaten genau einen Höhenwert zuweist. Da pro Arrayeintrag nicht mehrere Werte gespeichert werden, sind Höhlen oder Überhänge nicht modellierbar. Abschließend ist noch zu erwähnen, dass verschiedene Dateiformate für Höhenfelder etabliert wurden. Beispiele dafür sind DEM (digital elevation map), das von U.S. Geological Survey (USGS) entwickelt wurde, und DTED (digital terrain elevation data), das vom Militär der USA verwendet wird. [13] [10]

Bei den folgenden Terrain-Generierungs-Algorithmen handelt es sich, um Methoden, welche zunächst konzipiert waren, um die eindimensionale fBM erzeugen, und anschließend auf den zweidimensionalen Fall ausgedehnt wurden. Im Rahmen dieser Ausarbeitung werden der Zerteilungs-Algorithmus, die Spektralanalyse und der Midpoint Displacement Algorithmus besprochen.

3.1 Zerteilungs-Algorithmus

Die erste Simulation von fBM basierte auf dem Konzept der „unabhängigen Beunruhigung“ einer Funktion. Die Brownsche Molekularbewegung kann als die Summe von unabhängigen Impulsen angesehen werden, die auf ein Teilchen, das in einer Flüssigkeit schwimmt, einwirken. Jede Störung beeinflusst die Bewegung des Teilchens und das Auftreten dieser Störungen scheint zufällig zu sein, da das Teilchen einen sich ständig ändernden, nicht vorhersagbaren Bewegungszustand besitzt. Dies kann nun dadurch simuliert werden, dass man zu Beginn eines iterativen Algorithmus eine lineare Funktion F(t) mit der Steigung 0 einzeichnet, wobei t die Zeit beschreibt. Anschließend wird mit einem Zufallsgenerator, der Poisson verteilte Zufallszahlen generiert, ein beliebiger Zeitpunkt t_i ausgewählt. Alle Funktionswerte für $t > t_i$ werden um 1 erhöht, die restlichen Funktionswerte bleiben unverändert. Diese Prozedur wird nun so oft wiederholt, bis man optisch die Form einer fBM-Kurve hinreichend genau approximiert hat. [12]

Bei der Erweiterung dieser Methode auf das Höhenfeld wird das Koordinatennetz durch eine zufällige Linie in zwei Hälften geteilt. Der Höhenwert eines Teils wird um eins erhöht, der des anderen Teils erniedrigt. Durch oftmaliges Wiederholen dieser Prozedur entsteht beim Betrachter der Eindruck, dass die Erhebungen eine real existierende Landschaft darstellen.

Der Nachteil dieses Algorithmus liegt darin, dass man die beschriebene Erzeugungsvorschrift unbegrenzt oft anwenden kann, was zu einem theoretisch unbegrenzten Rechenaufwand führt. Erfahrungswerten zufolge sind nach 100 bis 200 Iterationen Ansätze einer Landschaft erkennbar. Gute Ergebnisse sind aber erst ab 500 Durchläufen zu erwarten.

Ein mögliches Abbruchkriterium kann man aber aus der Tatsache ableiten, dass wenn sich der Höhenwert an einem Koordinatenpunkt von den meisten Höhenwerten seiner Nachbarpunkte unterscheidet, eine weitere Iteration wenig am Erscheinungsbild des Höhenfelds ändert. Somit könnte man den Algorithmus terminieren lassen, wenn sich 50-60 % der Nachbareinträge jeder Arrayadresse vom aktuellen Eintrag unterscheiden. Allerdings ist der Aufwand der Überprüfung dieses Abbruchskriterium viel zu hoch und steigt quadratisch mit der Seitenlänge des Höhenfelds. Außerdem kann aufgrund der rein zufälligen Wahl der Trennungslinien nicht garantiert werden, dass dieses Abbruchskriterium jemals erfüllt wird.

Ein weiterer Umstand, der die Rechenzeit erhöht, ist dass bei jedem Durchlauf alle Höhenwerte geändert werden. Das bedeutet, dass auf jeden Arrayeintrag bei jeder Iteration zugegriffen wird.

Die fraktale Dimension der Landschaft beträgt vor dem ersten Zerlegungsschritt 2.0, sofern das Koordinatennetz nicht mit verschiedenen Höhenwerten vorbelegt wurde. Durch die wiederholte Anwendung der Zerteilungsprozedur nähert sich die fraktale Dimension immer mehr dem Wert 2.5. Sie wird aber nicht größer als 2.5. Abbildung 7 zeigt Ergebnisse dieser Methode nach verschiedenen Iterationsschritten. [13]

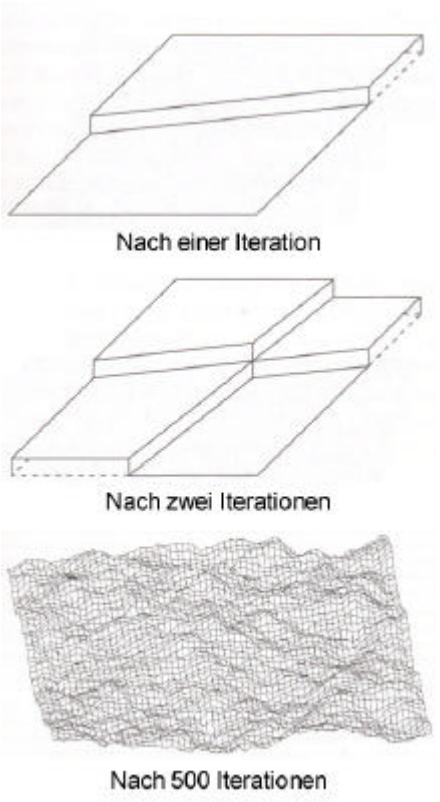

Abbildung 7: Zerteilungs-Algorithmus

Ein Vorteil dieses Prinzips ist es, dass es leicht auf andere euklidische Objekte wie Zylinder oder Kugeln übertragen werden kann. Bei Kugeln wird mit Hilfe des Zufallsgenerators eine Koordinate r_i auf der Einheitskugel gewählt. Der Höhenwert aller Positionen r, für die $r*r_i > 0$ gilt, wird um 1 erhöht. So können Planeten mit einem Oberflächenprofil erstellt werden.

Grundsätzlich wird mit dieser Methode immer ein fraktales Objekt mit dem Parameterwert H = 0.5 entstehen, da das physikalische Prinzip der Brownschen Molekularbewegung simuliert wird. Allerdings kann man die fraktale Dimension

beeinflussen, in dem man den Wert, um den die Höhenwerte vergrößert werden, ändert. Die Höhen auf der einen Seite der Entscheidungsgrenze werden um $t_i^{H-0.5}$ erhöht, die anderen um diesen Wert verringert. [12]

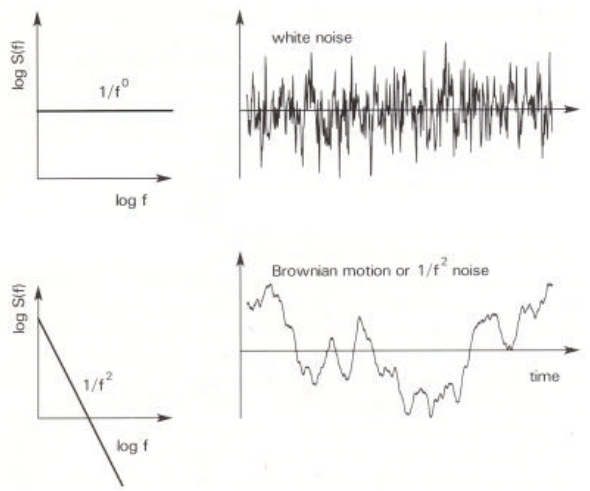

Abbildung 8: Spektrale Dichtefunktionen

3.2 Spektralsynthese

Dies ist eine weitere Methode, die das Prinzip der fBM zu simuliert. Dafür ist allerdings das Spektrum an mathematisch beschreibbaren Frequenzen zu untersuchen, die notwendig sind, um fBM-Kurven anzunähern oder darzustellen. Um eine fBM-Funktion exakt zu erzeugen, müsste man unendlich viele Funktionen mit jeweils verschiedenen Schwingungsfrequenzen erzeugen, sie entsprechend gewichten und anschließend kombinieren. Für den praktischen Gebrauch entsteht aber ein Abbruchfehler, da nur begrenzt viele Funktionen mit limitierten Frequenzen in die Berechnung eingehen können. Dies stört aber nicht weiter, da bisher für die Darstellung jedes Fraktals der Iterationsprozess abgebrochen wurde, nachdem eine gewünschte Genauigkeitsvorgabe erfüllt worden ist. In Abbildung 8 ist eine Gegenüberstellung von fBM-Funktionen in der rechten und deren spektralen Dichtefunktion in der linken Spalte zu sehen. Die erste Funktion wird als White Noise bezeichnet. Diese Kurve könnte durch eine Pseudo-Zufallszahlengenerator, der für jeden Zeitpunkt eine Zahl ausgibt, approximiert werden. Jeder Funktionswert des White Noise ist unkorreliert mit seinen Nachbarwerten und kann daher durch eine fBM-Kurve mit H = 0 nur angenähert werden. Das Frequenzspektrum ist dargestellt, in dem in logarithmischen Skalen die Frequenzen auf der x-Achse und die spektrale Dichtefunktion auf der y-Achse dargestellt sind. Das Diagramm zeigt, dass für die Erstellung einer White Noise-Funktion alle Frequenzen des Spektrums gleich stark in die Berechnung eingehen müssen. Im Gegensatz dazu werden für die Brownsche Molekularbewegung hauptsächlich niedrige Frequenzen benötigt. Die Abnahme des Einflusses der höheren Frequenzen kann hier mit der Formel $1/f^2$ beschrieben werden. Um aber auch hier die fraktale Dimension D mit dem Parameter H zu steuern, verwendet man für die Spektralsynthese die Formel $1/f^{ß}$, wobei die Gleichung

$$D = E + 1 - H = E + \frac{3 - \beta}{2}$$

gilt. Daraus folgt die Relation

$$H = \frac{\beta - 1}{2},$$

die für die Brownsche Molekular Bewegung mit H = 0.5 die Bedingung ß = 2 erfüllt. ß nimmt somit Werte zwischen 3 (für H = 1) und 1 (für H = 0) an. Je größer also ß wird, umso kleiner ist die fraktale Dimension D.
Für die Generierung einer fBM-Kurve wird nun nach einer zufälligen Gewichtung A_k von Cosinus-Schwingungen und B_k von Sinusschwingung im Frequenzraum gesucht. Die zufällige Wahl der Koeffizienten A_k und B_k wird durch die Bedingung $1/f^{ß}$ eingeschränkt, so dass die Frequenzen entsprechend der spektralen Dichtefunktion einer gewünschten fBM-Kurve in die Berechnung eingehen. Abschließend werden die ausgewählten Frequenzen in den Ortsraum mit Hilfe der diskreten inversen Fast-Fourier-Transformation zurückgerechnet. Die so entstehende Funktion X(t) entspricht der Summe der gewichteten Sinus- und Cosinusfunktionen. Formal bedeutet dies:

$$X(t) = \sum_{k=1}^{N/2} (A_k * \cos kt + B_k * \sin kt)$$

Die Zeit wird von der Variable t repräsentiert. N gibt die maximale Frequenz an und limitiert somit die Laufvariable k, die alle Frequenzen in dem vorgegebenen Bereich durchläuft.

Bei der Erweiterung dieser Methode auf ein Höhenfeld müssen N^2 Cosinus- und Sinusschwingungskoeffizienten ermittelt werden, da nun die zweidimensionale diskrete inverse Fourier-Transformation zur Anwendung kommt. Es muss also jetzt eine zusätzliche Koordinatenachse berücksichtigt werden, wenn die Zufallswerte für die Gewichtskoeffizienten im Frequenzraum festgelegt werden. Ebenso ist das Entscheidungskriterium, das angibt welche Frequenzen verstärkt berücksichtigt werden, um eine Dimension zu erhöhen. Die spektrale Dichtefunktion hängt nun nicht mehr von einem sondern von zwei Parametern k und l ab. Die einschränkende Bedingung lautet daher $(l^2 + k^2)^{-(H+1)}$. [12]

Im Gegensatz zu den bisher besprochenen fraktalen Generierungsmethoden ist die Spektralsynthese nicht iterativ. Es wird in einem durchgehenden Prozess ein Landschaftsprofil erstellt, weshalb auch nicht nach Bedarf zusätzliche Details eingefügt werden können. Man kann allerdings durch die Erhöhung des Frequenzbereichs mehr Einzelheiten im Relief erzeugen. Dafür muss allerdings die Berechnung der Oberfläche von Beginn an gestartet werden. Weitere Nachteile dieser Methode sind die Tatsachen, dass kaum Vorgaben gemacht werden können, um das Landschaftsprofil zu beeinflussen, und dass der Rechenaufwand höher als jener des Zerteilungs-Algorithmus ist. Die Methode liefert aber sehr realitätsnahe Bilder, wie sie in Abbildung 9 exemplarisch dargestellt sind.

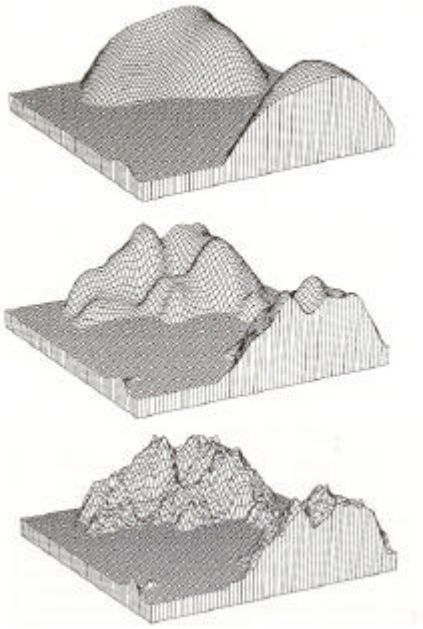

Abbildung 9: Spektrale Synthese

3.3 Midpoint Displacement

Die am meisten verwendete Methode, um eine fBM-Kurve zu generieren, ist der Mittelpunktverschiebungs-Algorithmus oder auch Midpoint Displacement genannt. Dabei wird eine gerade Strecke betrachtet, deren Endpunkte eindeutig definiert sind. Anschließend wird der Mittelpunkt zwischen den beiden Punkten berechnet und um einen normalverteilten Zufallswert mit Mittelwert 0 verschoben. Dieser Vorgang wird mit den so entstehenden Teilstrecken wiederholt, wobei die Varianz der Zufallsvariablen der Streckenlänge angepasst wird. [13] Diese Anpassung erfolgt gemäß der Beschreibung in Kapitel 2, wobei die fraktale Dimension wieder durch den Parameter H gesteuert wird. Die eindimensionale Generierung der fBM mit diesem Algorithmus ist in Abbildung 10 zu sehen.

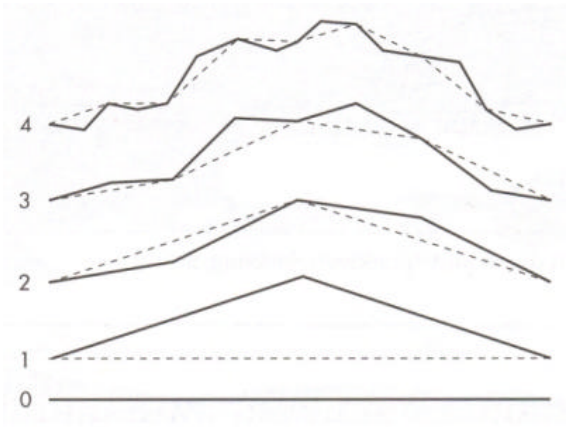

Abbildung 10: 1D Midpoint Displacement

Bei der Erweiterung des Mittelpunktverschiebungs-Algorithmus auf das zweidimensionale Höhenfeld, das für diese Methode die Form eines Quadrats mit der Seitenlänge $2^n + 1$ aufweisen muss, werden zunächst die Höhenwerte der Eckeinträge des Höhenfelds beliebig gesetzt. Anschließend berechnet man den Zentrumseintrag des zweidimensionalen Arrays, in dem man die Höhenwerte der Ecken mittelt und mit einer normalverteilten Zufallszahl mit Mittelwert 0 addiert. Im nächsten Schritt werden die Höhenwerte in der Mitte der Ränder analog unter Bezugnahme der angrenzenden Ecken und des Mittelpunktes des Höhenfelds ermittelt. Somit sind neue kleinere Quadrate entstanden, deren Eckhöhenwerte nun schon vorliegen. Der Algorithmus wird daher rekursiv auf jedes dieser Teilquadrate angewandt, wobei die Varianz der Zufallsvariablen entsprechend verringert wird. In Abbildung 11 sieht man die einzelnen Schritte dieses Algorithmus nochmals für einen Doppelarray mit einer Seitenlänge von 5, was zugleich das kleinste Höhenfeld ist, das alle Erzeugungsschritte benötigt, skizziert. Man kann aus dieser Grafik auch erkennen, dass das Höhenfeld die besondere quadratische Form besitzen muss, damit alle Einträge errechnet werden können. Man nennt diesen Füllalgorithmus auch Diamond-Sqare-Algorithm, da abwechselnd Karo- und Quadratmuster durch die bereits errechneten Werte auftreten.

Die Methode terminiert, sobald die gewünschte Genauigkeit erreicht worden ist. Sollten mehr Details benötigt werden, so kann der Algorithmus fortgesetzt werden. Der Detailreichtum ist allerdings nur solange erweiterbar bis jeder Position des Höhenfelds ein Wert zugewiesen worden ist. Überschreitet die Größe des Höhenprofils die Auflösung des Bildschirms, so muss lediglich jener Teil des Reliefs, der angezeigt wird, bis zur gewünschten Tiefe durchiteriert werden. Dafür muss der Algorithmus auf das gesamte Höhenfeld solange angewendet werden, bis die Höhenwerte der Ecken des relevanten Bereichs feststehen.

Anschließend können die nicht benötigten Teilquadrate vernachlässigt werden. [6] [13]

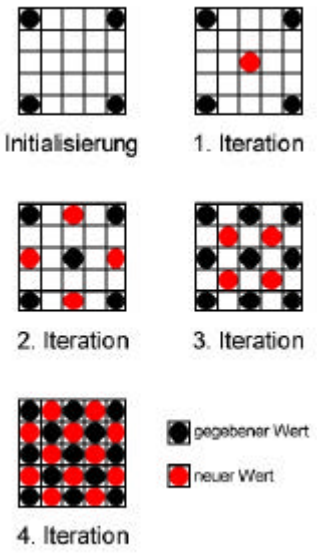

Abbildung 11: Diamond-Sqare-Algorithm

Für jeden Eintrag werden maximal 3 Additionen, eine Division und ein Aufruf einer Funktion, die normalverteilte Pseudo-Zufallszahlen generiert, benötigt. Wenn ein Höhenwert ermittelt worden ist, so wird dieser auch nicht mehr verändert im Gegensatz zum Zerteilungs-Algorithmus, was eine große Aufwandsminderung darstellt. Abgesehen von diesen Vorteilen bezüglich der Rechenzeit kann es zu Artefakten aufgrund unnatürlich großer Höhenunterschiede kommen, was beispielsweise bei der Spektralanalyse nicht auftritt. [6] In Abbildung 12 ist ein Geländebeispiel zu sehen, das mit Midpoint Displacement erstellt wurde. Gebiete deren Höhenwerte kleiner als Null sind werden auf Null zurückgesetzt und man kann die als Oberfläche eines Gewässers betrachten.

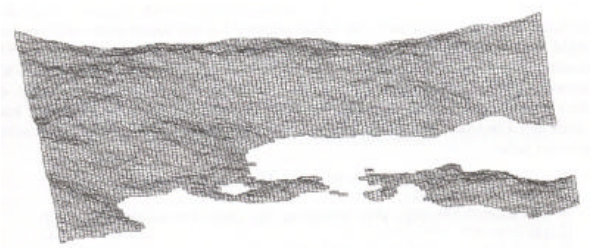

Abbildung 12: 2D Midpoint Displacement

4 Multifraktale

Bisher wurden in dieser Ausarbeitung lediglich Objekte mit einer einheitlichen fraktalen Dimension, so genannte Monofraktale, erwähnt. Alle besprochenen stochastischen Fraktale wie fBM und die erstellten Geländemodelle sind statistisch gesehen homogen und isotropisch. Homogen bedeutet, dass sie an jeder Stelle dieselben statistischen Eigenschaften besitzen, während isotropisch darauf hinweist, dass in allen Richtungen dieselben Charakteristika aufzufinden sind. So weisen beispielsweise jene Terrains die via Midpoint Displacement generiert wurden, an jeder Stelle im Höhenfeld eine Dimension von 3-H auf.

Natürliche Objekte sind im Allgemeinen nicht so simpel aufgebaut. Sie besitzen verschiedene fraktale Dimensionen, die von der Skalierung des Objekts abhängig sind. Wenn man die Erde vom Weltall aus betrachtet, so gleicht sie einer blauen Kugel. Beobachtet man aber die Erdoberfläche genauer, erkennt man Hügellandschaften oder Gebirge, die eine bestimmte fraktale Dimension besitzen. Konzentriert man sich auf eine flache Felswand eines Berges, so weist diese eine Dimension, die nahe bei 2 liegt, auf, während der Berg selbst ein sehr raues Relief besitzen kann, weshalb seine fraktale Dimension nahe bei 3 liegt. Fraktale, die unterschiedliche Dimensionen (in Abhängigkeit von der Skalierung) besitzen, nennt man Multifraktale. [9]

Multifraktale werden daher benötigt, um den einheitlichen Charakter von künstlich generierten Terrains aufzubrechen. Wie eingangs bereits im Beispiel angeführt wurde, dient es einer realistischeren Darstellung von Bergen, wenn man lokal die Oberflächeneigenschaften bezüglich deren Rauheit abändert. Natürliche Phänomene wie Erosion führen dazu, dass bestimmte Gebiete eines Gebirges stärker zerfurcht sind als andere. Demzufolge müssten diese Regionen auch andere fraktale Dimensionen aufweisen. Ein weiteres Problem, das aus der einheitlichen Rauheit der Oberflächenmodelle entsteht, ist, dass man keinen Übergang zwischen einer Ebene und einem Gebirge generieren kann. Weitere Beispiele für nicht künstlich erzeugbare Terrains sind Gebirge mit vorgelagerten Hügellandschaften, wie sie die Alpen oder die Rocky Mountains aufweisen, sowie Taleinschnitte in Gebirgszügen, die eine komplett andere Oberflächencharakteristik als die Berge, die sie umgeben. Täler verlaufen flach und steigen an den Rändern nur langsam an. [10] Ein Beispiel für die Generierung von multifraktalen Oberflächen wird im nächsten Abschnitt beschrieben.

5 Erosion

Erosion bezeichnet die Abtragung und Zerstörung der Erdoberfläche durch verschiedenste natürliche Einwirkungen, wie zum Beispiel Wind, Wasser und die sich ständig ändernden Temperaturwerte. Um nun extrem realitätsgetreue Abbildungen der Natur zu erreichen, muss man natürlich auch diesen Effekt in die Bildberechnungen mit einbeziehen. Wirklich

notwendig wird dies allerdings erst bei der Erzeugung von Bildserien und Animationen, die über Jahrhunderte oder Jahrtausende hinweg die Oberflächenveränderungen einer Landschaft simulieren sollen, da die Erosion ein sehr langsam wirkender Prozess ist. Zusätzlich benötigt man sehr viel an Rechenleistung, um wirklich alle Aspekte der Erosion zu berücksichtigen. Schließlich hängt sie von Wettereinflüssen ab, die ja bekanntlich chaotisch verlaufen und deren Berechnung nur stark angenähert werden kann.

In der Modellierungsphase wird zuerst das Höhenfeld des Terrains mit schon bekannten Algorithmen konstruiert und im zweiten Durchlauf schließlich die physikalischen Gesetze der Erosion darauf angewandt. Grundsätzlich kann man dabei vereinfacht zwei Arten der Erosion unterscheiden: erstens die Abtragungen, die vom fließenden Wasser respektive fallendem Regen verursacht werden, und zweitens die temperaturbedingte Witterung, die Material an Steilhängen abbröckeln und am Fuß des Hangs ansammeln lässt. [7]

5.1 Hydraulische Erosion

Die Grundidee besteht darin, dass man auf jedes Vertex im Höhenfeld des Terrains eine bestimmte Menge an Wasser „regnen“ lässt, und in Folge dann auf alle benachbarten niedriger gelegenen Vertices ein Teil des Wassers inklusive ein Teil des Sediments, das durch die Wassereinwirkung abgetragen wurde, abgegeben wird.

Zu einem bestimmten Zeitpunkt t werden einem Vertex mehrere Eigenschaften zugeordnet, die bequem im Höhenfeld des Modells gespeichert werden können:

- die momentane Höhe a_t^v, die zu Beginn durch das Höhenfeld festgelegt wird (das hochgestellte v steht für das aktuelle Vertex)
- die Menge an Wasser w_t^v, die bereits auf dem Vertex vorhanden ist
- die Menge an Sediment s_t^v, die sich in dem Wasser befindet

Für jedes benachbarte Vertex u wird die Menge an Wasser Δw berechnet, die dann an dieses übergeben wird.

$$\Delta w = \min(w_t^v, (w_t^v + a_t^v) - (w_t^u + a_t^u))$$

Falls das Vertex u höher liegen beziehungsweise die Summe der Höhe und des Wassergehalts größer als beim Vertex v sein sollte und somit die zu übergebende Menge an Wasser kleiner oder gleich null ist, dann lagert sich einfach ein Teil des Sediments am Untergrund an und erhöht somit den Wert des Höhenfelds dieses Vertex.

$$a_{t+1}^v = a_t^v + K_d s_t^v$$
$$s_{t+1}^v = (1 - K_d) s_t^v$$

(K_d bezeichnet den Anteil des in einer Einheit Wasser enthaltenen Sediments, der sich an der Oberfläche des Vertex anlagert.)

Falls der obige Fall nicht zutreffen sollte, werden die Wasserwerte für den nächsten Zeitpunkt $t+1$ ganz intuitiv korrigiert, und zwar indem der übergebene Wasseranteil vom Wert des aktuellen Vertex subtrahiert und zu dem des benachbarten Vertex hinzuaddiert wird. Weiters wird mit folgender Gleichung die Menge an Sediment im Wasser berechnet, die vom Vertex v zum Vertex u wandert.

$$c_s = K_c \Delta w$$

(K_c beschreibt die maximale Menge an Sediment, die in einer Einheit Wasser transportiert werden kann)

Aufgrund dieser Berechnung entstehen zwei Möglichkeiten der Fortsetzung. Einerseits kann c_s kleiner sein als s_t^v, was dazu führt, dass dieser Teil des Sediments mit dem Wasser an das benachbarte Vertex übergeben wird. Der zurückgelassene Anteil wird nach Anwendung der Konstante K_d teilweise angelagert, der größere Prozentsatz des Sediments bleibt aber im Wasser enthalten.

$$s_{t+1}^u = s_t^u + c_s$$
$$a_{t+1}^v = a_t^v + K_d (s_t^v - c_s)$$
$$s_{t+1}^v = (1 - K_d)(s_t^v - c_s)$$

Natürlich kann auch der Fall eintreten, dass c_s größer oder gleich der vorhandenen Sedimentmenge s_t^v ist, und genau hier tritt der eigentliche Erosionsprozess in Kraft. Nun wird ein Teil des festen Untergrunds in Sediment verwandelt und im Wasser weitergeschwemmt. Damit das Gleichgewicht respektive die Gesamtmenge an Material erhalten bleiben, muss sich natürlich die Höhe des Vertex v verringern.

$$s_{t+1}^u = s_t^u + s_t^v + K_s (c_s - s_t^v)$$
$$a_{t+1}^v = a_t^v - K_s (c_s - s_t^v)$$
$$s_{t+1}^v = 0$$

(K_s beschreibt die Härte des zuvor angelagerten bzw. bereits von Anfang an vorhandenen Materials, das heißt, diese Konstante könnte man als Maßzahl für die Stärke der Erosion bezeichnen.)

Mit Hilfe dieses Erosionsmodells werden Teile des Erdmaterials von hohen Punkten der Landschaft zu tieferen ebeneren Gebieten transportiert, wo es dann schließlich auf dem schon vorhandenen Untergrund zur Ruhe kommt. Obwohl die oben genannten

Berechnungsgrundlagen die natürlichen Begebenheiten nur sehr grob annähern, kann man damit sehr zufrieden stellende Resultate erzielen. Man kann zusätzlich noch die Tatsache berücksichtigen, dass höher gelegene Regionen wie zum Beispiel Berggipfel eine wesentlich größere Menge des Niederschlags abbekommen als niedrigere. Der Grund dafür liegt in den komplizierten Luft- und Wärmezirkulationen, auf die aber in dieser Arbeit nicht näher eingegangen wird. Um sich diesen Tatsachen zumindest ansatzweise zu nähern, macht man die Menge des Niederschlags von der Höhe des Vertex abhängig. Es ergeben sich dadurch schon wesentlich bessere Resultate, auch wenn man nur eine lineare Abhängigkeit verwendet. [7]

In Abbildung 13 kann man den Effekt der Erosion durch Wassereinwirkung beobachten, wobei allerdings auf die Darstellung des Wassers verzichtet wurde. Man erkennt klar die Furchen und Spuren, die der Regen und der anschließende Wasserfluss an den Berghängen verursacht haben. Für die Erosionskonstanten wurden folgende Werte verwendet: $K_c = 5.0$, $K_s = 0.3$, $K_d = 0.1$. In Intervallen zu 65 Zeitschritten "regnete" es auf jedes Vertex eine bestimmte Menge an Wasser, und zwar ein Tausendstel der Höhe jedes Vertex.

Nagashima [11] untersucht die Bildung eines Flussbetts genauer und entwickelt dafür vereinfachende Algorithmen. Anfangs bahnt sich das Regenwasser seinen Weg von einer höher gelegenen Region in ein Tal hinunter, wobei aufgrund der Naturgesetze schon vorhandene kleine Mulden den Lauf des Wassers bestimmen. Durch die Geschwindigkeit und die daraus resultierende Kraft des Stroms wird das Material am Grund und am unteren Teil der Seitenwände der Mulde abgelagert und weggeschwemmt, das heißt erodiert, weshalb sich das momentan noch kleine Flussbett vertieft. Aufgrund der im nächsten Kapitel besprochenen temperaturbedingten Witterung und des direkt einwirkenden Regens bröckeln die oberen Seitenteile ab und stürzen in den Wasserbach, der das Material gleich mitnimmt. Danach wird wiederum der untere Bereich erodiert, was zur Folge hat, dass sich das Bett auch oben weiterhin verbreitert. Dieser iterative Prozess dehnt die Mulde in kurzer Zeit zu einem großen Flussbett aus, wenn ein ständiger Wasserfluss vorhanden ist. Am Beispiel des Grand Canyon kann man eine stufenförmige Erosion beobachten, die auch in den übrigen Regionen der Erde nicht selten vorkommt. Die Ursache dafür liegt in den unterschiedlichen Härtegraden der Gesteinsschichten. (Abbildung 14)

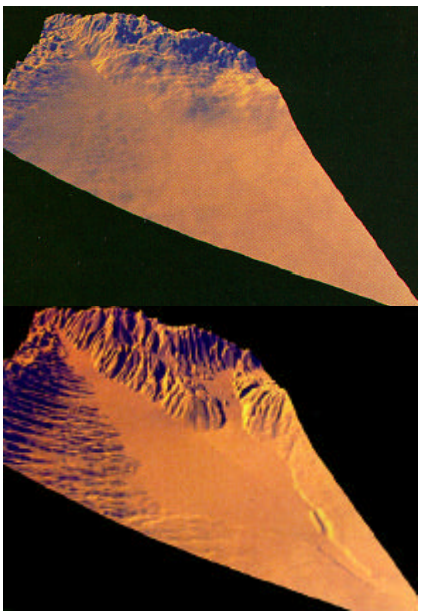

Abbildung 13: Anwendung der hydraulischen Erosion

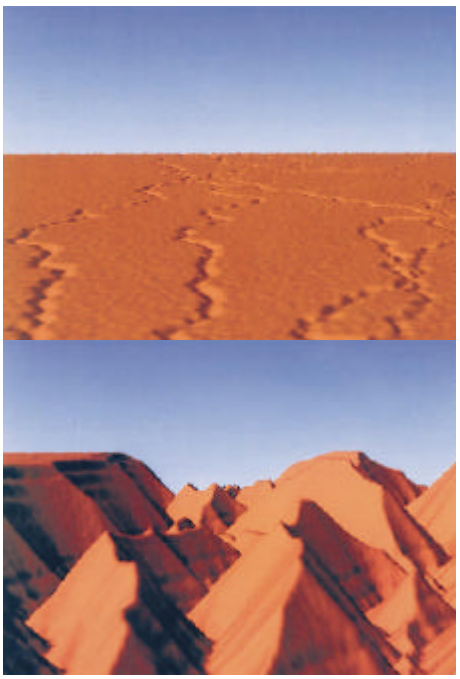

Abbildung 14: Hydraulische Erosion im Flussbett

5.2 Temperaturbedingte Witterung

Diese zweite Art der Erosion behandelt den Prozess des Abbröckelns von Erdmaterial durch jegliche Art von Temperatureinwirkung. Je wärmer die Erdoberfläche wird, desto mehr trocknet sie bei unzureichender Wasserversorgung aus, und durch diesen Volumenverlust wird Erde und Gestein spröder und verliert den Zusammenhalt. Auch starke Temperaturschwankungen zum Beispiel zwischen Tag und Nacht können dieses Phänomen auslösen. Das

gelockerte Material fällt anschließend am steilsten Abhang in seiner Umgebung hinunter und sammelt sich in ebeneren Regionen. [1] Durch diesen Vorgang entstehen Abhänge mit einheitlichem Winkel und zwar mit Hilfe folgender sehr einfach zu implementierenden und schnell berechenbaren Überlegungen.

Man berechnet die Differenz der Höhen zweier benachbarter Vertices und vergleicht diese mit einem vorher festgelegten Böschungswinkel *T*. Ist der Höhenunterschied größer als der Winkel, wird das niedrigere Vertex um einen gewissen Prozentsatz c_t der Differenz erhöht.

$$a_t^v - a_t^u > T : a_{t+1}^u = a_t^u + c_t(a_t^v - a_t^u - T)$$
$$a_t^v - a_t^u \leq T : a_{t+1}^u = a_t^u$$

In Abbildung 15 kann man die Anwendung dieser Art von Erosion deutlich sehen. Es entsteht eine gleichmäßige Landschaft ohne außergewöhnliche Höhen und Tiefen. Die Berggipfel und -grate wurden durch die Abtragungen extrem zugespitzt und auch nahezu alle anderen Rundungen aus dem Originalbild wurden durch scharfe geradlinige Kanten ersetzt. Bei der Simulation von weiten Wüstenregionen liefern diese Berechnungen sehr gute Ergebnisse, da die Sanddünen ja fast vollständig aus losen Sandkörnern bestehen und deshalb genau den Grundbedingungen oben behandelten Witterung entsprechen. [7]

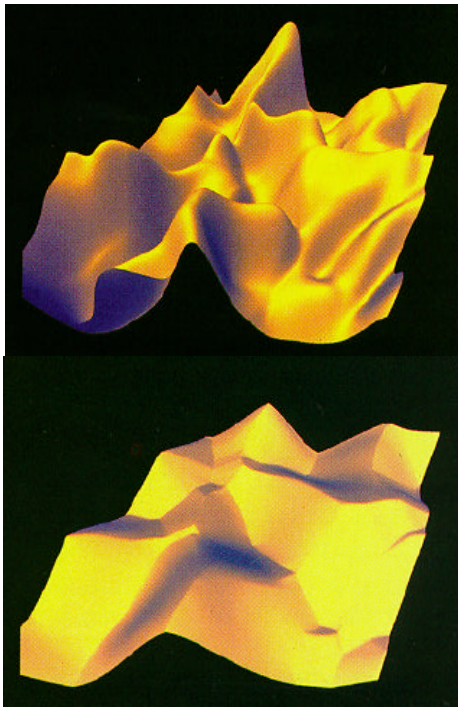

Abbildung 15: Anwendung der temperaturbedingten Witterung

Eine geringfügig differierende Art der Berechnung wird in [1] erklärt und in der Abbildung 16 demonstriert. Der Unterschied liegt darin, dass hier gleichzeitig alle acht benachbarten Vertices betrachtet werden und dadurch die Menge an Material, die an ein bestimmtes Vertex weitergegeben wird, von allen anderen sieben abhängt. Auch hier gibt es wieder einen Böschungswinkel *T*, der sozusagen die Viskosität des Materials beschreibt, denn falls die Höhendifferenz zwischen zwei benachbarten Vertices zu einem Winkel führt, der kleiner als *T* ist, werden keine Materialteile bewegt.

Man bildet zuerst die Menge *A* der Höhen h_i derjenigen benachbarten Vertices, die einerseits niedriger liegen als das Vertex in der Mitte und andererseits die noch stärkere Bedingung des Winkels, der größer als *T* sein muss, erfüllen. Der Abstand zwischen den Vertices, der normalerweise über das gesamte Terrain konstant gehalten wird, wird mit *d* bezeichnet. *h* beschreibt die Höhe des eingeschlossenen Vertex.

$$A = \{h_i \mid h_i - h < 0, (h - h_i)/d > \tan^{-1} T, i = 1,..,8\}$$

Bevor jedes Vertex aus *A* seinen Anteil an Material bekommt, muss noch festgelegt werden, wie viel Volumen ΔS überhaupt zu vergeben ist. Dazu nimmt man die Fläche *a* eines Terrainelements, wobei jeweils ein Element einem Vertex entspricht, und multipliziert sie mit der Hälfte der maximalen Höhendifferenz *H* zwischen dem mittleren Vertex und seinen acht Nachbarn. Es wird deswegen die Hälfte genommen, damit der Algorithmus nicht in einer Endlosschleife landet und ständig Material hin und wieder zurückschiebt. Jedes der Nachbarelemente, die in *A* enthalten sind, erhält nun einen zum Höhenunterschied proportionalen Anteil von ΔS.

$$\Delta S = a\frac{H}{2}$$
$$\Delta S_i = \Delta S \frac{h - h_i}{\sum_{\forall h_k \in A} h - h_k}$$

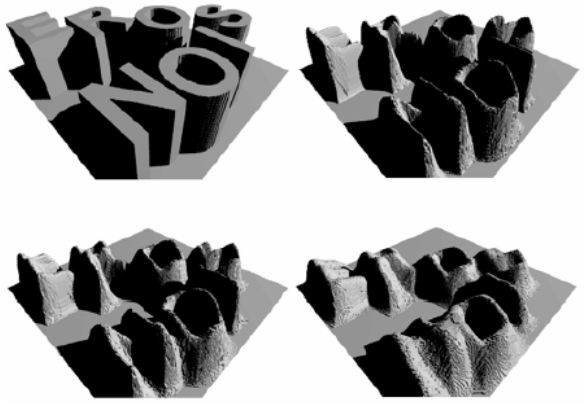

Abbildung 16: Temperaturbedingte Witterung mit Böschungswinkel 45°

5.3 Datenstrukturen

Alle Konstanten, die in den Erosionsmodellen verwendet wurden, müssen natürlich geeignet gewählt werden, um zu realistischen Resultaten zu gelangen. Weiters kann man auch nicht voraussetzen, dass eine zu erstellende Landschaft überall aus genau dem gleichen Material besteht. Deshalb muss eine geeignete Datenstruktur gefunden werden, in der für jedes Vertex die dazugehörigen Materialeigenschaften gespeichert werden. Das Höhenfeld als zweidimensionales Array würde sich für diese Zusatzinformationen anbieten, doch muss man darauf achten, dass dadurch nicht zuviel Speicher verschwendet wird. Die Materialien sind zwar nicht überall gleich, doch zwischen einem Vertex und seinem Nachbarn wird es auch nicht gerade große Differenzen geben. Ein eigenes Array, das nicht so feinkörnig aufgeteilt ist wie das Höhenfeld, eignet sich deshalb in vielen Fällen besser.

Genauso wie sich die Oberflächeneigenschaften in horizontaler Ausdehnung ändern können, differieren die Gesteins- und Erdschichten auch vertikal voneinander. Wenn ein Vertex im Laufe des Erosionsprozesses an Höhe gewinnt oder verliert, kann es durchaus sein, dass eine andere Gesteinsschicht erreicht wird, deren Erosionskonstanten sich von der vorigen drastisch unterscheiden. Eine Möglichkeit, um dieses Fakt im Modell zu implementieren, wäre ein eindimensionales Array, in dem zu jeder Höhe eine Reihe von Materialeigenschaften gespeichert wird. Wenn sich nun die Höhe eines Vertex durch eine Abtragung oder Ansammlung von Gestein ändert, so erfährt man die neuen Untergrundeigenschaften aus dem vorher festgelegten Array. Ein großer Nachteil dabei liegt jedoch darin, dass sich im gesamten generierten Terrain in gleicher Höhe die gleichen Materialien befinden, was nur selten der Realität entspricht. [7]

Ein sehr rechen- und speicherintensiver Ausweg aus dieser Misere besteht in der Verwendung von so genannten Voxels. Bei dieser Datenstruktur ist das gesamte Terrain aus diesen Voxels, das heißt aus dreidimensionalen Würfeln aufgebaut. Für jeden dieser Würfel werden alle zugehörigen Daten, wie zum Beispiel die Art des Materials, aus dem das Voxel aufgebaut ist, und dessen physikalische Eigenschaften, gespeichert. Wählt man nun eine genügend detaillierte Auflösung für die Landschaft, dann erhält man eine perfekte Nachbildung der Natur, in der jedes Sandkörnchen seinen eigenen Charakter hat. Der damit verbundene Speicheraufwand ist natürlich extrem groß und zusätzlich eignen sich die Voxels auch nicht sehr gut für die an den Terrainerzeugungsprozess anschließende Renderingphase, die dadurch sehr rechenintensiv wird. Aus diesen beiden Gründen ist von der Verwendung dieser Datenstruktur stark abzuraten.

In [1] wird eine Struktur entworfen, die ein Mittelding zwischen den zu simplen Höhenfeldern und den aufwendigen Voxels darstellt. Dabei erhält jedes Vertex des zweidimensionalen Terrainarrays ein zusätzliches eindimensionales Array, in dem die verschiedenen vertikal angeordneten Materialschichten des Untergrunds gespeichert sind. Ein Element dieses zusätzlichen Arrays besteht aus mehreren Informationen, wie zum Beispiel die Gesteinsart, die Dichte des Materials, die Menge an enthaltenem Wasser und Gas und vor allem die Dicke respektive die Höhe der aktuellen Schicht. Man kann diese Datenfelder bei Bedarf ohne großen Aufwand erweitern. Um zu vermeiden, dass an vielen Stellen in diesem erweiterten Höhenfeld dieselben Daten, wie die besonderen Charakteristiken eines Gesteins, redundant abgespeichert werden, kann man für diese Art von Informationen eine externe Tabelle heranziehen, in der alle vorkommenden Materialien und deren Eigenschaften aufgelistet werden.

Je dicker die Schichten mit gleichartigem Gestein sind, desto günstiger verhält sich diese Datenstruktur gegenüber den Voxels. Gemäß den schon oft durchgeführten geologischen Messungen kann man davon ausgehen, dass dem in den meisten Fällen auch so ist. In den übrigen Situationen, das heißt wenn die Schichthöhe schon fast der kleinsten Maßeinheit des Landschaftsmodells entspricht, nähert sich der dafür nötige Speicher- und Rechenaufwand schon sehr an den der Voxels an.

Ein großer Vorteil der Voxels besteht in der Möglichkeit, horizontale Löcher oder unterirdische Höhlen in das Terrain zu integrieren. Das war mit dem ursprünglichen Höhenfeld nicht möglich, da zu einem Punkt im 2D-Raster nur eine Höhe gespeichert werden konnte. Mit dem erweiterten Array können nun auch geologische Schichten kreiert werden, die nur aus Luft bestehen, und damit einen Hohlraum beschreiben. Hier muss allerdings beachtet werden, dass aufgrund der Schwerkraft auch Zusammenbrüche dieser Höhlen eintreten können. Auch unterirdische Seen, die das Ergebnis des Erosionsprozesses, sowohl des hydraulischen als auch des thermischen, sicherlich maßgeblich beeinflussen können, sind möglich.

Mit geringen Anpassungen sind die oben genannten grundlegenden Erosionsalgorithmen auch auf diese Datenstruktur anwendbar und es ergeben sich dadurch in der Praxis viel nützlichere Resultate.

5.4 Parallelverarbeitung der Erosionsalgorithmen

In großen Terrains mit einer Auflösung von zum Beispiel 1024x1024 kann die Erosionsberechnung eines einzigen Zeitschritts sehr viel an Rechenzeit in Anspruch nehmen, besonders wenn Rücksicht auf detaillierte Eigenschaften des zugrunde liegenden

Materials genommen wird. Aus diesem Grund besteht genügend Motivation, um sich Gedanken über eine Parallelisierung dieser Algorithmen zu machen.

Benes und Forsbach [2] behandeln genau dieses Thema ausführlich. Der gesamte 2D-Raster, der die Landschaft abdeckt, wird in eine bestimmte Anzahl gleich großer Streifen geteilt. Optimalerweise bekommt dann jede der zur Verfügung stehenden Hardwareressourcen, wie zum Beispiel Prozessoren oder überhaupt eigenständige über ein Netzwerk verbundene Computer, einen Streifen zugeteilt. Das für die Parallelverarbeitung Günstige an den verschiedenen Erosionsalgorithmen ist die Tatsache, dass man für eine Materialbewegung nur das aktuelle Vertex und seine acht Nachbarn benötigt. Deshalb kann jede Ressource seinen Streifen eigenständig berechnen und nur für den Übergang zu benachbarten Streifen muss mit anderen Prozessoren kommuniziert werden. Der Grund, warum für die Aufteilung Streifen gewählt werden, liegt in dem damit verbundenen geringeren Synchronisationsaufwand. Für eine Ressource existieren maximal zwei Kommunikationspartner, das es nur links und rechts Grenzen zu anderen Regionen gibt. Der erste und der letzte Streifen haben sogar jeweils nur einen Nachbarn. Ein Erosionsschritt für einen inneren Streifen sieht folgendermaßen aus:

- Alle Vertices auf diesem Streifen werden mit einem der schon vorher genannten Algorithmen erodiert.
- Die Daten der Materialmengen, die über die linke Grenze hinausbewegt werden müssen, werden in einer Datei abgespeichert.
- Es wird eine Nachricht an die Ressource, die für die linken Nachbarstreifen verantwortlich ist, geschickt, in der mitgeteilt wird, dass die Daten fertig berechnet wurden und ab jetzt zur Verfügung stehen.
- Derselbe Prozess wird für die rechte Grenze durchgeführt.
- Die Ressource wartet auf eine Nachricht des linken Partners.
- Die Übergangsinformationen werden ausgelesen und darauf basierend dann die linke Grenze erodiert.
- Gleiches gilt für die rechte Seite.
- Der nächste Erosionsschritt wird gestartet…

Wichtig ist, dass die Prozessoren zuerst die eigenen Daten abspeichern, die Nachrichten verschicken und danach erst in den Wartezyklus eintreten. Falls diese Reihenfolge nicht eingehalten wird, entsteht eine Deadlock-Situation.

Die Zeit, die benötigt wird, um die Erosion eines Vertex zu errechnen, bleibt durchschnittlich auf das ganze Terrain bezogen konstant. Aus diesem Grund ist es von Vorteil, Ressourcen zu verwenden, die ungefähr die gleiche Leistungsfähigkeit aufweisen, um damit die Wartephasen relativ kurz zu halten. Ein homogenes Mehrprozessorsystem mit gleich schnellen Prozessoren würde sich dafür sehr gut eignen.

Doch aufgrund der Tatsache, dass die Synchronisation und Kommunikation über Nachrichten und Dateien passiert, ist es auch möglich, ein heterogenes Netzwerk mit diesem Algorithmus zu füttern. Auch komplett unterschiedliche Betriebssysteme können mit geeigneter Clientsoftware an dem verteilten Rechnen teilnehmen.

Laufzeittests in [2] haben gezeigt, dass in einem Mehrprozessorsystem mit zehn CPUs der Geschwindigkeitsgewinn bei einem Faktor von 8.4 gegenüber einem Prozessor liegt, das heißt der Kommunikationsoverhead hält sich in Grenzen. Dabei wurde ein Terrain erodiert, das über einen Raster mit 24 Millionen Zellen respektive Vertices verfügt.

Abschließend sei noch kurz erwähnt, dass auch völlig andere Ansätze zur Implementierung der Erosion existieren, wie es zum Beispiel Chiba et al. [4] beweisen. In diesem Paper werden als Grundlagen die physikalischen Gesetze der Kraft eingesetzt. Bei der hydraulischen Erosion besteht das Wasser aus vielen kleinen Partikeln, die mit den sie umgebenden Materialien zusammenstoßen. Mit einer einfach implementierten Kollisionsabfrage wird nun die Kraft berechnet, die auf eine bestimmte Stelle des Terrains ausgeübt wird, und dementsprechend wird dann ein Teil der Landschaft erodiert.

Fazit

Dieses Paper soll einen Überblick über die Möglichkeiten der computerunterstützten Generierung von fraktalen Terrains bieten und deren Grundlagen vermitteln. Dabei wird zuerst mit der allgemeinen fraktalen Geometrie und der brownschen Bewegung eingeleitet und anschließend werden drei der am häufigsten verwendeten Algorithmen zur Terrainerstellung beschrieben. Um den Ansprüchen der Realität noch eher gerecht zu werden, wird mit den Gesetzen der Erosion und den Möglichkeiten zu deren Implementierung abgeschlossen.

Welcher der erwähnten Algorithmen nun tatsächlich am besten geeignet ist, hängt von der vorliegenden Situation respektive Problemstellung ab. Allgemein gilt jedoch der Grundsatz, je realistischer man ein Terrain entwerfen will, desto aufwendiger und leistungshungriger ist der Weg dorthin.

Da hier nur die Ansätze von Lösungsmöglichkeiten näher gebracht werden, reicht es für ein präzises Problem nicht aus, sich nur mit dieser Abhandlung zu beschäftigen. Detailliertere Informationen zu den einzelnen Algorithmen kann man den erwähnten Referenzen entnehmen.

Abbildungen

Abbildung 1: Java-Applet, Stand: 02.05.2003, http://www.jjam.de/Java/Applets/Fraktale/Koch_Kurve.html

Abbildung 2: selbst angefertigt

Abbildung 3: Java-Applet, Stand: 02.05.2003, http://www.astro.utoronto.ca/~mudryk/professional/cupc/fBm3/fBm2.html

Abbildung 4: Java-Applets von Abbildung 1 und 2

Abbildung 5: Peitgen H.-O., Saupe D., *The science of fractal images*, Springer, 1988, p.44

Abbildung 6: Java-Applet, Stand: 02.05.2003, http://www.astro.utoronto.ca/~mudryk/professional/cupc/fBm3/fBm2.html

Abbildung 7: Pfeiffer R., Scholl O., *Natur als fraktale Grafik*, Markt und Technik, 1991, p.63, 65

Abbildung 8: Peitgen H.-O., Saupe D., *The science of fractal images*, Springer, 1988, p.40

Abbildung 9: Peitgen H.-O., Saupe D., *The science of fractal images*, Springer, 1988, p.106, 107

Abbildung 10: Pfeiffer R., Scholl O., *Natur als fraktale Grafik*, Markt und Technik, 1991, p.67

Abbildung 11: selbst angefertigt

Abbildung 12: Pfeiffer R., Scholl O., *Natur als fraktale Grafik*, Markt und Technik, 1991, p.69

Abbildung 13: Kolb C., Mace R., Musgrave K., *The Synthesis and Rendering of Eroded Fractal Terrains*, ACM Computer Graphics: Proceedings SIGGRAPH, Volume 23, Number 3, 1989, p.48,49

Abbildung 14: Nagashima K.: *Computer generation of eroded valley and mountain terrains*, The Visual Computer, Volume 13, Number 9-10, 1998, p.462

Abbildung 15: Kolb C., Mace R., Musgrave K., *The Synthesis and Rendering of Eroded Fractal Terrains*, ACM Computer Graphics: Proceedings SIGGRAPH, Volume 23, Number 3, 1989, p.48

Abbildung 16: Benes B., Forsach R., *Layered Data Representation for Visual Simulation of Terrain Erosion*, ITESM Campus Ciudad de Mexico, IEEE SCCG2001, p.81

Referenzen

[1] Benes B., Forsbach R., *Layered Data Representation for Visual Simulation of Terrain Erosion*, ITESM Campus Ciudad de Mexico, IEEE SCCG2001, p.80-86

[2] Benes B., Forsbach R., *Parallel Implementation of terrain erosion applied to the surface of Mars*, ACM Computer Graphics: Proceedings AFRIGRAPH, pp.53-58, 2001

[3] Carpenter L.C., *Computer rendering of fractal curves and surfaces*, ACM Computer Graphics: Proceedings SIGGRAPH, Volume 13, Number 3, pp.9-15, 1980

[4] Chiba N., Fujita K., Muraoka K., *An Erosion Model Based on Velocity Fields for the Visual Simulation of Mountain scenery*, The Journal of Visualization and Computer Animation, Volume 9, Number 4, pp.185-194, 1998

[5] Dudgeon J.E., Gopalakrishnan R., *Fractal-based modelling of 3D terrain surfaces*, IEEE Press, pp. 246-252, 1996

[6] Fournier A., Fussell D., *Stochastic Modeling in Computer Graphics*, ACM Computer Graphics: Proceedings SIGGRAPH, Volume 14, Number 3, pp.108, 1980

[7] Kolb C., Mace R., Musgrave K., *The Synthesis and Rendering of Eroded Fractal Terrains*, ACM Computer Graphics: Proceedings SIGGRAPH, Volume 23, Number 3, pp.41-50, 1989

[8] Mandelbrot B.B., *Die fraktale Geometrie der Natur*, Birkhäuser, 1991

[9] Musgrave K.F., *Texturing & Modeling: A Procedural Approach, Third Edition*, Morgan Kaufmann, 2002, pp.431-445

[10] Musgrave K.F., *Texturing & Modeling: A Procedural Approach, Third Edition*, Morgan Kaufmann, 2002, pp.489-506

[11] Nagashima K.: *Computer generation of eroded valley and mountain terrains*, The Visual Computer, Volume 13, Number 9-10, pp.456-464, 1998

[12] Peitgen H.-O., Saupe D., *The science of fractal images*, Springer, 1988

[13] Pfeiffer R., Scholl O., *Natur als fraktale Grafik*, Markt und Technik, 1991

[14] Traxler C., http://www.cg.tuwien.ac.at/courses/Fraktale/PDF/fractals1.pdf, Stand: 19.05.2003, TU Wien, Institut für Computergraphik und Algorithmen