
Unterschrift des Betreuers

Vorberechnete interaktive physikalisch-basierte Animation von Vegetation am Beispiel Gras

Ausgeführt am
Institut für Computergrafik und Algorithmen
der Technischen Universität Wien

unter der Anleitung von:
Univ.Prof. Dipl.-Ing. Dr.techn. Werner Purgathofer

assistiert durch:
Projektass. Dr. Stefan Jeschke
Projektass.(FWF) Dipl.-Phys. Ralf Habel

von

Georg Selig
Lustkandlgasse 26-28/1/16
1090 Wien

Datum

Unterschrift

Inhaltsverzeichnis

1	ABSTRAKT	3
2	EINLEITUNG	4
3	PROBLEMSTELLUNG UND ZIELE DER ARBEIT	7
4	ÜBERBLICK	9
5	VERWANDTE LITERATUR	14
6	GRUNDLAGEN & DEFINITIONEN.....	17
6.1	ÜBERBLICK	17
6.2	FORMALE GRAMMATIKEN.....	18
6.2.1	Grundlagen	19
6.2.2	Definition	20
6.2.3	Arten der formalen Grammatiken.....	22
6.3	L-SYSTEME.....	26
6.3.1	Grundlagen	27
6.3.2	Definition	29
6.3.3	Arten der L-Systeme	30
6.4	RAUSCHEN	35
6.4.1	Grundlagen	36
6.4.2	Definition	40
6.4.3	Arten von Noise.....	43
6.5	DYNAMISCHE SYSTEME	47
6.5.1	Grundlagen	48
6.5.2	Definition	51
6.5.3	Arten von dynamischen Systemen.....	53
6.6	FEDER-DÄMPFER-MASSE-SYSTEME.....	55
6.6.1	Grundlagen	57
6.6.2	Definition	60
6.6.3	Arten von Feder-Dämpfer-Masse-Systemen.....	62
7	PRÄPROZESS: ERSTELLUNG PHYSIKALISCH-BASIERTER ANIMATION	64
7.1	VEGETATIONSMODELL	66
7.1.1	Analyse.....	66
7.1.2	Realisierung.....	68
7.1.3	Aufbau des L-Systems	70
7.1.4	Überführung in ein dynamisches System mittels Partikelsysteme	74
7.2	KRÄFTEMODELL.....	77
7.2.1	Analyse.....	77
7.2.2	Realisierung.....	79
7.2.3	Aufbau der Kräfte	81
7.2.4	Überführung in das dynamische System.....	81
7.3	DYNAMISCHES FEDER-DÄMPFER-MASSE-SYSTEM	83
7.3.1	Analyse.....	83

7.3.2	Realisierung.....	85
7.3.3	Aufbau des dynamischen Feder-Dämpfer-Masse-Systems	88
7.3.4	Überführung der Daten zur Visualisierung	89
8	LAUFZEITSYSTEM: ANIMATION VON VEGETATION IN ECHTZEIT	91
8.1	ANALYSE	91
8.2	REALISIERUNG	92
8.3	AUFBAU DER VISUALISIERUNG	93
9	IMPLEMENTIERUNG DES PRÄPROZESSES.....	95
9.1	IMPLEMENTIERUNG DER VEGETATION	96
9.2	IMPLEMENTIERUNG DER KRÄFTE	100
9.3	IMPLEMENTIERUNG DES DYNAMISCHEN SYSTEMS	103
10	IMPLEMENTIERUNG DES LAUFZEITSYSTEMS.....	109
11	RESULTATE	113
12	ZUSAMMENFASSUNG	124
13	FUTURE WORK	126
14	DANKSAGUNG.....	128
15	LITERATURVERZEICHNIS.....	129
16	ABBILDUNGSVERZEICHNIS	137

1 Abstrakt

In dieser Arbeit wird ein Ansatz zur Simulation der Bewegung von Vegetation (Gräser, Sträucher und Bäumen) vorgestellt. Die Hauptneuerung in diesem Verfahren ist die Trennung der Berechnung der Bewegungen (Präprozess) von der tatsächlichen Durchführung der Animation (Laufzeitsystem). Durch diese Teilung muss der Präprozess nicht in Echtzeit ablaufen, wodurch aufwendigere physikalisch-basierte Berechnungen möglich werden. Der Laufzeitprozess benutzt die vorberechneten Daten, um die Bewegung der Vegetation durchzuführen. Die Visualisierung wird durch Anwendung der Daten auf Billboards umgesetzt.

Beim Präprozess wird besonderes Gewicht auf den modularen Aufbau gelegt, wodurch eine leichte Anpassung an verschiedene Szenarien möglich wird. Der Präprozess gliedert sich in drei Hauptkomponenten. Der erste Bereich ist das parametrisierte 3D-D0L-System zur Erstellung der Vegetation, welche zu Simulationszwecken in Partikelsysteme überführt werden. Der zweite Bereich betrifft die normalverteilte 4D-Perlinnoisestruktur zur Simulation verschieden wirkender Kräfte. Zusammengeführt werden diese beiden Bereiche in einem dynamischen Feder-Dämpfer-Masse-System zur Berechnung der Bewegungsdaten. Die Speicherung erfolgt in einer visuell kontrollierbaren 3D-Textur.

Mit dieser Anordnung ist es möglich, realistische Vegetationsbewegungen zu erstellen, aktuelle Probleme des ‚State of the Art‘ zu lösen und diese in einer schnell zugreifbaren Datenstruktur für verschiedene Anwendungen zu erhalten.

Schlüsselwörter:

L-Systeme, Fraktale, Perlin Noise, Dynamisches System, Grammatik, Parametrisierung, Speicherung

2 Einleitung

Menschen waren seit jeher von der Natur und ihrer Gestaltungsformen fasziniert. Vor allem die große Vielzahl der Farben und Formen ist immer wieder bestaunenswert (siehe Abbildung 1). Immer wieder stoßen wir auf beeindruckende Vorgänge, die schon lange in unserem Umfeld stattgefunden haben, von denen aber lange Zeit keine Notiz genommen wurde. Erst einmal entdeckt, strebt der Mensch allerdings stets danach diese Vorgänge zu analysieren und sich zunutze zu machen.



Abbildung 1: Vielfalt der Natur

Besonders schwierig stellen sich das Leben und dessen Interaktion mit dessen Umwelt dar. Die Komplexität ist zumeist schon bei nur kleinen und unbedeutender scheinenden Prozessen weit größer als angenommen. Versucht man nun, diesen Prozess zu imitieren, kann man sehr leicht auf Hindernisse stoßen. So können Prozesse zu aufwendig und komplex sein, um sie mit den vorhandenen Ressourcen genau zu berechnen.

Daher müssen andere Wege entwickelt werden, um den in der Natur gefunden Prozess zu simulieren. Durch Approximationen können gewünschte Prozesse imitiert werden. Natürlich muss die Approximation so genau wie möglich sein, sodass sie nicht von der Wirklichkeit zu unterscheiden ist. Meist werden jedoch Kompromisse eingegangen, die einen vertretbaren Aufwand einem brauchbaren Ergebnis gegenüberstellen.

Daher kämpfen solche Imitationen in der Computergrafik immer mit den gleichen Problemen. Der Zwiespalt zwischen Effizienz und Korrektheit der Darstellung ist eines davon. Meist brauchen physikalisch genauer Visualisierungen mehr Ressourcen aufgrund von Berechnungen. Dies führt dazu, dass die Effizienz und Geschwindigkeit unannehmbar reduziert würden. Zudem ist es in der Computergrafik nicht notwendig,

vollständig korrekte Berechnungen durchzuführen. Vielmehr ist es wichtig, Verfahren zu finden, die realistisch wirkende Visualisierungen ermöglichen.

Bei Nachahmungen sind dabei immer drei Bereiche wichtig. Erstens sollte die Berechnung möglichst nahe an der Wirklichkeit liegen. Zweitens sollte die Darstellung möglichst plausible sein. Und drittens sollte eine ausreichend gute Effizienz vorliegen. Im Bereich der Animation von Vegetation gibt es bereits verschiedene Verfahren. Besonders der Realismus wird aufgrund von besserer Effizienz meist vernachlässigt.



Abbildung 2: Bewegung von Gräsern

Um eine natürliche Bewegung (siehe Abbildung 2) umzusetzen, muss zuerst die Bewegung selbst untersucht werden. So ist ein sanftes Schaukeln der Gräser einer Wiese bei einem böigen Wind nicht ein einheitlicher, standardisierter Fluss. Vielmehr ist es eine Interaktion einer Vielzahl von Einflussgrößen mit der Vegetation. So ist zum Beispiel nicht an jeder Stelle die Kraft eines Windes gleich stark, sondern sie variiert. Alle Größen können sich von einem Zeitpunkt zum anderen ändern und mit ihnen das gesamte Umfeld.

Weiters sind die Größen selbst äußerst mannigfaltig. Verschiedene Elemente spielen dabei eine Rolle. Winde sind dabei die aktiven Elemente. Sie sind sich zwar in ihrer Art ähnlich, können aber unterschiedlichste Stärken und Ausprägungen annehmen. Die passiven Elemente entsprechen der Vegetation selbst. Die verschiedenen Pflanzen reagieren auf die Winde und setzen sich durch diese in Bewegung. Natürlich können nicht alle Einflüsselemente und deren detaillierte Eigenschaften in die Berechnung miteinbezogen werden, wodurch eine entsprechende Abstraktion durchgeführt werden muss. Schwierig ist dabei jedoch, auf welche Weise diese Abstraktion durchgeführt werden soll.

Der „State of the Art“ in diesem Bereich ist hierbei klar auf Effizienz ausgerichtet. Für die Bewegung der Vegetation in Echtzeit stehen meist auch nur wenige Ressourcen zur Verfügung. So werden Bewegungen meist einfach im einheitlichen Takt auf eine Vegetation übertragen. Nur durch Zufallswerte werden Abweichungen eingeführt, die von dieser einfachen Form der Bewegung ablenken sollen. Die Bewegung basiert dabei meist auf einer einfachen Sinuswelle. In aufwändigeren Fällen kann diese Bewegung auch auf mehreren überlagerten Sinuswellen basieren. Aber selbst in diesem Fall kommt die Approximation einer natürlichen Bewegung nicht nahe.

Diese Vorgehensweise bringt keine realistischen Ergebnisse, da sie nicht einmal im Ansatz auf einem physikalischen Hintergrund basiert. Im schlimmsten Fall ähneln die Ergebnisse einem im Wasser schwankendem Seetang oder im Gleichschritt wippende Brettern. Im besten Fall gleichen sie einer durchgehend zufälligen Bewegung ohne jegliches globales Zusammenspiel. In den umgesetzten Programmen werden die Berechnungen meist direkt während der Laufzeit ausgeführt. Begründet wird diese Herangehensweise, dass die Berechnungen selbst ohnehin nur wenige Ressourcen verbrauchen.

Die konträre Herangehensweise konzentriert sich voll und ganz auf den Einsatz physikalischer Berechnungen. Dadurch wird eine möglichst realistische Simulationen von Bewegungen und deren Zusammenspiel erschaffen. Diese Simulationen sind jedoch sehr aufwendige. Sie erfordern zeitintensive Berechnungen, die nur durch den Einsatz von entsprechenden Ressourcen gewährleistet werden müssen. Diese Ressourcen stehen aber in Echtzeit größtenteils nicht zur Verfügung, da diese Bewegungen zumeist nur im Hintergrund laufen.

Die Darstellung zeichnet sich durch guten Realismus aus. Die Bewegungen der verschiedenen Teile einer Vegetation kann hier sehr gut aufeinander abgestimmt werden. Damit bietet diese Herangehensweise eine äußerst plausible Visualisierung, jedoch ist diese sehr aufwandsintensiv und bietet keine direkte Möglichkeit Änderungen im Aufbau des Szenarios vorzunehmen. Die Bewegung unterscheidet bei vielen „State of the Art“ Verfahren weder zwischen unterschiedlichen Vegetationsformen, noch können innerhalb einer Struktur lokale Adaptionen vorgenommen werden.

Das Ziel jener Methoden ist allerdings immer dasselbe, nämlich die Natur in ihren Bewegungen und in ihrem Aussehen zu imitieren. Eine Vereinbarung beider Methoden zu einer einzigen, die nur die Vorteile der jeweiligen Methoden teilt, sollte in der Durchführung ein erreichbares und angestrebtes Ziel sein.

3 Problemstellung und Ziele der Arbeit

Die Probleme bei der Vegetationsanimation (Gräsern, Sträucher, Bäume) sind vielfältig und nicht immer durch einen simplen Weg lösbar. Viele Ansätze in diesem Bereich sehen aus, als ob sich Seetang anstatt Vegetation im Wind bewegen würde (siehe Abbildung 3). Das Ziel einer realistischen Bewegung von Vegetation steht daher im Vordergrund dieser Arbeit.

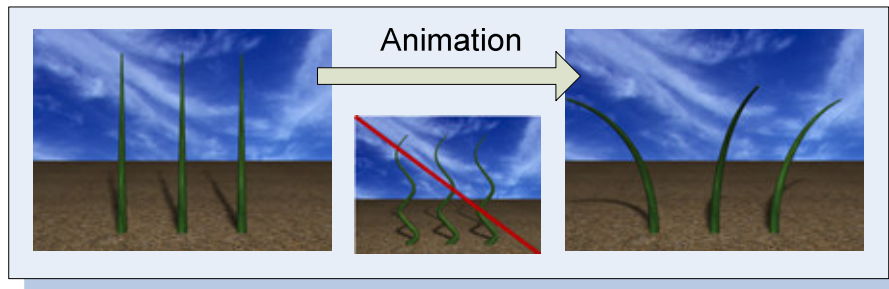


Abbildung 3: Animation von Vegetation

Das Hauptproblem ist die Erstellung einer realistischen, physikalisch-basierten Bewegung, ohne dabei zu viele Ressourcen zu verwenden. Meist schließen sich diese beiden Bereiche gegenseitig aus. Daher muss ein Ansatz gefunden werden, der Berechnung und tatsächlichen Animation trennt. Nur so können beide Bereiche vollständig verwirklicht werden.

Zusätzlich ist die große Anzahl der unterschiedlichen Anforderungen an das zu simulierende Szenario zu beachten. Eine Vielzahl der unterschiedlichen Vegetationsstrukturen erfordert einen flexiblen Aufbau in diesen Szenarien. Jede Vegetation hat unterschiedliche Eigenschaften, welche definiert werden müssen. Weiters beinhaltet jedes Szenario unterschiedliche Kräfte, welche auf die Strukturen wirken. Diese müssen ebenfalls unterschiedlich definierbar sein. Beide Bereiche sollen verändert werden können.

Unweigerlich rücken damit die Formen der Definitionen der einzelnen Abschnitte und deren Wirkungen in den Brennpunkt. Jedes anpassungsfähige Verfahren muss auch in der Lage sein, unterschiedlichste Daten in eine allgemein verwendbare Form zu bringen. Dabei sollte eine Unterteilung in Kernbereiche stattfinden. Das würde die Verwendung von Definition in mehreren unabhängigen Szenarien gewährleisten. Eine derartige Wiederverwendung sollte zudem ohne Adaptionen oder eventuelle Umrechnungen möglich sein. So können Strukturen und Kräfte in unterschiedlichen Umgebungen beobachtet und deren Verhalten verglichen werden.

Die Definition der Strukturen einer Vegetation erfolgt auf einem grammatikalischen Weg. Dieser ermöglicht es, innerhalb der Grammatik seine Struktur frei zu definieren.

Außerdem gibt er der Struktur gleichzeitig eine Form, die eine leichte Weiterverwendung ermöglicht. Eine zusätzliche Attribuierung der Grammatik ermöglicht es, die Struktur mit ihren Eigenschaften in das System einzubinden. Grammatiken sind extrem flexible in ihrer Beschaffenheit, wodurch sie sich sehr gut an unterschiedliche Anforderungen anpassen können.

Bei der Definition von Kräften treten parametrisierte, zeitabhängige, vorberechnete Zufallszahlen in Aktion. Mit Hilfe der Parameter können die Kräfte je nach Bedarf in ihrer Zufälligkeit bearbeitet werden. Durch die Verwendung von normalverteilten Zufallszahlen können die Definition der Parameter noch besser kontrollierbar gemacht werden. So kann ein möglichst breites Spektrum an unterschiedlichen Kräften definierbar gemacht werden. Durch die Kombination mehrerer Kräfte können sehr komplexe Systeme erstellt werden, die an die jeweilige Situation angepasst werden können.

Weiters sollten die Ergebnisse eine Form besitzen, die eine schnelle und visuelle Kontrolle möglich machen. Damit sind die Daten leichter miteinander vergleichbar. Die Ergebnisse müssen daher visuell aufbereitet werden und in einem flexiblen, standardisierten Format gespeichert werden. Damit können die Daten leichter wieder verwendet werden. Auch direkte Änderungen an den Daten wären damit möglich.

Durchgehend sollten alle Schritte ab der Eingabe visuell dargestellt werden, von der Definition der Vegetationsstrukturen über die Berechnung der Daten bis hin zur Protokollierung der Ergebnisse. Damit wird ein größtmögliches Feedback gewährleistet, wodurch eine optimale Kontrollierbarkeit und Steuerbarkeit der Prozesse gewährleistet wird. Unnötige und zeitaufwendige Wiederholungen könnten somit besser vermieden werden.

Mehrere Möglichkeiten sind hierfür möglich, doch muss in jedem Fall darauf geachtet werden, dass alle Methoden sich zu einem Ganzen zusammenfügen lassen.

4 Überblick

Das Konzept der Arbeit gründet sich auf zwei Hauptbereiche, dem physikalisch-basierten Präprozess und einer anschließenden Echtzeitvisualisierung. Der langsame, berechnungsintensive Präprozess berechnet dabei die physikalisch-basierte Animation. Aufgrund dieser Daten arbeitet der folgende Laufzeitprozess und vollzieht die Animation in Echtzeit.

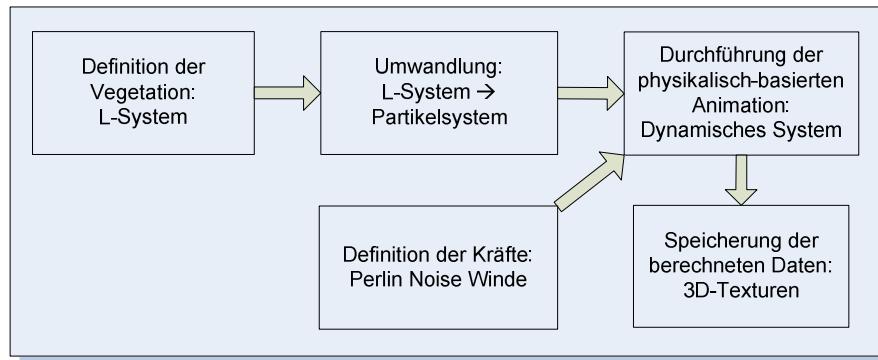


Abbildung 4: Präprozess der Animation von Vegetation, Überblick

Der Präprozess basiert auf drei Hauptbereichen (siehe Abbildung 4), die in modularer Form zu einem ganzen System zusammengefügt sind. Die Definition der Strukturen von Vegetation und die Definition der Kräfte bilden dabei die Grundlage für die Dateneingabe. Das dynamische System fügt diese beiden in einem System zusammen und vollzieht die notwendigen Berechnungen. Die durch L-Systeme definierten Strukturen müssen vorher in Partikelsysteme umgewandelt werden. Diese Partikelsysteme werden dann in das dynamische System eingebunden. Das dynamische System durchläuft die Schritte der Simulation, während die Daten parallel gespeichert werden. Die Daten werden dann für die angestrebte Visualisierung verwendet.

Die definierten vegetativen Strukturen finden sich im dynamischen System als umgewandelte Partikelsysteme wieder. Die Definition selbst erfolgt auf grammatikalische Weise mit Hilfe von L-Systemen. Dadurch können sowohl einfache als auch komplexe Datenstrukturen sehr flexibel definiert werden. Der Speicheraufwand bleibt dabei sehr gering. Die Strukturen der Vegetation sind sowohl untereinander als auch vom Gesamtsystem unabhängig. Mit Hilfe von Parametern werden Eigenschaften der Strukturen mit in die anschließende Berechnung eingebracht.

Die Kräfte (im speziellen Winde) werden ebenfalls unabhängig vom Rest des Systems definiert. Sie basieren auf einem „Perlin Noise Zufallsprinzip“. Um die Kräfte trotz des Zufalls besser steuerbar zu machen, basieren die Zufallszahlen auf einer definierbaren Normalverteilung. Es wird sowohl für die Hauptrichtung als auch die beiden

orthogonalen Richtungen (Nebenwindrichtungen) ein unabhängiger „Perlin Noise“ angelegt. Die Nebenwindrichtungen sind als jeweilige orthogonale Richtung zur Hauptwindrichtung definiert. Wie auch schon bei der Definition der vegetativen Strukturen sind auch hier die Kräfte unabhängig voneinander und auch unabhängig vom dynamischen System. Die Kräfte sind dadurch frei mit allen Strukturen einer Vegetation und anderen Kräften kombinierbar.

Beide Definitionen fließen beim Festlegen eines dynamischen Systems ineinander und bilden damit die Grundlage für die Simulation. Das dynamische System beruht auf einem Feder-Dämpfer-Masse-Prinzip. Um die vegetativen Strukturen in eine für das dynamische System nutzbare Form zu bringen, werden sie in Partikelsysteme umgewandelt. Auf diese Partikelsysteme werden die Kräfte aller Winde im System angewendet. Die daraus resultierende Schwingung der Partikel stellen die notwendigen Daten für eine Laufzeitvisualisierung dar. Sie werden durch das System in jedem Zeitschritt festgehalten und für den folgenden Laufzeitprozess in Texturen festgehalten.

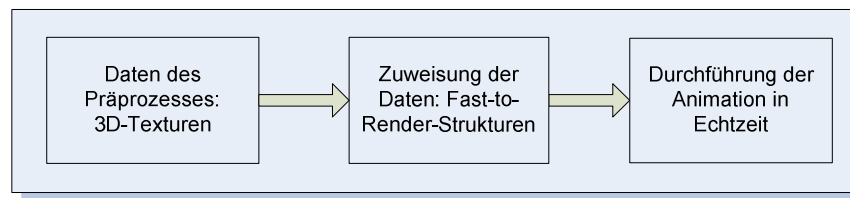


Abbildung 5: Laufzeitsystem der Animation von Vegetation, Überblick

Der Laufzeitprozess schließt direkt an die Erstellung der Daten an (siehe Abbildung 5). Er ist jedoch unabhängig von der Berechnung. Er arbeitet ausschließlich auf den erhaltenen Daten und führt keine weiteren Berechnungen durch. Er verwendet die Texturen des Präprozesses und führt mit diesen die Animation auf verschiedenen geometrischen Strukturen aus. Besonders eignen sich dafür Fast-to-Render-Geometrie, wie Billboards oder Impostors. Alternativ kann statt einfacher auch komplexere Geometrie verwendet werden. Die Daten aus den Texturen müssten nur den entsprechenden Vertices in der Geometrie zugeordnet werden. Die Animation läuft dabei mit einem Minimum an Rechenleistung ab und besitzt trotzdem die Vorzüge einer physikalisch-basierten Bewegung.

Die Kombination der drei Präprozessbereiche zu einem dynamischen System wird in Abbildung 6 illustriert. Im ersten Abschnitt links oben ist zuerst die Darstellung eines einfachen L-Systems zu sehen. Dieses wird in Folge in ein Partikelsystem mit drei Partikeln und zwei Verbindungen zwischen diesen Partikeln überführt. Das unterste Partikel (Wurzelpartikel) dient zur späteren Verankerung in der Simulationsfläche. Im Abschnitt links unten sind die Kräfte (Winde) zu sehen, welche in der Simulation wirken. Sie werden direkt über die gesamte Simulationsfläche gelegt und beeinflussen damit jedes Partikelsystem auf der Simulationsfläche. Die Simulationsfläche (Abschnitt rechts) stellt jene Fläche dar, auf welcher die Simulation abläuft. Die Partikelsysteme sind auf dieser Fläche verankert. Die Unterteilung dieser Fläche bestimmt die

Auflösung der gespeicherten Texturen im späteren Verlauf. Je zahlreicher die Unterteilung, desto größer sind auch die Texturen, die später gespeichert werden (6x6 Unterteilungen führen zu einer 6x6 Textur).

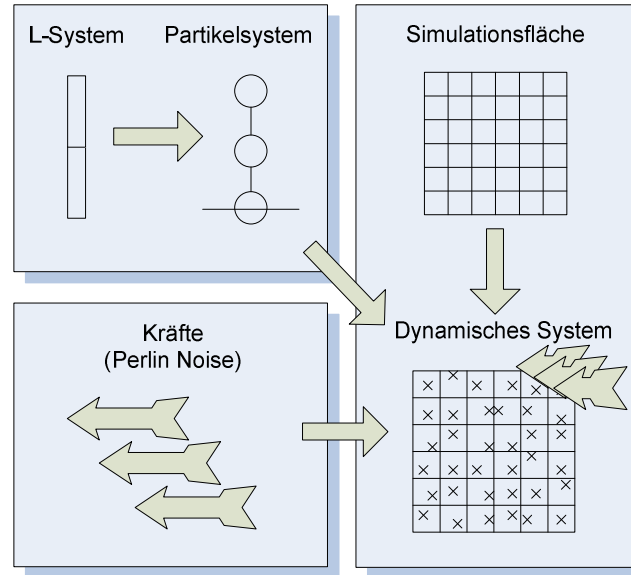


Abbildung 6: Aufbau des Präprozesses

Die Partikelsysteme werden auf der Simulationsfläche verteilt. Jede Unterteilung der Simulationsfläche beinhaltet in Abbildung 6 ein Partikelsystem. Die Positionierung des Partikelsystems innerhalb einer Unterteilung erfolgt dabei zufällig. Das jeweilige Partikelsystem ist für die Daten der jeweiligen Unterteilung verantwortlich. Dies ist gleichbedeutend mit dem jeweiligen Pixel in der Textur. Mehrere Partikelsysteme pro Unterteilung auf die Simulationsfläche machen den Einsatz einer zufälligen Verteilung sinnvoll. Dadurch wird die Information in einem Pixel durch mehrere Partikelsysteme erstellt. Dadurch ergibt sich eine bessere Verteilung, jedoch wird dadurch der Berechnungsaufwand größer.

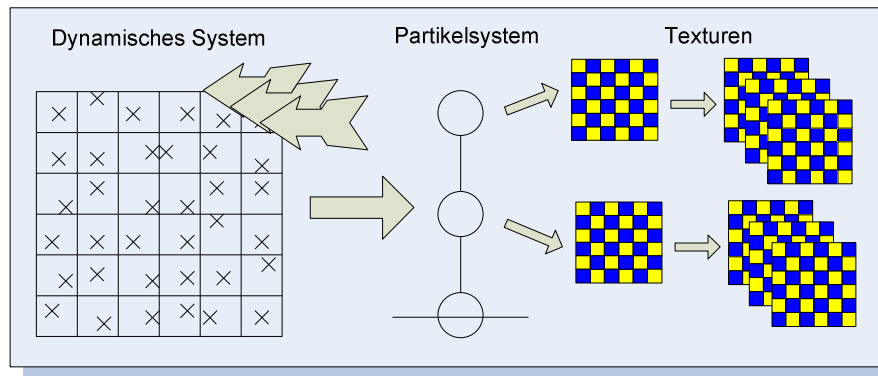


Abbildung 7: Speicherung der Daten des dynamischen Systems

Die Simulation wird dann aufgrund der definierten Partikelsysteme und Kräfte durchgeführt. In jedem Schritt werden die Texturen des gesamten Simulationsfeldes erstellt. Die Anzahl dieser Texturen hängt von der Anzahl der verschiedenen Partikel in der Simulation ab. In Abbildung 7 ist ein Partikelsystem mit drei unterschiedlichen Partikeln zu sehen. Da sich das Wurzelpartikel nicht bewegt und nur der Verankerung dient, bleiben zwei Partikel übrig, für die in jedem Zeitschritt eine Textur angelegt wird. Ein Pixel wird dabei von dem in dem zugehörigen Bereich befindlichen Partikel und dessen Bewegung festgelegt. Für jeden Zeitschritt wird dabei eine weitere Textur erstellt. Diese hält die zeitliche Folge der Bewegungen des jeweiligen Partikels fest. Am Ende erhält man dadurch eine 3D-Textur für jedes Partikel des Partikelsystems.

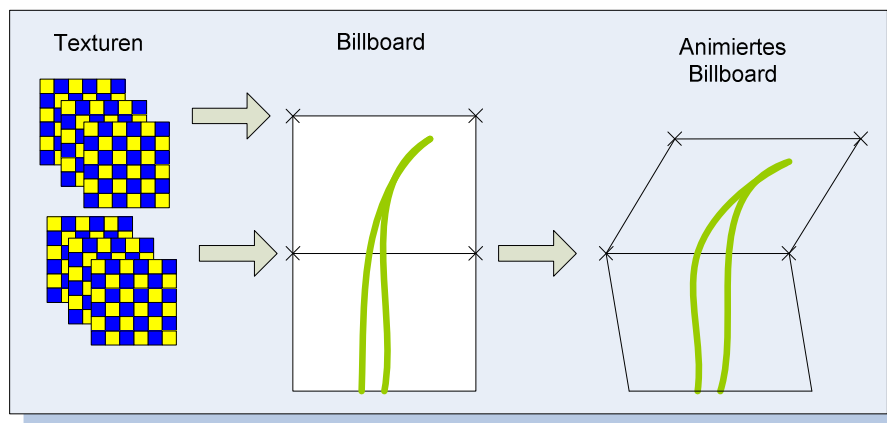


Abbildung 8: Aufbau des Laufzeitsystems

Mit dieser 3D-Textur ist es dann möglich (siehe Abbildung 8) eine Animation in Echtzeit durchzuführen. Dabei werden die 3D-Texturen des Präprozesses den entsprechenden Vertices zugeordnet. Die Position der geometrischen Struktur im Raum gibt die Koordinaten für die Information in der Textur an (Grashalm1 auf Position x,y , entsprechend Texturkoordinaten $u,v = x,y$ in zugeordneter Textur). Jeder Vertex hat

somit eine Zuordnung zu einer Textur und eine Positionsangabe für diese Textur. Damit beinhaltet jeder Vertices die notwendigen Informationen, um die Animation zu durchlaufen. Entsprechend den gespeicherten Bewegungen werden die Vertices dann im Raum versetzt. Wurden alle Schichten einer 3D-Textur durchlaufen, startet der Durchlauf von neuem. Zwischen den einzelnen Schritt der gespeicherten Bewegungen wird interpoliert, um eine fließende Bewegung zu sichern.

5 Verwandte Literatur

Die Arbeit ist durch ihre flexible Gliederung und ihrem Aufbau sehr gut mit ähnlicher Literatur vergleichbar. Da sie aber mehrere Thematiken behandelt, kann kein gesamtes Werk zu einem direkten Vergleich herangezogen werden. Die Bereiche der Arbeit spiegeln sich nicht nur nach ihrer lokalen Effizienz und Leistung wieder sondern ebenfalls nach ihrer Anpassungsfähigkeit im Gesamtsystem.

Im Allgemeinen können Niederschriften wie „Randomly Accessible Procedural Animation of Physically Approximate Turbulent Motion“ [A2] oder „Stochastic Motion - Motion Under the Influence of Wind“ [A1] herangezogen werden. Sie behandeln die grobe Thematik im Überblick und sind von den Details der Arbeit völlig unabhängig. Vor allem „Stochastic Motion - Motion Under the Influence of Wind“ [A1] stellt eine sehr schöne Einführung in die Problematik des Gebiets dar.

Für das Kapitel 6.2 eignen sich vor allem die Werke von Przemyslaw Prusinkiewicz und/oder Aristid Lindenmayer wie „Animation of Plant Development“ [L5], „The Algorithmic Beauty of Plants“ [L2] und „The Artificial Life of Plants“ [L1]. Przemyslaw Prusinkiewicz ist ein wahrer Meister auf dem Gebiet der Grammatiken (insbesondere der L-Systeme) und vieles aus seinen Ideen ist auch direkt oder als Anstoß in diese Arbeit eingeflossen.

Innerhalb der Thematik der parametrisierten Zufallsschemata sind vor allem die Werke „Texturing & Modeling, A Procedural Approach, Third Edition“ [P2] und „Perlin noise pixel shaders“ [P8] zu empfehlen. Sie behandeln die unterschiedlichen Thematiken sehr gut und geben immer wieder Anregungen wie eine Herangehensweise überarbeitet werden kann.

Im Bereich der Dynamischen Systeme zeigen Werke wie „Dynamics: Numerical Exploration“ [M13] und „Nonlinear Dynamics and Chaos“ [M12] Wege und Möglichkeiten auf, die in diese Arbeit mit eingeflossen sind. Durch sie werden auch einige Bereiche weitaus detaillierter behandelt und können so ebenfalls als weiterführende Literatur genutzt werden.

Eine zu erwähnende Engine in diesem Bereich ist die neue „Crysis Engine 2.0“ von Crytek. Die Neuerung dieser Engine beruht mehr auf der Interaktion der Benutzer mit der Vegetation (Zerstören von Bäumen, etc.) als auf einer realistischen Bewegung der Vegetation. Sie verwendet für die Vegetationsbewegung einen physikalisch-basierten Ansatz, der jedoch zur Laufzeit ausgeführt und nicht vorberechnet wird. Dadurch kann nur ein relativ simples System verwendet werden, ohne die Rechenleistung zu sehr zu beanspruchen.

Der Ansatz von John Isidoro und Drew Card [G3] beschreibt wie man Gras mit Pixeln und Vertex Shadern effektiv und realistisch bewegen kann. Diese Bewegung selbst

basiert in ihrer Arbeit allerdings nur auf einer einfachen Sinusberechnung. Es werden dafür vier Sinuswellen mit unterschiedlichen Frequenzen zur Animation der Gräser vereint, was den Realismus der Bewegung durchaus steigert. Jedoch kommt keine Bewegung auf Basis von Sinuswellen einem physikalisch-basierten Ansatz an Realismus nahe. Zudem werden nur einfache Billboards verwendet, die weiche und realistische Bewegungen von Gras kaum ermöglichen.

Der Ansatz von Sylvain Guerraz et al. [G6] beschäftigt sich mit der automatischen Erstellung eines Grasbewuchses in einer Szene, um diese dann durch Wind und anderen lokalen Phänomenen, wie animierte Objekte, etc., zu beeinflussen. Der Wind wird dabei mit einer 2D-Maske über das Gras bewegt. Diese Bewegungen sind aufgrund der eingeschränkten Dimension relativ einfach in der Auswirkung. Die verwendeten Grasbillboards sind zwar 3-teilig, werden jedoch nur von dieser einen 2D-Maske bewegt. Individuelles Bewegen der verschiedenen Teile ist dadurch nicht möglich.

Der Ansatz von Frank Perbet und Marie-Paule Cani [G7] arbeitet ebenfalls wie diese Arbeit auf physikalisch-basierten Daten der Bewegung von Gräsern. Der Wind wird in einer einzigen 2D-Maske über die Szene gezogen. Dadurch können nur begrenzt Bewegungsdaten auf die Billboards in der Szene angewendet werden. Die Eigenschaften der Gräser sind bereits vordefiniert, wodurch individuelle Anpassungen gar nicht möglich sind. Die Darstellung und Handhabung von Gras beschränkt sich nämlich auf ein einziges Modell eines einfachen Grashalms, ohne das dieses Modell geändert werden könnte.

Die Ansätze von Brook Bakay [G9][G13] verwenden ebenfalls einen physikalisch-basierten Ansatz, um Gras zu animieren. Dieser Ansatz arbeitet mit einer einzigen Textur, um die vorberechneten Bewegungen von Gras zu übertragen, wodurch eine uniforme Biegung der Gräser entsteht. Die Darstellung der verwendeten Grasmodelle zur Visualisierung ist geometrisch sehr aufwendig und wirkt zudem nicht sehr realistisch. Die Bewegung der Gräser entspricht einer einheitlichen Funktion die für alle Gräser zu jeder Zeit gleich ist. Dadurch werden die Bewegungsmöglichkeiten der Gräser stark eingeschränkt. Individuelle Bewegungen sind mit diesem Ansatz nicht umzusetzen. Wie auch bei anderen Ansätzen wird hier ein festes Grasmodell verwendet, ohne die Möglichkeit zu geben, das Modell nach bestehenden Anforderungen zu ändern.

Der Ansatz von Sven Banisch und Charles A. Wuthrich [G12] beschäftigt sich ebenfalls mit der Anwendung von Feder-Masse-Systemen zur Bewegung von Gras oder Fell. In dieser Arbeit werden jedoch nur einfache Feder-Masse-Systeme (nur ein Partikel und eine Verbindung) verwendet und die Animation wird nicht vorberechnet. Wie auch bei [G9] wird die Biegung der Gräser implizit verwendet, ohne diese ändern zu können. Dadurch werden die Möglichkeiten der Bewegung stark eingeschränkt. Diese Vereinheitlichung der Bewegungen führen zu geringerem Realismus und zu eingeschränkter Flexibilität des Ansatzes.

Der Ansatz von Thomas Di Giacomo et al. [G15] befasst sich mit der Generierung und Bewegung von Bäumen. Die Bewegung der Bäume durch Wind basiert auf einem Feder-Dämpfer-Masse-Systeme, das direkt in der Szene berechnet wird. Dadurch werden mehr Ressourcen während der Laufzeit gebraucht. Der Wind selbst ist in seiner Form (einfache sinus-basierende Kraft) sehr einfach gehalten. Wie auch bei anderen Ansätzen fehlt hier durch den sinus-basierten Wind der Realismus der Bewegung.

Der Ansatz dieser Arbeit basiert auf einem physikalischen Ansatz, der auf einer Trennung der Berechnung der Daten und Animation der Gräser beruht. Dadurch werden sinus-basierte Bewegungen durch realistische, physikalisch-basierte Bewegungen ersetzt und gleichzeitig werden während der Laufzeit keine zusätzlichen Ressourcen verbraucht. Die Gräsermodelle können individuell angepasst werden. Zusätzlich können die Daten der Berechnung direkt in verschiedenen Detailstufen auf die Gräsermodelle während der Laufzeit angewendet werden. Es werden implizite Funktionen zur Biegung der Gräser vermieden.

6 Grundlagen & Definitionen

6.1 Überblick

In den folgenden Kapiteln werden alle notwendigen und grundlegenden Kenntnisse dieser Arbeit besprochen. Sie geben einen Einblick in die verwendeten Verfahren und Berechnungsmethoden. Diese Vorkenntnisse sind nur für den aufwendigeren Präprozess notwendig. Wie auch der Präprozess können die folgenden Kapitel in drei Bereiche unterteilt werden. Diese können dann jenen Kapiteln des Präprozesses zugeordnet werden.

Die Kapitel 6.2 und 6.3 sind dem Kapitel 7.1 zugeordnet. Die formalen Grammatiken bieten eine Einführung für das Kapitel 6.3. Sie geben kompakt wieder, wie Grammatiken funktionieren und welche Arten von Grammatiken es gibt. Dadurch können die L-Systeme leicht aus dem Kontext der Grammatiken heraus verstanden werden. Die L-Systeme werden später bei der Definition der Vegetation verwendet.

Das Kapitel 6.4 legt die Grundlage für das Kapitel 7.2. Im Kapitel 6.4 werden alle notwendigen Vorkenntnisse vermittelt, um Noise im Allgemeinen zu verstehen. Außerdem wird Perlin Noise genauer in diesem Kapitel behandelt. Dieser Noise wird später zur Definition der Kräfte verwendet.

Die beiden Kapitel 6.5 und 6.6 sind für das Kapitel 7.3 notwendig. Die dynamischen Systeme geben dabei einen allgemeinen Einblick. Darauf aufbauend liefern die Feder-Dämpfer-Masse-Systeme Kenntnisse über die spezielle Art von Feder-Dämpfer-Masse-Systeme. Diese werden später zur Durchführung der Bewegungsberechnungen benötigt.

6.2 Formale Grammatiken

Der Mensch bedient sich jeden Tag seiner Sprache, um die Dinge in seiner Umgebung zu beschreiben und sich mit anderen auszutauschen. Dieser Informationsaustausch kann aber nur erfolgen, wenn andere die verwendete Sprache verstehen. Schwieriger wird dieses Unterfangen, wenn der Informationsempfänger die Sprache nicht versteht. Im schlimmsten Fall verfügt der Empfänger allerdings gar nicht über die Möglichkeit, eine solche Sprache zu verstehen. Ein Beispiel dafür wäre ein Computerprogramm. Im Bereich der formalen Grammatiken kann auf [L12], [L13], [L14], [L15] und [L16] verwiesen werden.

In diesem Fall ist es notwendig die Sprache zu formalisieren. So kann die Möglichkeit geschaffen werden, die die Sprache in ein logisches Umfeld kleidet. Damit wird die Sprache in einen brauchbaren Zustand für eine Verarbeitung gebracht. Diese Sprachen werden dann Formale Sprachen genannt und schon seit langem in der Informatik eingesetzt. Im Grunde handelt es sich dabei um Mengen von Zeichenketten. Bei diesen ist nur gefordert, dass die Zeichen untereinander unterscheidbar sind. Im Unterschied zu natürlichen Sprachen ist mit Formalen Sprachen lediglich eine Syntax, jedoch keine Semantik verbunden.

Einer der wichtigsten Begründer der Formalen Sprachen ist Noam Chomsky. Ihm ist es wie keinem anderen gelungen, Sprache allgemein durch ihre Formalisierung in eine für die Informatik verwendbare Form zu bringen. Im Bereich des Compilerbaus aber auch in der Komplexitätstheorie werden seine grundlegenden Erkenntnisse genauso verwendet wie auch in der Programmierung. Durch ihre Nähe zu natürlicher Sprache sind sie leicht verständlich und in vielen Bereichen einsetzbar. Besonders in der Definition von großen Mengen werden sie oft verwendet.

Grundlegend definiert sich eine Formale Sprache S durch

$$S = \{A, W\}$$

wobei A das Alphabet ist und W die daraus gebildeten Wörter. Das Alphabet besteht dabei aus verschiedenen Zeichen der Sprache. Wörter aus W setzen sich aus einer Folge von Zeichen aus diesem Alphabet zusammen. Voraussetzung ist dabei, dass das Alphabet A nicht leer ist.

Formale Sprachen sind jedoch nicht immer ganz ausreichend, um eine Vegetation zu definieren. Es können zwar die Zeichen und zugehörigen Wörter definiert werden, jedoch keine zusätzlichen Regeln, welche deren Anordnung angibt. Weiters können fehlt in formalen Sprachen die Möglichkeit eine zeitliche Komponente mit einzubinden. Daher muss eine umfangreiche Spezifikation gefunden werden. Diese findet sich in der Welt der Formalen Grammatiken.

Bei formalen Grammatiken lassen sich zusätzlich zu den Zeichen und Symbolen, Regeln für die Erstellung von Worten angeben. Dadurch kann der Verlauf in der Zeit als auch die Möglichkeiten der Anordnung kontrolliert werden. Die Anpassung an jeweilige Strukturen einer Vegetation wird damit möglich. Formalen Grammatiken lassen sich in zwei große Gruppen unterteilen, die analytischen und die generischen Grammatiken. Für die Definition von solchen Strukturen ist nur der Bereich der generischen Grammatiken interessant. Diese können die Strukturen mit Hilfe der Grammatik erzeugen bzw. nachbilden. Bei der analytischen Grammatik kann nur eine bestehende Sprache analysiert werden. Aus diesem Grund werden im folgenden Verlauf nur mehr Formale Grammatiken in Bezug auf die Generierung betrachtet werden.

6.2.1 Grundlagen

Eine formale Grammatik beschreibt mit Hilfe einer Menge von Symbolen und Regeln eine Sprache in ihrer Gesamtheit. Das beinhaltet neben den Zeichen der Sprache (Menge der Symbole) auch alle Wörter. Die Wörter einer Grammatik umfassen alle die mit den Symbolen bildbaren Kombinationen. Weiters können auch Wörter kombiniert werden. Die Kombinationen von Wörtern werden Sätzen genannt. Die Regeln einer formalen Grammatik werden dabei Produktionen genannt. Die Menge der Symbole bilden das Alphabet. Diese sind die Basiselemente einer Grammatik, mit welchen nur durch ihre Definition unterschiedlichste Sprachen generiert werden können.

Die Regeln oder Produktionen werden dabei auf ein Symbol angewendet. Dadurch erzeugen sie ein Wort, einen Satz oder auch ein neues Symbol des Alphabets. Bei der Anwendung einer Produktion handelt es sich schlicht um eine Ersetzung des Symbols. Ein Beispiel für eine einfache Regel wäre

$$A \rightarrow BB$$

Wird die Regel angewandt, wird das Symbol A durch die Symbole B und B ersetzt. Bei einem Alphabet von {A, B} könnten die Ergebnisse {B, BB} erzeugt werden. Ausschlaggebend für das Ergebnis ist hierbei die Wahl des Anfangssymbols. Wird bei diesem Beispiel ein B als Anfangssymbol gewählt, so würde die Regel nicht zum Einsatz kommen.

Da sich die zu beschreibenden Sprachen meist aus unendlich großen Mengen an Wörtern und Sätzen zusammensetzen, muss dies durch die Definition einer Grammatik beschreibbar sein. Rekursionen machen dies möglich. Die Rekursion ist im Grundbegriff nichts anderes als eine Wiederholung. Die Regeln können immer wieder auf die Wörter und Sätze der Sprache angewendet werden und bringen dadurch immer neue Wörter und Sätze hervor. Dadurch beschränkt sich die Sprache nicht in ihrer Endlichkeit. Ein Beispiel dafür wäre

$$A \rightarrow AB$$

Diese Regel angewendet auf das Symbol A ergibt in Folge ein Wort, das wiederum das Symbol A beinhaltet. Dadurch kann die Regel immer wieder angewendet werden, um ein neues Wort zu generieren. So entstehen in Folge die Wörter

AB, ABB, ABBB, ABBBB, ABBBBB, usw.

Hieraus ist leicht ersichtlich, warum Grammatiken sehr einfach unendliche große Sprachen beschreiben können.

Formale Grammatiken können auf vielerlei Arten und Weisen definiert werden. Zu beachten ist dabei, dass die Definition immer eindeutig und nachvollziehbar bleiben sollte. Dies erleichtert die Rückverfolgung und das System bleibt klar deterministisch. Natürlich können formale Grammatiken nicht eindeutig definiert werden, wie zum Beispiel die Grammatik

$$\begin{aligned} A &\rightarrow AB \\ B &\rightarrow AB \end{aligned}$$

Drei alternative Wege zur Erreichung des Wortes AABB wären

A	A → AB	AB	A → AB	ABB	B → AB	AABB
A	A → AB	AB	B → AB	AAB	A → AB	AABB
B	B → AB	AB	A → AB	ABB	B → AB	AABB

In diesem Beispiel kann das Wort AABB auf unterschiedlichem Weg erzeugt werden. Dadurch ist nicht klar ersichtlich, auf welchem Weg es tatsächlich erzeugt worden ist. Da die formale Grammatik zur Beschreibung von vegetativen Strukturen Verwendung finden soll, sollte diese eindeutig und damit klar in ihrem Aufbau gestaltbar sein.

6.2.2 Definition

Bisher wurde nur eine vereinfachte Form der Definition einer Grammatik verwendet, um grundlegende Begriffe und grundlegendes Verhalten zu erklären. Exakterweise wird das Alphabet einer formalen Grammatik nämlich in zwei Gruppe aufgeteilt, in die Terminalsymbole und die Nonterminalsymbole. Der Unterschied zwischen diesen beiden Symbolgruppen besteht darin, dass Nonterminalsymbole immer mit Hilfe einer Produktion ersetzt werden können. Terminalsymbole können keine weitere Ersetzung erfahren.

Weiters beinhaltet jede formale Grammatik ein Startsymbol. Dieses Symbol ist eines der Symbole aus den Nonterminal- oder Terminalsymbolen. Mit diesem Symbol startet

der gesamte Vorgang der formalen Grammatik. Nur über dieses Symbol können alle Wörter der Grammatik erreicht werden. Bei dem Startsymbol kann es sich auch um ein Terminalsymbol handeln. Da Terminalsymbole nicht ersetzt werden können, beinhaltet die aus diesem Terminal entstehende formale Grammatik folglich nur dieses eine Symbol.

Die mathematische Definition einer formalen Grammatik ist

$$G = (N, T, P, S)$$

wobei N der Menge aller Nonterminalsymbole entspricht, T die Menge aller Terminalsymbole angibt, P die Menge aller Ersetzungsregeln bzw. Produktionen umfasst und S das Startsymbol ist. Dabei zu beachten ist, dass N und T niemals die gleichen Symbole umfassen dürfen, sondern immer disjunkt sein müssen.

Im vorhergehenden Kapitel 6.2.1 wurden Produktionen schon in ihrer Grundbedeutung erklärt. Die genaue Definition einer Produktion ist wie folgt

$$\alpha \rightarrow \beta$$

wobei α von der Form $(N \cup T)^* N (N \cup T)^*$ und $\beta (N \cup T)^*$. Dies bedeutet nichts anderes, als dass α zumindest ein Nonterminalsymbol enthalten muss. Damit kann erst eine Produktion eingesetzt werden. Das Ergebnis β muss wiederum nur Terminal- und Nonterminalsymbole enthalten, die in der Grammatik definiert sind.

Ableitungen können wiederum auf verschiedene Art und Weise geschehen. Die erste mögliche Art einer Ableitung ist die Linksableitung. In diesem Fall erfolgt die Ableitung von links und ersetzt immer das erste, ganz linke Nonterminal mit Hilfe der Produktionen. Bei der Rechtsableitung hingegen wird das letzte Nonterminal also das ganz rechte ersetzt. Bei der Parallelableitung werden alle Nonterminal gleichzeitig in einem Schritt ersetzt.

Die Definition der drei Ableitungen lautet

Linksableitung

$$A \alpha B \rightarrow_L A \beta B$$

wobei $\alpha \rightarrow \beta \in P$, $A \in T^*$ und $B \in (N \cup T)^*$.

Rechtsableitung

$$A \alpha B \rightarrow_R A \beta B$$

wobei $\alpha \rightarrow \beta \in P$, $A \in (N \cup T)^*$ und $B \in T^*$.

Parallelableitung

$$A_0 \alpha_1 A_1 \dots A_{n-1} \alpha_n A_n \rightarrow_p A_0 \beta_1 A_1 \dots A_{n-1} \beta_n A_n$$

wobei $\alpha_1 \rightarrow \beta_1 \dots \alpha_n \rightarrow \beta_n \in P$ und $A_0 \dots A_n \in T^*$.

Weiters gilt ein Wort w ableitbar in einer formalen Grammatik G aus einem anderen Wort v dieser formalen Grammatik, gleichbedeutend mit $v \Rightarrow w$, wenn

$$\begin{aligned} &w, v \in (N \cup T) \\ &\text{falls } \exists x, y \in (N \cup T) \text{ und } \exists \alpha, \beta \in P, \\ &\text{sodass } v = x\alpha y \text{ und } w = x\beta y \text{ gilt} \end{aligned}$$

Dies bedeutet, dass man durch Anwendung von Produktionen auf das Ausgangswort v das Wort w bilden kann.

Eine formale Grammatik erzeugt eine bestimmte Sprache die aus allen Wörtern besteht, die diese formale Grammatik erzeugen kann. Diese Sprache besteht jedoch nur aus Wörtern, bei denen keine weiteren Ersetzungen durchgeführt werden können. Damit dürfen diese Wörter keine Nonterminalsymbole mehr beinhalten. Auch das Leerwort kann in dieser Menge enthalten sein. Die formale Definition dazu lautet

$$L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$$

wobei $L(G)$ die erzeugte Sprache von G nur aus Wörtern w , welche sich aus Terminalsymbolen T zusammensetzen, besteht und aus dem Startsymbol S ableitbar sind.

6.2.3 Arten der formalen Grammatiken

Noam Chomsky ist der Pionier der formalen Sprachen und Grammatiken. Er gliederte die formalen Grammatiken in vier Untergruppen, die allgemeine Grammatik (Typ 0), die kontextsensitive Grammatik (Typ 1), die kontextfreie Grammatik (Type 2) und die reguläre Grammatik (Typ 3). Die Unterteilung wird die „Chomsky Hierarchie“ genannt. Die „Chomsky Hierarchie“ beruht auf einer immer stärkeren Einschränkung der Grammatiken bezüglich ihrer Produktionen. Je höher der Typ einer formalen Grammatik wird, desto stärker ist die Einschränkung. So ist eine Typ 0-Grammatik völlig frei bezüglich der Wahl ihrer Produktionen. Eine Typ 3-Grammatik ist hingegen am stärksten beschränkt in der Form ihrer Produktionen. Zudem bildet die Menge aller übergeordneten Grammatiken immer eine echte Teilmenge der untergeordneten Grammatiken.

Typ 0-Grammatik, allgemeine Grammatik

Typ 0-Grammatiken erlauben jede Art von Produktion. Diese Form der Grammatik kann durch ihre Produktionen jegliche Terminal-Nonterminalsymbol-Folge in eine andere Terminal-Nonterminalsymbol-Folge überführen. Sie ist damit die mächtigste aller Grammatiken. Alle formalen Grammatiken erfüllen damit die Voraussetzung, eine allgemeine Grammatik zu sein. Produktionen müssen der Form

$$\alpha \rightarrow \beta \text{ und } \alpha, \beta \in (N \cup T)^+$$

genügen.

Die allgemeine Grammatik ist zwar äußerst vielseitig, jedoch in der tatsächlichen Verwendung sehr ineffizient und nicht deterministisch. Dies entsteht dadurch, dass unterschiedliche Produktionen für ein Nonterminalsymbol erlaubt werden. Ebenso können unterschiedliche Produktionen zu unterschiedlichen Zeitpunkten verwendet werden. Diese Problematiken können jedoch umgangen werden, wenn Restriktionen eingeführt werden. Bei Grammatiken höheren Typs wird dies durchgeführt.

Typ 1-Grammatik, kontextsensitive Grammatik

Bei kontextsensitiven Grammatiken werden die Produktionen bezüglich ihrer Ersetzungsmöglichkeit eingeschränkt. Bei allgemeinen Grammatiken können mehrere Gruppen von Terminal- und Nonterminalsymbolen ersetzt werden. Im Gegensatz dazu, können Produktionen von Typ 1-Grammatiken nur mehr genau ein Nonterminalsymbol ersetzen. Trotzdem ist die Ersetzung immer noch von der Umgebung des zu ersetzenden Nonterminals abhängig. Die Produktionen einer kontextsensitiven Grammatik haben das Aussehen

$$\alpha A \beta \rightarrow \alpha B \beta \text{ und } \alpha, \beta \in (N \cup T)^+, B \in (N \cup T)^+ \text{ und } A \in N$$

Das Zeichen A ist hier das Nonterminalsymbol, das durch B ersetzt wird. α und β sind die jeweilige geforderte Umgebung (Kontext), damit die Ersetzung stattfinden kann. Das gleiche Nonterminalsymbol kann damit in unterschiedlichen Kontexten unterschiedlich ersetzt werden.

Weiters muss bei kontextsensitiven Grammatiken noch beachtet werden, dass die Produktion

$$S \rightarrow \varepsilon$$

mit S gleich dem Startsymbol und ε dem Leerwort, enthalten kann. Allerdings darf in diesem Fall das Startsymbol S bei allen anderen Produktionen der Grammatik auf der rechten Seite auftauchen. Diese Produktion findet vor allem dann Verwendung, damit das Leerwort mit Hilfe der Grammatik hergeleitet werden kann.

Für jede Produktion gilt außerdem, dass das Ergebnis in der Anzahl der Symbole größer oder gleich groß wie die ersetzten Nonterminalsymbole. Das heißt

$$\forall p(\alpha, \beta) \in P \text{ gilt } |\alpha| \leq |\beta|$$

wobei α die rechte Seite der Produktion und β die linke Seite darstellt. Dadurch wird sichergestellt, dass sich die Länge des Resultats niemals verringern kann. Deshalb wird die kontextsensitive Grammatik auch monotone Grammatik genannt, da sie durch eine Produktion immer gleich große oder größere Worte hervorbringt.

Typ 2-Grammatik, kontextfreie Grammatik

Kontextfreie Grammatiken sind noch mehr in der Auswahl ihrer Produktionen eingeschränkt als kontextsensitive Grammatiken. Nur mehr Produktionen die genau ein Nonterminalsymbol ersetzen, sind in Typ 2-Grammatiken erlaubt. Bei dieser Form ist auch der Kontext in dem das Nonterminalsymbol auftaucht, nicht mehr ausschlaggebend. Produktionen von kontextfreien Grammatiken haben die Form

$$A \rightarrow \alpha, \text{ wobei } A \in N \text{ und } \alpha \in (N \cup T)^*$$

A ist hierbei das Nonterminalsymbol, das durch den Ausdruck α ersetzt wird, welcher sich wiederum aus Nonterminal- und Terminalsymbolen zusammensetzt. Zudem ist es erlaubt, α als Leerwort zu definieren. Dadurch ist es bei Bedarf möglich, Nonterminalsymbole einfach aus dem Ergebnis zu entfernen. Weiters muss innerhalb einer kontextfreien Grammatik bei gleich gewähltem Startwort unabhängig von der Art der verwendeten Ableitung das exakt gleiche Wort der Sprache entstehen.

Durch die Kontextfreiheit in dieser Form der Grammatik können Produktionen für ein Nonterminalsymbol unabhängig gewählt werden. Dadurch sind sie weitaus effizienter und leichter einsetzbar. Allerdings tauchen auch bei kontextfreien Grammatiken diverse Entscheidungsprobleme auf. Vergleiche mit anderen kontextfreien oder auch kontextsensitiven Grammatiken gestalten sich schwierig. Es kann nämlich nicht immer klar entschieden werden, ob die verglichenen Grammatiken ein und dieselbe Sprache erzeugen oder nicht.

Typ 3-Grammatik, reguläre Grammatik

Reguläre Grammatiken sind bezüglich ihrer Produktionen die eingeschränkteste Art der Grammatiken. Bei dieser Form der formalen Grammatik sind nur jeweils ein Nonterminalsymbol auf der rechten Seite und ebenfalls nur ein Nonterminalsymbol auf der linken Seite erlaubt. Auch die Anordnung dieses Nonterminalsymbols in der Ersetzung unterliegt strikten Regeln. Reguläre Grammatiken werden bezüglich dieser Position in zwei Untergruppen eingeteilt, in linksregulären und rechtsregulären Grammatiken.

Bei linksregulären Grammatiken können die Produktionen nur wie folgt gestaltet werden

$$A \rightarrow B\alpha, A \rightarrow \varepsilon, \text{ mit } A, B \in N \text{ und } \alpha, \varepsilon \in T$$

A und B sind hierbei die einzigen Nonterminalsymbole und B, als das Nonterminalsymbol auf der rechten Seite steht immer ganz links. Das Terminalsymbol α ist das einzige Terminalsymbol und steht immer ganz rechts. ε ist wiederum das Leerwort.

Im Gegensatz dazu stellen sich die Produktionen bei rechtsregulären Grammatiken so dar

$$A \rightarrow \alpha B, A \rightarrow \varepsilon, \text{ mit } A, B \in N \text{ und } \alpha, \varepsilon \in T$$

Lediglich die Position des Nonterminalsymbols B hat sich von ganz links nach ganz rechts verändert. Ansonsten ist die Definition mit der linksregulären Grammatik äquivalent. Beide Formen der regulären Grammatik dürfen jedoch innerhalb einer Grammatik nicht vermischt werden. Andernfalls wäre die Grammatik nicht mehr regulär. Zu erwähnen ist, dass zu jeder linksregulären Grammatik eine äquivalente rechtsreguläre Grammatik existiert.

Die reguläre Grammatik kann zusätzlich bei Bedarf erweitert werden. Man kann das Terminalsymbol α sowohl in der rechtsregulären als auch in der linksregulären Grammatik durch ein Wort aus Terminalsymbolen der Grammatik ersetzt. Dadurch können mehrere Terminalsymbole innerhalb eines Ersetzungsschrittes in das Resultat mit eingebracht werden. Natürlich dürfen keine weiteren Nonterminalsymbole hinzugefügt werden. Die Position des einzigen Nonterminalsymbols auf der rechten Seite der Produktion muss ebenfalls erhalten bleiben.

6.3 L-Systeme

Lindenmayer Systeme (L-Systeme) wurden nach ihrem Schöpfer dem ungarischen Biologen Aristid Lindenmayer benannt. Dieser wirkte zwischen 1925 und 1989. In dieser Zeit versuchte er unter anderem die Biologie und die Informatik im Bereich der Flora zu vereinen. 1968 gelang ihm der erste und wichtigste Schritt in diesem Wirken. Er entwickelte die L-Systeme. Zu diesem Zeitpunkt kannte er noch nicht das volle Ausmaß der Einsatzgebiete. Diese eröffnet sich erst später in der Zusammenarbeit mit Peremyslaw Prusinkiewicz. L-Systeme werden am besten durch [L1], [L2], [L7] und [L9] abgedeckt.

Zur Seite standen ihm bei diesem Wirken viele Wissenschaftler, die alle die Bedeutung der L-Systeme erahnten. Vor allem jedoch sind der Informatiker Peremyslaw Prusinkiewicz und der Mathematiker Grzegorz Rozenberg mit der Geschichte von Aristid Lindenmayer und seinen L-Systemen verweben. „The Algorithmic Beauty of Plants“ war das Glanzstück von Peremyslaw Prusinkiewicz und Aristid Lindenmayer, welches sie gemeinsam verfassten. Leider konnte Aristid Lindenmayer die Veröffentlichung dieses Werkes nicht mehr miterleben.

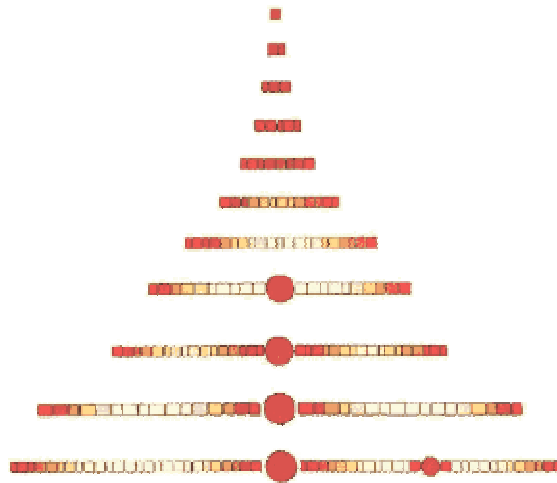


Abbildung 9: Algenwachstumssimulation mit L-System

Die ersten Anwendungsgebiete der L-Systeme waren die Simulationen von simplem, pflanzlichem Leben (siehe Abbildung 9). Mit Hilfe von L-Systemen versuchte Lindenmayer das Leben von Algen so genau wie möglich künstlich nachzubilden. Vom Beginn ihres Lebens, der Teilung dieser Algenzellen, über deren Wachstum bis hin zur ihrem Tod wurde alles nachgestellt. Da diese Versuche sehr erfolgreich waren, begann man sie ebenfalls für höhere Pflanzenstrukturen und deren Verhalten zu verwenden. Die L-Systeme waren für diesen Bereich grundsätzlich nicht ausgelegt, aber es stellten sich trotzdem Erfolge ein.

Ermöglicht wird dieses anpassungsfähige und leicht dehbare Anwendungsgebiet durch die Tatsache, dass L-Systeme mit einer hohen Informationsdichte ausgestattet sind. Dadurch erklärt sich ihre Anpassungsfähigkeit und ihre unabhängig zum ursprünglichen Kontext. L-Systeme beschränken sich lediglich auf die Erfassung des Entwicklungs- und Wachstumsgedanken und formen ihn zu einer mathematischen Theorie um. Diese lässt sich einfach auf alle Gebiete anwenden, in denen eine Entwicklung stattfindet.

Mit dem Aufkommen der Computertechnologie waren auch der Visualisierung der L-Systeme keine Grenzen mehr gesetzt. Durch die besondere Form der L-Systeme war es sogar sehr leicht möglich, die Visualisierung in einer universellen Art und Weise umzusetzen. In Form der „Turtle Grafik“ wurde dies umgesetzt. Später im Kapitel 6.3.2 wird diese Grafik noch ausführlicher behandelt. Damit ist sowohl eine einfache Darstellung eines L-Systems in einem statischen Zustand möglich als auch eine dynamische Betrachtung der vollständigen Entwicklung des L-Systems. Damit sind auch die zugrunde liegenden Strukturen einer Vegetation grafisch darstellbar.

L-Systeme stellen eine Mischung zwischen fraktaler Geometrie und formalen Grammatiken dar. Dadurch werden einige wichtige, vorteilhafte Merkmale von beiden Seiten zu einem einzigen System vereint. Durch die Verwandtschaft zu Fraktalen besitzen L-Systeme den Vorteil mit nur wenigen Ausgangsinformationen, umfangreiche Gebilde zu erschaffen. Da in der Natur die Selbstähnlichkeit weit verbreitet ist, kann sich dieser Selbstähnlichkeit zwischen verschiedenen Vegetationsstrukturen leicht bedient werden. Von den Grammatiken ererben sich die L-Systeme die leichte Anwendbarkeit, da Grammatiken und auch L-Systeme einem simplen Ersetzungsprinzip unterliegen.

6.3.1 Grundlagen

L-Systeme basieren in ihrer Bearbeitung auf einer Mischung aus fraktaler Geometrie und Grammatiken. Durch ihren rekursiv zyklischen Verlauf leiten sich die fraktalen Eigenschaften ab. Auf der anderen Seite stehen die Ersetzungsregeln, die sie aus den formalen Grammatiken erhalten. L-Systeme haben jedoch nicht alle Eigenschaften einer vollständigen formalen Grammatik, wie Noam Chomsky sie klassifiziert hat. Auch besitzen sie nicht alle Merkmale eines Fraktals. Allerdings kann man L-System trotz dieses Mangels an Eigenschaften gut in die Klassen beider Bereiche eingliedern. Die Grundlagen der L-Systeme wurden durch [L6] und [P1] ergänzt.

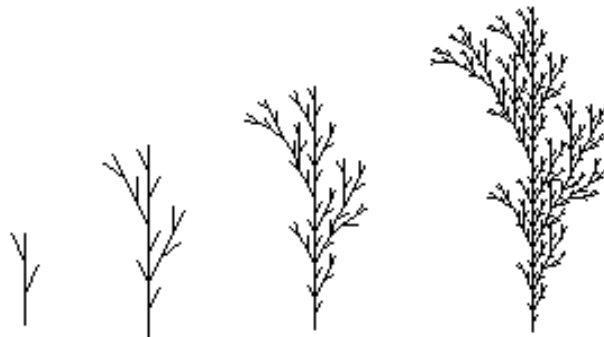


Abbildung 10: Ableitung eines L-System

So besteht in grammatikalischer Sicht eine deutliche Abweichung bezüglich der Anwendung der Produktionen. Bei formalen Grammatiken im üblichen Sinn wird bei Ersetzungen gefordert, dass sowohl eine Rechtsableitung, eine Linksableitung als auch eine Parallelableitung dieselbe Sprache erzeugt. Bei L-Systemen wird nur eine Parallelableitung durchgeführt. Alle anderen Ableitungsarten haben keinerlei Bedeutung für ein L-System. Diese Ableitung wird zudem auf alle Symbole gleichzeitig angewandt, die in einem Iterationsschritt vorhanden sind (siehe Abbildung 10).

Möglich ist dies nur, da bei L-Systemen auf eine Trennung in Terminal- und Nonterminalsymbole verzichtet wird. Dies ist der zweite Unterschied zu einer formalen Grammatik. Bei L-Systemen werden Symbole als sowohl Terminal als auch Nonterminal gesehen. Damit ist die Möglichkeit gegeben, die erhaltenen Symbole in jedem Iterationsschritt als Wort der Sprache des L-Systems zu sehen. Auch kann es einer weiteren Ableitung unterzogen werden. Dadurch wird die Sprache insgesamt größer und jedes Wort hat einen direkten Vorgänger. Jedes Wort in einem L-System ist damit automatisch ein terminales Wort.

Jedes Symbol, das in der Menge der Produktionen keine explizite Ableitung enthält, erhält die idente Abbildung also

$$A \rightarrow A$$

als Produktion hinzugefügt. Dadurch lässt sich auch in solch einem Fall die vollständige, parallele Ersetzung durchführen.

Im Bereich der Fraktale ist die geforderte Selbstähnlichkeit des L-Systems nicht immer gegeben. Sie kann durchaus bei entsprechender Definition des L-Systems erfüllt werden. Allerdings ist dies, besonders im Bereich der Erstellung von L-Systemen von Vegetation nicht immer wünschenswert und zielführend. Pflanzen weisen durchaus im Allgemeinen in ihrer Struktur eine gewisse Selbstähnlichkeit auf. Diese ist jedoch weitaus geringer in ihrem Ausmaß als bei einem typischen Fraktal.

Zudem haben L-Systeme keinen fraktaltypischen Konvergenzpunkt und keine unendlich feine Struktur. L-Systeme sind nicht wie echte Fraktale kontraktiver Natur. Pflanzen haben im Allgemeinen die Tendenz sich durch ihr Wachstum und ihre Entwicklung auszudehnen. Dabei würde eine Kontraktion keinerlei Sinn machen. Daher muss auf dieses Merkmal der Fraktale bei der Verwendung von L-Systemen zur Erstellung natürlicher Strukturen ebenfalls verzichtet werden. Der Vergleich mit anderen Fraktalen ist aber trotz dieser Änderungen nicht behindert.

6.3.2 Definition

Die Definition der L-Systeme wurde durch [L4], [L5] und [L8] ergänzt. L-Systeme definieren sich ähnlich wie typische formale Grammatiken. Allerdings besitzen sie keine Unterteilung in Terminal- und Nonterminalsymbole. Daher beschränkt sich die Definition auf

$$L = \{A, P, S\}$$

wobei sich das L-System L aus A dem Alphabet, der Menge aller Symbole, P den Produktionen (der Menge aller Ersetzungsregeln) und S dem Startsymbol zusammensetzt. Alle Symbole aus dem Alphabet A besitzen ebenfalls eine zugehörige Produktion in der Menge P.

Da L-Systeme nicht zwingend die Monotonieeigenschaft einer formalen Grammatik besitzen müssen, ist auch automatisch die Produktion mit Leerwort erlaubt

$$\alpha \rightarrow \varepsilon$$

wobei α ein Symbol aus A darstellt und ε das Leerwort, sowie Produktionen der Gestalt

$$\alpha \rightarrow \beta$$

wobei $\alpha \rightarrow \beta \in P$ und $|\alpha| > |\beta|$ ist.

Definiert werden L-Systeme meist im Rahmen einer Typ-1 (kontextsensitiv) oder Typ-2 (kontextfrei) formalen Grammatik. Wie bereits im Kapitel 6.2 beschrieben, verhalten sich auch L-Systeme wie die entsprechenden formalen Grammatiken. Allerdings muss auf die bereits erklärten Abweichungen geachtet werden. Kontextfreie L-Systeme werden im Allgemeinen mit einem vorangehenden 0L gekennzeichnet und kontextsensitive L-Systeme mit dem Grad X ihrer Sensitivität XL.

Durch die zusätzliche Definitionsmöglichkeit der Produktionen und den beschriebenen Abweichungen, sind L-Systeme immer mächtiger als vergleichbare Grammatiken. Dies betrifft sowohl den kontextsensitiven als auch den kontextfreien Fall. Dadurch sind die

erstellten Sprachen immer mächtiger und auch die Menge aller durch L-Systeme definierten Sprachen immer größer. Die vergleichbare grammatikalisch erstellte Menge von Sprachen ist damit immer eine Teilmenge der durch L-Systeme erzeugten Sprachenmenge. Im weiteren Verlauf wird von kontextsensitiven bzw. kontextfreien L-Systemen gesprochen. Diese sollten nicht mehr mit kontextsensitiven bzw. kontextfreien, formalen Grammatiken nach Chomsky verwechselt werden.

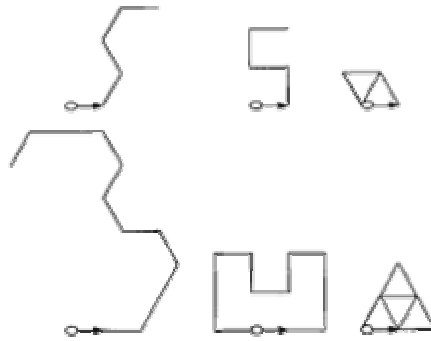


Abbildung 11: Beispiel für die Bewegungen einer Turtle

Dargestellt werden L-Systeme mit Hilfe der sogenannten „Turtle Geometrie“ oder auch „Turtle Grafik“ (siehe Abbildung 11). Bei dieser Art der Darstellung wird eine Turtle (Positionshalter) über die Zeichenfläche bewegt. Die zurückgelegte Strecke wird dabei durch geometrische Formen dokumentiert. Die so entstehende Geometrie stellt die Visualisierung des L-Systems dar. Durch einfache syntaktische Zeichen, die man in das Alphabet eines L-Systems einfügt, kann man der Turtle diverse Befehle zum Zeichnen eines L-Systems vermitteln. Diese Zeichen haben jedoch immer nur die idente Abbildung als Produktion. Die Möglichkeiten reichen hier von einfachen Vorwärtsbewegungen, über Rotationen und Skalierungen, bis hin zu rekursiv definierten Abzweigungen. Vereinfacht bedeuten Abzweigungen für die Turtle nichts weiter als einen Rücksprung innerhalb des zurückgelegten Weges.

6.3.3 Arten der L-Systeme

L-Systeme lassen sich durch ihre Merkmale in verschiedene Gruppen unterteilen. All diese Merkmale sind untereinander kombinierbar und ergeben so eine große Fülle an verschiedensten L-Systemen. Eines dieser Merkmale ist ihre Dimension, die sie im Raum einnehmen. So können einfache L-Systeme bereits in einem 1D Raum dargestellt werden, wogegen komplexere L-Systeme zwei, drei oder sogar mehr Dimensionen einnehmen können. Dimensionsgrößen über drei können natürlich nicht mehr ganz einfach dargestellt werden und sind daher nur extrem selten in ihrer Verbreitung.



Abbildung 12: Stochastisches L-System

Ein weiteres Kriterium zur Unterteilung stellt die Tatsache dar, ob ein L-System deterministisch ist oder einer stochastischen Verteilung unterliegt (siehe Abbildung 12). Deterministische L-Systeme eignen sich besonders gut, wenn Reproduzierbarkeit gebraucht wird. Sie liefern immer die exakt gleichen Ergebnisse, wenn sie durchlaufen werden. Deterministische L-Systeme werden mit einem D in der Bezeichnung des L-Systems als solche gekennzeichnet. Stochastische L-Systeme dagegen können dazu genutzt werden, ähnliche Modelle eines Typs hervorzubringen. Dabei müssen keine weiteren Änderungen bei der Definition der einzelnen Modelle vorgenommen werden.

Weiters ist die Verwendung von Parametern bei der Erstellung des L-Systems ein Merkmal zur Unterscheidung. Durch die Ausstattung mit Parametern in der Definition der Ersetzungsregeln, besitzen L-Systeme eine größere Mächtigkeit bezüglich ihrer Form und ihres Aussehens. Die Parameter selbst können auf das L-System und dessen Entwicklung Einfluss nehmen. Auch können Parameter verwendet werden, die keinen direkten Einfluss auf das L-System haben.

Ebenfalls als eine zu unterscheidende Eigenschaft ist die Beeinflussung der Entwicklung des L-Systems durch ihre Umgebung. Diese Eigenschaft unterscheidet kontextsensitive und kontextfreie L-Systeme. Kontextsensitive L-Systeme lassen ihre Beschaffenheit als Einflusskriterium in ihren Ersetzungen zu. Kontextsensitive L-Systeme sind dabei an kontextsensitive Grammatiken angelehnt, die ebenfalls in ihrer Entwicklung an ihre Umgebung gebunden sind. Sie werden auch als „Offene L-Systeme“ bezeichnet. Kontextfreie L-Systeme hingegen sind von ihrer Beschaffenheit und Umgebung vollständig frei.

1D, 2D, 3D-L-Systeme

1D-L-Systeme sind in einer Dimension darstellbar. Sie enthalten keinerlei dimensionale Größen oder Bewegungen in mehr als einer Dimension. Dabei ist es unabhängig wie

viele Iterationen das L-System durchläuft. Verwendung finden solche L-Systeme besonders im Bereich der Modellierung wachsender Größen wie der Simulation der Geschwindigkeit von Pflanzenwachstum. Da sie jedoch zumeist nur eine Größe über die Zeit hinweg simulieren können, sind sie sehr eingeschränkt in ihrer Verwendbarkeit.

Für Ersetzungsregeln P jedes 1D-L-Systeme in den Reellen Zahlen \mathbb{R} gilt:

$$P: \mathbb{R}^1 \rightarrow \mathbb{R}^1$$

2D-L-Systeme verfügen bereits über dimensionale Änderungen, die sie wesentlich flexibler in ihrem Gebrauch machen. Fraktale Geometrien, wie die „Koch Kurve“ oder die „Drachen Kurve“ können mit diesem L-System bereits hervorragend visualisiert werden. Durch die einfache Handhabung und bereits gute Visualisierbarkeit sind 2D-L-Systeme die gebräuchlichsten ihrer Gruppe.

Für Ersetzungsregeln P jedes 2D-L-Systeme in den Reellen Zahlen \mathbb{R} gilt:

$$P: \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

3D-L-Systeme bieten die umfangreichste Möglichkeit ein L-System grafisch darzustellen. Durch die freie Beweglichkeit und Modellierungsmöglichkeit im Raum können sehr komplexe L-Systeme immer noch sehr übersichtlich dargestellt werden. Durch die höhere Dimension müssen Rotationen im Raum umgesetzt werden. Dadurch finden diese L-Systeme nicht ganz so oft Verwendung. Sie werden oft durch einfachere L-Systeme ersetzt, die sich in einer niedrigeren Dimension befinden.

Für Ersetzungsregeln P jedes 3D-L-Systeme in den Reellen Zahlen \mathbb{R} gilt:

$$P: \mathbb{R}^3 \rightarrow \mathbb{R}^3$$

Stochastische L-Systeme

Stochastische L-Systeme beinhalten, zusätzlich gegenüber deterministischen L-Systemen, Wahrscheinlichkeiten für die Wahl von Ersetzungsregeln für ein Symbol. Dadurch können mehrere Ersetzungsregeln für ein und dasselbe Symbol in der Menge aller Ersetzungsregeln vorhanden sein. Die durch diese Systeme erstellten Modelle sind nicht mehr identisch, sondern nur mehr stochastisch gleich. Dies ist eine Ähnlichkeit, wie sie bei fraktalen Strukturen ebenfalls zu finden ist.

Bei stochastischen L-Systemen $L = (A, P, S)$ gilt

$$\forall a \in A \text{ gilt, } \forall p \in P \text{ mit } a \rightarrow \alpha$$

und einer Wahrscheinlichkeitsverteilung von

$$\pi_a: P \rightarrow (0,1], \text{ dass} \\ \sum \pi_a = 1$$

Durch diese Vergabe von Wahrscheinlichkeiten bei der Wahl der Ersetzungen können unterschiedliche Strukturen erzeugt werden. Diese besitzen nach den gewählten Wahrscheinlichkeiten mehr oder weniger Ähnlichkeit.

Parametrische L-Systeme

Durch die Parametrisierung eines L-Systems kann das L-System selbst in seiner Entwicklung beeinflusst werden. Auch Parameter für eine eventuelle Weiterverwendung können inkludiert werden. Diese müssen die Entwicklung des L-Systems jedoch nicht beeinflussen. Jeder Parameter in einem derartigen L-System ist an ein Symbol gebunden. Wird eine Iteration durchgeführt so vererben sich die Parameter weiter. Sie können zudem bei jeder Iteration verändert werden. Diese Vererbungen und Veränderungen können nach Belieben konditional erfolgen.

Parametrische Symbole aus dem Alphabet A haben folgende Form:

$$a \in A, \text{ mit } a(d_1, d_2, \dots, d_n), \text{ wobei } d_1, d_2, \dots, d_n \in R$$

mit Produktion P, wobei für alle $p \in P$ gilt

$$p : a(d_1, d_2, \dots, d_n) : k \rightarrow \alpha$$

mit Kondition k abhängig von Parameter d_1, d_2, \dots, d_n und $\alpha \in A^+$. α kann dabei beliebige Werte der Parameter der jeweiligen eingefügten Symbole beinhalten.

Ersetzungen von Symbolen können daher nur vorgenommen werden, wenn diese auch über die entsprechende Anzahl von Parametern verfügen. Andererseits können dadurch auch Symbole mit gleichem Namen in einem Alphabet A eines L-Systems bestehen, wobei jedes dieser Symbole natürlich eine unterschiedliche Anzahl von Parametern besitzen muss, damit eine eindeutige Identifikation des jeweiligen Symbols möglich ist.

Kontextsensitive L-Systeme

Bei einem kontextsensitiven L-System fließt der Kontext in dem sich das jeweilige L-System befindet mit in die weitere Entwicklung des L-Systems ein. Dabei können sowohl der dem Symbol vorangegangene Bereich (linksseitig) als auch der nachfolgende Bereich (rechtsseitig) wie auch beide als Kontext gesehen werden. Kontextfreie L-Systeme werden mit 0L, einseitig kontextsensitive mit 1L und beidseitig kontextsensitive mit 2L gekennzeichnet.

Dabei geben die kontextsensitiven L-Systeme aber nicht die Anzahl der Vor- oder Nachfolger an, an die sie gebunden sind. Es existiert auch die Möglichkeit, sie

detaillierter zu definieren und zu beschreiben. Dies passiert in der Form $(l, r)L$ -System. Dabei gibt l die maximale Länge des Kontexts in Symbolen vor dem entsprechenden Symbol und r die maximale Länge des Kontexts danach an.

Die Ersetzungsregeln mit Kontextbindung haben die Form

$$\begin{aligned} a_l < a &\rightarrow \alpha \text{ für linksseitige Bindung} \\ a > a_r &\rightarrow \alpha \text{ für rechtsseitige Bindung} \\ a_l < a > a_r &\rightarrow \alpha \text{ für beidseitige Bindung} \end{aligned}$$

wobei $a_l \in A^+$, $a_r \in A^+$ und A Alphabet des L -Systems.

6.4 Rauschen

Die Natur zeigt immer wieder, wie vielfältig und unberechenbar sie ist. Viele Phänomene und Ereignisse beruhen dabei auf Zufälligkeiten. Um nun aber natürliche Phänomene wie Winde zu simulieren, muss man diese Zufälligkeiten verstehen. Der „Zufall“ in der Natur ist nämlich nicht völlig wahllos. Nur durch dieses Wissen ist es möglich, eben jene Zufälligkeiten nachzubilden und sie für physikalisch korrekte Simulationen zu nutzen. Die Informationen zu diesem Kapitel beruhen vor allem auf [P2], [P4], [P5] und [P6].

Eine dieser Zufälligkeiten ist Rauschen oder auch Noise genannt (siehe Abbildung 13). Erstmals wurde Noise als solches 1918 von Walter Schottky erkannt. In den „Annalen der Physik“ wurde Noise erstmals namentlich festgehalten. Schottky fand Noise erstmalig als messbare Stromschwankung. Er konnte diese auch mit Hilfe eines Verstärkers sowie eines Lautsprechers hörbar machen. In dieser Zeit war Noise noch sehr spezifisch definiert. Erst später sollte Noise in vielen verschiedenen anderen Gebieten der Natur ebenfalls gefunden werden. Dadurch entstand letztendlich eine allgemeine Definition für Noise.

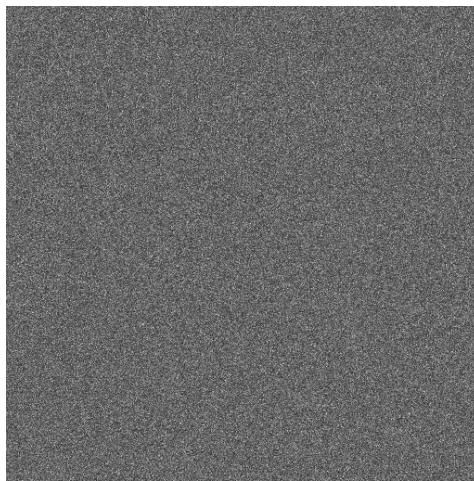


Abbildung 13: Visualisierung von Rauschen

Aber selbst Zufälligkeiten haben untereinander gewisse Unterschiede. Sie können dadurch in verschiedene Gruppen unterteilt werden. Diese Unterschiede basieren auf der unterschiedlichen Korrelation zwischen den einzelnen Zufallswerten. Noise ist damit nicht immer völlig wahllos, sondern folgt durchaus gewissen Vorgaben. Diese Vorgaben müssen erkannt werden, um sie nutzbar zu machen. Aufgrund dieser Vorgaben kann Noise auf korrekte Art und Weise simuliert werden.

Selbst die Vielfalt der Natur basiert auf einer Zufälligkeit, die unter dem Namen „Pink Noise“ bekannt ist. „Pink Noise“ ist nicht völlig zufällig, sondern es besteht eine Korrelation zwischen den einzelnen Zufallswerten. Daher sind Kenntnisse über den Aufbau von „Pink Noise“ notwendig, um das natürliche Phänomen Wind richtig zu simulieren. Nur so können Kräfte und Winde möglichst genau wiedergegeben werden. Da Approximationen auf Basis von unkorrelierter Zufälligkeit arbeiten, ist das Wissen über „Pink Noise“ umso wichtiger.

Die unkorrelierte Verteilung ist unter dem Namen „White Noise“ bekannt. Sie ist neben „Pink Noise“ eine der bekanntesten Noisearten. „White Noise“ ist im eigentlichen Sinne nur theoretisch möglich. Er kann aber leicht künstlich in einem begrenzten Frequenzbereich hergestellt werden. Dadurch ist sie eine der häufigst verwendeten Noisearten. In diesem Zusammenhang sollte ebenfalls „Brown Noise“ erwähnt werden. Bei „Brown Noise“ oder „Brown'sche Bewegung“ handelt es sich allerdings um eine in die Welt der Fraktale fallende Noiseart.

Es empfiehlt sich in diesem Bereich, Näherungsverfahren zu verwenden. Diese passen sich soweit wie möglich der natürlichen Verteilung der Natur an. Für die angestrebte Bewegungsanimation müssen diese Verfahren ausreichend genau sein und trotzdem effizient berechnet werden können. Damit ist sowohl die Wahl der verwendeten Verteilung als auch das Verfahren zur Approximation dieser Verteilung wichtig. Das Verfahren sollte möglichst exakt und ausreichend effizient sein. Zudem ist es notwendig, die Approximation in entsprechenden Bahnen steuern zu können. Dadurch sollte das Verfahren eine Möglichkeit zur Parametrisierung beinhalten.

6.4.1 Grundlagen

Noise wird allgemein als Signal oder Welle dargestellt. Dieses leitet sich vor allem von den Gebieten ab, in der Noise erstmalig auftrat. Außerdem ermöglicht es eine einfachere Bearbeitung von Noise und einer besseren Möglichkeit der Darstellung. Noise wird in der Fachliteratur sowohl als Störgröße als auch Verteilung betrachtet. Von zentraler Bedeutung für die Zuordnung der verschiedenen Verteilungen in Noisearten ist die spektrale Leistungsdichte. Diese beschreibt als messbare physikalische Größe Noise am besten. Die Grundlagen wurden durch [P1] und [P7] ergänzt.

Noise kann als Signal oder Funktion mit einer zugehörigen spektralen Leistungsdichte. Die spektrale Leistungsdichte gibt die Leistung des Signals in einem (infinitesimal kleinen) Frequenzband an. Dieses Frequenzband ist ein Teilbereich des Spektrums des Signals.

Die Definition von spektraler Leistungsdichte $S_{xx}(w)$ ist die Fouriertransformation der zeitlichen Autokorrelationsfunktion

$$R_{XX}(\tau) = \lim_{T_F \rightarrow \infty} \frac{1}{T_F} \int_{-T_F/2}^{T_F/2} x(t) \cdot x(t + \tau) dt$$

wobei die Autokorrelation die Korrelation des Signals mit sich selbst angibt. Mit der Zeit t , der zeitlichen Verschiebung τ , dem Zeitsignal $x(t)$ im Zeitabschnitt T_F ergibt sich die Autokovarianz R_{XX} . Die Fouriertransformation wird durch

$$S_{XX}(\omega) = \int_{-\infty}^{\infty} R_{XX}(\tau) e^{-i\omega\tau} d\tau$$

umgesetzt, wobei τ die zeitlichen Verschiebung, ω der Frequenz und i die imaginäre Einheit angibt. Die Fouriertransformation einer Integraltransformation zerteilt dabei den kontinuierlichen, aperiodischen Vorgang des Noisesignals in ein kontinuierliches Spektrum.

Durch diese Form der Beschreibung von Noise ist es ebenfalls möglich, gleichartige Noiseformen zu erkennen und in die gleiche Form einzuordnen. Dies ist wichtig, da sehr viele Benennungen in diesem Gebiet auf gleiche Noiseformen verweisen. Durch die Beschreibung mit Hilfe der spektralen Leistungsdichte kann somit Noise eindeutig festgelegt und benannt werden.

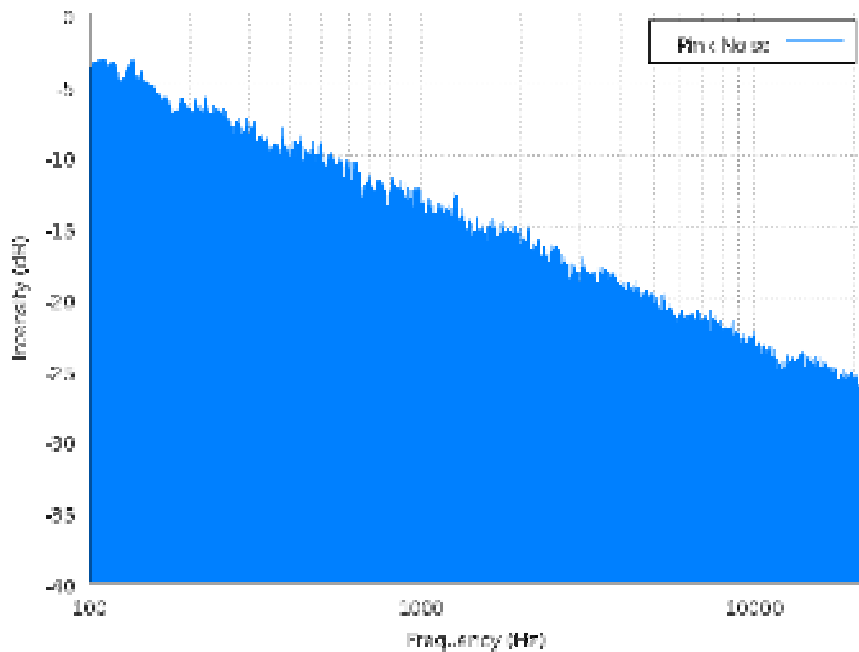


Abbildung 14: Amplitudenverteilung von Pink Noise

„Pink Noise“ kennt man auch unter dem Begriff $1/f$ -Noise (siehe Abbildung 14). $1/f$ bedeutet dabei, dass die Änderungen der spektralen Leistungsdichte mit steigender

Frequenz f abnehmen. Erklärbar ist dies durch die Betrachtung der Frequenz selbst. Je höher die Frequenz, desto höher ist auch der Energieaufwand, um eine Änderung zu erzeugen. Da die Energie über die Frequenzbereiche gleich bleibt, muss die spektrale Leistungsdichte bei höherer Frequenz abnehmen.

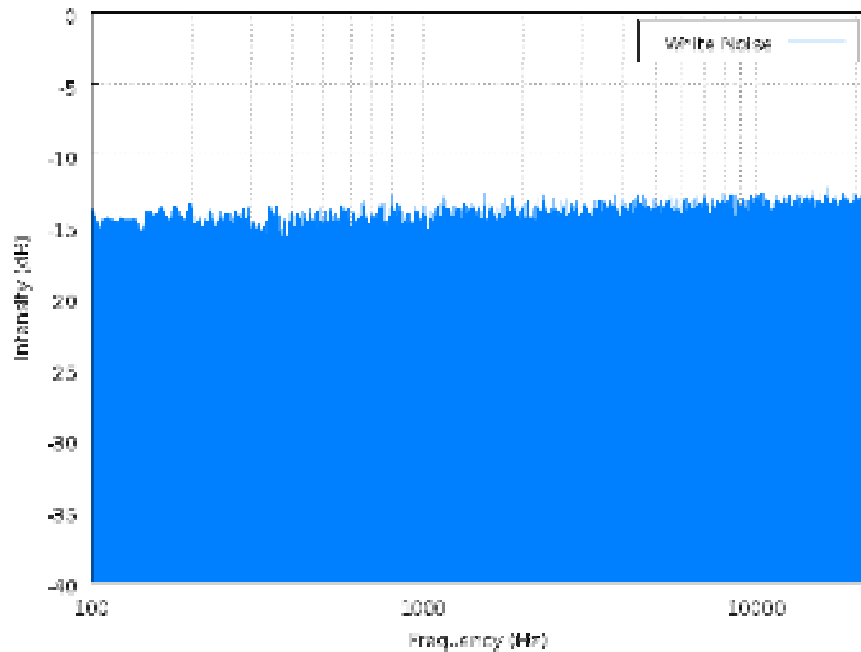


Abbildung 15: Amplitudenverteilung von White Noise

„White Noise“ hingegen ist ein völlig unkorreliertes Signal. Bei diesem Noise bleibt die spektrale Leistungsdichte über das gesamte Signal gleich (siehe Abbildung 15). Sie verringern sich nicht wie bei „Pink Noise“, was einem stetigen Energiezuwachs entsprechen würde. Echter „White Noise“ kommt gegenüber „Pink Noise“ nicht wirklich in der Natur vor, sondern ist eine rein theoretische Größe. Diese Form des Noise müsste eine unendliche große Gesamtleistung haben, da die Leistung über die Frequenzen stetig ansteigt. „White Noise“ tritt, wie auch andere Noiseformen, nur als „Quasi-White Noise“ auf und ist immer nur auf einen bestimmten Frequenzbereich begrenzt.

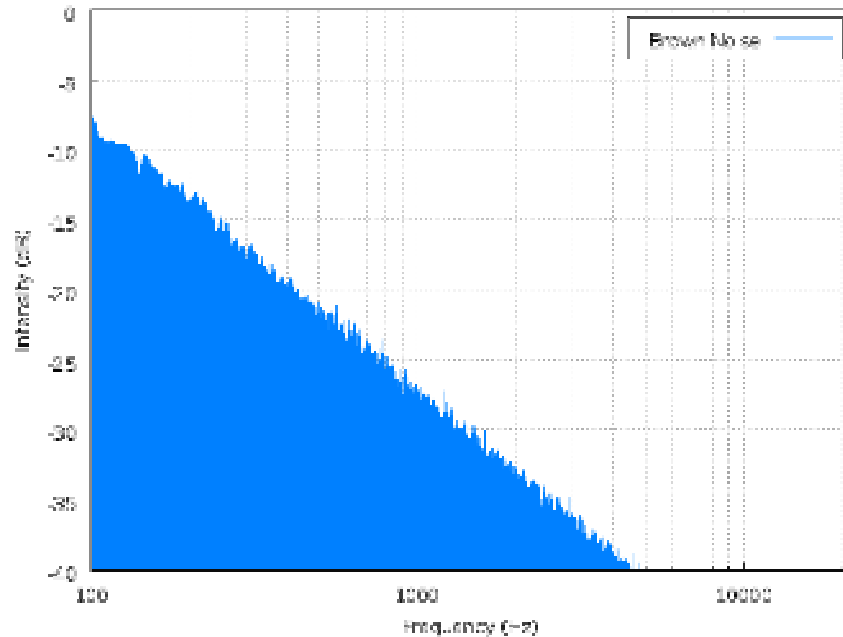


Abbildung 16: Amplitudenverteilung von Brown Noise

„Brown Noise“ oder „Brown’sche Bewegung“ ist ein stärker korrelierter Noise als „Pink Noise“. Er wurde durch seinen Erfinder Robert Brown benannt und wird auch als $1/f^2$ -Noise bezeichnet. Die spektrale Leistungsdichte nimmt bei dieser Noiseform mit steigender Frequenz sehr viel stärker (f^2) ab als bei „Pink Noise“ (siehe Abbildung 16). „Brown Noise“ wird oftmals als Ersatz für „Pink Noise“ verwendet. Grund dafür ist seine simple Erstellung durch die Integration von „White Noise“ und seine Eigenschaft der Korrelation. Langläufig wird diese Form von Noise auch „Red Noise“ genannt.

Um sich nun aber eine dieser Noiseformen nutzbar zu machen, muss auf ein Verfahren zugegriffen werden. Dieses bringt die entsprechende Noiseform in die gewünschte verwendbare Form. Eines dieser Verfahren nennt sich „Perlin Noise“. „Perlin Noise“ basiert für gewöhnlich auf „White Noise“. Trotz dieser völlig unkorrelierten Basis erzeugt „Perlin Noise“ Zufallszahlen, die sich für die Beschreibung von natürlichen Phänomenen gut verwenden lassen. Dieses Verfahren stellt eine Korrelation zwischen den Zufallswerte her, indem es zwischen diesen interpoliert.

„Perlin Noise“ wurde von Ken Perlin in den achtziger Jahren entwickelt. Der eigentliche Zweck von „Perlin Noise“ war im Bereich Computergrafik die Generierung von natürlich wirkenden Texturen. Die Verwendungsmöglichkeiten dieses Verfahrens blieben aber nicht auf dieses Gebiet beschränkt. Auch die Generation von Terrains, Geräuschen, Winden oder Wolken können mit „Perlin Noise“ simuliert werden. Mittlerweile wird selbst die Mimik von animierten Gesichtern durch „Perlin Noise“ imitiert.

Anwendung findet „Perlin Noise“ in allen Bereichen, in denen interpolierte Zufallszahlen benötigt werden. Dabei wird „Perlin Noise“ ein-, zwei-, drei- oder vierdimensional gebraucht. Die vierte Dimension wird meist als zeitliche Dimension verwendet. An sich ist der Algorithmus jedoch nicht dimensional gebunden und kann jederzeit auf einer beliebig hohen Dimension durchgeführt werden. Dadurch wird lediglich die Berechnungsdauer erhöht.

6.4.2 Definition

Der Algorithmus von „Perlin Noise“ beruht auf zwei Bereichen, der verwendeten Noisefunktion und der Art der Interpolation. Die Noisefunktion selbst basiert auf der Noiseform („White Noise“, „Pink Noise“, „Brown Noise“), die approximiert werden soll. Dabei können diese Werte durch den Einsatz von Verteilungsfunktionen noch zusätzlich steuerbar gemacht werden. Die Art der Interpolation reicht von einfacher linearer über Kosinus- bis hin zu kubischer Interpolation. Es kann jedoch auch jeder andere bekannte Interpolationsalgorithmus für „Perlin Noise“ genutzt werden. Im Interpolationsschritt kann zusätzlich ein Smoothing verwendet werden. Das Smoothing gestaltet den letztendlich erzeugten „Perlin Noise“ weicher. Auch bei den Definition wurde auf [P1] ergänzend zurückgegriffen.

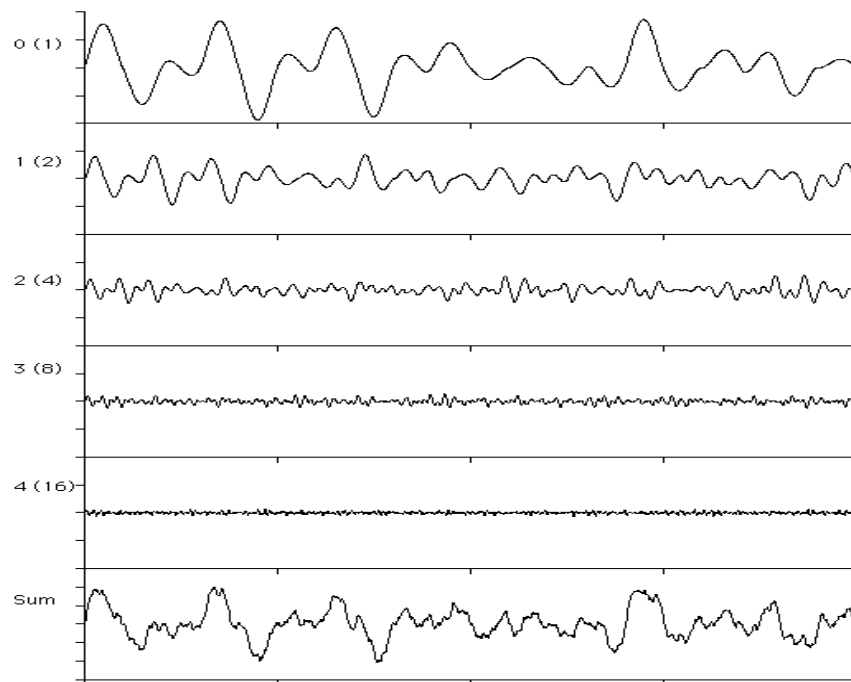


Abbildung 17: Aufbau von Perlin Noise auf Basis von mehreren Funktionen, beginnend mit niedriger Frequenz und hoher Amplitude bis hoher Frequenz und niedriger Amplitude, sowie der Summation

Die Noisefunktion beruht auf einer Reihe von zufälligen Zahlen. Diese werden aus der gewählten Noiseform entnommen. Dabei liegen diese Zahlen in einem Intervall zwischen $[-1, 1]$. Durch die Verwendung einer festgelegten Frequenz und der Wahl einer Amplitude werden diese Zufallszahlen in die Form einer Funktion gebracht. Die Frequenz gibt den Abstand zwischen den Zufallszahlen an. Die Amplitude wird mit den verschiedenen Zahlen multipliziert und entspricht der Auslenkung der Funktion.

Um nun eine für den „Perlin Noise“ brauchbar Funktion zu erhalten, muss dieser Schritt einige Male ausgeführt werden. Dabei werden die Frequenz und die Amplitude bei jedem Mal modifiziert. Die Frequenz erfährt jedes Mal eine Erhöhung und die Amplitude wird in jedem Schritt gesenkt. Die erhaltenen Funktionen werden danach zu einer einzigen Funktion aufaddiert. Große Änderungen treten dadurch nur selten auf. Geringere Frequenz und kleine Änderungen scheinen sehr oft auf. Dadurch wird eine natürlich wirkende Verteilung auf stochastisch, fraktale Weise erzeugt.

Jede dieser im Bereich Frequenz und Amplitude verschobenen Funktionen werden Oktaven genannt. Wie groß die Verschiebung gewählt wird, kann beliebig festgelegt werden. Es existieren jedoch Richtwerte, welche sehr gute Resultate liefern. Diese Richtwerte halbieren bei jeder Oktave die Amplitude der Funktion und verdoppeln gleichzeitig die Frequenz (siehe Abbildung 17). Die Anzahl, wie oft dies durchgeführt wird, hängt vor allem von der gebrauchten Genauigkeit ab. Dabei gilt, dass die Funktion nicht genauer sein sollte als die Möglichkeit, das Erzeugnis abzubilden. Die ursprünglichen Zufallszahlen bilden jedoch nicht von alleine eine Funktion.

Durch Interpolation zwischen diesen Zahlen wird eine durchgehende Funktion erreicht. Je nach Art der Interpolation weist diese Unterschiede auf. Jede existierende Form der Interpolationsverfahren trägt Vor- und Nachteile in sich. Diese Interpolationsverfahren können nicht lineare und nicht lineare Interpolationsverfahren unterteilt werden. Sie sind in Effizienz und Geschwindigkeit sehr unterschiedlich.

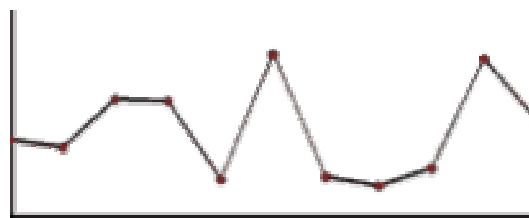


Abbildung 18: Kurve einer einfachen linearen Interpolation zwischen Zufallszahlen

Die einfache lineare Interpolation ist die schnellste und simpelste Methode, um die generierten Zufallszahlen zu einer Funktion zu vereinen (siehe Abbildung 18). Die Zufallszahlen werden dabei als Stützpunkte der Funktion verwendet. Die X-Koordinate kann dabei als Verteilung der Punkte aufgrund der Frequenz und die Y-Koordinate als Auslenkung der Funktion Anwendung finden. Dementsprechend wird jedes Punktepaar

(x, y) zwischen diesen Stützpunkten aufgrund der jeweiligen benachbarten Punktepaaren (ax, ay) und (bx, by) wie folgt berechnet

$$\forall (x, y) \text{ mit } x \in (ax, bx) \text{ und } f = \frac{x - ax}{bx - ax}$$

$$y = (ay * f) + (by * (1-f))$$

wobei f den Anteil des Einflusses von a und b widerspiegelt. Der Nachteil dieser Methode ist jedoch, dass die Funktion nicht sehr weich in den Übergängen ist und sehr hölzern wirkt.

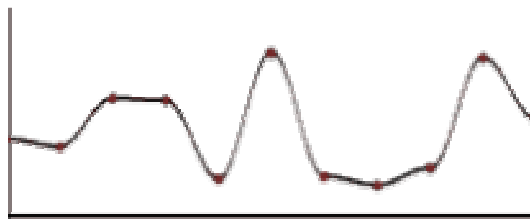


Abbildung 19: Kurve einer Kosinusinterpolation zwischen Zufallszahlen

Der Aufwand der Kosinusinterpolation gegenüber der einfachen linearen Interpolation ist geringfügig höher. Sie ist allerdings sehr viel weicher und eignet sich daher besser für die Imitation natürlicher Phänomene (siehe Abbildung 19). Geht man wiederum von der Anordnung der Zufallszahlen als Koordinaten aus, so gilt hier für (x, y)

$$\forall (x, y) \text{ mit } x \in (ax, bx), f = \frac{x - ax}{bx - ax} \text{ und } r = \frac{1 - \cos(f * \pi)}{2}$$

$$y = (ay * r) + (by * (1-r))$$

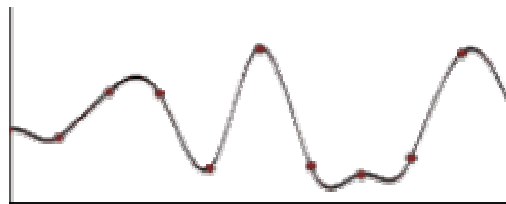


Abbildung 20: Kurve einer kubischen Interpolation zwischen Zufallszahlen

Bei der kubischen Interpolation werden im Unterschied zur linearen und Kosinusinterpolation nicht nur die zwei nächsten benachbarten Stützpunkte zur Berechnung herangezogen. Jeweils vier Stützpunkte dienen zur Berechnung der Kurve. Dabei werden je zwei Punkte beider Richtungen verwendet (siehe Abbildung 20). Durch diese Maßnahme wird die Interpolation sehr viel weicher, allerdings erhöht sich dadurch das Ausmaß des Aufwands. So gilt nun für (x, y) mit den Nachbarpunkten

$A1(a1x, a1y)$, $A2(a2x, a2y)$ und $B3(b3x, b3y)$, $B4(b4x, b4y)$, wobei $A1$ und $B4$ die jeweils weiter entfernten Nachbarn von x sind

$$\forall (x, y) \text{ mit } x \in (ax, bx) \text{ und } f = \frac{x - ax}{bx - ax}$$

$$P = (b4y - b3y) - (a1y - a2y)$$

$$Q = (a1y - a2y) - P$$

$$R = b3y - a1y$$

$$S = a2y$$

$$y = Pf^3 + Qf^2 + Rf + S$$

Nach der Anwendung eines Interpolationsverfahrens kann noch zusätzlich ein Schritt zur Glättung der Kurve angeschlossen werden. Durch dieses Verfahren können eventuelle Ausreißer abgeschwächt werden und der Kurve insgesamt zu einem schöneren Verlauf verholfen werden. Dadurch wirkt die Kurve weniger zufällig und stärker korreliert. Jegliches bekannte Verfahren zur Glättung von Funktionen oder auch geometrischen Formen kann hier Anwendung finden.

Natürlich ist es auch wichtig, das Ergebnis, vorher zu sehen bzw. die Herstellung selbst steuerbar zu machen. Dabei sind vor allem die verwendete Frequenz und die Amplitude der generierten Funktion betroffen. Auch die Änderungen bei der Erzeugung der jeweiligen Oktaven sind dabei ausschlaggebend. Außerdem beruhen die Zahlen immer noch auf einer Zufälligkeit. Diese muss ebenfalls einer entsprechenden Steuerung unterworfen werden können. Dafür können statistische Verteilungsfunktionen, wie eine Normalverteilung oder auch eine Poissonverteilung, verwendet werden. Damit wird die Zufälligkeit der Zahlen eingegrenzt.

6.4.3 Arten von Noise

Noise tritt in vielen Bereichen der Natur auf. Es ist möglich, die künstlich generierten und auch die natürlichen Noisearten zu strukturieren. Im Folgenden werden verschiedene Arten von Noise vorgestellt, die alle für die Generierung von „Perlin Noise“ verwendet werden können. [P9] und [P10] wurden zur Erstellung dieses Abschnittes verwendet.

Noise kann in verschiedener Dimensionalität erzeugt werden. Anwendungsgebiete bieten sich von eindimensionalen über zweidimensionalen bis hin zu vierdimensionalen Noise. Noise mit geringer Dimensionalität ist simpler und schneller zu berechnen als Noise in höherer Dimension. Bei vielen Berechnungsverfahren steigert sich der Umfang auf ein polynomielles Ausmaß. Dadurch muss sehr genau auf das ausgewählte Verfahren geachtet werden.

Lattice Noise

Noise kann je nach Dimensionalität in einem Gitter der gleichen Dimensionalität angeordnet werden. Diese Gruppe von Noise wird „Lattice Noise“ genannt. Das Gitter entspricht einer Anordnung von Zufallszahlen im Raum. Diese Zufallszahlen können bereits als Tabelle vorberechnet oder in eine Funktion gepackt werden. Um nun diese Zahlen zu einem Gitter zu formen, müssen sie miteinander verbunden werden. Die unter Kapitel 6.4.2 erklärte Methode ist ein „Lattice Noise“. „Lattice Noise“ kann als „Value Noise“, „Gradient Noise“ oder „Value-Gradient Noise“ aufgebaut werden.

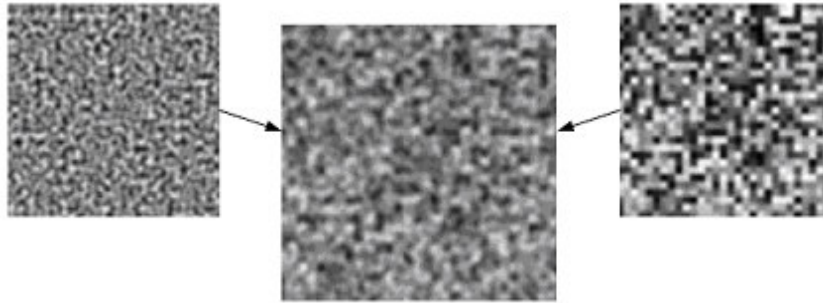


Abbildung 21: Vergleich „Gradient Noise“ (links) und „Value Noise“ (rechts), Kombination zu „Value-Gradient Noise“ (Mitte)

Value Noise

„Value Noise“ ist die schnellste aller Noisegruppen und basiert auf Zufallszahlen zwischen -1 und 1, zwischen welchen interpoliert wird (siehe Abbildung 21). Auf welche Weise diese Interpolation erfolgt, hat dabei ebenso Einfluss auf die Einordnung in eine Gruppe. Dabei kann zwar jede Interpolationsmethode herangezogen werden, wodurch es aber zu einer Änderung in der Zuordnung kommen kann. Trotzdem hat die Wahl der Interpolation enormes Gewicht bezüglich der Form des erhaltenen Noise. Die folgenden „Lattice Convolution Noise“ stellt eine Untergruppe dar. Dieser verwendet spezielle Interpolationen, um bessere Ergebnisse zu erhalten. Die unter das Kapitel 6.4.2 von Noise vorgestellte Methode fällt in die Gruppe von „Value Noise“.

Gradient Noise

„Gradient Noise“ ist die zweite große Gruppe von Noise (siehe Abbildung 21). Dieser Noise basiert zum Unterschied zu „Value Noise“ auf zufälligen Gradientenvektoren. Diese treten an die Stelle der Zufallszahlen als Basispunkte für die Berechnung der Noisefunktion. An allen Integerwerten der Funktion beträgt der Wert der Gradienten 0. Die Gradienten beeinflussen daher nur das Verhalten zwischen diesen Punkten. Interpoliert wird bei dieser Methode mit den direkten Nachbarn und auch mit den diagonalen Nachbarn im Gitter bei zwei und mehr Dimensionen. Bei dieser Form des

Noise wird die Richtung der Funktion bei jedem Basispunkt geändert und sie ist damit höher frequent als „Value Noise“.

Value-Gradient Noise

Beim „Value-Gradient Noise“ werden „Value Noise“ und „Gradient Noise“ zu einem Noise kombiniert (siehe Abbildung 21). Dies wird dadurch erreicht, dass die gewichtete Summe eines „Value Noise“ und eines „Gradient Noise“ erstellt wird. Diese Form des Noise ist deshalb entstanden, weil Gradient Noise an allen Integerwerten sichtbare Muster erzeugt. Diese entstehen dadurch, dass an allen Integerwerten Gradientwerte von 0 auftreten. Um diesen Nachteil zu umgehen und trotzdem hoch frequenten Noise zu behalten, wurde eine Kombination dieser zwei Noisegruppen herangezogen.

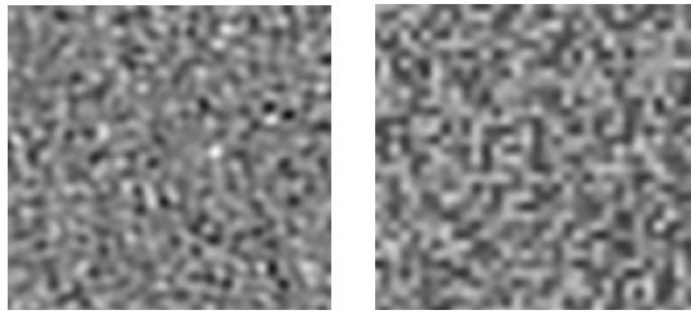


Abbildung 22: Vergleich „Sparse Convolution Noise“ (links), „Lattice Convolution Noise“ (rechts)

Lattice Convolution Noise

Eine Untergruppe von Value Noise stellt die Gruppe des „Lattice Convolution Noise“ dar (siehe Abbildung 22). Das Problem bei normalem „Lattice Value Noise“ ist, dass oft achsen-ausgerichtete Artefakte aufgrund von anisotropen Interpolationsmethoden zwischen den Basispunkten entstehen. Dadurch nimmt die Natürlichkeit der Verteilung ab. Daher verwendet man bei „Lattice Convolution Noise“ eine diskrete Faltungstechnik, um zwischen den Basispunkten zu interpolieren. Hierfür wird eine symmetrische Filterfunktion verwendet. Für die Berechnung eines Punktes wird die Summe der Produkte jedes Basispunktes mit dem Wert der Filterfunktion basierend auf der Entfernung des errechneten Punktes zum jeweiligen Basispunkt erstellt. Dadurch wird leicht ersichtlich, dass eben jene Artefakte durch einen größeren Einflussbereich nicht mehr so leicht entstehen können. Dies wird jedoch auf Kosten von umfangreicheren Berechnungen erkaufte.

Sparse Convolution Noise

Um Artefakte dieser Art zu umgehen, kann auch „Sparse Convolution Noise“ verwendet werden (siehe Abbildung 23). Dieser unterscheidet sich von „Lattice

Convolution Noise“ durch die Einschränkung der Zufälligkeit. Die Wahl der Zufallszahlen zur Berechnung eines Punktes in einem Gitter wird nicht regulär durchgeführt, sondern es wird auf Basis einer Poissonverteilung gearbeitet. Diese Poissonverteilung entnimmt zufällige Werte aus dem Gitter für die Berechnung eines jeden Punktes. So haben alle Punkte in dem Gitter die Möglichkeit, für die Berechnung eines Punktes herangezogen zu werden. Diese Art der Berechnung ist am aufwendigsten von allen Noisegruppen. Sie weist aber dafür am wenigsten Artefakte von allen auf.

6.5 Dynamische Systeme

Dynamische Systeme sind in dieser Arbeit auf einen Teilbereich eingegrenzt. Sie werden als ein mathematischer Formalismus gesehen. Dieser besteht aus einer formalen Sprache und den zugehörigen Ableitungsregeln (siehe Kapitel 6.2). Der Formalismus spannt dabei einen Raum auf, dessen Punkte in deren zeitlichem Verhalten beschrieben werden. Durch die große Vielfalt der Definitionsmöglichkeiten dieser Sprache und der anpassungsfähigen Ableitungsregeln, ist ein solches System sehr flexibel einsetzbar. Auch die Eigenschaften der Zeit und des Raumes in einem solchen System können beliebig festgelegt werden. Vor allem [M2], [M3], [M6] und [M9] wurden bei der Informationssammlung für dieses Kapitel herangezogen.

Durch diese Flexibilität erlaubt einem dieses Konzept viele Gebiete und Bereiche mathematischer Probleme zu lösen. So arbeiten fast alle Bereiche der Physik mit dynamischen Systemen, aber auch in Bereichen der Mathematik oder der Biologie werden diese eingesetzt. Ein typisches Beispiel für die Anwendung eines dynamischen Systems ist die Erstellung eines Pendelmodells (siehe Abbildung 23). Aber auch biologische System oder gar menschliche Gesellschaften fallen in das Anwendungsgebiet dieser Systeme.

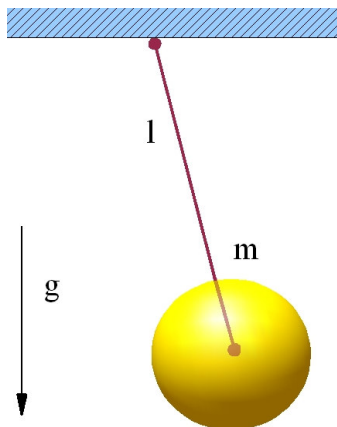


Abbildung 23: Dynamisches System, Pendel

Der Ursprung dieser Systeme liegt im 17. Jahrhundert, als Newton die Differentialgleichung, die Gesetze der Bewegung und der Gravitation gefunden hat. Er kombinierte diese mit Keplers Gesetz der planetaren Bewegung. Dadurch entstanden die ersten Beschreibungen von dynamischen Systemen. Die Geschichte führt weiter über Poincare, der die dynamischen Systeme erstmals mit Chaos und Fraktalen in Berührung brachte. Wie bei vielen anderen Bereichen brachte die Erfindung des Computers erst den Durchbruch. Berühmte Wissenschaftler aus der Welt der Fraktale

(Edward N. Lorenz, Mitchell J. Feigenbaum, Benoît B. Mandelbrot) brachten die Welt dieser dynamischen Systeme erst zum Blühen.

Durch dynamische Systeme können einfache und komplexe Sachverhalte und Probleme gelöst oder zumindest einer genaueren Betrachtung unterzogen werden. Daher spielen auch im Bereich der Fraktale die dynamischen Systeme eine wichtige Rolle.

6.5.1 Grundlagen

Das Wetter ist eines der bekanntesten komplexen Systeme (siehe Abbildung 24). Hierbei ist leicht ersichtlich, dass solche Systeme sehr schwer vorhersagbar sind. Dies beruht auf der Tatsache, dass sie von vielen Parametern diese Systems nicht bekannt sind und es nicht lineares Verhalten zeigt. Um nun aber so ein System in seinem Verhalten zu erforschen, ist ein Modell notwendig. Ein solches Modell ist eine Abstraktion und stellt eine bearbeitbare Form der Definition eines Systems dar. Aufgrund eines Modells kann der Ansatz des dynamischen Systems greifen. Zusätzlich wurden in diesem Kapitel [M7], [M8] und [M12] herangezogen.



Abbildung 24: Wetteraufnahme

Ein dynamisches System spannt einen mathematischen Raum auf, der Phasenraum genannt wird. Dieser wird im Verlauf noch genauer erklärt. Zusätzlich ist über diesem Raum eine glatte Evolutionsfunktion F definiert. Eine glatte Funktion liegt dann vor, wenn sie beliebig oft differenzierbar ist. Diese stellt eine Abbildungsfunktion dar. Die

Funktion F bildet im diskreten Fall bei Eingabe eines Punktes aus dem Phasenraum und einer zeitlichen Angabe den eingegebenen Punkt auf einen neuen Punkt im selben Raum ab. Für die Angabe der Zeit können diskret oder kontinuierlich erfolgen. Im Folgenden wird nur mehr der diskrete Fall behandelt.

Dabei können zwei grundsätzliche Typen von dynamischen Systemen unterschieden werden. Der erste Typ von Systemen arbeitet in dessen Entwicklung auf Basis kontinuierlicher Zeit. Der zweite findet in einem diskreten Zeitraum statt. Dynamische Systeme mit kontinuierlicher Zeit werden mit Hilfe von Differentialgleichungen beschrieben. Bei dynamischen Systemen mit diskreter Zeit werden Differenzgleichungen verwendet.

Dynamische Systeme können auf unterschiedlichen Zahlenräumen definiert werden. Dadurch werden sie auch unterschiedlich benannt. So sind dynamische Systeme als Flow bekannt, wenn es den Raum der reellen Zahlen für die Zeitangabe verwendet. Semi-Flow wird bei einer Basis von positiven reellen Zahlen verwendet. Cascade oder Map heißen dynamische Systeme im ganzzahligen Raum. Semi-Cascade oder Semi-Map wird wiederum verwendet, wenn das dynamische System bei positiven ganzen Zahlen bleibt.

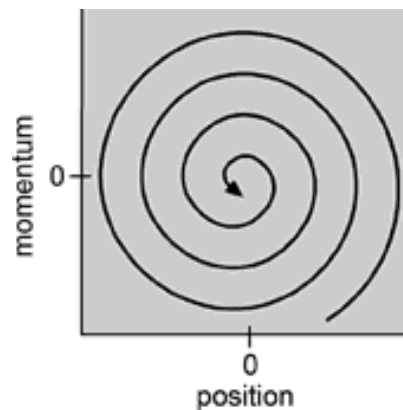


Abbildung 25: 2D-Phasenraums eines Pendels

Ein weiterer erklärungsbedürftiger Punkt ist der Phasenraum (siehe Abbildung 25). Er stellt den Raum dar, in dem alle möglichen Zustände des dynamischen Systems enthalten sind. Jeder einzelne Zustand stimmt mit einem eindeutigen Punkt in diesem Raum überein. In einem Phasenraum werden die Achsen aufgrund der Freiheitsgrade bzw. der Parameter gebildet. Das heißt, dass jede Achse in einem Phasenraum einem Freiheitsgrad entspricht. Damit stellt jeder Parameter des dynamischen Systems eine Achse in diesem Phasenraum dar. Alle Zustände im Phasenraum können durch einen Phasenvektor n -ter Ordnung dargestellt werden, wobei n für die Dimension des Raums steht. Ein derartiger Raum kann dadurch sehr schnell sehr hochdimensional werden und ist damit meist nicht mehr einfach grafisch darstellbar.

Wird ein Punkt in diesem Raum einzeln betrachtet, so legt dieser einen Weg durch den Raum über die Zeit hinweg zurück. Dieser Weg oder auch Kurve wird mit Hilfe eines zeitlich geordneten Menge von Zuständen dargestellt. Jeder Zustand wiederum besteht aus einer Menge von festgelegten Parametern bzw. Freiheitsgraden. Wie der Punkt sich nun im Raum bewegt, hängt von der Abbildungsfunktion bzw. Evolutionsfunktion des dynamischen Systems ab. Dabei hängt die Vorhersage der Entwicklung eines Punktes nicht unbedingt von der Komplexität dieser Funktion ab. Auch schon bei einfachen dynamischen Systemen kann es zu unvorhersagbaren Verhalten kommen.

Der Phasenraum kann zudem in verschiedene Orbits unterteilt werden. Dies kann allerdings in jedem dynamischen System unterschiedlich sein. Ein Orbit ist dabei nichts anderes als eine Sammlung von Punkten aus dem Phasenraum, die durch die Evolutionsfunktion verbunden sind. Ein Orbit ist daher immer eine Teilmenge des Phasenraums. Alle Orbits eines Phasenraums bilden immer eine Partition. Eine Partition beschreibt Unterteilung einer Menge in Teile, die sich selbst nicht überlappen und die ganze Menge abdecken.

Bei Orbits gibt es eine Reihe unterschiedlicher Arten, die bei dynamischen Systemen auftreten können. Ihr Auftreten ist jedoch nicht zwingend. Der einfachste Orbit ist der konstante Orbit. Er besteht nur aus einem Punkt und wird daher auch Fixpunkt genannt. Periodische Orbits bestehen aus einer Reihe von Punkten im Phasenraum, die sich bei Anwendung der Evolutionsfunktion immer wiederholen. Beide dieser Orbits sind geschlossene Orbits, da die Menge der beinhalteten Punkte endlich ist. Daneben existieren jedoch noch weitere Orbits, die diese Eigenschaft nicht aufweisen. Sie werden offene Orbits genannt. Darunter fallen asymptotisch periodische Orbits. Diese nähern sich einem periodischen Orbit an, erreichen ihn aber nie ganz. Die chaotischen Orbits fallen ebenfalls in die Gruppe der offenen Orbits. Diese Orbits sind sehr unterschiedlich. Daher ist der umfasste Bereich nur schwer vorhersagbar. Die Chaostheorie beschäftigt sich mit dieser Gruppe.

Das Verhalten von dynamischen Systemen ist aber nicht immer derart festlegbar. In weiten Bereichen dieser Systeme ändern sich Orbits, in ihrer Position und auch in ihrem Verhalten. Dies passiert teilweise auch, wenn sich Parameter nur geringfügig verändern. Trotz dieser minimalen Änderung der Parameter können die Änderungen des Verhaltens des Systems extrem sein. Dieses Verhalten von dynamischen Systemen ist unter dem Begriff Bifurkation bekannt (siehe Abbildung 26). Geprägt wurde der Begriff erstmalig von Poincare innerhalb seiner Arbeit in der Chaostheorie. Systeme die derartige Verhaltensänderungen zeigen, tendieren zu unendlich komplexem Verhalten.

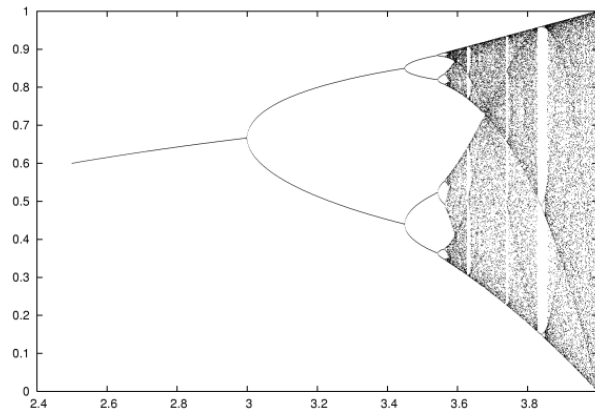


Abbildung 26: Bifurkationsdiagramm, Feigenbaumdiagramm

Es können zwei Typen von Bifurkationen unterschieden werden, je nachdem wie umfangreich die Änderungen im Verhalten ausfallen. Der erste Gruppe sind die lokalen Bifurkationen. Sie ändern das Verhalten des dynamischen Systems nur lokal und können daher einfach untersucht und identifiziert werden. Lokale Bifurkationen können vollständig erfasst werden. Im Gegensatz dazu stehen die Bifurkationen des zweiten Typs, die globalen Bifurkationen.

Um nun auch die Idee des dynamischen Systems verwenden zu können, muss ein Modell des zu betrachtenden Systems entwickelt werden. Dadurch muss zuerst das System selbst betrachtet werden. Die Eigenschaften und Parameter eines Systems sind dabei das Ziel dieser Betrachtung. Erst dann kann ein Modell entworfen werden, dass die Komplexität des Systems widerspiegelt.

6.5.2 Definition

Die Definition wurde durch [M6], [M13] und [P1] ergänzt. Die allgemeine Definition von dynamischen Systemen ist der Definition der formalen Grammatiken sehr ähnlich. Auch bei den dynamischen Systemen existiert eine Evolutionfunktionen F , die den Produktionen entspricht. Jedes Element einer Menge M (Phasenraum) wird bei Anwendung der Funktion F wiederum in den Phasenraum M abgebildet. Hierbei wird allerdings zusätzlich die Algebra T definiert, in welcher sich das System bewegt. Definiert wird daher ein dynamisches System durch (T, M, F) , wobei gilt

$$F: U \subset T \times M \rightarrow M$$

und

$$I(x) = \{t \in T : (t,x) \in U\}$$

$$F(0, x) = x$$

$$F(t_2, F(t_1, x)) = F(t_1 + t_2, x) \text{ für } t_1, t_2, t_1 + t_2 \in I(x),$$

wobei t den Evolutionsparameter (zeitlichen Verlauf) darstellt und x dem Anfangszustand im System entspricht. Leicht zu erkennen ist hierbei, dass die Form der Algebra einem Monoid entspricht. Um einem Monoid zu entsprechen, müssen ein neutrales Element und eine zweistellige Operation vorhanden sein. Das neutrale Element hierbei ist 0 und die zweistellige Operation ist die Addition.

Weiters können aufgrund dieser Definition weitere Begriffe formal definiert werden. So definiert sich der eingeschlagene Pfad aufgrund eines Anfangszustandes x als

$$F: I(x) \rightarrow M,$$

wobei diese Definition je nach zugrunde liegendem Zahlenraum benannt ist (siehe Kapitel 6.4.1). Für reellen Zahlen würde der Pfad als Flow bezeichnet werden. Die zugehörige grafische Darstellung wird als Kurve von x bezeichnet.

Weiters kann der Begriff des Orbits nun eindeutig erfasst werden. Ein Orbit O ist dabei nichts weiter als eine Teilmenge von M für die gilt

$$O := \{F(t, x) : t \in I(x)\},$$

wobei O der Orbit ist, der durch x aufgespannt wird. Geschlossen ist eine Teilmenge S von M dann, wenn gilt

$$\forall x \in S \text{ und } \forall t \in T$$

$$F(t, x) \in S,$$

wobei diese Regel sowohl für alle Orbits als auch für jede andere Teilmenge gilt. S wird dabei invariant bezüglich F genannt.

Diese Definition für dynamische Systeme ist sehr allgemein und für eine direkte Verwendung meist nicht sehr praktikabel. Deshalb ist es notwendig, weitere Definitionen innerhalb der dynamischen Systeme und ihrer Teilbereiche zu kennen. Diese machen einen Einsatz praktisch besser umsetzbar. Zu diesem Zweck muss zuvor eine Unterteilung der dynamischen Systeme erfolgen.

Dynamische Systeme werden in verschiedene Bereiche unterteilt. Diese Unterteilung ist abhängig von den Eigenschaften, die sie besitzen. So können dynamische Systeme linear oder nicht-linear sein, was sich auf deren Definition auswirkt. Lineare Systeme sind weitaus einfacher zu definieren und können komplett gelöst werden. Das Verhalten aller ihrer Orbits kann eindeutig klassifiziert werden. Nicht-lineare Systeme sind sehr viel problematischer in Verhalten und Definition. Nicht-lineare Systeme können in manchen

Abschnitten durch lineare Systeme approximiert werden, um diese Problematik zu umgehen.

Ein weiterer Punkt, welches die Definition von dynamischen Systemen unterteilt, ist die Anzahl der Dimensionen des dynamischen Systems. Je mehr Parameter Einfluss auf das dynamische System nehmen, desto komplexer wird es. Auch das Verhalten und die Definition dieser Systeme werden dadurch komplexer. Schon die Verwendung von einer Einflussgröße in einem nicht linearen System stellt eine Herausforderung dar, die nicht unterschätzt werden darf. Daher ist es während der Modellierung wichtig, möglichst umfangreich zu abstrahieren.

Besonders die linearen Systeme sind von großer Bedeutung für die gesamten dynamischen Systeme. Sie sind die einfachste Gruppe aus diesem Themenbereich. Durch die Kenntnis der linearen Systeme ist es einfacher, komplexe Systeme aufzubauen. Auch zur Approximation von komplexeren Systemen werden diese gerne herangezogen. Durch ihre Linearität haben sie den Vorteil, vollständig lösbar zu sein. Zudem ist ihr Verhalten vorhersehbar.

Die Evolutionsfunktion F ist in einem linearen dynamischen System selbstverständlich linear und definiert sich über die allgemeine lineare Abbildungsfunktion. Die Dimension dieser Funktion stellt die Dimension des dynamischen Systems dar.

Bei dynamischen Systemen gilt, dass die Lösung des Systems durch die Analyse des Verhaltens im Vordergrund steht. Im linearen Fall mag eine Analyse bei geringer Dimensionalität noch möglich und sinnvoll sein. Bei steigender Komplexität wird dies jedoch zu aufwendig oder mit den derzeit zur Verfügung stehenden Mitteln gar nicht möglich. Die Lösung eines komplexen Systems wird dabei durch die zusätzlichen Übertritte in den nicht-linearen Bereich verhindert.

6.5.3 Arten von dynamischen Systemen

Wie schon im vorhergehenden Abschnitt beschrieben wurde, können dynamische Systeme anhand ihrer Linearität und ihrer Dimension unterteilt werden. In jedem dieser Teilbereiche existiert eine Vielzahl von Vertretern, weshalb nur einige wenige herausgenommen und hier behandelt werden können. Vor allem auf die linearen Vertreter der dynamischen Systeme wird besonders eingegangen, da diese im späteren Verlauf Verwendung finden. [M12], [M13] und [P1] wurden zusätzlich als Informationsquellen für diesen Abschnitt verwendet.

Dynamische Systeme können von unterschiedlicher Dimension sein. Je nach Anzahl der Dimensionen wird ihre Komplexität erhöht. Bei dynamischen Systemen mit einem Parameter spricht man von Systemen erster Ordnung. Dies ist völlig unabhängig davon, ob es sich um lineare oder nichtlineare Systeme handelt. Sie haben die Form

$$xf = F(x),$$

wobei F die Evolutionsfunktion des Systems darstellt und x den Initialwert. xf steht für das Ergebnis der Anwendung der Funktion des dynamischen Systems auf den Wert x . Hierunter fallen Systeme wie der radioaktive Zerfall oder exponentielles Wachstum.

Im zweidimensionalen Fall sind zwei Parameter für die Entwicklung des Systems verantwortlich. Für Systeme zweiter Ordnung gilt

$$(t, xf) = F(t, x),$$

wobei F wiederum die Evolutionsfunktion darstellt, die aber in diesem Fall zwei Werte als Parameter verwendet. Beispiele für diesen Bereich sind Pendel oder Feder-Dämpfer-Masse-Systeme.

Die allgemeine Definition basiert auf einer unbegrenzten Anzahl von Parametern und lautet daher

$$(x_1, \dots, x_n) = F(x_1, \dots, x_n),$$

wobei x_1, \dots, x_n wiederum die Parameter der Evolutionsfunktion F sind. Sehr viele Beispiele lassen sich hierfür in der molekularen Dynamik finden.

Lineare Systeme sind die einfachere Gruppe der dynamischen Systeme. Sie sind vorhersehbar und daher auch berechenbar. Es kann sowohl ihr Verhalten als auch ihr Ergebnis berechnet werden. Nur mit sehr großem Aufwand können solche umfangreichen natürlichen Systeme analytisch gelöst werden. Bekannte Vertreter für lineare dynamische Systeme sind Feder-Dämpfer-Masse-Systeme, radioaktive Zerfallssysteme, exponentielle Wachstumssysteme oder auch lineare Pendel.

Im weiteren Verlauf werden lineare dynamische Systeme noch genauer im Kapitel 6.6 behandelt und zwar bezogen auf den Fall eines Feder-Dämpfer-Masse-Systems, das für das Kapitel 7.3 verstärkt gebraucht wird.

Nicht-lineare Systeme sind die weitaus kompliziertere Form der dynamischen Systeme. Sie können nur mehr durch die Beobachtung ihres Verhaltens analysiert werden. Das allgemeine Pendel, Chaos und Fraktale fallen in den Bereich der nicht-linearen Systeme. Diese Art der dynamischen Systeme wird allerdings in dieser Arbeit nicht weiter verwendet.

6.6 Feder-Dämpfer-Masse-Systeme

Die Übertragung eines dynamischen Systems aus der Natur in ein mathematisches Äquivalent erfordert mehr als nur die Kenntnis über die verschiedenen dynamischen Systeme. Vielmehr kommt es darauf an, eine sinnvolle Abstraktion des natürlichen Systems zu erstellen. Die Modellierung ist wohl die wichtigste und zugleich komplexeste Anforderung. Bei der Modellierung versucht man immer die optimalen Modelle zu identifizieren und umzusetzen. Sie soll sowohl die Natur möglichst genau simulieren und dabei aber ebenfalls effizient in der Berechnung sein. Durch [M1], [M2], [M4] und [P4] wird dieses Kapitel abgedeckt.



Abbildung 27: Bewegung von Gräsern

Vegetative Systeme sind zum Beispiel die Bewegung von Gräsern oder Sträuchern durch den Wind (siehe Abbildung 27). Zuerst beginnt man sich diesen natürlichen Systemen durch einfache dynamische Systeme anzunähern. Sollten die Möglichkeiten eines solchen Systems nicht ausreichend sein, tastet man sich zu komplexeren Systemen vor.

Ein sehr bekanntes und für natürliche Systeme gut anwendbares System stellt ein Feder-Dämpfer-Masse System dar. Die Feder stellt die oszillierende Komponente im System dar. Der Dämpfer vermindert die Energie im System und damit auch die Oszillation. Die Masse im System wird durch die Federn und Dämpfer miteinander verbunden. Durch diesen Zusammenhang wird mit der Zuführung von Energie das System in Bewegung versetzt. Dadurch kann die Interaktion der einzelnen Massen untersucht werden. Am Beispiel der Bewegung von Gräsern wäre der Wind die Energie, die dem System zugeführt wird. Genauere Erklärungen zu den Komponenten erfolgen im nachfolgenden Kapitel 6.6.1.

Ein solches System aus Federn, Dämpfern und Massen fällt unter die linearen dynamischen Systeme. Damit ist es im Hinblick auf die Komplexität noch verhältnismäßig simpel, wodurch das Verhalten einfach berechnet werden kann. Zudem

wird das System durch eine deterministische aufgebaut und dadurch noch zusätzlich vereinfacht. Ein stochastisches System wäre in diesem Fall auch nicht sinnvoll, da das Verhalten selbst untersucht werden soll. Eine große Anzahl von Verbindungen durch Federn und Dämpfern kann dadurch einfacher verarbeitet werden. Auch kann auf Änderungen bei den Verbindungen flexibler reagiert werden.

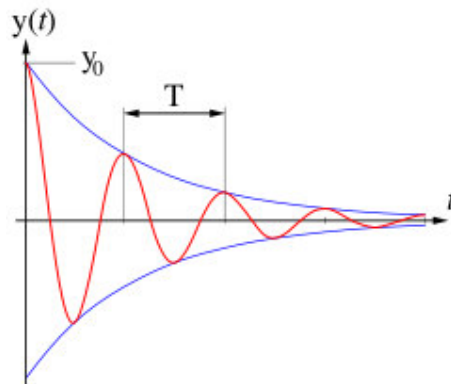


Abbildung 28: Beispiel eines richtig gedämpften Systems

Aber auch diese Systeme haben ihre Tücken. Bei der Modellierung eines stabilen und oszillierenden Systems muss auf einige Dinge Rücksicht genommen werden. (siehe Abbildung 28). Vor allem die Einstellung der Dämpfung ist immer wieder ein Problem. Wird nämlich das System zu stark gedämpft (Überdämpfung), dann kommt es zu keiner entsprechenden Oszillation. Umgekehrt führt eine zu schwache Dämpfung zu einem System, das schon bei geringer Energiezufuhr und ausreichend Zeit den nutzbaren Kontext verlässt. Es bewegt sich dann unkontrolliert und es sammelt sich immer mehr Energie im System.

Weiters kann in der Schrittberechnung der Grund für eine auftretende Instabilität liegen. Die Bewegungen der Massen in diesem System verlaufen nicht diskret, sondern kontinuierlich. Diese Bewegungen werden jedoch diskret approximiert. Wird nun die Approximation zu ungenau durchgeführt, potenzieren sich die Ungenauigkeiten bis zu einem Stabilitätsverlust. Dies kann durch zu große Berechnungsschritte oder ein zu ungenaues Verfahren passieren. Näheres zur Approximationen kann im Kapitel 6.6.2 nachgelesen werden.

Wie gezeigt wurde, sind selbst lineare dynamische Systeme nicht immer einfach in der Handhabung. Daher ist immer darauf zu achten, möglichst einfache dynamische Systeme zu verwenden.

6.6.1 Grundlagen

Ein Feder-Dämpfer-Masse-System besteht aus eben jenen erwähnten drei Komponenten: Federn, Dämpfern und Massen (siehe Abbildung 29). Alle drei Komponenten stehen in direktem Zusammenspiel und werden bei Zuführung von Energie in das System in ihrem Verhalten untersucht. Diese Energie ist gerichtet und kann daher von verschiedenen Richtungen auf das System wirken. Dadurch können unterschiedliche Reaktionen des Systems erzeugt werden. [M6], [M8] und [P1] ergänzen dieses Kapitel.

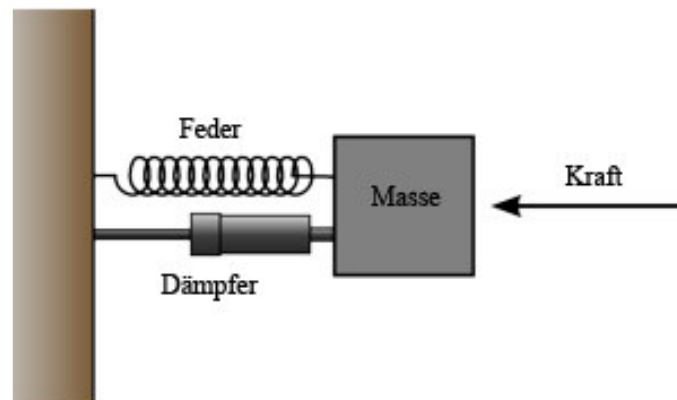


Abbildung 29: Grundlegendes Feder-Dämpfer-Masse-System

Die erste Komponente des Systems stellen die Federn dar. Sie ermöglicht es dem System einen schwingenden Zustand zu erreichen. Federn erzeugen bei einer Verschiebung ihrer Massen eine Gegenkraft. Dadurch wird die Masse wieder in die Richtung ihrer ursprünglichen Position bewegt. Je größer die Auslenkung, desto größer ist auch die Kraft der Gegenbewegung. Eine Feder erzeugt damit immer nur Veränderungen der Energie im System und entfernt keine Energie daraus. Nur alleine durch Federn würde sich damit bei einem Energiezufluss immer mehr Energie im System sammeln.

Erst durch den Einsatz von Dämpfern ist es möglich, in einen Ruhezustand zu gelangen. Die Energie des Systems wird durch Dämpfer aus diesem abgezogen. Dämpfer stellen Nachbildungen der natürlichen Form des Energieabflusses eines Systems dar. Diese können etwa Reibung, Luftwiderstand oder andere physikalische Gegebenheiten sein. Dämpfer haben keine andere Funktion als Bewegungen von mit ihnen verbundenen Massen zu reduzieren. Damit versuchen sie immer die Energie aus dem System zu ziehen. Sie verhindern, dass Systeme aus ihrem verwendbaren Kontext gleiten oder auch dass Systeme ewig schwingen würden.

Sowohl Federn als auch Dämpfer sind immer in ihrer Funktion an den mit ihnen verbundenen Massen gebunden. Die Größe der Massen beeinflusst die Reaktion des Systems. Aufgrund einer Energiezufuhr wird das System in Schwingung versetzt.

Durch die Trägheit einer Masse wirkt sich die Energiezufuhr auf kleinere Masse anders aus als auf große. Im Vergleich sind dabei kleinere Masse leichter in Schwingung zu versetzen als größere.

Neben den Komponenten ist aber auch das verwendete Approximationsverfahren wichtig. Dieses berechnet die Bewegung der Massen. Das Verfahren berechnet den Weg der Masse immer in bestimmten Zeitschritten. In Wirklichkeit bewegt sich die Masse natürlich kontinuierlich, wodurch bei jedem Berechnungsschritt ein Genauigkeitsfehler entsteht. Diese Fehler potenzieren sich bei jedem weiteren Berechnungsschritt auf und können zu Stabilitätsproblemen in der Berechnung führen. Außerdem führen sie dazu, dass die Ergebnisse im Verhalten ungenau werden. Kleine Berechnungsschritte führen in diesem Fall zu genaueren Ergebnissen. Allerdings wird dadurch der Berechnungsaufwand größer. Hier besteht also wieder ein Tradeoff zwischen Genauigkeit und Geschwindigkeit der Berechnung.

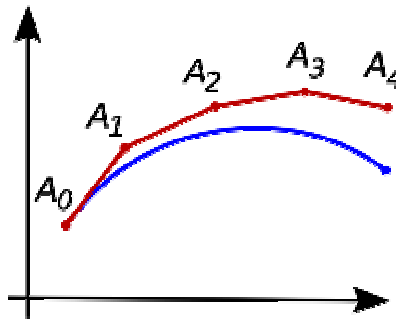


Abbildung 30: Annäherung einer Kurve mittels Euler Step-Verfahren

Eines der einfachsten, aber auch ungenauesten Verfahren ist der Euler Step (siehe Abbildung 30). Bei diesem Verfahren wird die Position einer Masse, mit Hilfe eines vorher festgelegten Zeitschrittes linear berechnet. Aufgrund der errechneten Position wird dann die nächste Position in der Zeitlinie berechnet und so weiter. Die Position weicht mit immer mehr Schritten immer mehr vom korrekten Weg ab. Nur durch kleinere Berechnungsschritte kann dies verhindert werden. Dadurch kann dieses Verfahren jedoch trotz der Einfachheit sehr schnell sehr langsam werden.

Um trotzdem ein genaueres Ergebnis zu erzielen, kann das „Runge-Kutta“ Verfahren verwendet werden. Dieses Verfahren verwendet die Steigungen der Bewegungskurve der betrachteten Masse an verschiedenen Punkten über den verwendeten Zeitschritt und bildet ein gewichtetes Mittel darüber (siehe Abbildung 31). Am Anfang steht immer die Berechnung der Steigung der Initialposition der Masse. Durch diese Steigung wird ein Punkt mit dem halben Zeitschritt berechnet. Dieser liegt zwischen Anfangs- und Endpunkt. Danach wird die Steigung dieses Punktes errechnet. Mit dieser Steigung wird wiederum ausgehend vom Anfangspunkt ein Punkt mit dem halben Zeitschritt berechnet. Dieser Schritt kann theoretisch beliebig oft wiederholt werden. Zuletzt wird

mit der letzten Steigung der genäherte Endpunkt berechnet und wiederum dessen Steigung ermittelt.

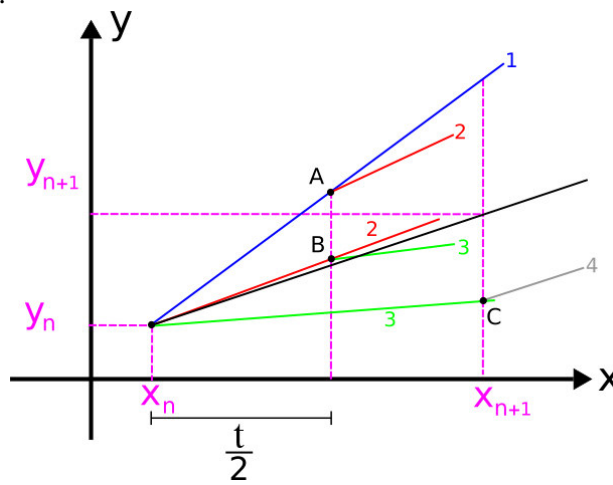


Abbildung 31: Grafische Darstellung des „Runge-Kutta“ Verfahrens

Alle Steigungen werden danach addiert und gemittelt. Dabei erfolgt meist eine Gewichtung, wodurch die Anfangs- und Endsteigung weniger Einfluss haben. Da der Zwischenschritt beliebig oft wiederholt werden kann, inkludiert man bei der Benennung des jeweiligen „Runge-Kutta“ Verfahrens auch immer die Ordnung. Die Ordnung des „Runge-Kutta“ Verfahrens entspricht der Anzahl der Steigungen, die zur Berechnung herangezogen wurden. Normalerweise führt man bei einer derartigen Berechnung zwei Zwischenschritte durch. Dabei handelt es sich dann um einen „Runge-Kutta“ der vierten Ordnung, da zwei Steigungen für die Zwischenschritte, eine Anfangssteigung und eine Endsteigung verwendet wurden.

Sowohl der „Euler Step“ als auch der „Runge-Kutta“ Verfahren basieren grundlegend auf fixen Zeitschritten. Eine im Vorhinein festgelegte Schrittgröße eignet sich aber nicht immer für die Berechnung. Es kann nämlich während der Berechnung zu unterschiedlichen Passagen kommen. Manche Passagen beinhalten kaum Änderungen und können mit großen Schritten durchgerechnet werden. Andere Passagen beinhalten sehr viele Änderungen und können damit nur mit kleinen Schritten richtig berechnet werden. Durch eine adaptive Schrittanpassung kann dies ermöglicht werden. Beide Verfahren können mit dieser Anpassung noch zusätzlich verfeinert werden.

Bei der Anpassung wird das jeweilige Verfahren dreimal angewendet. Einmal wird mit einem vorher festgelegten Zeitschritt ein Endpunkt berechnet. Danach wird noch einmal der gleiche Endpunkt berechnet. Dabei wird jedoch das gleiche Verfahren zweimal durchlaufen, wobei nur der halbe Zeitschritt verwendet wird. Danach wird der Unterschied zwischen den Ergebnissen ermittelt. Dieser Fehler gibt an, ob eine Unterteilung notwendig ist. Auch kann leicht berechnet werden, wie groß der jeweilige Schritt sein muss, um unter einem festgelegten Fehlergrenzwert zu liegen.

6.6.2 Definition

Zuerst werden die einzelnen Komponenten und deren Definition behandelt. Danach werden sie zu einem Ganzen zusammengesetzt. Dann erfolgen die Definition des gesamten Systems und die Behandlung der möglichen Berechnungsarten. Dieser Abschnitt wird zusätzlich durch [M5], [M8] und [P1] ergänzt.

Die Definition der Feder beruht auf einer beliebig festlegbaren Konstante s , der Federkonstanten. Die Konstante wird in Newtonsekunden pro Meter angegeben. Diese gibt Aufschluss über die Stärke der Feder. Die Federstärke ist ausschlaggebend für die Gegenbewegung einer mit der Feder verbundenen Masse aufgrund einer Positionsveränderung x . Definiert wird die Kraft F_s , welche die Feder auf die Masse ausübt durch

$$F_s = -sx,$$

wobei x die Form eines ein- bis dreidimensionalen Vektors annehmen kann. Dabei handelt es sich um eine lineare Rückstellkraft. F_s ist proportional zu Δx . F_s stellt damit eine direkte Kraft in entgegengesetzter Richtung zur Auslenkung dieser Masse dar. Die Feder versucht immer die mit ihr verbundenen Massen wieder in ihre Ausgangsposition zu versetzen.

Ein Dämpfer braucht ebenfalls, wie auch die Feder, eine Konstante, die Dämpferkonstante d . Auch die Dämpferkonstante wird in Newtonsekunden pro Meter angegeben. Diese gibt an, wie stark der Dämpfer die an ihn gekoppelten Massen in ihrer Bewegung dämpfen soll. Die Dämpfung vermindert dabei stetig Bewegungen in jegliche Richtung. Die Dämpfung beruht dabei auf der Geschwindigkeit v , in der sich die Masse befindet. Definiert wird die Kraft der Dämpfung F_d durch

$$F_d = -d \frac{\delta x}{\delta t}$$

gilt, wobei t die Zeit bzw. den Zeitschritt darstellt, der bei der Berechnung verwendet wird.

Die Kraft, die einem System zugeführt wird, hat meist die Form eines gerichteten Vektors. Um es in Schwingung zu versetzen, wird diese Energie dem System von außen zugeführt. Die erreichte Schwingung ist dabei von der Masse m abhängig, auf die die Energie trifft. Wenige kleine Massen beginnen schon bei geringer Energiezufuhr zu schwingen. Größere und zahlreichere Massen benötigen dazu mehr Energie, um die gleiche Amplitude zu erreichen. Die Energie wird dabei als Beschleunigung b der Masse gesehen. Die entstehende Kraft F_m definiert sich über

$$F_m = m \frac{\delta^2 x}{\delta t^2}.$$

Diese drei Kräfte können nun zu einem Ganzen kombiniert werden. Sie werden zu einer Gesamtkraft F_g addiert, die dann auf die Masse wirkt. Dadurch ergibt sich

$$F_g = -s x - d \frac{\delta x}{\delta t} + m \frac{\delta^2 x}{\delta t^2}.,$$

wobei diese Kraft im Ruhezustand gleich null ist. Bei numerisch bedingter Instabilität würde sich die Auslenkung immer weiter erhöhen, trotzdem keine zusätzliche Energie von außerhalb in das System zugeführt würde.

Besonders wichtig und empfindlich sind die Einstellungen der Konstanten für die Federn und Dämpfer. Falsch eingestellt, führt dies zu Überdämpfung oder Unterdämpfung. Überdämpfung verhindert dabei eine Schwingung des Systems. Das Verhältnis zwischen der Konstante der Federn und der Konstante der Dämpfer ist dafür ausschlaggebend. Die tatsächliche Größe der beiden Konstanten hat dabei weniger Einfluss. Alles in allem ist die Dämpfung jedoch für das jeweilige Feder-Dämpfer-Masse-System sehr spezifisch. Sie muss deshalb an die zu simulierende Umgebung angepasst werden.

Das System kann nun über die Zeit in dessen Verhalten beobachtet werden. Dazu braucht es ein Verfahren, das die Berechnung der neuen Positionen der Massen durchführbar macht. Die einfachste Methode stellt der „Euler Step“ dar. Bei dieser Berechnung wird mit Hilfe der wirkenden Kraft F_g und der Masse m , die Beschleunigung v ermittelt durch

$$v = \frac{F_g}{m} * t,$$

wobei t der gewählte Zeitschritt ist und m die Größe der Masse darstellt. Zur Erklärung des Verfahrens wird zu Beginn der Simulation davon ausgegangen, dass die Beschleunigung gleich null ist. Damit erfolgt erst im zweiten Durchlauf eine Änderung der Position. Die Position einer Masse ändert sich dann in jedem Zeitschritt. Eine andere Position hat zur Folge, dass auch andere Kräfte auf die Masse wirken. Daher muss in jedem Schritt die Kraft F_g neu berechnet werden.

Jede weitere komplexere Berechnungsart beruht auf dem „Euler Step“. Auch das „Runge-Kutta“ Verfahren baut auf dem „Euler Step“ auf. Das gebräuchlichste „Runge-Kutta“ Verfahren ist der „Runge-Kutta“ 4ter Ordnung (siehe Kapitel 6.6.1). Im ersten Schritt wird die Steigung k_1 über den gesamte Zeitschritt mit einem „Euler Step“ ES, wie folgt, berechnet

$$k1 = \text{ES}(p, t),$$

wobei p die Position der Masse und t der gewählte Zeitschritt ist. Sowohl Position als auch Zeitschritt werden dem „Euler Step“ übergeben. Danach wird mit Hilfe der errechneten Steigung ein Punkt innerhalb des halben Zeitschrittes ermittelt. Dann erfolgt die Berechnung einer Steigung innerhalb eines halben Zeitschrittes. Beides passiert durch

$$k2 = \text{ES}\left(p + \frac{t}{2} * k1, \frac{t}{2}\right),$$

wobei die Position der Masse p um einen halben Zeitschritt auf Basis der errechneten Steigung des ersten „Euler Step“ nach vorne gezogen wird. Es wird auch nur mehr einen halben Zeitschritt weiter gerechnet. Damit erhält man wiederum eine Steigung und damit auch einen alternativen Endpunkt. Als nächstes wird auf Basis der neu berechneten Steigung wiederum eine Steigung bzw. ein alternativer Endpunkt berechnet mit

$$k3 = \text{ES}\left(p + \frac{t}{2} * k2, \frac{t}{2}\right),$$

wobei auch hier wiederum die Position p zur Hälfte des Zeitschrittes vorgezogen wird und nur ein halber Zeitschritt weiter gerechnet wird. Zuletzt wird mit Hilfe der letzt berechneten Steigung ein gesamter Zeitschritt nach dem eigentlichen Zeitschritt getan. Die Steigung berechnet sich durch

$$k4 = \text{ES}(p + t * k3, t),$$

wobei $k4$ anders als alle Vorgänger bereits der Steigung des nächsten Schrittes entspricht. Um nun alle Werte in die Berechnung der neuen Position der Masse mit einfließen zu lassen, werden die Werte zu einem gewichteten Mittel zusammengefasst

$$k = (k1 + 2*k2 + 2*k3 + k4) / 6,$$

wobei die Anfangs- und Endsteigungen immer weniger Einfluss auf die Berechnung haben als die errechneten Steigungen der Zwischenschritte.

6.6.3 Arten von Feder-Dämpfer-Masse-Systemen

Feder-Dämpfer-Masse-Systeme werden in vielen Bereichen der Animation eingesetzt. In jedem Fall steht die physikalisch korrekte bzw. korrekt wirkende Animation von

unterschiedlichen Objekten, wie Kleidung, Fell, Haar oder auch Fahrzeuge im Vordergrund. Selbst die Nachbildung der Bewegung von Tieren oder Menschen kann auf so einem System aufgebaut werden. Durch [M5] und [M11] wurde dieser Abschnitt vor allem erstellt.

Animation von Bäumen und Gräsern

Natürlich kann auch jede andere Animation von natürlichen Bewegungen mit Hilfe von Federn-Dämpfer-Masse-Systeme umgesetzt werden. Auch die Bewegung von Pflanzen kann damit simuliert werden. Diese Systeme lassen sich leicht nachmodellieren (siehe Abbildung 32). Die Simulation der Energiezufuhr beruht dabei auf der Imitation von Wind oder Regen. Diese lassen sich gut in ihrer Wirkung in ein dynamisches System integrieren.



Abbildung 32: Animation von Bäumen

Diese Art der Animation ist der Brennpunkt dieser Arbeit und wird auch im Folgenden noch genauer erläutert.

7 Präprozess: Erstellung physikalisch-basierter Animation

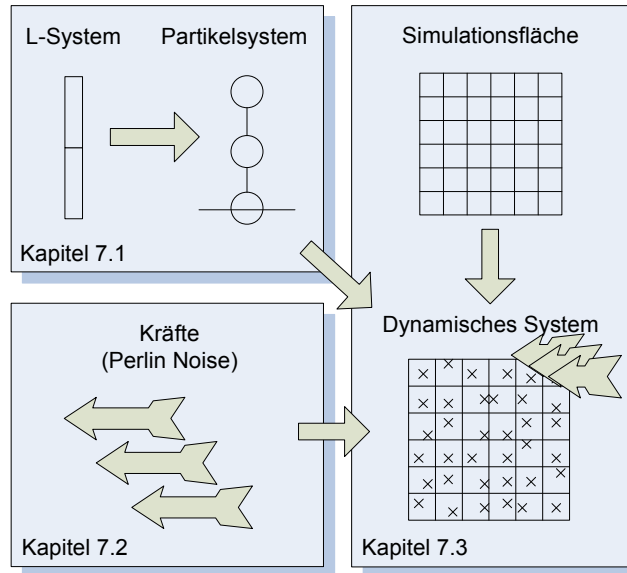


Abbildung 33: Aufbau des Präprozesses

Der Präprozess behandelt die Erstellung der physikalisch-basierten Animationsdaten und führt die dafür notwendige Simulation durch. Er unterteilt sich in drei Unterabschnitte (siehe Abbildung 33). Der erste Abschnitt (links oben) wird im Kapitel 7.1 behandelt. Dieses Kapitel gibt Aufschluss über die L-Systeme und die Überführung in die Partikelsysteme. In Kapitel 7.2 (links unten) werden alle wichtigen Punkte zur Erstellung von Wind behandelt. Die Zusammenführung der Vegetationsmodelle und der Kräftenmodelle erfolgt im Kapitel 7.3 (rechts). Auch die Durchführung der Simulation ist im Kapitel 7.3 zu finden.

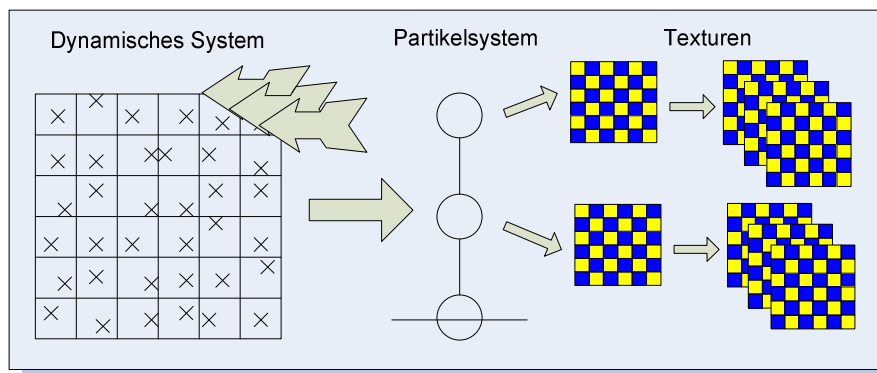


Abbildung 34: Speicherung der Daten des dynamischen Systems

Weiters behandelt das Kapitel 7.3 die Speicherung der Daten (siehe Abbildung 34). Als Abschluss in diesem Kapitel steht die Überführung der Daten aus dem dynamischen System in Texturen. Diese Texturen werden für das anschließende Laufzeitsystem (siehe Kapitel 8) benötigt.

7.1 Vegetationsmodell

Das Kapitel 7.1 behandelt die Erstellung von L-Systemen und deren Überführung in Partikelsysteme (siehe Abbildung 41). Die L-Systeme werden dabei zur Definition der Vegetation verwendet. Durch die Überführung der L-Systeme in Partikelsysteme werden diese für das dynamische System aufbereitet. Es gliedert sich in die vier Bereiche „Analyse“, „Realisierung“, „Aufbau des L-Systems“ und „Überführung in ein dynamisches System mittels Partikelsystem“.

7.1.1 Analyse

Die große Vegetationsvielfalt in der Umwelt bringt eine unüberschaubare Zahl an möglichen Ausgangspunkten mit sich. Diese gilt es sowohl zu erfassen als auch in eine allgemeine Form zu bringen. Erst eine solche Form macht es möglich, diese Strukturen zu bearbeiten. Einzig allein eine entsprechend ausgewählte Möglichkeit einer flexiblen Definition kann sich diesem Problem annehmen. Nur so kann es auf eine Art und Weise gelöst werden, die eine einfache Informationsweitergabe an die weiteren Bereiche ermöglicht. Daher muss die Definition in einer einfach weiterverwendbaren Form erfolgen.

Das Definitionsmodell sollte sehr anpassungsfähig sein. Wichtig ist dabei sowohl die Beschreibung der Vegetation als auch die Erstellung der Beschreibung selbst. Bei den zu beschreibenden vegetativen Strukturen kann es sich sowohl um einen kleinen Grashalm als auch um einen ausgewachsenen Baum handeln. Daher liegt es nahe, diese Anpassungsfähigkeit nicht nur im Bezug auf das Aussehen der Struktur zu limitieren. Es muss auch möglich sein, deren Eigenschaften mit in das Modell aufzunehmen. Dadurch kann es ebenfalls in den Prozess mit eingebunden werden.

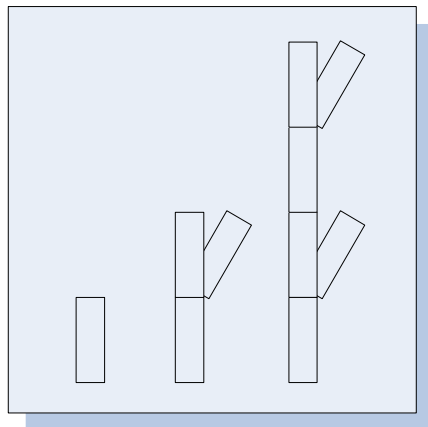


Abbildung 35: Iterationsstufen eines L-Systems

Die Flexibilität der Erfassung der Daten darf hier nicht enden. Bei genauerer Betrachtung der vegetativen Strukturen handelt es sich bei diesen nicht um völlig starre Systeme. Vielmehr wandeln sich diese im Verlauf der Zeit von kleineren Strukturen in größere, ähnliche Strukturen um. Das Wachstum dieser Systeme sollte daher ebenfalls Platz in der Definition dieser Strukturen finden (siehe Abbildung 35). So muss eine Struktur nur ein einziges Mal definiert werden und kann dann in unterschiedlichen Entwicklungsstadien in eine Simulation einfließen. Dabei müssen die Strukturen nicht erneut definiert werden. Daher muss die Definition selbst einen zeitabhängigen Faktor (Iterationen) in sich tragen, der diese verschiedenen Stadien zugänglich macht.

In diesem Stadium sollte bereits eine notwendige Abstraktion durchgeführt werden. Da nicht alle Daten einer vegetativen Struktur in die Berechnung miteinbezogen werden können, müssen bereits bei der Definition dieser Strukturen nur wichtige Informationen herausgefiltert werden. Nicht alle Daten einer vegetativen Struktur sind relevant für die Simulation. Diese Daten können dann in das System eingeflochten werden. So kann eine unnötige Datenlawine vermieden werden. Zudem ist dadurch eine bessere Transparenz über die Einflussdaten des Prozesses gegeben.

Die Natur ist in ihrer Schöpfung zwar sehr vielfältig, doch ähneln sich sehr viele Strukturen in ihrer Form. Diese Ähnlichkeit und auch die Selbstähnlichkeit vieler Strukturen kann ebenfalls ausgenutzt werden. Teile verschiedener Strukturen einer Vegetation können so wieder verwendet werden. Dadurch können modulare vegetative Strukturen erzeugt und auf diese Weise leichter und schneller in das System eingebunden werden. Außerdem können Strukturen in ihrem Verhalten in dem folgenden Prozess einfacher miteinander verglichen werden. Dadurch kann das Verhalten von ähnlichen Strukturen besser vorhergesehen werden.

Da man bei der Definition der vegetativen Strukturen bestehende Strukturen in der Wirklichkeit nachbildet, sollte man diese nicht nur auf deren numerische Eigenschaften reduzieren. Nicht nur eine Anhäufung von Zahlen und Nummern, sondern auch eine Form der Visualisierung sollte schon während der Definition zur Verfügung stehen. Dadurch können in diesem Schritt Abweichungen vom Original leichter festgestellt oder Auswirkungen von Änderungen genauer beobachtet werden.

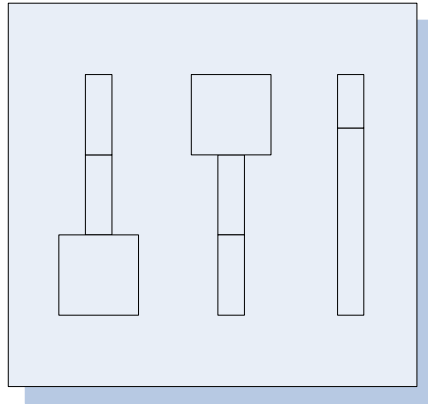


Abbildung 36: Unterschiedliche Eigenschaften eines L-Systems,
größere Breite unten (links), größere Breite oben (Mitte),
größerer Abstand zwischen den Abschnitten (rechts)

Grammatiken bieten jene flexible, anpassungsfähige Form der Definition (siehe Abbildung 36). Sie sind unabhängig von den Ausmaßen der vegetativen Struktur. Daher können sie leicht an die jeweilige Situation angepasst werden. Zudem kann sehr einfach ein zeitabhängiger Faktor in die Definition mit eingebracht werden. Um die Ähnlichkeiten und Selbstähnlichkeiten der Strukturen auszunutzen, können Fraktale Verwendung finden. Diese basieren nämlich genau auf diesem Grundsatz. So können sehr leicht vegetative Strukturen und deren Ähnlichkeiten ausgenutzt werden.

L-Systeme verbinden diese zwei Ideen. Sie verbinden Grammatiken und Fraktale und ermöglichen so eine gemeinsame Form der Definition von natürlichen Strukturen. Daher sind L-Systeme ideal, um die Definition solcher Strukturen einer Vegetation durchzuführen. Da jedoch L-Systeme in einer sehr informationskomprimierten Form vorliegen, kann nicht direkt mit ihnen gearbeitet werden. Sie können jedoch ohne großen Aufwand in ein Partikelsystem überführt werden. Dieses Partikelsystem wird durch die Daten des jeweiligen L-Systems konfiguriert.

7.1.2 Realisierung

Das Konzept basiert auf einem L-System für die Eingabe von Pflanzen und anderen natürlichen Strukturen jedweder Form. Dabei werden in einem Schritt sowohl dessen geometrische Form als auch die notwendigen Eigenschaften in das dynamische System integriert. So kann eine Definition mit einem L-System vollständig durchgeführt werden. Es müssen keine zusätzlichen Schritte in dessen Definition passieren und es ist kein weiteres zusätzliches System notwendig. Bei dem L-System handelt es sich um ein deterministisches 3-dimensionales kontextfreies System, kurz 3D-DOL-System. Zusätzlich kann dieses L-System mit optionalen fünf Parametern parametrisiert werden. Wird ein Parameter nicht definiert, wird ein Defaultwert angenommen. Die Anzahl der

Iterationen des L-Systems können frei festgelegt werden. Auch das Startsymbol unterliegt keinen Beschränkungen.

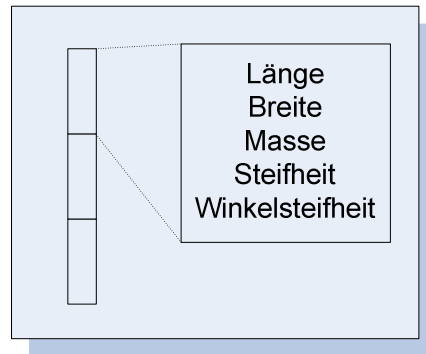


Abbildung 37: Lokale Parametrisierung des L-Systems

Die Definition der Eigenschaften erfolgt zudem lokal (siehe Abbildung 37). Jeder Teil einer vegetativen Struktur beinhaltet die Eigenschaften dieses Teils der Struktur. Dies passiert deshalb, weil jede Pflanze in unterschiedlichen Bereichen ihrer Struktur unterschiedliche Eigenschaften aufweist. So zum Beispiel ist der Stamm eines Baumes viel starrer und unflexibler als ein kleiner Zweig desselben Baumes. Wenn nun Wind auf ihn trifft, verhalten sich diese beiden Teile unterschiedlich. Der kleine Zweig beginnt zum Beispiel schon bei leichtem Wind zu schwingen. Der Stamm würde sich dabei nicht bewegen. Durch die Parametrisierbarkeit von L-Systemen kann diese lokale Definition gut umgesetzt werden. Sogar eine hierarchische Vererbung ist bei L-Systemen gegeben. Eine solche Vererbung der Eigenschaften und Änderungen derselbigen durch ein Voranschreiten der Iterationen des L-Systems ist eine Nachahmung des natürlichen Wachstumsprozesses. Dadurch können die Eigenschaften durch den Ausgangswert festgelegt werden. Zusätzlich dazu können Veränderung über die Iterationen ebenfalls mit eingebaut werden.

Der Determinismus des L-Systems wurde deshalb verwendet, um eine exakte Reproduzierbarkeit der Einstellungen zu gewährleisten. Die Definition der Vegetationsstrukturen bezieht sich auf eine Abbildung aus der Natur. Daher wird keine stochastische Verteilung gebraucht und gewünscht. Die reine Übernahme der Informationen und die Einbindung in ein dynamisches System sind das Ziel. Dies wird im folgenden Verlauf noch genauer besprochen.

Das System wurde mit keiner Bindung an einen Kontext ausgestattet. Solch eine Bindung erlaubt zwar komplexere L-Systeme, allerdings ist dies bei der Definition der Strukturen nicht notwendig. Es ist nicht das laufende Verhalten nach jeder Iteration wichtig sondern nur die Erlangung einer Ausgangsstruktur. Bei dieser reicht meist eine ein- oder zweifach iterierte Struktur für die weitere Verarbeitung. Zudem hätte eine Kontextbindung das System unnötig kompliziert, da eine Kontextbindung zusätzlich bei

der Definition einer Struktur mit einfließen müsste. Dies wäre nur durch weitere Parameter möglich gewesen.

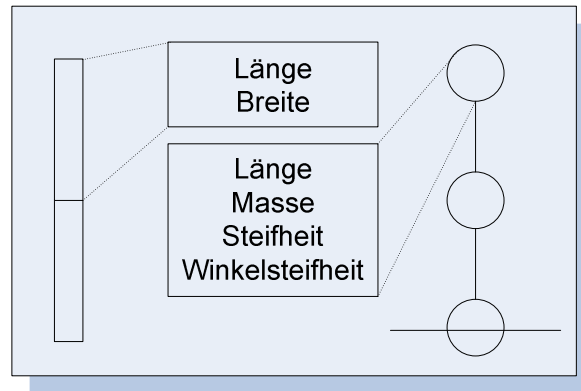


Abbildung 38: Parameterverwendung in L-System und Partikelsystem

Die Parametrisierung des L-Systems erlaubt es, weitere Größen in das L-System mit einzubinden. Diese Größen dienen dazu, die Vegetation zusätzlich in ihren Eigenschaften (siehe Abbildung 38) zu beschreiben. Dabei werden zwei der fünf Parameter für die Entwicklung des L-Systems selbst genutzt. Vier der fünf Parameter werden für die spätere Erstellung des Partikelsystems verwendet. Bei jedem Parameter können Änderungen vorgenommen werden. Diese sind trotz ihrer unterschiedlichen Verwendung aber völlig äquivalent.

7.1.3 Aufbau des L-Systems

Die Syntax des L-Systems besteht aus den Zeichen A-Z, a-z und den Ziffern 0-9 sowie den Sonderzeichen „| () [] { } - , “. Die Zeichen A-Z, a-z sind für die Definitionen des Namens eines Literals reserviert. Zahlen und Sonderzeichen sind für die Namen des Literals nicht erlaubt. Die Ziffern 0-9 werden zur Definition der Parameter sowie zur Änderung der Winkel im Raum verwendet. In jedem Fall darf einer Zahl das Sonderzeichen „-“ vorangestellt werden, wodurch diese Zahl negativ wird.

Das Sonderzeichen „|“ trennt zwei Literalnamen voneinander, wenn keine anderen Sonderzeichen zwischen beiden Literalen stehen. Dies wird benötigt, um eine eindeutige Zuordnung zu den Produktionen zu ermöglichen.

Beispiel:

$$\begin{aligned} A &\rightarrow AA \\ AA &\rightarrow A|A \end{aligned}$$

Hier ist deutlich zu erkennen, dass man Literale nur dann eindeutig zuordnen kann, wenn auch das Ende eines Literalnamens bekannt ist. Nur so ist die eindeutige Identifikation gewährleistet.

Die Sonderzeichen „()“ werden für die Eingabe von Winkeländerungen im Raum für den Aufbau des L-Systems verwendet. Änderungen erfolgen sowohl in der X-, Y- als auch in der Z- Koordinate, wobei X und Z die horizontale Ebene aufspannen und Y die vertikale Achse darstellt. Zu Beginn ist der Winkel auf null, wobei dies gleichbedeutend mit der Entwicklung in Y-Richtung ist.

Beispiele:

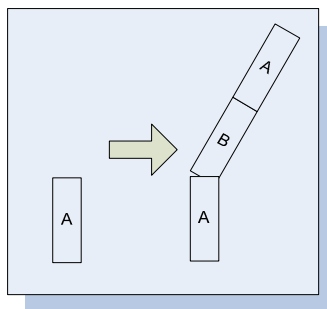


Abbildung 39: Beispiel eines L-Systems

$$A \rightarrow A(30,0,30)B|A$$

Das Beispiel zeigt eine Winkeländerung von 30° in X und 30° in Z sowie keine Änderung in Y. Alle Literale, die dieser Winkeländerung nachfolgen, werden mit diesen veränderten Winkeln dargestellt. Im Beispiel wären dies B|A sowie alle deren Ersetzungen.

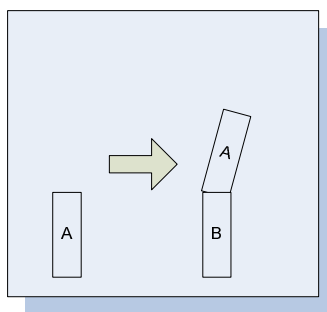


Abbildung 40: Beispiel eines L-Systems (2)

$$B \rightarrow B(.,10)A$$

Bei der Eingabe der Winkel muss kein Wert angegeben werden. Eine Eingabe der Trennzeichen reicht aus, um eine eindeutige Zuordnung zur Achse zu ermöglichen.

Die Sonderzeichen „[]“ markieren einen Kreuzungspunkt, bei dem das Zeichen „[“ den Punkt markiert und „]“ den Rücksprungzeitpunkt angibt. An diesem wird die Darstellung des L-Systems an dem vorher festgelegten Kreuzungspunkt fortgesetzt.

Beispiel:

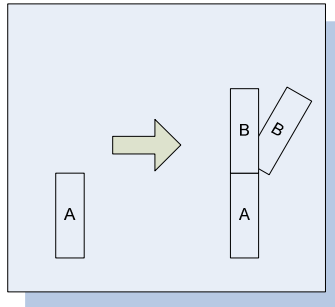


Abbildung 41: Beispiel eines L-Systems (3)

$$A \rightarrow A[(30,,)B]B$$

In diesem Beispiel wird zuerst das Symbol A gezeichnet. Dann wird das erste B mit einer Winkeländerung von 30° gezeichnet. Mit „]“ erfolgt ein Rücksprung zu A und das nachfolgende zweite B wird direkt nach A gezeichnet. Dies würde einer einfachen Astgabelung entsprechen.

Jedes Symbol kann mit bis zu fünf Parametern versehen werden. Diese Parametrisierung erfolgt durch die Symbole „{ }“ und besteht auch, wenn keine individuelle Einstellung vorgenommen wird. Es können sowohl positive als auch negative Zahlen zur Änderung der Parameter herangezogen werden. Die Parametrisierung erfolgt immer direkt nach dem Auftreten des Symbols. Die Änderung kann sowohl bereits bei der Definition des Literalnamens als auch innerhalb der Produktionen erfolgen. Die Parameter, die im Folgenden noch genauer erklärt werden, sind von links nach rechts, Länge, Breite, Masse, Steifheit und Winkelsteifheit.

Beispiel:

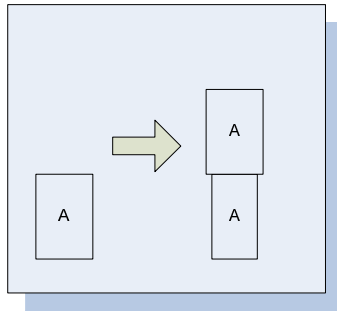


Abbildung 42: Beispiel eines L-Systems (4)

$$A\{0,100,0,0,0\} \rightarrow A\{0,-10,0,0,0\}A$$

Anhand des Beispiels ist die Möglichkeit der Definition leicht ersichtlich. A wird bei seinem ersten Auftreten mit 100 Prozent mehr Breite angelegt. Bei jedem Durchlauf der Produktion werden 10 Prozent abgezogen, was dazu führen wird, dass die Breite nach mehreren Iterationen null wird. Damit wäre das Symbol zwar noch im L-System vorhanden, aber nicht mehr im grafischen Display zu sehen.

Die Vererbung des obigen Beispiels tritt immer dann ein, wenn ein Symbol bereits vor der nächsten Iteration vorhanden ist. Ist es nach der Iteration immer noch vorhanden, vererben sich die Parameter. Nur in diesem Fall ist eine Vererbung der Parameter möglich (siehe Abbildung 43). Andernfalls wird das Symbol immer zu seinem Ausgangswert zurückgesetzt, im Beispiel oben wäre dies $A\{0,100,0,0,0\}$.

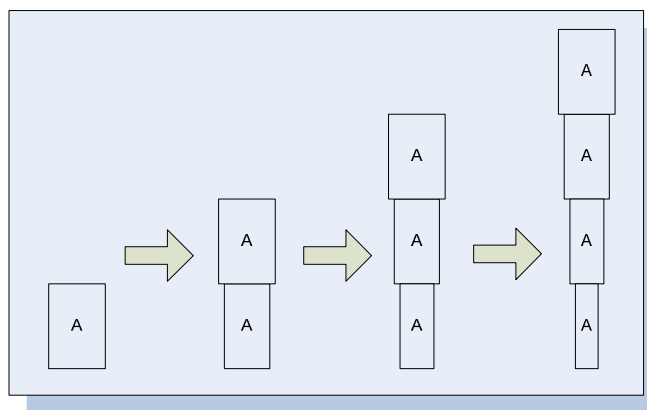


Abbildung 43: Parametervererbung, Weiterführung von Abbildung 42

Die beiden Parameter Länge und Breite sind die einzigen beiden Parameter, die für die Erstellung des L-Systems zuständig sind. Sie haben Auswirkungen auf die Form des L-Systems und dessen Darstellung. Die Länge beeinflusst den Abstand zwischen den

verschiedenen Elementen des L-Systems und die Breite ändert die Form der Darstellung. Auswirkungen auf die weitere Verarbeitung hat nur die Länge. Die Breite hingegen hat nur darstellungstechnischen Nutzen.

Die Parameter Masse, Steifheit und Winkelsteifheit sind für die Entwicklung des L-Systems von keinerlei Bedeutung. Sie finden erst später im dynamischen System ihre Verwendung. Die Definition dieser Parameter ist die hauptsächliche Schwierigkeit, um realistische Simulationen des dynamischen Systems zu erhalten. Da meist ein Mangel an diesen Daten von vergleichbaren vegetativen Strukturen besteht, erhält man diese Einstellungen meist nur durch mehrmaliges manuelles Ausprobieren.

7.1.4 Überführung in ein dynamisches System mittels Partikelsysteme

Nach der Erstellung des L-Systems kann die Vegetationsstruktur in das dynamische System übernommen werden. Im dynamischen System ist aber die direkte Bearbeitung des L-Systems nicht mehr praktikabel. Daher wird jedes L-System in ein aus Partikeln bestehendes System umgewandelt (siehe Abbildung 44). Dieses ist mit Verbindungen zwischen den Partikeln ausgestattet und mit einem Wurzelpartikel in der Simulationsfläche verankert. Durch diese Umwandlung ist es möglich, das L-System wie ein Feder-Dämpfer-Masse-System zu behandeln. Dies ist notwendig, um Kräfte auf das System wirken zu lassen. Im Kapitel 7.3 wird auf diesen Bereich genauer eingegangen.

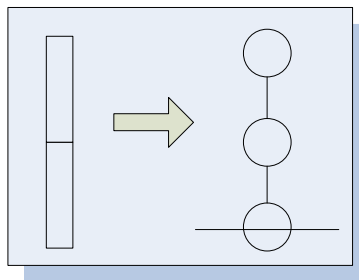


Abbildung 44: Überführung des L-System in Partikelsystem

Umgewandelt wird das L-System, indem aufbauend auf einem unbeweglichen Wurzelpartikel für jedes Symbol im L-System ein Partikel erstellt wird. Dieses Partikel trägt die Eigenschaften der Parameter des jeweiligen Symbols im Bereich Länge, Masse, Steifheit und Winkelsteifheit (isotrop). Die Partikel werden dann mit dem jeweiligen Vorgänger mit einer Feder und einem Dämpfer verbunden. Dadurch entsteht ein in sich verbundenes System. Durch diese Umwandlung ist das System nicht mehr von dem ursprünglichen L-System abhängig. Das L-System hatte nur den Zweck der Definition einer vegetativen Struktur.

Die Eigenschaft Länge entscheidet dabei, wie weit das Partikel von seinem Vorgänger entfernt ist. Dies kann bei der grafischen Darstellung des L-Systems gesehen werden. Größere Entfernungen bewirken, dass auf zwei benachbarte Partikel unterschiedlichere Kräfte wirken. Auf sehr nahe gelegene Partikel hingegen wirkt fast die gleiche Kraft. Dadurch wird sich ihr Verhalten zueinander sehr ruhig und stabil gestalten.

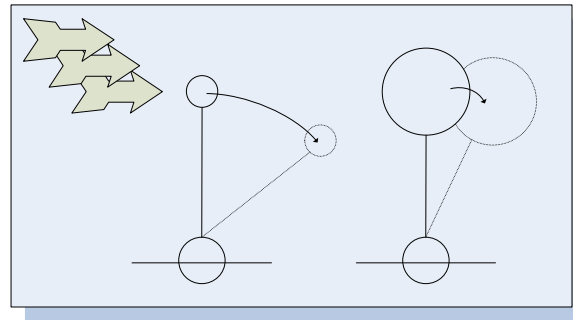


Abbildung 45: Trägheit der Masse, Auslenkung kleiner Masse (links), Auslenkung großer Masse (rechts)

Die Masse eines Partikels beeinflusst die Annahme der Kräfte und bewirkt so die Trägheit eines Partikels (siehe Abbildung 45). Partikel mit größerer Masse werden dabei von Kräften zu Beginn nur gering in ihrer Position beeinflusst. Es wird allerdings auch viel mehr Kraft und Widerstand gebraucht, um sie in ihrer Bewegung zu bremsen. Weiters wird durch die Masse die Schwerkraft beeinflusst aufgrund von

$$F = m \cdot g.$$

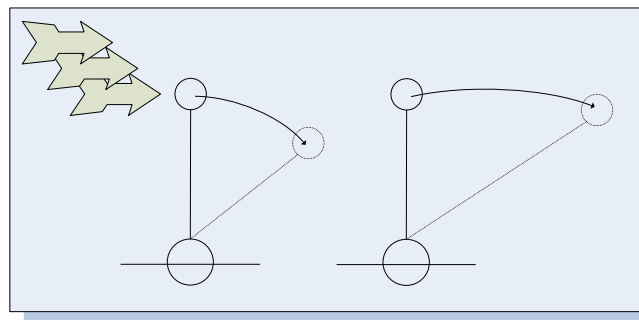


Abbildung 46: Steifigkeit der Verbindung, starre Verbindung (links), elastische Verbindung (rechts)

Die Eigenschaft der Steifigkeit eines Partikels zielt nicht auf das Partikel selbst, sondern auf seine Verbindung zu seinem Vorgänger ab (siehe Abbildung 46). Die Steifigkeit gibt an, wie unnachgiebig die ursprüngliche Entfernung des Partikels zu seinem Vorgänger

gehalten wird. Diese Entfernung ist dabei völlig unabhängig von dem Winkel, indem die Partikel zueinander stehen. Nur die reine Entfernung ist davon betroffen mit

$$D_m = D_a * S + D_o * (1-S),$$

wobei D_m die neue Distanz, D_a die aktuelle Distanz, S die Steifheit und D_o die ursprüngliche Distanz angibt. Dadurch wird die strukturelle Stabilität einer einzelnen Verbindung mit nur begrenztem Blick auf die gesamte vegetative Struktur simuliert. Ursprünglich waren die Steifheit und Winkelsteifheit ein einzelner Parameter, jedoch wurde dieser aus später genannten Gründen aufgeteilt.

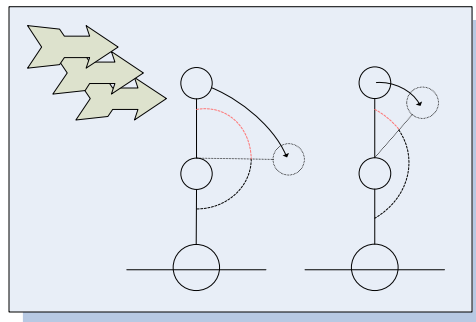


Abbildung 47: Winkelsteifheit der Verbindung, flexibler Winkel (links), starrer Winkel (rechts)

Um eine gesamte strukturelle Stabilität der Grundstruktur des Systems zu erhalten, braucht es einen weiteren Parameter, die Winkelsteifheit (siehe Abbildung 47). Dieser Parameter besagt, wie stark der Partikel und dessen Verbindung den ursprünglichen Winkel zu ihrem Vorgänger halten. Dies passiert, um die beiden Parameter Steifheit und Winkelsteifheit völlig unabhängig voneinander zu erhalten. Durch eben diese Unabhängigkeit können beide Parameter völlig frei gesetzt werden. Dies ist zur Simulation vieler verschiedener Strukturen einer Vegetation notwendig. Die Winkelsteifheit wird impliziert durch die Verwendung von Vektoren und Rotation in die Simulation eingebunden. Im Kapitel 7.3.3 wird die genaue Anwendung erläutert.

Alle notwendigen Punkte zur Erstellung einer Vegetation sowie die Umwandlung in eine Struktur zur weiteren Verarbeitung sind damit gegeben. Fortgeführt wird die Verwendung dieser vegetativen Strukturen im Kapitel 7.3. Zuvor allerdings muss noch das Kapitel 7.2 für die Definition und Wirkung der verschiedenen Kräfte im dynamischen System Erklärung finden.

7.2 Kräftemodell

Dieses Kapitel behandelt die Definition von Kräften mit Hilfe von „Perlin Noise“. Der Aufbau der „Perlin Noise“ Kräfte wird in diesem Kapitel erklärt. Die Überführung der Kräfte in das dynamische System schließlich beschreibt, wie die Kräfte im dynamischen System verwendet werden. Das Kapitel gliedert sich, wie auch das Kapitel 7.1, in vier Bereiche, die „Analyse“, die „Realisierung“, der „Aufbau der Kräfte“ und die „Überführung in das dynamische System“.

7.2.1 Analyse

Die Anforderungen bei den Kräften liegen im Bereich der Flexibilität, der Kombinierbarkeit, der visueller Darstellung, der Definition der Zufallsbasis und der Interaktionsmöglichkeit. Daher muss die Definition dieser Kräfte, im Speziellen Winde, äußerst flexibel und anpassungsfähig an die verschiedenen Szenarien sein.

Durch diese flexible Gestaltung müssen jedoch unterschiedlichste Winde mit denselben Parametern erstellbar sein. Dabei darf es keinen Unterschied machen, ob es sich nun um einfachen, turbulenzlosen Wind oder einen komplexeren, mit großen Turbulenzen versehenen Wind handelt. Optionale Parameter kommen diesem Konzept am besten entgegen.

Im Falle von aufwendigen Szenarien treten mehrere verschiedene Winde innerhalb eines Szenarios auf. Diese Winde müssen in der Lage sein, sich mit anderen kombinieren zu lassen. Das kann von einer Verstärkung der Winde bis zu einer teilweisen oder völligen Aufhebung von Kräften führen. Je nachdem, wie die Hauptwindrichtung gewählt wird, kann es zu diesen Effekten kommen. Die Kombination muss vor der Anwendung der Kräfte auf die Strukturen einer Vegetation erfolgen. Ansonsten würden unnötige kostenintensive Bearbeitungsschritte mehrmals durchlaufen werden.

Die Kraft eines Windes und die Kombination von mehreren Winden braucht eine visuelle Unterstützung. Erst dadurch kann man sich über dessen Ausmaß und Wirkung im Klaren sein. Eine Visualisierung der aktuellen Kraft des Windes bzw. der kombinierten Kraft von mehreren Winden an allen möglichen Positionen kann eine bessere Kontrolle bringen. Wichtig ist nicht nur die Kraft selbst, sondern auch die aktuelle Auswirkung der Kraft auf die verschiedenen vegetativen Strukturen in der Simulation. Dadurch wird sichtbar, wie stark eine Kraft tatsächlich auch Wirkung zeigt.

Kräfte und Winde sind in der Natur zwar eingrenzbar in ihrer Wirkung, jedoch unterliegen sie immer gewissen Zufälligkeiten. Da diese Zufälligkeiten auch in einer realistischen Simulation notwendig sind, dürfen bei Winden nur Grenzen durch

Parameter festgelegt werden. Das Grundprinzip muss immer auf Zufälligkeiten basieren. So bleibt es möglichst nahe am Original. Diese Zufälligkeiten müssen sich auf die Windstärke selbst und auf dessen Richtung als Abweichungen bzw. Turbulenzen des Windes beziehen.

Verschiedene Winde kommen nicht nur mit Turbulenzen in deren Definitionen aus. Winde können sich auch in ihrer Richtung komplett drehen. Eine solche Möglichkeit der Verhaltensänderung kann ebenfalls in das System miteinbezogen werden. Die Winde sollten während einer Simulation interaktiv in deren Richtung zu ändern sein. So können Kräfte in deren Richtung leicht angepasst werden und sogar eine völlig entgegengesetzte Richtung einschlagen, ohne die Simulation abubrechen.

„Perlin Noise“ ist ein Verfahren, dass für die Erstellung von Feldern von Zufallszahlen verwendet werden kann. Zufällige Werte bestimmen hierbei ein Feld von Vektoren, die durch Interpolationen in Verbindung stehen. Durch die zusätzliche Verwendung von normalverteilten Zufallszahlen können zusätzlich Steuerungsmechanismen in das Vektorenfeld eingebracht werden. Dabei wird das Verfahren selbst nicht verändert. Das Verfahren bietet sich damit als ideales Werkzeug an, um Kräfte bzw. Winde zu simulieren.

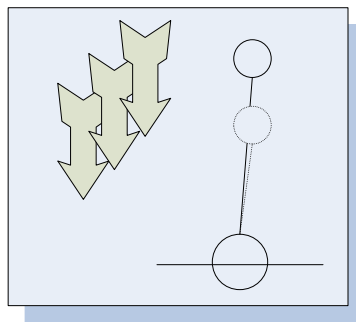


Abbildung 48: Auswirkungen der Schwerkraft

Eine weitere Kraft ist die in jedem Szenario herrschende Schwerkraft, die einer konstanten Kraft nach unten gleicht (siehe Abbildung 48). Sie kann nicht geändert werden und ist standardmäßig in jedem Szenario vorhanden. Abhängig von der Masse der einzelnen Partikel hat sie unterschiedlich starke Auswirkungen auf die verschiedenen vegetativen Strukturen. Um ein Greifen der Schwerkraft gleich zu Beginn zu ermöglichen, sind die Partikel bereits zu Anfang leicht zu ihrer ursprünglichen Position versetzt. Wenn alle Partikel exakt übereinander platziert sein würden, könnte die Schwerkraft aufgrund der Numerik nicht greifen.

7.2.2 Realisierung

Die Idee der Kräfte bzw. des Windes beruhen auf einer Kombination aus drei gleichartigen „Perlin Noise“ Strukturen. Diese sind in einem Gitter angeordnet, das über der Simulationsfläche der Vegetation verankert ist. Damit ist es für die Wirkung auf die Vegetation im Raum platziert. Das Gitter basiert auf Zufallszahlen und zählt damit zur Untergruppe der „Lattice Value Noise“ Strukturen. Der erste „Perlin Noise“ wird für die Hauptwindrichtung verwendet. Die beiden anderen stehen für die Turbulenzen in der jeweiligen orthogonalen Richtung zur Verfügung.

Die Turbulenzen sind damit sowohl unabhängig von der Hauptwindrichtung als auch zu der jeweiligen anderen Turbulenzrichtung. Diese Unabhängigkeit betrifft sowohl deren Ausmaß als auch deren Parametrisierung. Dadurch kommt es zu keiner Änderung in den entsprechend anderen Bereichen, wenn eine der drei Bestandteile verändert wird. Alle drei „Perlin Noise“ Strukturen werden bei der Erstellung zu einer einzigen Kraftwirkung kombiniert. Sie liegen im Gitter über den jeweiligen anderen Gittern gleich auf.

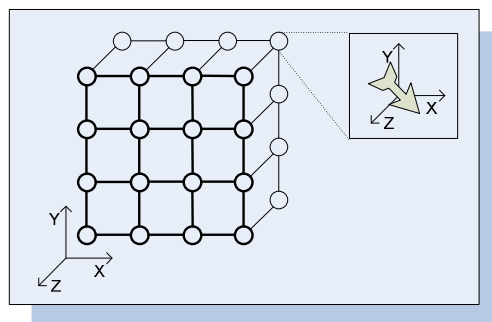


Abbildung 49: 3D „Perlin Noise“ Gitter

Die „Perlin Noise“ Strukturen sind vierdimensionale Gitter, wobei drei Dimensionen für die räumlichen Parameter des Windes verantwortlich sind (siehe Abbildung 49). Die vierte Dimension entspricht der zeitlichen Abänderung der Vektoren des Gitters (siehe Abbildung 50). Für jeden Knoten des Gitters wird ein dreidimensionaler Vektor berechnet. Dieser Vektor bestimmt die Wirkung des Windes auf die in der Nähe befindlichen Partikel der in der Simulation vorhandenen vegetativen Strukturen. Nur wenn sich ein Partikel direkt an der Position eines Knotens befindet, wird diese Wirkung ohne weitere Berechnung übertragen.

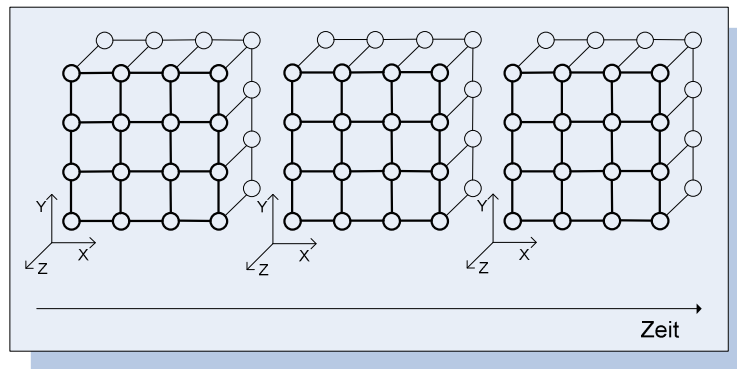


Abbildung 50: 4D-Gitter, „Perlin Noise“

Wenn sich Partikel ein wenig abseits eines solchen Knotens befinden, muss zwischen den nächsten vier Knoten, die das Partikel einschließen, eine Interpolation durchgeführt werden. Bei dieser Interpolation wird eine Kosinusinterpolation verwendet. Die Kosinusinterpolation ist nicht so schnell wie eine lineare Interpolation. Sie besteht aber mit guten Ergebnissen und der Aufwand bleibt trotzdem geringer als bei einer kubischen Interpolation. Dadurch ist es möglich, ohne weitere Algorithmen zur Glättung der Werte auszukommen.

Um die „Perlin Noise“ nicht völlig willkürlich einer „White Noise“ Zufallszahlenberechnung zu überlassen, unterliegen die Basiszufallszahlen für jede „Perlin Noise“ Struktur einer Normalverteilung. Diese Normalverteilung wird durch Parameter an die jeweiligen Wünsche angepasst. Durch diese Steuerung der Zufallszahlen werden Winde besser kontrollierbar und können so den Nachteil der Basierung auf „White Noise“ anstatt „Pink Noise“ besser überbrücken. Die Zufallszahlen sind durch die Parametrisierung nicht mehr völlig unabhängig, wie es bei einfachem „White Noise“ der Fall wäre.

Alle „Perlin Noise“ Strukturen sind auf Zufallszahlen aufgebaut. Diese werden zum Zweck der Nachvollziehbarkeit deterministisch erzeugt. Dies gewährleistet eine Wiederholbarkeit der Simulationen. Dieser Determinismus bezieht sich jedoch nur auf die Zufallszahlen der Winde, die vom System geladen werden. Davon sind nicht Winde betroffen, die die gleichen Parameter besitzen. Diese sind in ihren Basiszufallszahlen unterschiedlich. Das heißt, dass ein Wind immer dieselben Kräfte erzeugt. Dabei ist es egal, in welchem Simulationsdurchlauf er eingesetzt wird. Werden zwei Winde mit den gleichen Parametern in das System eingefügt, so sind diese aufgrund der unterschiedlichen Basiszufallszahlen zueinander unterschiedlich.

7.2.3 Aufbau der Kräfte

Die gesamten Kräfte, die im System aktiv sind und auf die im System befindliche Vegetation wirken, ordnen sich in einem Vektorengitter an. Dieses Gitter ist je nach Szenario unterschiedlich dicht an Gitterknoten. Der Vektor eines Knoten berechnet sich anhand aller Kräfte und deren Kombination, sowohl in deren Hauptrichtung als auch aufgrund der Turbulenzen. Errechnet werden die Kräfte einzeln aus der Entnahme von normalverteilten Zufallszahlen aus den „Perlin Noise“ Strukturen. Dies erfolgt aus Geschwindigkeitsgründen aus zuvor berechneten normalverteilten Zufallstabellen.

Die Vektoren der Knoten stehen in Form von Polarkoordinaten zur Verfügung, da das Hinzufügen der Turbulenzen dadurch effizienter ist. Durch die Verwendung von Polarkoordinaten sind die zweite und dritte Dimension, die Winkel Polar und Azimut. Die erste Dimension entspricht der Länge des Vektors und damit der Stärke des Windes. Dies ist eine reelle Zahl und wird direkt angegeben. So können die einzelnen Bereiche des Vektors unabhängig zum Rest verändert werden. So kann zum Beispiel die Windstärke erhöht werden, ohne die Auslenkung des Windes zu berühren.

Winde haben zu Beginn eine Default Hauptrichtung, die vor oder während der Simulation interaktiv geändert werden kann. Die Turbulenzen drehen sich mit der Hauptwindrichtung und sind daher stets gleich in ihrem Verhältnis zur Hauptwindrichtung. Winde können sich sowohl in der Hauptwindrichtung als auch innerhalb ihrer Turbulenzen beeinflussen. So können sich Winde mit den gleichen Parametern und der gleichen Hauptwindrichtung in ihren Turbulenzen aufheben oder auch verstärken.

7.2.4 Überführung in das dynamische System

Die Kräfte werden im Gegensatz zur Vegetation nicht bei der Überführung in das dynamische System umgewandelt, sondern direkt in ihrer Form verwendet. Das Gitter der Vektoren wird direkt über das Szenario gelegt (siehe Abbildung 51). Die notwendigen Kräfte für die Bewegung der vegetativen Strukturen werden daraus entnommen. Die Einstellung der Hauptwindrichtung kann erst nach der Übernahme in ein System vollzogen werden. Diese Richtung ist unabhängig von der Parametrisierung eines Windes. Bei mehreren Winden addieren sich die Werte der einzelnen Winde, wenn diese Winde aktiv sind.

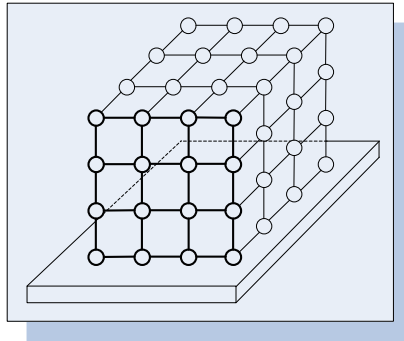


Abbildung 51: Platzierung des Kräftegitters auf der Simulationsfläche

Die Aktivierung eines Windes kann zu Beginn eines Simulationslaufes erfolgen oder auch währenddessen. So können Winde in eine Simulation aufgenommen werden, ohne dass auch ihre Kräfte in das System einfließen. Auch kann der Einfluss von Winden zeitlich begrenzt werden. Grundlegend können so Szenarien ausgetestet werden und überflüssige Winde einfach beiseite gelassen werden. Dabei muss das Szenario nicht neu definiert werden. Ein weiterer Vorteil dabei ist, dass Winde zugeschaltet werden können, um eventuelle Änderungen im Gesamtszenario zu simulieren.

7.3 Dynamisches Feder-Dämpfer-Masse-System

In diesem Kapitel werden die Vegetationsmodelle und die Kräftemodelle zu einem einzigen Simulationssystem zusammengefügt und die Zusammenführung der beiden Komponenten behandelt. Auch wird der Aufbau und die Durchführung der physikalisch-basierten Simulation erklärt. Das Kapitel unterteilt sich in vier Bereiche, „Analyse“, „Realisierung“, „Aufbau des dynamischen Feder-Dämpfer-Masse-Systems“ und „Überführung der Daten zur Visualisierung“.

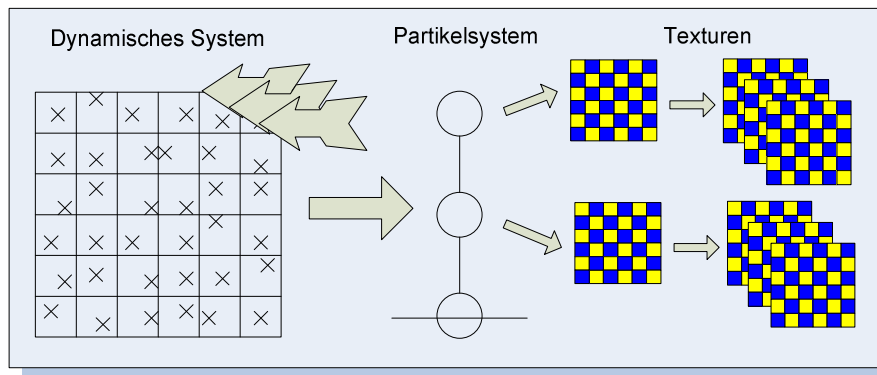


Abbildung 52: Speicherung der Daten des dynamischen Systems

Danach wird die Speicherung der Daten erklärt (siehe Abbildung 52). Es wird in „Überführung der Daten zur Visualisierung“ die Aufbereitung der Daten in Texturen erklärt. Die Texturen werden für das anschließende Laufzeitsystem verwendet.

7.3.1 Analyse

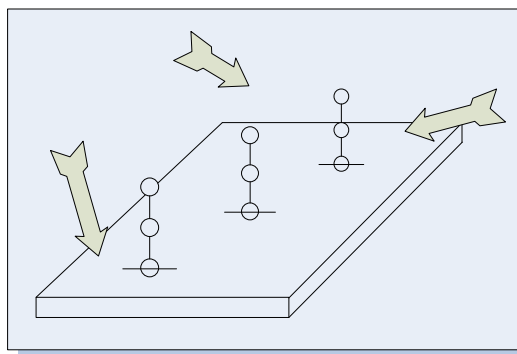


Abbildung 53: Verteilung der Vegetation und Kräfte auf der Simulationsfläche

Die Gestaltung der Szenarien sollte flexibel bezüglich des Ausmaßes der Absicht der zu sammelnden Daten sein. Aufbauend auf einer Basisfläche sollen vegetative Strukturen und Kräfte verteilt werden können (siehe Abbildung 53). Die Aufmerksamkeit liegt dabei auf der Größe der Simulationsfläche als auch auf der Feinheit des Kräftegitters. Das Kräftegitter ist dabei ein Gitter aus Richtungsvektoren. Diese Richtungsvektoren werden durch die Kombination aller Winde in der Simulation gebildet. Das Kräftegitter liegt dabei über der Simulationsfläche. Eine größere Fläche erzeugt ein größeres Ausmaß an Daten, wodurch eine Visualisierung umfangreicher gestaltet werden kann. Die Feinheit des Kräftegitters erhöht die Unterschiede in den Bewegungsmustern. Damit lässt sich die Bewegung realistischer und auch abwechslungsreicher gestalten.

Nicht nur die Größe eines solchen Szenarios, sondern auch die freie Festlegung der Anzahl an unterschiedlichen Vegetationsstrukturen und Kräften muss innerhalb des Systems möglich sein. Dies ermöglicht eine Anpassung an verschiedenste natürliche Phänomene in der Natur. Daher ist die Kombination unterschiedlicher vegetativer Strukturen und Kräfte notwendig. Auch untereinander müssen diese zwei Komponenten miteinander kombinierbar bleiben. Die Anzahl und die Unterschiede zwischen den unterschiedlichen Komponenten eines Szenarios dürfen dabei keine Hindernisse aufwerfen.

Die Verarbeitung der Eingaben sollte einer allgemeinen Art und Weise folgen und nicht von Spezialfällen und Ausnahmen abhängig sein. Damit werden unvorhersehbare Grenzen vermieden. Eine Teilung auf eine lokale Anwendung der Kräfte auf die vegetativen Strukturen sollte eine derartige Verarbeitung ermöglichen. Dabei soll durch die fehlende Interaktion der Vegetationsstrukturen untereinander kein Nachteil bezüglich des Realismus entstehen.

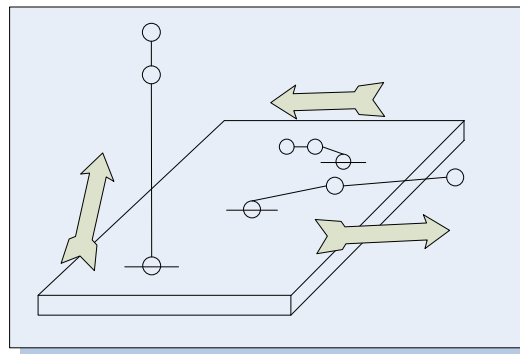


Abbildung 54: Visualisierung zu Adaptionzwecken

Während des gesamten Vorganges der Simulation wäre eine Visualisierung des Vorganges wünschenswert (siehe Abbildung 54). Erst durch eine solche Visualisierung können bereits im Vorfeld ungenaue oder falsche Voraussetzungen erkannt und entsprechende Adaptierungen vorgenommen werden. Probleme, wie zum Beispiel ein zu starker Wind, eine zu flexible Vegetation, etc., können gleich zu Beginn behoben

werden. Andernfalls wäre dies erst nach der Auswertung der Daten möglich. Die zusätzliche Belastung des Vorganges durch die Visualisierung kann somit durch die Einsparung von Zeit bei der Adaption durchaus gerechtfertigt werden.

Durch die Berechnung entsteht eine Fülle von Informationen. Eine Speicherung der Daten kann schon während der Berechnung erfolgen. So können bereits sich nicht mehr ändernde Daten aus dem System genommen und für eine später Verwendung gesichert werden. Damit würden sie das System nicht mehr weiter belasten oder sich in unnötiger Weise sammeln. Ebenfalls sollten die Daten auch aufbereitet und bearbeitbar bleiben.

7.3.2 Realisierung

Umsetzung finden diese Forderungen in einem dynamischen System, genauer gesagt in einem Feder-Dämpfer-Masse System. Die Vegetationsdefinition wird dabei in Form von Partikelsystemen in das System eingeführt und als solche während der gesamten Berechnung behandelt. Das System selbst baut auf diesen beiden Eingaben die Berechnung auf. Dabei basiert die Berechnung aufgrund diskreter Zeitschritte. Die Bewegung der Vegetationsstrukturen wird aufgrund der Kräfte mit Hilfe eines Einschrittverfahrens („Runge-Kutta“ Verfahren) berechnet.

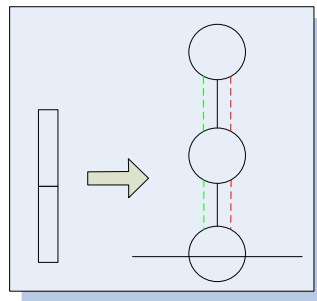


Abbildung 55: Umsetzung des L-Systems in Partikelsystem, Dämpfer (grün), Federn (rot)

Die Partikelsysteme sind dabei die exakte Umsetzung der L-Systeme, die im Vorfeld definiert wurden (siehe Abbildung 55). Diese sind so aufgebaut, dass jeweils ein Literal im L-System einem Partikel mit zugehöriger Masse entspricht. Die Partikel werden mit einer Feder-Dämpfer-Verbindung mit dem jeweiligen vorhergehenden Partikel (Elternpartikel) verbunden. Damit werden sie zu einer vegetativen Struktur zusammengebunden. Das erste Partikel (Wurzel) ist ohne Masse und dient nur der Verankerung des Partikelsystems auf der Simulationsfläche. Die Wurzel wird zu keiner Zeit durch Gravitation oder Winde bewegt. Partikelsysteme beeinflussen sich nicht untereinander, etwa durch Kollisionen oder Windschatten. Sie agieren völlig

unabhängig voneinander und beeinflussen sich nur durch ihre Feder-Dämpfer-Verbindungen gegenseitig. Diese sind ausreichend, um die Bewegung zu simulieren.

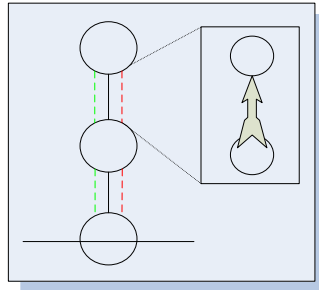


Abbildung 56: Feder-Dämpfer-Verbindung als Vektor

Die Feder-Dämpfer-Verbindungen zwischen den einzelnen Massepartikeln werden in Form von Vektoren verwendet (siehe Abbildung 56). Es wird dabei sowohl der Abstand zwischen den Partikeln als auch die strukturelle Beschaffenheit des Systems erhalten. Das System versucht immer trotz des Kräfteeinflusses, in seine Ausgangslage zurückzukehren. Diese Ausgangslage muss daher in entsprechender Form im System vorhanden sein. Bloße Entfernungen wären nicht ausreichend, um die Struktur in gleicher Form wiederherzustellen. Selbst im Falle des Einsatzes von Mehrfachbindungen zu anderen Partikeln wäre dies nicht möglich. Erst durch die Bewahrung von Winkeln zwischen den Verbindungen ist es möglich, auch die Struktur zu erhalten.

Da jedoch die Materialien der Vegetationsstrukturen nur begrenzt dehnbar sind, muss ein weiteres Verfahren zum Erhalt der Länge der Verbindungen angewendet werden. Ansonsten würden starke Winde und Kräfte die Verbindungen unrealistisch weit dehnen. Dieses Verfahren greift erst nach der im jeweiligen Zeitschritt erfolgten Berechnung. Es stellt damit sicher, dass sich Verbindungen zwischen Partikeln nur entsprechend ihrer Möglichkeiten dehnen. Alle Überschüsse an Ausdehnung werden verhindert und als strukturelle Kräfte an die Vegetationsstruktur weitergeleitet. Diese weitergeleiteten Kräfte werden teilweise von der Struktur als Bewegung verarbeitet oder durch Dämpfer in den weiteren Verbindungen absorbiert.

Die Berechnung innerhalb der Zeitschritte erfolgt mit einem 4-dimensionalen „Runge-Kutta“ Verfahren. Dieses Verfahren bietet weitaus größere Genauigkeit als einfache Euler Verfahren und trägt damit wesentlich zur Stabilität des Systems bei. Da das Verfahren vier Eulerschritte in einem Schritt berechnet, ist es aufwendiger in der Berechnung. Die Schrittberechnung ist der Hauptgrund dafür, dass das System bei einer großen Anzahl an Partikeln nicht in Echtzeit laufen kann. Das „Runge-Kutta“ Verfahren bietet ein ausreichend exaktes Verfahren, um die Stabilität des Systems zu gewährleisten.

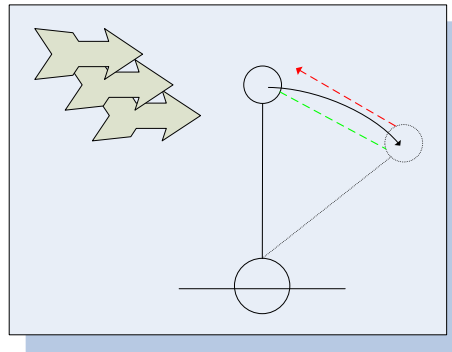


Abbildung 57: Kräftewirkung auf Partikel, Dämpfer reduziert Energie (grün), Feder versucht originalen Zustand wiederherzustellen (rot)

Die Kräfte wirken auf die jeweiligen Partikel und bringen so das System in Bewegung (siehe Abbildung 57). Aufgrund der Federn im jeweiligen System stehen diesen Kräften entgegengesetzte Kräfte gegenüber. Abhängig von dem aktuellen Ausmaß der Auslenkung der Partikel und der Stärke der Feder variieren diese Gegenkräfte.

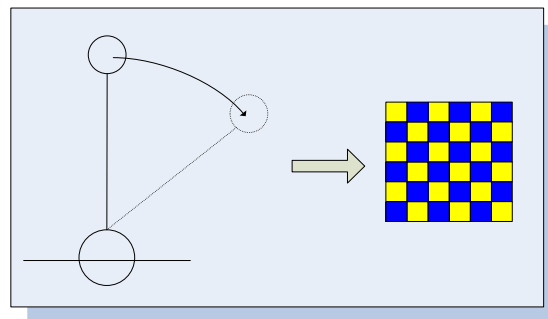


Abbildung 58: Speicherung der Bewegungsdaten

Die Speicherung der Daten erfolgt in einem Bild, das die jeweilige Abweichung der Partikel von dessen Ausgangsposition im System als Farbe darstellt (siehe Abbildung 58). Jeweils ein Farbkanal entspricht einer räumlichen Koordinate. Für jedes Partikel in einer Struktur wird dabei ein eigenes Bild angelegt. Für jedes Pixel in diesem Bild wird eine Struktur im Gesamtsystem platziert. Nach Belieben können ein oder mehrere Partikelstrukturen für ein Pixel dieses Bildes positioniert werden. Dadurch besitzt jedes Pixel eines Bildes genau zugehörige Partikel, die nur für die Daten in diesem Pixel zuständig sind. Sollten mehrere Strukturen für ein Pixel zur Verfügung stehen, wird der Mittelwert ihrer Positionsänderung in der Textur gespeichert.

7.3.3 Aufbau des dynamischen Feder-Dämpfer-Masse-Systems

Auf der Simulationsfläche werden die vegetativen Strukturen und die Vektoren der Kräfte platziert (siehe Abbildung 59). Diese dient grundsätzlich der Darstellung als auch der Feinheit des Kräftegitters. Sie können zu jeder Zeit einfach in ihrer Hauptwindrichtung verändert werden.

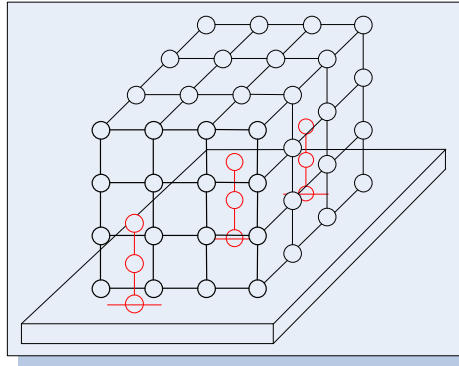


Abbildung 59: Simulationsfläche mit Kräftegitter und Vegetation

Zusätzlich wird diese Fläche aufgrund der Größe der zu speichernden Bilder unterteilt. Je nachdem wie viele Pixel das Bild hat, ist auch die Anzahl der Unterteilungen. Die Strukturen einer Vegetation werden mit einer festgelegten Anzahl in den entsprechenden Bereichen der Pixel verteilt. Diese Vegetationsstrukturen werden innerhalb eines Pixelbereichs zufällig gesetzt, sodass sie auch von unterschiedlichen Kräften betroffen sind. Werden zum Beispiel pro Pixel jeweils zwei vegetative Strukturen platziert, so sind diese in ihrer Position leicht unterschiedlich. Damit ist ihre Bewegung nicht völlig identisch. Alle Daten für ein Pixel werden dann aus dem Durchschnitt dieser beiden berechnet.

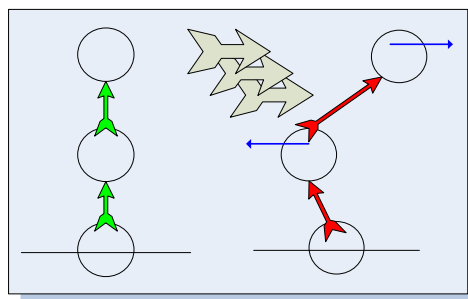


Abbildung 60: Kraftberechnung, ursprüngliche Distanz (grün), aktuelle Distanz (rot), Beschleunigung (blau)

Um die Systeme in einem brauchbaren Kontext zu halten, sind die Konstanten für die Federn und Dämpfer immer konstant in ihrem Verhältnis zueinander. Dabei ist die Konstante der Feder immer doppelt so groß wie die Konstante des Dämpfers, um eine ausreichende Oszillation zuzulassen. Berechnet wird die Kraft für jedes Partikel wie folgt

$$F = (D_R - D) * s - v * d,$$

wobei F der Kraftvektor ist, die auf den Partikel und den Elternpartikel wirkt. Die D_R entspricht der ursprünglichen Distanz als Vektor (siehe Abbildung 60). Dieser Vektor wird aufgrund von Rotationen der Struktur angeglichen. D ist die tatsächlich aktuelle Vektordistanz zwischen dem aktuellen Partikel und seinem Elternpartikel. Jeder Partikel hat zudem nur einen einzigen Elternpartikel. Die Variable s repräsentiert die Federkonstante und d ist die Dämpferkonstante. v entspricht der aktuellen Beschleunigung des Partikels.

Zusätzlich zu den Feder-Dämpfer Verbindungen verfügt das System über einen zusätzlichen Strukturerehalt. Die Länge der Verbindungen zwischen zwei Partikeln wird dadurch überwacht. Dieser Schritt schließt an die gesamte Auswertung der Kräfte in dem System an. Der Strukturerehalt ändert im Nachhinein je nach Grad der Flexibilität die Position eines Partikels. Dabei werden die Kräfte teilweise in die aktuelle Bewegung des Partikels als Rotation überführt. Diese Bewegung entspricht einer strukturell bedingten Ausweichbewegung. Teilweise werden die Bewegungen auch von der Struktur selbst absorbiert. Dadurch wird die vegetative Struktur in ihrer Form erhalten.

Die Bewegungen der Vegetation des Systems werden in einem konstanten Schritt durch das „Runge-Kutta“ Verfahren berechnet. Sie können daher auch in jedem dieser Schritte festgehalten werden. Das System benötigt sehr kleine Schritte, um einen ausreichenden Grad an Genauigkeit zu erhalten. Daher ist eine Speicherung in jedem Schritt nicht unbedingt notwendig. Am Ende der Simulation werden alle Einzelbilder zu einem einzigen 3D-Bild zusammengeführt. Zusätzlich werden auch zu jedem Bild die wirkenden Kräfte aufgezeichnet. Zu jedem Partikel wird der Positionsoffset zu dem Wurzelpartikel in der zugehörigen Struktur der Vegetation gespeichert.

7.3.4 Überführung der Daten zur Visualisierung

Die gespeicherten Bilder in dem 3D-Format entsprechen den Daten für die Positionsänderung der zu animierenden Vegetation. Dabei enthalten diese auch die Positionsdaten der vegetativen Strukturen. Diese Positionsdaten werden zur Zuordnung der Positionsänderungen und der richtigen Struktur benötigt (siehe Abbildung 61). So kann Gras animiert werden, indem die gespeicherten Daten direkt über dieses Feld gelegt werden. Die einzelnen Strukturen der Vegetation werden mit den jeweiligen überlagernden Daten animiert.

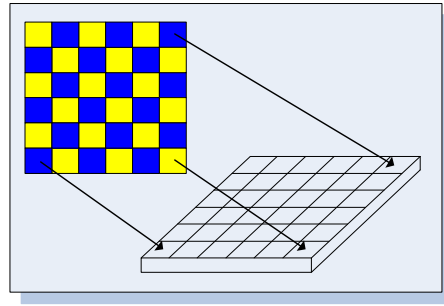


Abbildung 61: Implizite Positionsdaten der Textur

Die Speicherung der Daten erfolgt immer pro Partikel in der Berechnung (siehe Abbildung 62). Die Daten eines Partikels sind daher immer getrennt von den restlichen Partikeldaten. Um nun eine komplette Struktur in einer Visualisierung zu animieren, müssen die Einzeldaten auf die Struktur übertragen werden. Sie werden so zu einer einzigen Animation zusammen gebaut. Dies hat den Vorteil, dass die Daten einzeln verwendet werden können. Dadurch können sie auf Teilstrukturen oder Näherungen angewendet werden. So können Verfahren wie Billboarding oder LOD verwendet werden, ohne zusätzliche Berechnung für einfachere Strukturen durchführen zu müssen.

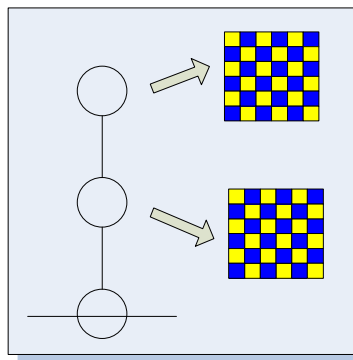


Abbildung 62: Speicherung der Daten pro Partikel

Die Daten werden direkt aus dem Bildformat für die Animierung verwendet und benötigen keine weitere Bearbeitung oder zusätzlichen Rechenaufwand. Somit können diese Daten auch für die Vegetation verwendet werden. Da der Bereich der Vegetation meist nur über wenige Ressourcen verfügt, können so leicht Vegetationsstrukturen in einer komplexen Szene mit diversen anderen Objekten animiert werden.

8 Laufzeitsystem: Animation von Vegetation in Echtzeit

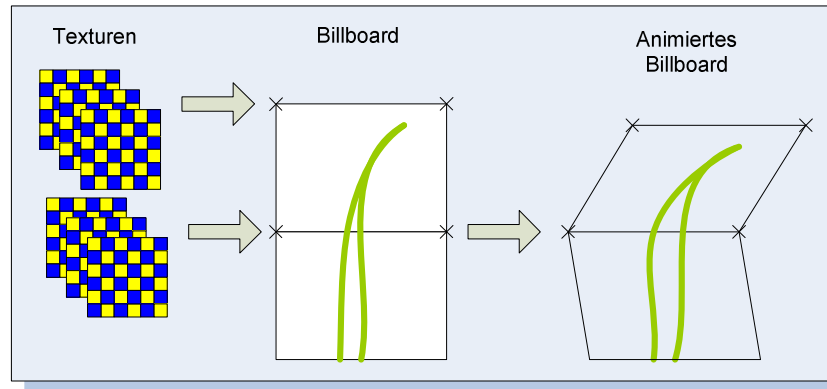


Abbildung 63: Aufbau des Laufzeitsystems

Im Kapitel 8 wird die Visualisierung der Präprozessdaten erläutert (siehe Abbildung 63). Es wird erklärt, wie sich diese Visualisierung aufbaut. Die Anwendung der Texturen auf die Visualisierungsstrukturen (Billboards) wird dabei behandelt. Das Kapitel unterteilt sich in drei Bereiche: „Analyse“, „Realisierung“ und „Aufbau der Visualisierung“.

8.1 Analyse

Die Visualisierung der Daten stellt die Nutzung der Präprozessberechnungen dar. Die Präprozessdaten sollen in einer Form genutzt werden, die möglichst schnell und effizient ist. Dabei sollen zudem möglichst wenige Ressourcen verbraucht werden. Auf diese Art kann die Animation im Hintergrund laufen, was bei der Bewegung von Gräsern und Bäumen wünschenswert ist. Weitere Berechnungen in der Szene dürfen dadurch nicht behindert werden. Nur so ist es möglich, eine derartige Animation auch in eine Szene zu integrieren, ohne diese unnötig mit Berechnungen zu belasten.

Die Visualisierung selbst muss auf die Datenform (3D-Texturen) zugeschnitten sein und muss diese möglichst ideal ausnutzen. So können Bildformate nicht nur als Datenpakete genutzt werden, sondern auch deren spezielles Format durch Shader. Diese arbeiten zumeist schneller und effizienter als ein direktes Auslesen und anschließendes Übertragen der Daten auf herkömmlichen Wegen. Zudem kann auf diese Weise die Verarbeitung der Daten besser vom restlichen System abgeschlossen werden. Dies erleichtert sowohl die Bearbeitung als auch die Portierung der Daten.

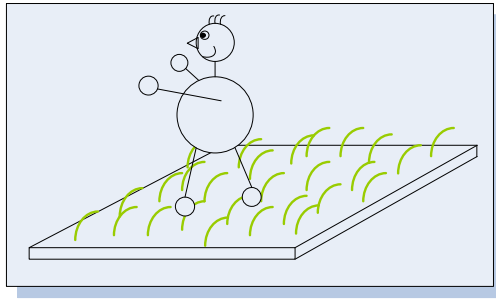


Abbildung 64: Anpassung der Animation an die Szene

Die Daten sollten trotz der Präprozessberechnung immer noch modifizierbar und an die Szene anpassbar bleiben (siehe Abbildung 64). So können leicht das Ausmaß der Bewegung und die Animationsgeschwindigkeit erhöht werden. Dabei bleiben die Relationen zwischen den einzelnen Datenteilen konstant. Bewegungen können damit auf die entsprechende Skalierung der Szene gebracht werden.

8.2 Realisierung

Für die Auslesung der Daten und Durchführung der Animation wird ein Vertex Shader verwendet. Ein Shader kann äußerst effektiv mit den Daten arbeiten. Er hat die Möglichkeit eines direkten und schnellen Zugriffs auf die Bilddaten und kann gleichzeitig die ausgelesenen Daten durch einen Vertex Offset auf die ihm unterliegende Vegetation anwenden. Für diesen Fall kann nur ein Shader 3.0 und höher verwendet werden. Die Auslesung aus dem Bildformat muss nämlich bereits im Vertexshader erfolgen, um sie auf die vegetativen Strukturen anwenden zu können.

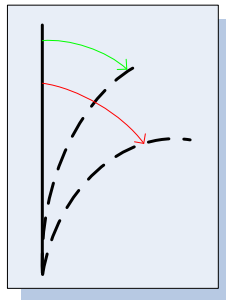


Abbildung 65: Skalierung der Bewegung, keine Skalierung (grün), doppelte Auslenkung (rot)

Zusätzlich gibt es zwei Parameter. Der erste stellt einen Multiplikator für die Auslenkung dar und ermöglicht eine Skalierung der Bewegung (siehe Abbildung 65). Der zweite Parameter ist ein Multiplikator für die Zeit (siehe Abbildung 66). Durch

diesen Zeitparameter können die Bewegungen gedrosselt oder beschleunigt werden. Diese Parameter werden direkt im Shader zur Modifizierung der Bewegungen verwendet. Damit kann die Animation noch besser an die Szene anpassen werden. Durch geschicktes Verwenden dieser Parameter kann auch leicht die Simulation eines Anschwellens der Windstärke während der Animation erreicht werden. Dafür müssen ebenfalls keine zusätzlichen Berechnungen mehr durchgeführt werden.

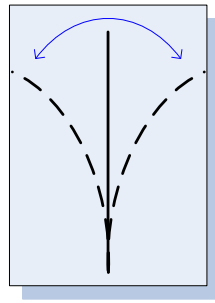


Abbildung 66: Zeitanpassung der Bewegung

8.3 Aufbau der Visualisierung

Die Animation passiert auf einer quadratischen Fläche. Auf dieser Fläche werden Billboards in unterschiedliche, zufällige Rotationen und in regelmäßiger Anordnung gesetzt. Die Anordnung wird deshalb in dieser Art durchgeführt, weil dadurch die Bewegung der einzelnen Strukturen einer Vegetation besser zu Geltung kommt. Außerdem können die Übergänge besser beobachtet werden. Selbst bei sehr turbulenten Winden kann hierbei immer noch der Zusammenhang der Bewegung gesehen werden und dass es sich um eine einzige Simulation handelt.

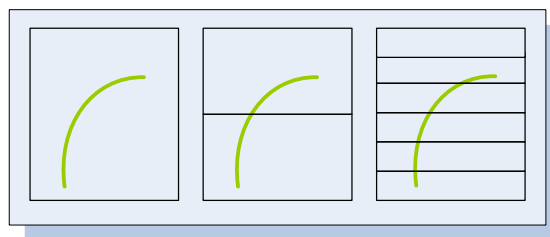


Abbildung 67: Unterteilung der Billboards

Die Billboards sind aufgrund der verwendeten Daten nicht einfache Rechtecke sondern setzen sich aus sechs aufeinander gestapelten Rechtecken zusammen (siehe Abbildung 67). Diese sind miteinander verbunden. Die Grastextur wird über diese gesamte Billboardstruktur gelegt, indem die Texturkoordinaten nach Position des jeweiligen

Rechtecks übergeben werden. Bei der Animation der Bewegung werden keine Texturveränderungen durchgeführt. Nur die Vertices der Geometrie werden bewegt. Während der Bewegung werden die einzelnen Bilderdaten durchgegangen und zwischen den Werten interpoliert. Dies erzeugt eine flüssige und weiche Bewegung der Billboards. Am Ende der Datenreihe beginnt die Animation wieder am Beginn und verwendet das letzte und erste Bild als Interpolationsgrundlage. Werden das Anfangs- und Endbild dabei entsprechend gewählt, können sich die Animationen in einer nahtlosen Endlosschleife bewegen.

9 Implementierung des Präprozesses

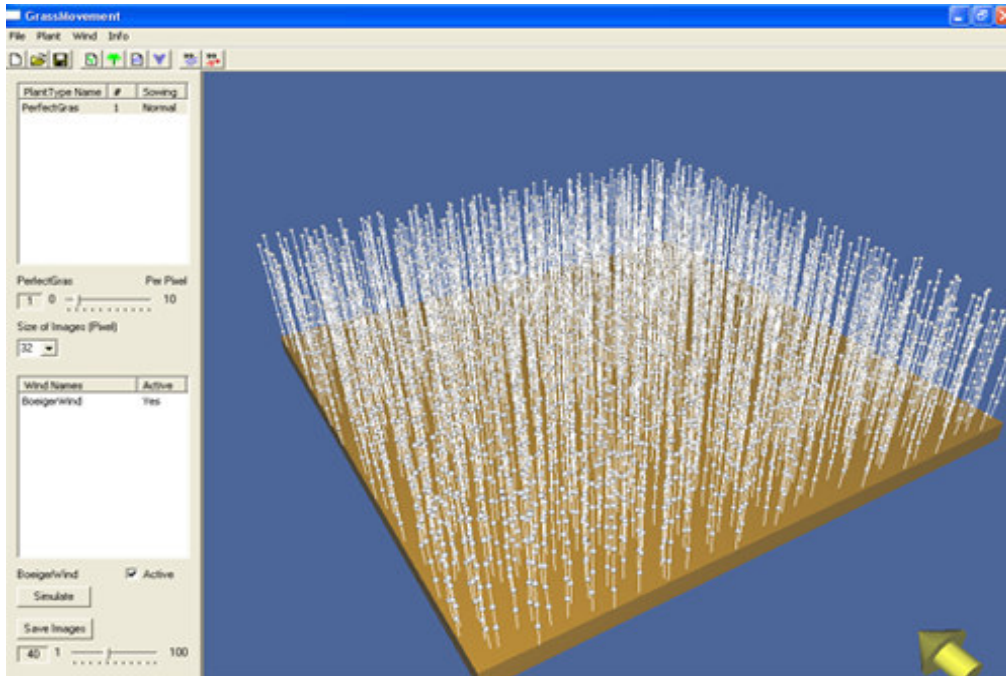


Abbildung 68: Implementierung des Präprozesses (GrassMovement)

Der Präprozess erstellt seine Berechnungen völlig unabhängig vom Laufzeitprozess. Es handelt sich um ein völlig eigenständiges Programm namens „GrassMovement“ (siehe Abbildung 68). Die physikalisch-basierte Simulation wird in diesem Programm durchgeführt. Zur Implementierung wurde Visual C++ verwendet. „wxWindows“ wurde zur Erstellung des Interface genutzt. Die Darstellung basiert auf OpenGL. Das benötigte Betriebssystem ist Microsoft Windows XP (Service Pack 2). Es wurde in Microsoft Visual Studio .NET 2003 (Service Pack 1) implementiert.

Wie auch das Konzept wird die Implementierung des Präprozesses in drei Bereiche untergliedert, die „Implementierung der Vegetation“, die „Implementierung der Kräfte“ und die „Implementierung des dynamischen Systems“.

9.1 Implementierung der Vegetation

Die Umsetzung des L-Systems ist, wie auch alle anderen Teile des Programms, in einem grafischen Interface realisiert. Damit gestaltet sich die Bedienung so übersichtlich wie möglich. Ein eventuelles Feedback kann direkt während der Erstellung des individuellen Systems übermittelt werden. Dieses Feedback wird sowohl grafisch als auch durch diverse Meldungen des Systems umgesetzt.

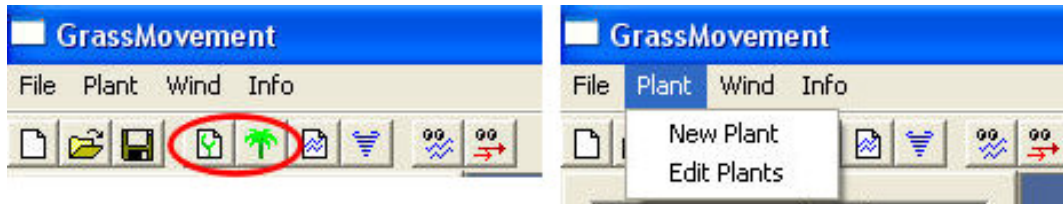



Abbildung 69: Interface, „New Plant“ und „Edit Plants“

Neue Systeme im Programm werden mit der Funktion „New Plant“  erstellt (siehe Abbildung 69). In der Eingabemaske „Create New Plant“ ist es möglich, neue L-Systeme vollständig zu erstellen. Es beinhaltet zusätzlich ein grafisches Fenster, welches sich rechts oben befindet. In diesem Fenster wird das aktuelle L-System dargestellt. Die Eingabemaske (siehe Abbildung 70) wird im Folgenden noch genauer erklärt.

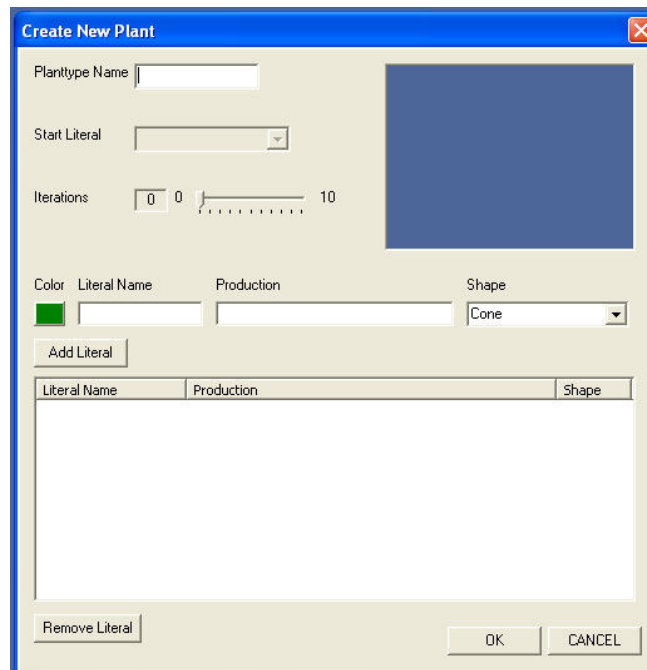


Abbildung 70: Interface, Erstellen einer neuen vegetativen Struktur

Im Parameter „Planttype Name“ muss der einmalige Name des L-Systems angegeben werden. Es sind nur Buchstaben und Zahlen erlaubt.

Der Parameter „Start Literal“ legt das Anfangswort für das L-System fest. Die Auswahl erfolgt aus den angegebenen Literalen. Es ist daher nicht aktiv, bis zumindest ein Literal eingegeben wurde. In jedem Fall wird das erste eingegebene Literal als Startliteral verwendet.

Der Parameter „Iterationen“ legt die Anzahl der Iterationen fest, die das L-System durchlaufen soll. Bei einer Anzahl von null Iterationen wird nur das Startliteral verwendet. Die Anzahl von zehn Iterationen ist deshalb limitiert, weil Grammatiken sehr schnell wachsen und daher zu umfangreich werden könnten. Die Leistung des Programms wäre sonst nicht gewährleistet.

Die Eingabe von Literalen umfasst die Parameter „Color“, „Literal Name“, „Production“ und „Shape“. „Color“ legt die Farbe fest. Im Parameter „Literal Name“ muss der Name des Literals festgelegt werden. Dieser darf nur aus Zeichen (A-Z, a-z) bestehen und mit den fünf optionalen Eigenschaften versehen werden. Diese fünf Eigenschaften und der Parameter „Produktion“ werden im Kapitel 7.1.3 genauer erklärt. „Shape“ legt schließlich die Form der grafischen Darstellung des Literals fest.

Der Parameter „Add Literal“ kann das Literal in den Pool übernommen werden. Dadurch geht es zu den verwendeten Literalen des L-Systems über. Durch einen Doppelklick auf das Literal wird dieses zur Änderungen wieder aus dem Pool genommen und in die Bearbeitungsleiste gezogen. Ist ein Literal im Pool selektiert kann es durch „Remove Literal“ vollständig aus dem Pool gelöscht werden.

Mittels „OK“ wird das L-System überprüft und gespeichert oder eine Fehlermeldung ausgegeben.

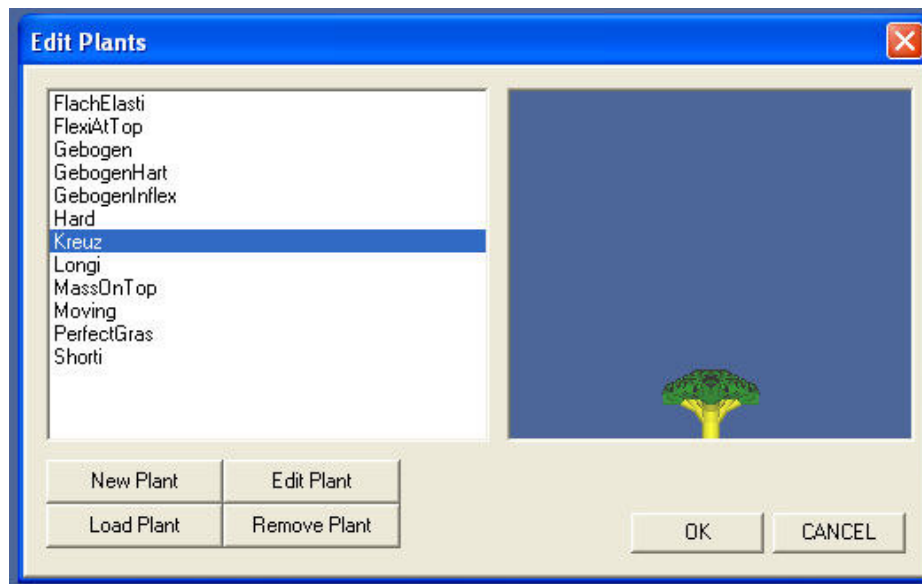



Abbildung 71: Interface, Editieren bereits erstellter vegetativen Struktur

In der Eingabemaske „Edit Plants“  (siehe Abbildung 71) können L-Systeme in jeder Art und Weise geändert werden, die das Programm zur Verfügung stellt. Links befinden sich alle L-Systeme. Rechts ist ein grafisches Fenster, das das aktuell selektierte L-System darstellt.

Durch „New Plant“ wird ein neues L-System angelegt, mit „Edit Plant“ wird das selektierte L-System erneut bearbeitet, durch „Remove Plant“ wird das aktuell selektierte L-System gelöscht und durch „Load Plant“ werden Definitionen geladen, die als Textfile vorliegen.

Der Aufwand bezüglich des Erstellens eines komplett neuen Szenarios hängt von der Komplexität der vegetativen Strukturen ab. Alle restlichen Einstellungen können schnell getätigt werden und brauchen kaum länger als eine Stunde. Die Strukturen einer Vegetation hingegen sind durch die Parametereinstellungen der kritischste Bereich und auch jener, der die meiste Zeit benötigt. Einfache Grasstrukturen können innerhalb einer Stunde, komplexe Bäume und Sträucher innerhalb eines Tages definiert werden. Voraussetzung dabei ist natürlich das Verständnis von L-Systemen.

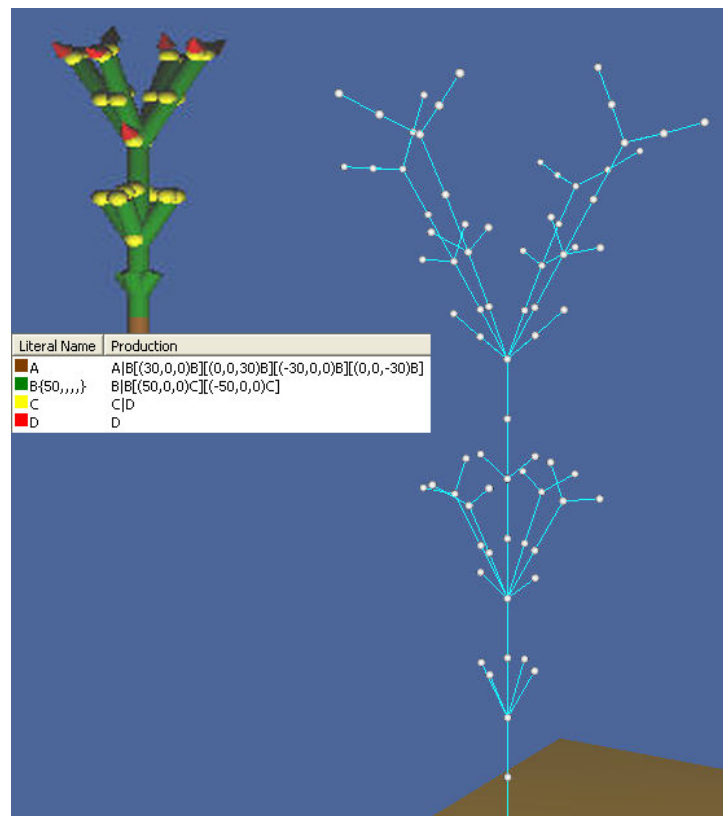


Abbildung 72: Komplexeres L-System (links) mit äquivalentem zugehörigem Partikelsystem (rechts)

Die Erstellung von komplexen vegetativen Strukturen ist deshalb zeitkritisch, weil der Eingabe kaum Grenzen gesetzt sind. Damit besteht eine Vielzahl von Möglichkeiten (siehe Abbildung 72). Durch die Vererbung der Eigenschaften kommt zusätzlich ein Faktor hinzu, der die Komplexität bei der Erstellung vergrößert. Diese Komplexität ist jedoch notwendig und äußerst hilfreiche bei der Definition von Vegetation. Auf der Abbildung 86 ist eine komplexere vegetative Struktur zu sehen, einmal bei der Definition als L-System und dann als übertragenes Partikelsystem. Hier kann leicht gesehen werden, dass Einstellungen wie die Masse, Steifheit oder Winkelsteifheit sehr umfangreich werden können, da komplexe L-Systeme sehr viele Knoten besitzen, die solche Eigenschaften tragen können.

9.2 Implementierung der Kräfte

Die Implementierung des „Perlin Noise“ wurde ebenfalls in einem grafischen Interface umgesetzt. Die Kräfte sind auf numerische Angaben beschränkt. Um den Wind grafisch betrachten zu können, muss die Simulation des dynamischen Systems durchgeführt werden. Dadurch wird der Wind mit den eingegebenen Werten berechnet.



Abbildung 73: Interface, „New Wind“ und „Edit Winds“





Auf die Menüs zur Erstellung oder zur Bearbeitung eines auf „Perlin Noise“ beruhenden Windes wird über die Funktionen „New Wind“  oder „Edit Winds“  zugegriffen (siehe Abbildung 73).



Abbildung 74: Interface, Icons „Show Winds“ und „Show Forces“

Zudem kann während der Simulation der Wind durch die Option „Show Winds“  und die wirkenden Kräfte der Partikel durch „Show Forces“  angezeigt werden (siehe Abbildung 74).

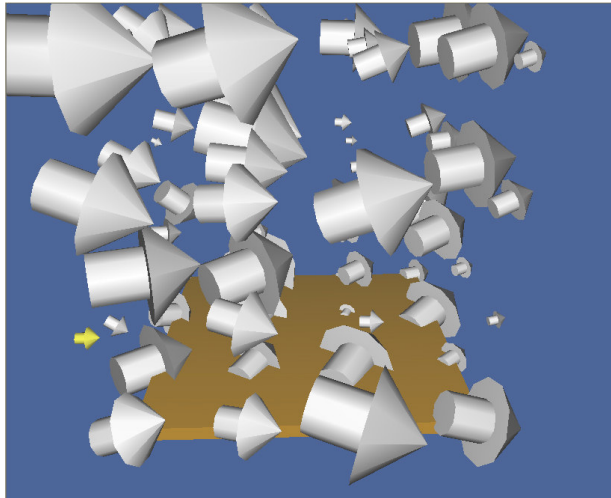


Abbildung 75: Interface, Visualisierung „Show Winds“

In Abbildung 75 ist die grafische Darstellung der Kräfte („Show Winds“) zu sehen. An jedem ganzzahligen Punkt im Kräftegitter wird die Kraft als Pfeil dargestellt. Die Krafrichtung entspricht dabei der Richtung und die Stärke der Größe des Pfeils.

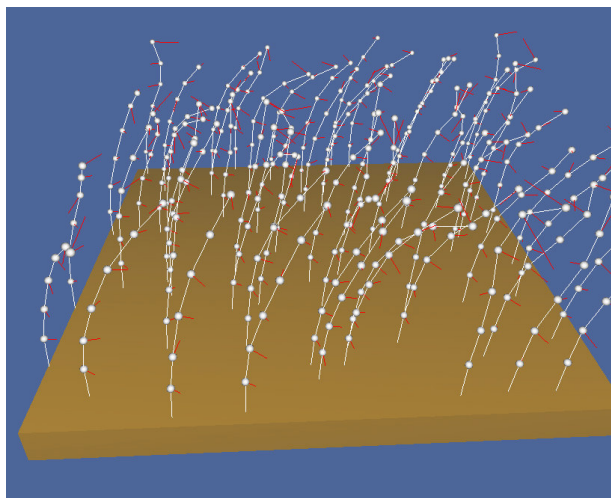


Abbildung 76: Interface, Visualisierung „Show Forces“

In Abbildung 76 kann man die Kräfte sehen, die auf das jeweilige Partikel wirken („Show Forces“). Sie sind durch rote Linien an das jeweilige Partikel geknüpft. Die Richtung der Linie gibt die Orientierung der Kraft und die Länge die Stärke der Kraft an.

Mit der Funktion „New Wind“ gelangt man zur Eingabemaske, um die Parameter eines neuen Windes angeben zu können. Die eingegebenen Werte werden auf Fehler überprüft.

„Wind Name“ steht für die Benennung des jeweiligen Windes. Da der Name die Identifikation des Windes ausmacht, kann nur jeweils ein Wind mit demselben Namen im System bestehen.

Die Bereiche „Wind Power“, „Polar Angle“ und „Azimut Angle“ untergliedern die Eingabe in drei Abschnitte. „Wind Power“ die generelle Windstärke. „Polar Angle“ gibt den Winkel in der Y-Windrichtungsebene und „Azimut Angle“ gibt den Winkel in der Ebene an, die zu dieser Ebene orthogonal liegt. Initialisiert werden alle diese Werte mit 0 und müssen daher nicht alle bei der Eingabe definiert werden.

„Power Offset“ gibt an, wie groß die Stärke des Windes zu jeder Zeit an jedem Punkt minimal sein muss. Dadurch erfolgt bei dem Wind jedoch kein Clipping sondern lediglich eine Verschiebung der errechneten Werte um den Offsetwert. Dadurch erhöht sich die gesamte Windstärke bei Erhöhung dieses Faktors ebenfalls. Dies ist notwendig, um Winde zu verhindern, die unnatürlich konstant bei einer gewissen Amplitude reagieren. Eine bessere Imitation eines echten Windes wird dadurch ebenfalls gewährleistet (siehe auch Kapitel 7.2).

„Power Amplitude“ gibt an, wie groß die größte Differenz bei der Modulation des Windes sein darf. Der Minimalwert der Stärke des Windes ist der Wert, der bei „Power Offset“ angegeben wurde. Der Maximalwert ist der Wert der „Power Amplitude“ plus dem „Power Offset“. Bei den beiden „Polar Amplitude“ und „Azimut Amplitude“ verhält es sich ähnlich, nur dass bei diesen kein Offset hinzu addiert wird. Sie sind nicht einem Stärkewert sondern in einer Winkelangabe in Rad angegeben. Bis zu diesem Wert wird der Wind maximal abgelenkt.

Über „Frequency“ sind die Werte in allen drei Bereichen gleich. Sie geben an, wie schnell Änderungen eintreten und wieder verschwinden. Hochfrequente Winde reagieren schneller und haben damit insgesamt eine höhere Dynamik als niedrigfrequente. Die Änderungen werden im System jedoch mit einer gewissen Trägheit aufgenommen. Dadurch können sich die Impulse die in das System gelangen sehr leicht gegenseitig aufheben, wenn es sich um einen hoch turbulenten Wind handelt.

„Deviation“ ist ebenfalls in allen Bereichen äquivalent und gibt die Abweichung der generierten Zufallswerte durch den Einsatz einer Normalverteilung an. Bei hohen Abweichungen erfolgt die Generierung im Zuge eines simplen „White Noise“. Bei geringer Abweichung hingegen tendieren die Werte zum Durchschnitt. Dadurch ist es sehr einfach möglich, den Wind mit Wahrscheinlichkeiten in einen gewünschten Bereich zu drängen. Definitionen von Winden liegen, wie auch schon bei L-Systemen, als lesbare Textdateien vor.

9.3 Implementierung des dynamischen Systems

Die Implementierung des dynamischen Systems baut auf die beiden vorhergehenden Teile auf und kombiniert sie zu der gewünschten Simulation. Wie auch die beiden anderen Teile ist die Erstellung des dynamischen Systems grafisch gelöst. Alle Bereiche können interaktiv gesteuert werden. Grafische Texturelemente dienen dem dynamischen System als Datenbehälter für die Speicherung der erzeugten Daten.

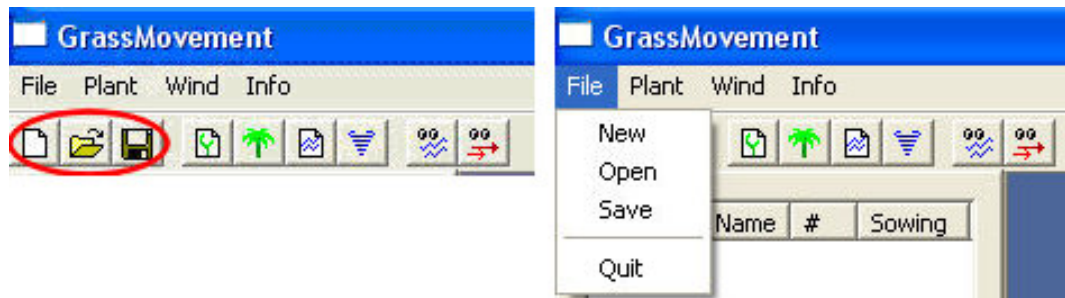





Abbildung 77: Interface, „New“, „Open“ und „Save“

Die Funktion „New“  erstellt ein neues dynamisches System. Mit „Open“  kann auf bestehende dynamische Systeme zugegriffen werden und „Save“  speichert das aktuelle dynamische System (siehe Abbildung 77).

Mit der Funktion „New“ gelangt man zur Eingabemaske eines neuen dynamischen Systems „Create New System“ (siehe Abbildung 78). Um jedoch ein sinnvolles System anlegen zu können, müssen bereits vorher Pflanzen und Winde definiert worden sein (siehe dazu Kapitel 7.1 und Kapitel 7.2). Die Maske besteht aus Parametern zur Festlegung der Größe des Feldes, einem grafischen Display zur Darstellung der selektierten Pflanze und zwei großen Bereichen. Der obere ist für die Auswahl der Pflanzen und der untere ist für die Auswahl bestehender Winde. Durch diese Maske ist es möglich in die Eingabemasken für „Edit Plants“ und „Edit Winds“ zu wechseln.

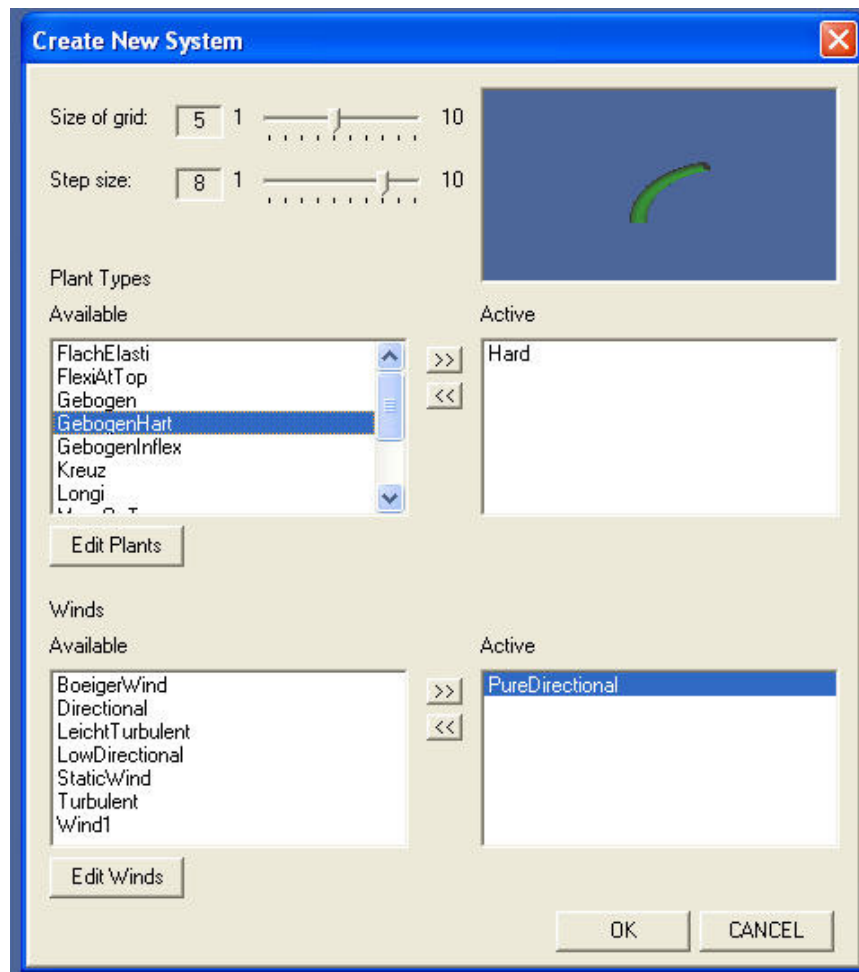


Abbildung 78: Interface, Erstellen eines neuen Dynamischen Systems

Die Parameter „Size of grid“ und „Step size“ sind ausschließlich zur Einstellung der Größe der verwendeten Ebene zuständig. Dabei gibt der Schieberegler „Size of grid“ die Anzahl der Rasterfelder an und „Step size“ die Größe eines einzelnen Rasterfeldes. Beide sind sowohl für die Ausdehnung in x sowie z-Richtung zuständig und ergeben damit immer eine quadratische Grundfläche.

Mittels „Plant Types“ werden die Pflanzenkomponenten festgelegt. Alle bereits erstellten Pflanzen scheinen links und alle verwendeten rechts auf. Mittels „Winds“ werden die Windkomponenten festgelegt. Alle bereits erstellten Winde scheinen links und alle verwendeten rechts auf. Mit „Ok“ wird zum Simulationsmodus gewechselt.

Bei Bestätigung eines neuen dynamischen Systems oder dem Laden eines bereits bestehenden Systems werden die Interfaceelemente in der Hauptansicht aktualisiert. Durch diese Aktualisierung können die Elemente spezifisch auf die entsprechenden

Durchläufe der Simulation eingerichtet werden. Aktuelle Einstellungen können über „Save“ dauerhaft abgespeichert werden.

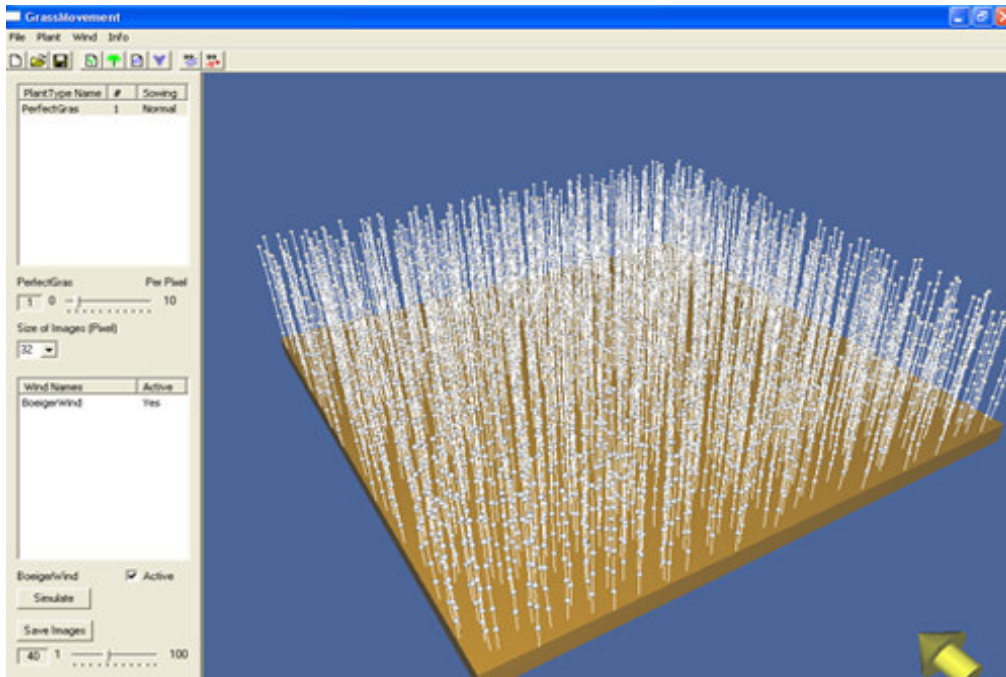


Abbildung 79: Interface, Hauptfenster

Das Hauptfenster besteht aus zwei Bereichen, dem grafischen Hauptfenster und dem Parameterfenster (siehe Abbildung 79). Im Hauptfenster ist die Simulation zu sehen. Im Parameterfenster können diverse Einstellungen vorgenommen werden.

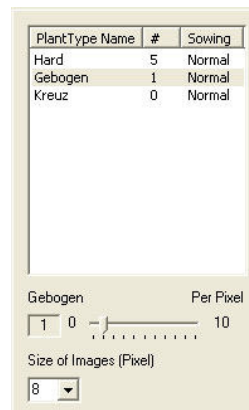


Abbildung 80: Interface, Hauptfenster, Parametereinstellung Pflanzen

Die Einstellungsmöglichkeiten beginnen bei den Pflanzenkomponenten, die ausgewählt wurden (siehe Abbildung 80). Für jeden Pflanzentyp kann angegeben werden, wie viele Pflanzen pro Pixel in der Simulation vorgesehen sind. Diese Einstellung wird durch Selektion der Pflanze und dem darunter liegenden Schieberegler vorgenommen. Außerdem werden alle Daten eines jeden Pflanzentyps eigens gespeichert, unabhängig von den anderen Pflanzentypen in der Simulation.

Der Parameter „Size of Image“ bestimmt, wie groß die errechnete Datenmenge sein soll. Hier wird festgelegt, wie viele Pixel in Höhe und Breite das errechnete Bild der Daten hat. Je größere diese Zahl, desto größer ist der Aufwand der Simulation. Dieser Wert sollte Anfangs klein gewählt werden, um die Simulation schneller (fast Echtzeit) ablaufen lassen zu können. Damit können die erhaltenen Daten visuell nachkontrolliert werden. Erst bei korrekten Einstellungen sollte auf eine höhere Anzahl gewechselt werden.

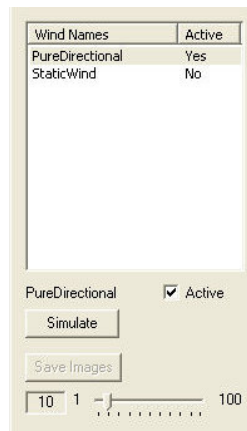


Abbildung 81: Interface, Hauptfenster, Parametereinstellung Winde

Der nachfolgende Bereich ist für die Einstellung der Winde zuständig (siehe Abbildung 81). Hier kann man die Winde durch Selektion und Aktivierung der Checkbox „Active“ aktivieren oder deaktivieren.

„Simulation“ startet die Simulation mit den vorgenommenen Einstellungen. Mit „Save Image“ beginnt das System die Daten zu speichern. Die Speicherung erfolgt innerhalb eingestellter Zeitschritte, ganz unten, und speichert jeweils ein Bild pro eingestellte Zeitschritte. Wird „Save Image“ ein weiteres Mal betätigt, werden alle bis zu diesem Zeitpunkt erzeugten Einzelbilder zu 3D-Bildern zusammengefasst.

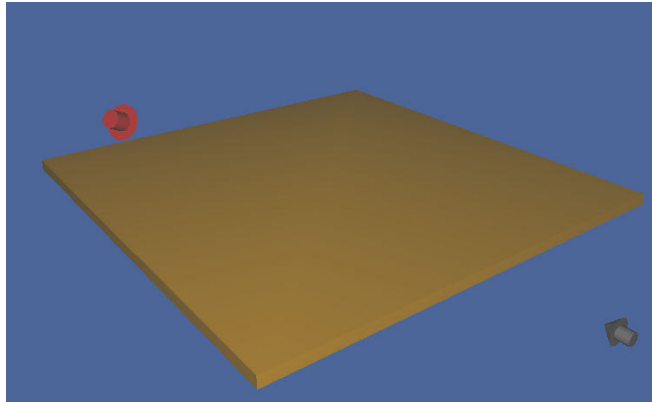
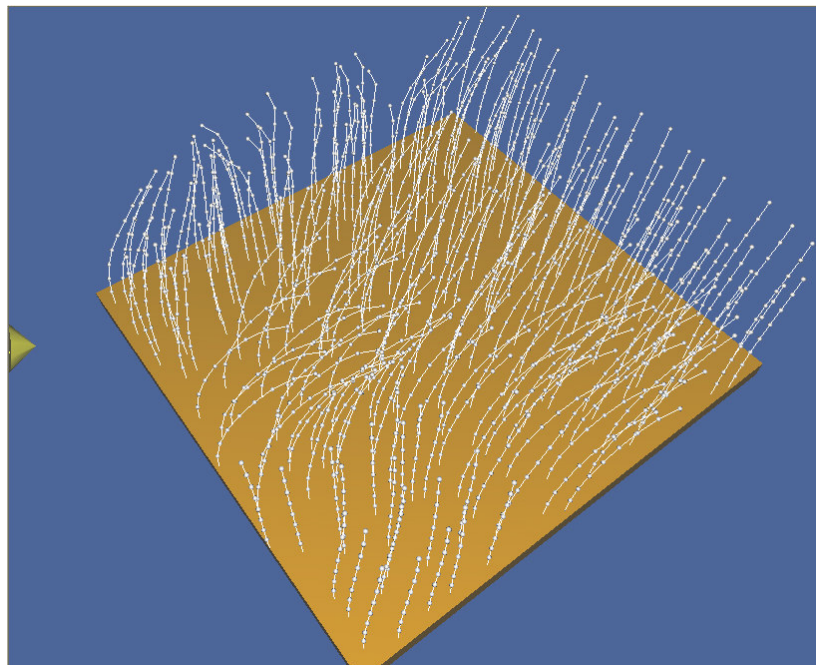


Abbildung 82: Interface, Einstellung der Krafthauptrichtungen

Weiters kann die Windrichtung innerhalb des grafischen Fensters festgelegt werden (siehe Abbildung 82). Dabei muss der jeweilige Wind in der Box selektiert sein und dann kann mit der linken Maustaste der Wind gezogen werden. Diese Einstellung kann auch während der Simulation verändert werden und kommt einer Drehung der Windrichtung gleich.



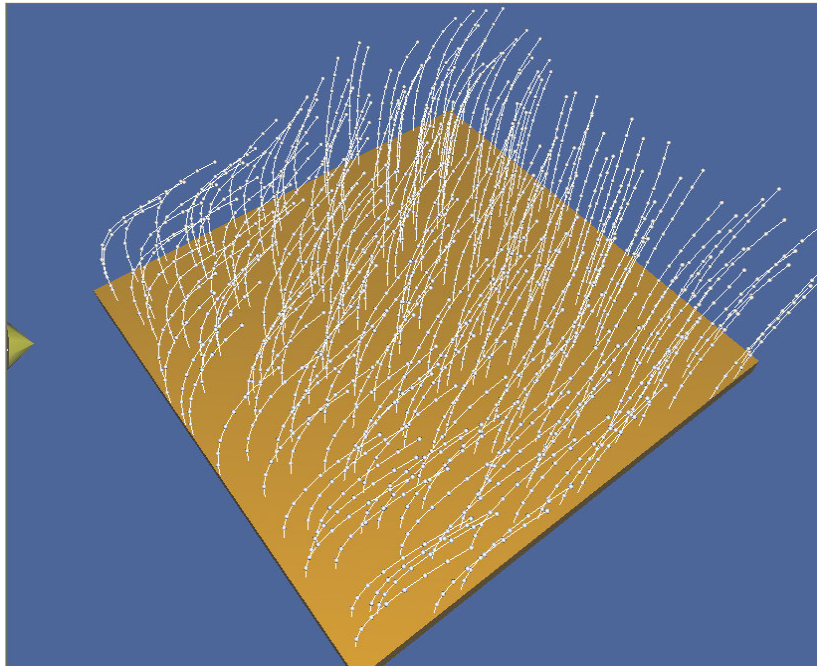


Abbildung 83: Durchführung der Simulation

In Abbildung 83 sind zwei Schritte derselben Simulation zu sehen. Sie zeigen, wie sich die Simulation über die Zeit entwickelt. Der gelbe Pfeil gibt die Windrichtung an. Die Partikel in der Simulation bewegen sich aufgrund der herrschenden Kräfte im System.

10 Implementierung des Laufzeitsystems

Das Laufzeitsystem ist ein eigenständiges Programm („GrassDisplay“), das in OGRE implementiert ist. Es arbeitet mit den Daten des Präprozesses und setzt diese Daten mit Hilfe eines Vertexshaders um. Zur Implementierung wurde Visual C++ verwendet. Die Darstellung erfolgt in DirectX. Das benötigte Betriebssystem ist Microsoft Windows XP (Service Pack 2). Es wurde in Microsoft Visual Studio .NET 2003 (Service Pack 1) implementiert.

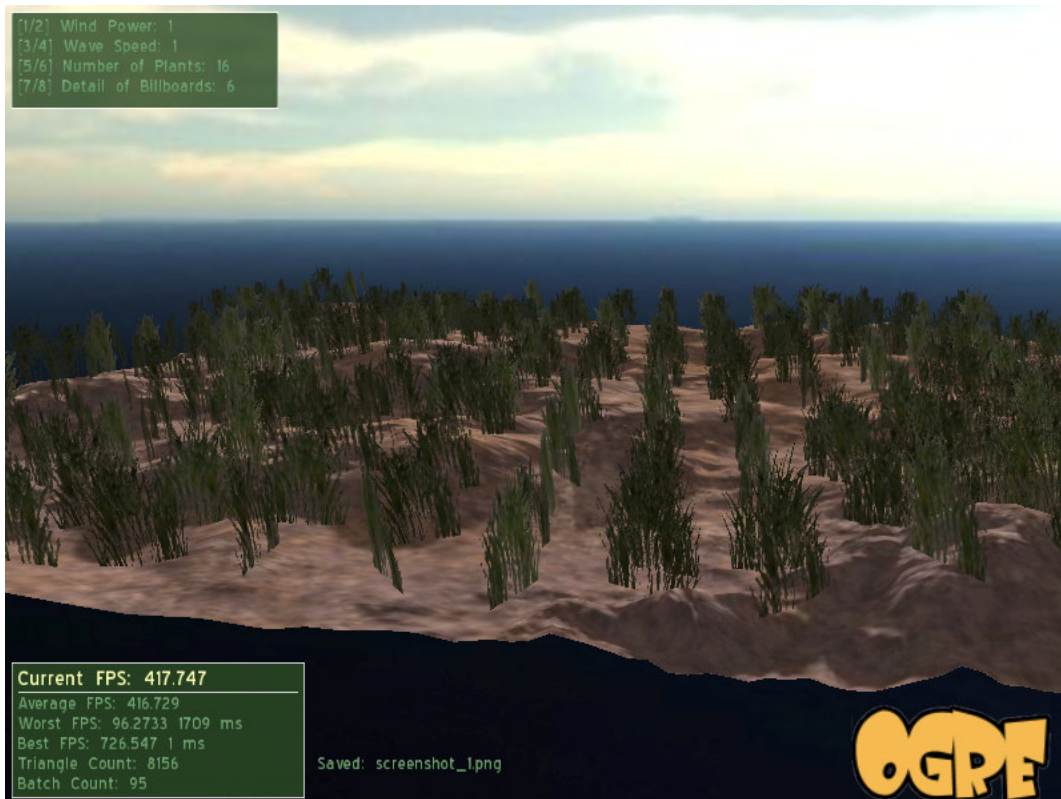
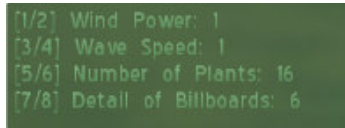


Abbildung 84: Implementierung des Laufzeitsystems

Die Visualisierung der erstellten Daten durch das dynamische System ist sehr einfach gehalten und die grafische Oberfläche beschränkt sich dabei auf einige wenige interaktive Möglichkeiten zur Veränderung der Auswirkungen der laufenden Visualisierung (siehe Abbildung 84). Es handelt sich nur um die reine Darstellung der Daten, wie sie genutzt werden können. Diese Visualisierung ist zudem datenabhängig und kann nur mit Daten bestimmter Pixelgröße arbeiten. Die Visualisierung wird anhand von einfachen Gräsern durchgeführt, die aus mehrstufigen Bitmapstrukturen und einer Grastextur basieren. Gestartet wird die Simulation für Pflanzen erst während

des Programms, sodass zuerst Änderungen bezüglich des Umfangs vorgenommen werden können.



```
[1/2] Wind Power: 1
[3/4] Wave Speed: 1
[5/6] Number of Plants: 16
[7/8] Detail of Billboards: 6
```

Abbildung 85: Interface, Parametereinstellung,
Zahlen neben dem Parameter zur Änderung
zB 1 erhöht „Wind Power“, 2 reduziert „Wind Power“

Eine Möglichkeit der Interaktion (siehe Abbildung 85) besteht aus der Einstellung der Windstärke „Wind Power“, die einen konstanten Multiplikationsfaktor darstellt, der mit der jeweiligen Positionsänderung aus den Daten multipliziert, damit die Stärke der Änderung der Position beeinflusst werden kann. Dadurch ist es möglich mit den gleichen Daten aus einer Simulation mit unterschiedlichen Stärken von Wind mit den gleichen Verhältnissen der Positionsänderungen zueinander zu vergleichen. Da sich die Verhältnisse zueinander nicht ändern, bleibt die Windart unverändert, wie zum Beispiel turbulenter Wind bleibt turbulenter Wind, lediglich die Stärke des Windes ändert sich.

Die Einstellung der Geschwindigkeit unter „Wave Speed“ beeinflusst die Folge, in der zwischen den Daten der einzelnen Positionsänderungen interpoliert wird. Eine höhere Geschwindigkeit hat zur Folge, dass die Positionsänderungen schneller durchgeführt werden und sich das Gras damit auch schneller bewegt als in der ursprünglichen Simulation. Damit können sanftere Winde begrenzt in turbulentere Winde verwandelt werden und umgekehrt. Zu schnelle Bewegungen haben jedoch zur Folge, dass die Darstellung kantig und unrealistisch wirkt, zu langsame Bewegungen ähneln einer Zeitlupenbewegung, wodurch diese Funktion nur begrenzt zur Änderung der Windart herangezogen werden kann.

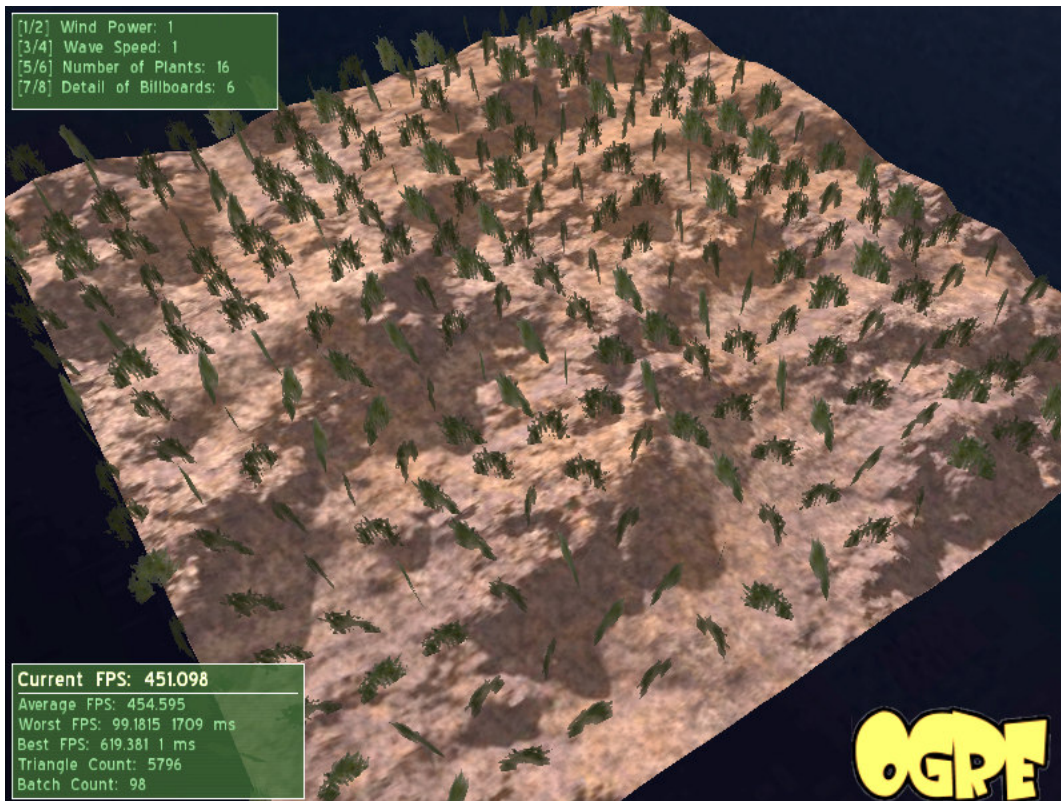


Abbildung 86: Billboards, Verteilung

Weiters kann die Anzahl der Billboards auf der Visualisierungsfläche durch den Parameter „Number of Plants“ eingestellt werden. Diese werden dann regelmäßig über die gesamte Fläche verteilt. Sie werden außerdem zufällig rotiert. In Abbildung 86 wird diese Verteilung gut ersichtlich, da sich nur wenige Billboards auf der Fläche befinden.

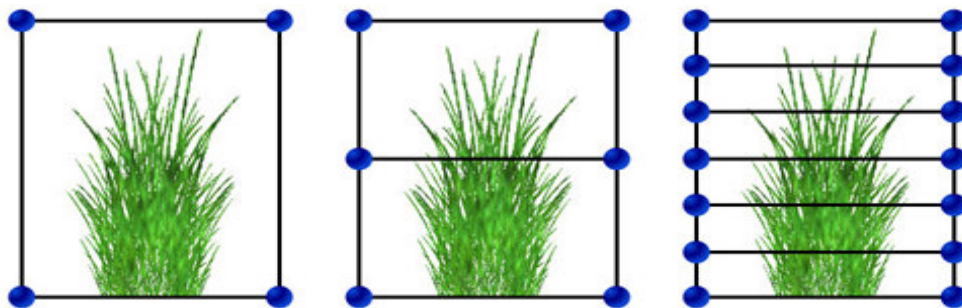


Abbildung 87: Billboards, einfach, 2-teilig, 6-teilig

Die Anzahl der Unterteilungen der Billboards kann durch „Detail of Billboards“ festgelegt werden. Dies erfolgt von einfachen über 2-teiligen bis hin zu 6-teiligen

Billboards (siehe Abbildung 87). Die Fläche der Billboards wird dabei durch weitere Vertices unterteilt. So unterteilt können die Billboards umfangreicher bewegt werden. Bewegungen von 6-teiligen Billboards sehen dadurch sehr viel realistischer aus.

Die Visualisierung selbst läuft automatisch aufgrund der basierenden Daten ab und kann nicht auf andere Weise beeinflusst werden. Sie stellt die Funktionsweise dar, wie die erstellten Daten verwendet werden können. Die Änderungen während der Visualisierung passen die Daten besser an die jeweilige Situation an, ohne den Realismus zu zerstören.

11 Resultate

Die Problematik der Seetangbewegung von Gräsern konnte durch das Verfahren sehr gut gelöst werden. Durch die Parametrisierung der Steifheit und Winkelsteifheit bewegt sich die Vegetation nun realistischer. Die unterschiedliche Vegetationsbewegung an unterschiedlichen Positionen der Fläche verhindert, dass Synchronitäten in den Bewegungen auftreten. So kann im gleichmäßigen Takt schwingende Vegetation verhindert werden, was zum Realismus der Animation beiträgt. Außerdem erleichtert das Verfahren die Animation unterschiedlichster Szenarien durch flexible Eingabemöglichkeiten.

Die Visualisierung der Daten erfolgte auf einem AMD Athlon 64 X2 Dual mit einem Core Prozessor 4200+ mit 2.21 GHz und 1.00 GB RAM. Die verwendete Grafikkarte war eine Nvidia GeForce 8800 GTX 768 MB PCI Express x16. Das unterliegende Betriebssystem war Microsoft Windows XP (Service Pack 2) und die verwendete Grafikkarte OGRE 1.4.3 für Visual C++.Net 2003, Version 7.1. Für die Visualisierung wurde ein Vertex Shader 3.0 verwendet, der die Anforderungen eines Texturelookups innerhalb eines Vertex Shaders erfüllt.



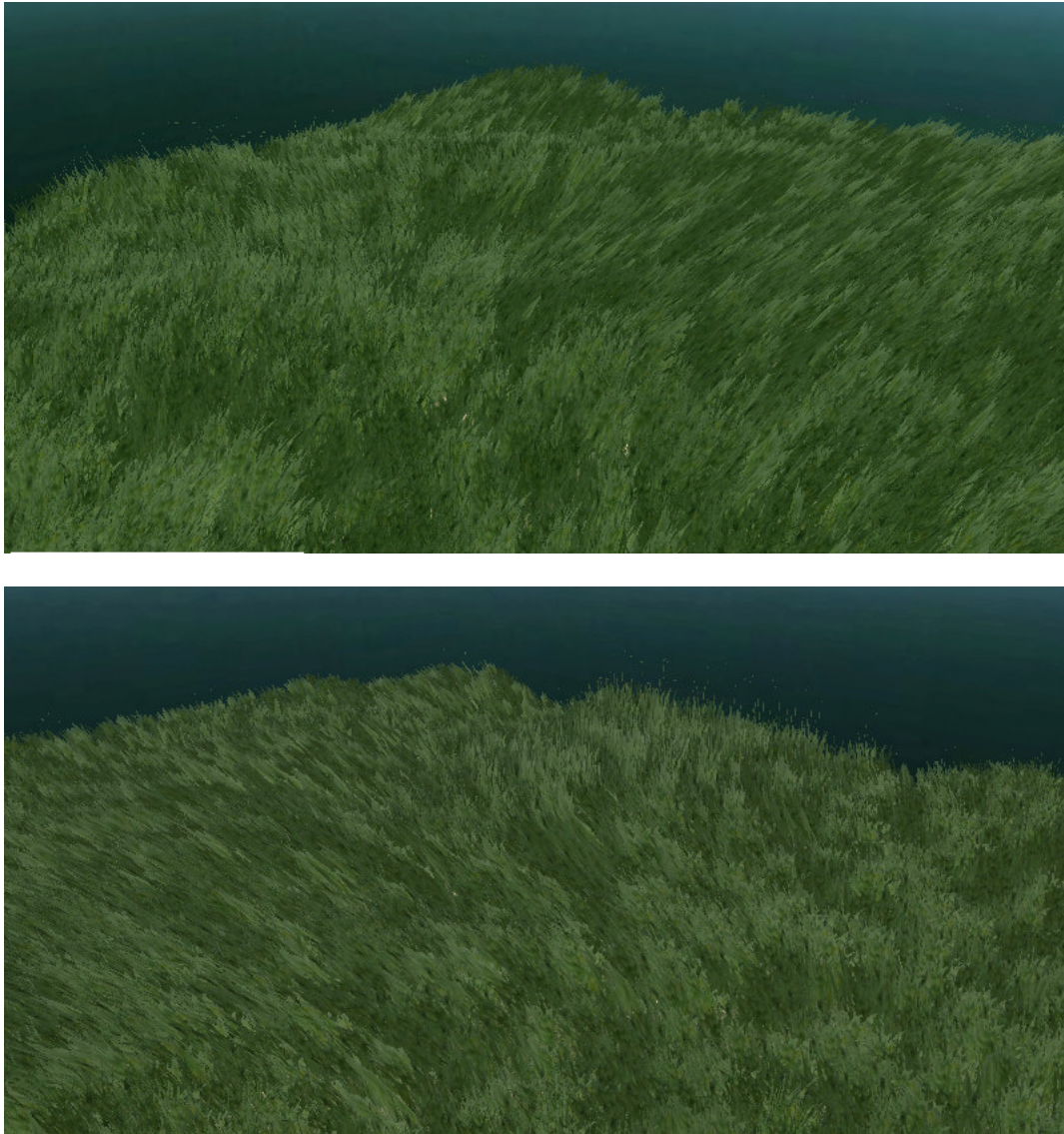


Abbildung 88: Echtzeitvisualisierung der Daten

In Abbildung 88 sind Aufnahmen des Laufzeitsystems zu sehen. Auf dem ersten Bild weht der Wind stark nach rechts und beginnt sich dann in seiner Richtung nach links zu drehen. Dabei nimmt die Vegetation am mittleren Bild eine kurze Ruheposition ein, um sich dann vollends am untersten Bild nach rechts zu drehen. Zudem ist auch klar zu sehen, dass nicht alle Gräser diesem Rhythmus folgen, sondern nur jeweils ein kleiner Bereich davon betroffen ist. Es wurde ein böiger Wind verwendet (siehe Abbildung 98), damit Änderungen deutlicher in Erscheinung treten.

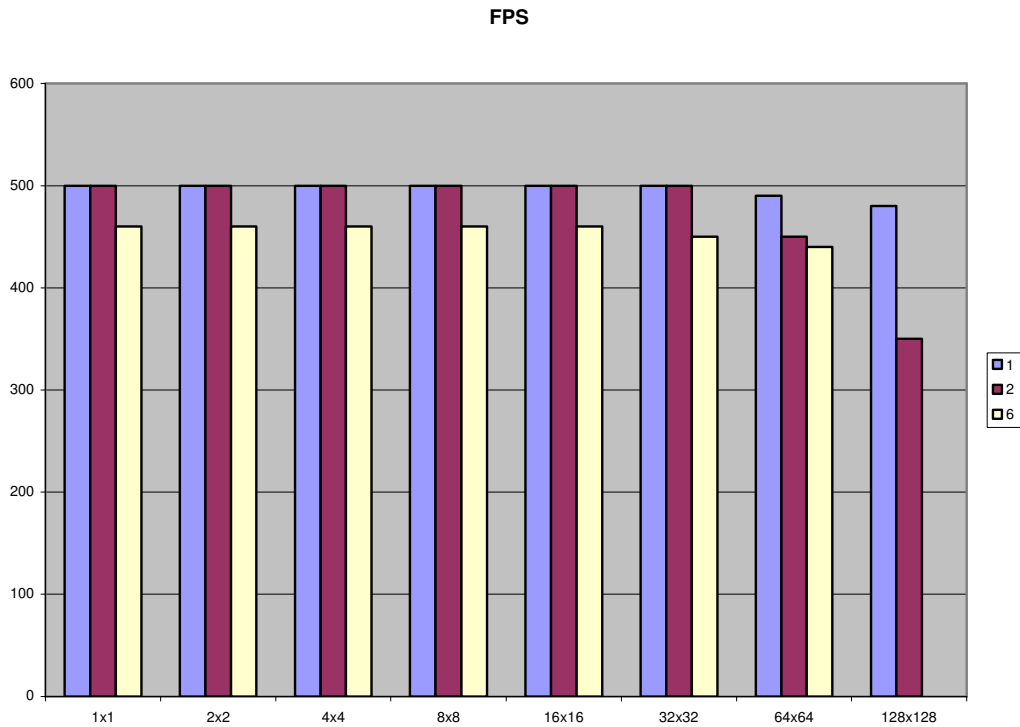


Abbildung 89: Performance der Echtzeitvisualisierung (in Frames pro Sekunde), FPS-Variation +/- 5

Performance	1-teilig	2-teilig	6-teilig
1x1	500	500	460
2x2	500	500	460
4x4	500	500	460
8x8	500	500	460
16x16	500	500	460
32x32	500	500	450
64x64	490	450	440
128x128	480	350	*

Abbildung 90: Zugehörige Tabelle der Performance

In Abbildung 89 kann der Test der Performance beobachtet werden. Sowohl die Abbildung als auch die Werte in der Tabelle (siehe Abbildung 90) basieren auf einer unterschiedlichen Anzahl von Billboards, von 1x1 bis 128x128 (16384), als auch in unterschiedlichem Detail von einfachen Billboards (1-teilig) bis zu Billboards (2-teilig, 6-teilig). Die vertikalen Unterteilungen erzeugen eine geschmeidigere Bewegung der Vegetation. Der letzte Test 128x128 mit 6-teiliger Unterteilung, welcher auch mit einem

* markiert wurde, konnte aufgrund von Limitation der verwendeten Grafikkarte nicht durchgeführt werden.

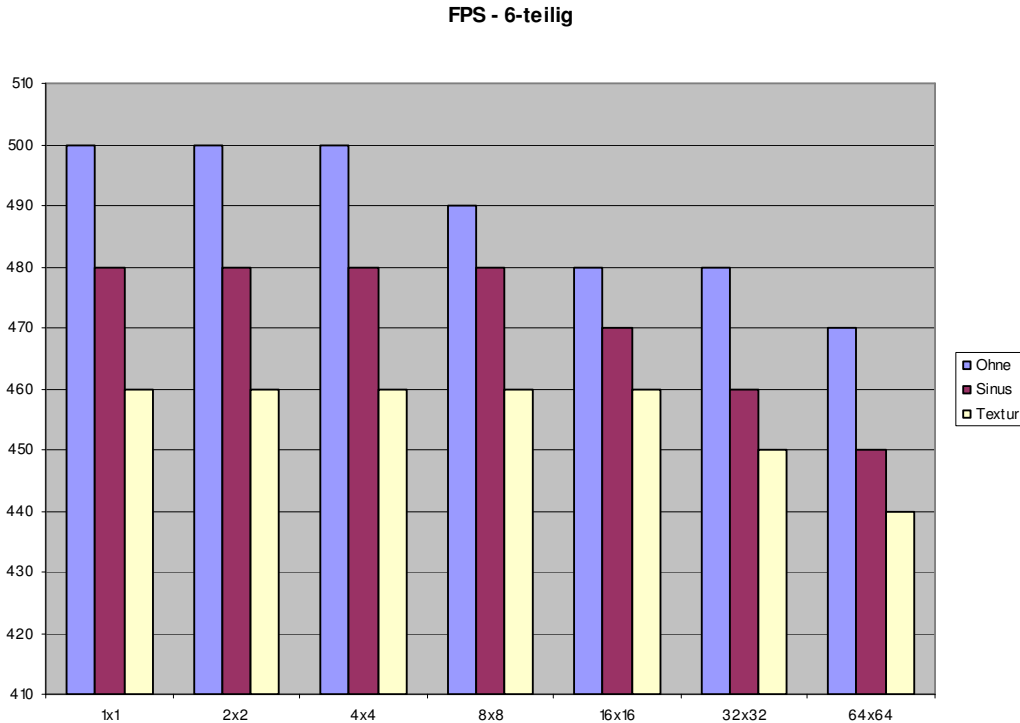


Abbildung 91: Performancevergleich, ohne Animation, mit Sinusbewegung, mit Texturlookup (in Frames pro Sekunde), FPS-Variation +/- 5

FPS - 6-teilig	Ohne	Sinus	Textur
1x1	500	480	460
2x2	500	480	460
4x4	500	480	460
8x8	490	480	460
16x16	480	470	460
32x32	480	460	450
64x64	470	450	440

Abbildung 92: Zugehörige Tabelle der Performancevergleich

In Abbildung 91 (zugehörige Tabelle Abbildung 92) kann das Ergebnis eines Performancevergleichs gesehen werden. Es wurde eine Szene bestehend aus 6-teiligen Billboards ohne jegliche Animation, mit einfacher Sinusanimation und mit der besprochenen Texturanimation durchgeführt. Ohne Animation wurde die Szene am schnellsten durchlaufen. Sowohl die Sinusberechnungen als auch die Texturlookups im

Vertex Shader führten zu einem Performanceverlust. Dabei hat die Methode des Texturlookups leicht schlechter abgeschnitten als die Methode mit Hilfe der Sinusberechnung. Im Hinblick auf den Gewinn an Realismus durch die Texturen kann jedoch der Mehraufwand durchaus als gerechtfertigt angesehen werden.

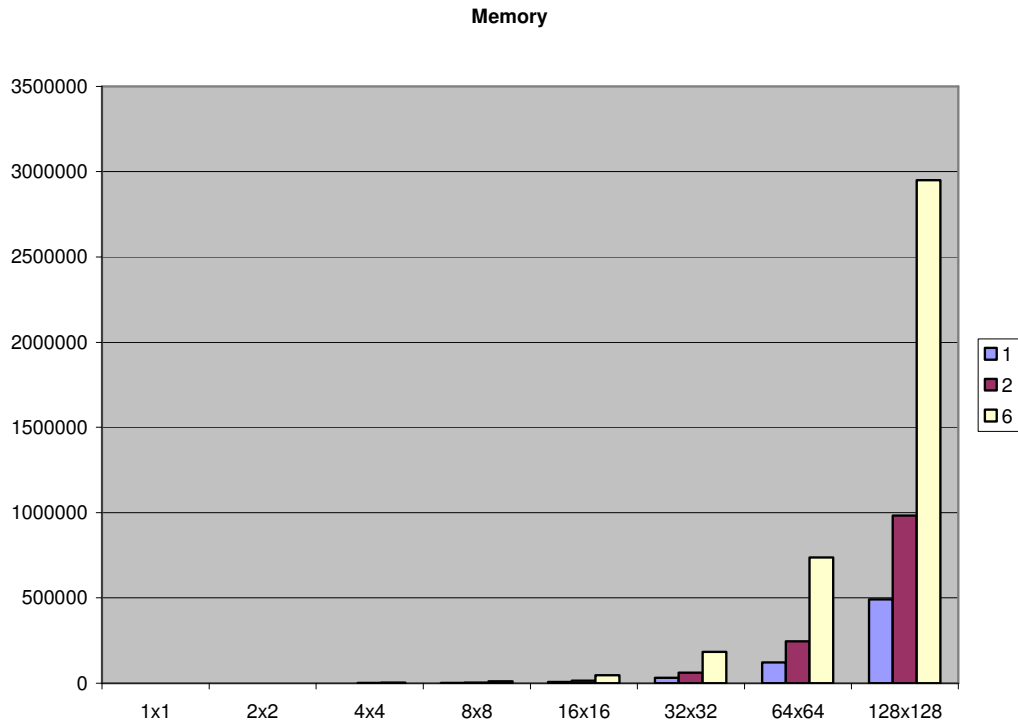


Abbildung 93: Speicherbedarf der Echtzeitvisualisierung (in Byte)

Memory	1-teilig	2-teilig	6-teilig
1x1	30	60	180
2x2	120	240	720
4x4	480	960	2880
8x8	1920	3840	11520
16x16	7680	15360	46080
32x32	30720	61440	184320
64x64	122880	245760	737280
128x128	491520	983040	2949120

Abbildung 94: Zugehörige Tabelle des Speicherbedarfs

Der Speicherbedarf in Form von Texturen ist, wie auch schon der Test der Performance, auf unterschiedlicher Anzahl von Billboards und Detailtiefe errechnet worden (siehe Abbildung 93 und 94). Die Berechnung basiert darauf, dass für jedes einzelne Billboard

in der Szene eine entsprechende Berechnung im Präprozess stattgefunden hat. Dies ist jedoch nicht unbedingt notwendig, sondern es können auch kleinere Texturen für eine größere Anzahl verwendet werden. Es muss lediglich zwischen den Pixeln interpoliert werden. Die Spalten 2-teilig und 6-teilig können aufgrund ihrer Unterteilung mehr Daten des Präprozesses in die Bewegung einbringen, wodurch die Anzahl der Texturen um den Faktor 2 bzw. den Faktor 6 steigt. Mehr Speicher wird hierbei für mehr Realismus in der Bewegung aufgewendet.



Abbildung 95: Vergleich Unterteilung Billboards, einfache Billboards (oben), 6-teilige Billboards (unten)

In Abbildung 95 ist der Unterschied zwischen einfachen und 6-teiligen Billboards zu sehen. Am oberen Bild kann man sehen, dass einfache Billboards in ihren Bewegungen

sehr eingeschränkt sind. Nur einfache Schrägstellungen sind dadurch möglich und der Realismus leidet dadurch. Im Gegensatz dazu stehen die 6-teiligen Billboards, die am unteren Bild zu sehen sind. Sie können weitaus detailliertere Bewegungen ausführen und wirken damit weitaus realistischer.

Die erhaltenen Ergebnisse sprechen durch die Visualisierung und Vergleichbarkeit mit anderen Animationen von Gras für sich. Die Parametrisierbarkeit gibt die Möglichkeit, sehr realistische Animationen zu berechnen und auch Probleme bei der Bewegung von Vegetation zu lösen.

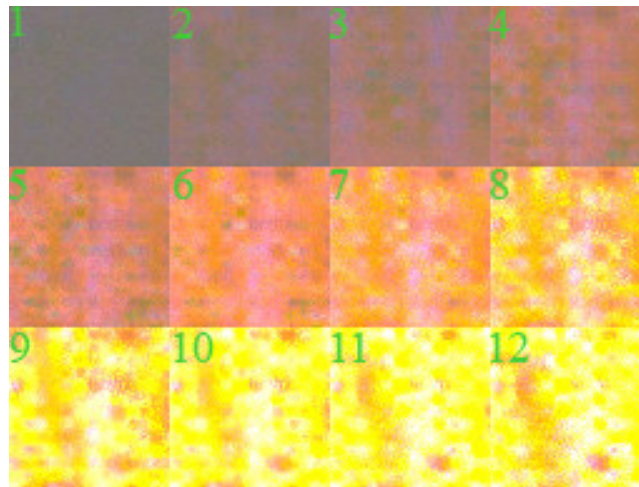


Abbildung 96: Einschwingen eines dynamischen Systems (gesehen horizontal oben links nach unten rechts)

An den Bilderfolgen in Abbildung 96 kann gut erkannt werden, wie sich die verschiedenen Bewegungen der Gräser im zeitlichen Verlauf entwickeln. Zu Beginn ist das System in einem Ruhezustand und keine Bewegung der Gräser erfolgt. Danach erfolgt von Bild 1 bis Bild 8 eine Einschwingung des Systems, bei dem die Winde erstmals auf die Vegetation wirken und diese auslenken. Diese Startphase ist, abhängig von den verwendeten Winden und den vegetativen Strukturen, unterschiedlich lang. In den nachfolgenden Bildern befindet sich das System bereits im eingeschwungenen Zustand, welche erst zur Animation herangezogen werden sollten.

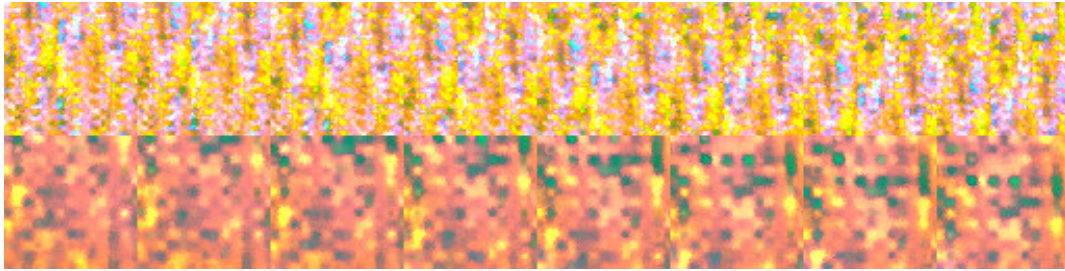


Abbildung 97: Vergleich eines stark turbulenten Windes (oben) mit einem weniger turbulentem Wind (unten) und deren Auswirkungen auf eine vegetative Struktur (gesehen von links nach rechts)

Durch die Speicherung der Daten in einem Bildformat können auch Winde und durch die von ihnen bewegten Gräser miteinander verglichen werden. Wie anhand Abbildung 97 zu sehen ist, handelt es sich bei den Winden um einen sehr sanften und gleichmäßigen Wind und einen sehr turbulenten, wilden Wind. Die Unterschiede sind deutlich zu sehen und auch die Art des Windes kann aus den Bildern herausgelesen werden. So können verschiedene Winde und deren Auswirkungen auf dieselbe Vegetation einfach untersucht und verglichen werden. Einfache Winde können so als Standardwinde zum Vergleich mit komplexeren Winden verwendet werden.

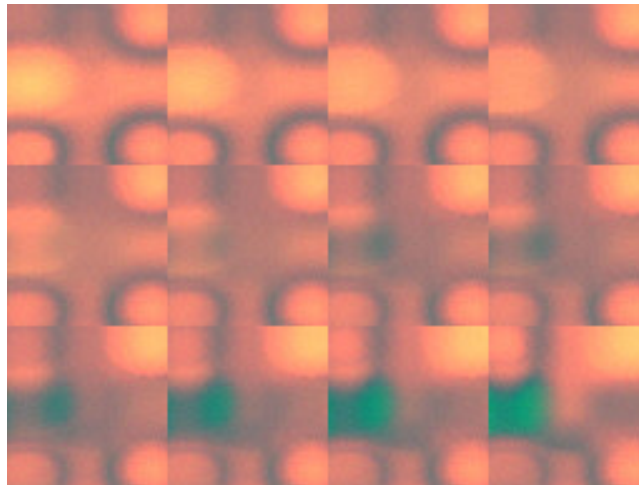


Abbildung 98: Bewegung von vegetativen Strukturen durch einen böigen Wind (gesehen horizontal oben links nach unten rechts)

Durch die visuelle Speicherung der Daten können auch Winde in ihrem Verhalten sehr gut beobachtet werden. In Abbildung 98 kann man die Änderung der Positionen der Strukturen einer Vegetation aufgrund eines böigen Windes erkennen. Der Wind selbst wirkt in einem begrenzten Bereich, wo die Böe auf die Vegetation trifft, und auch nur für kurze Zeit, jedoch sind die Auslenkungen und deren Schwingungen weitaus länger.

Die wirkenden Kräfte und deren verzögerte Auslenkungen stehen im direkten Zusammenhang mit den Eigenschaften der Vegetation.

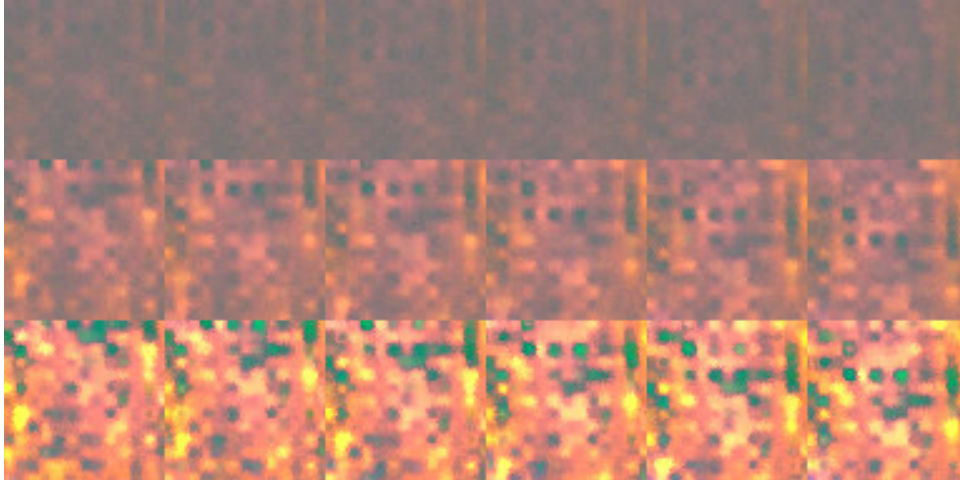


Abbildung 99: Auswirkungen eines leicht turbulenten Windes auf die verschiedenen Ebenen eines Partikelsystems (Entfernung von der Wurzel von oben nach unten)

Da vegetative Strukturen unterschiedlich komplex sein können, erfolgt bei der Speicherung der Daten eine Aufteilung in die jeweiligen Partikel der Struktur. Für jedes dieser Partikel werden Auslenkungsdaten und deren Abstand zum Wurzelknoten gespeichert. In Abbildung 99 kann man drei Partikel einer Struktur sehen, die in unterschiedlichen Abständen zum Wurzelpartikel stehen. Partikel näher bei der Wurzel sind weitaus weniger von einer Auslenkung betroffen als weiter entfernte. Das entspricht einer Imitation der Natur, da sich bei Wind Gräser und andere Strukturen zuerst an der Spitze biegen und erst bei stärker werdendem Wind auch die unteren Bereiche betroffen sind.

Durch die Speicherung der Daten in Bildformaten können auch nachträglich noch leicht Änderungen vorgenommen werden. Direktes Eingreifen in die Farbbereiche ist dadurch möglich, falls einfache Nachbesserungen notwendig sein sollten. Man kann dadurch eine bestimmte Bewegung erzeugen, eine ungewollte Bewegung reduzieren oder ganz entfernen. Ein Beispiel dafür wäre eine Videosequenz, in der animiertes Gras verwendet wird. In diesem Gras steht aber in einem Abschnitt des Videos ein Objekt, so kann das Gras in diesem Bereich platt auf den Boden gelegt werden.

Die Winkelsteifheit ist aufgrund von Vereinfachung richtungsunabhängig. Die Erstellung der L-Systeme würde sonst zu aufwendig werden und eine entsprechende Einstellungsmöglichkeit trägt kaum zur Verbesserung des Realismus bei, da solche Eingaben wiederum auf den Benutzer des Systems zurückzuführen wären. Um das System so einfach wie nötig und so realistisch wie möglich zu halten, wurde daher auf

diesen Bereich verzichtet. Er stellt durchaus eine interessante Erweiterung des Systems dar.

Um einen Einblick in die Möglichkeiten der Resultate zu erhalten, werden die Anforderungen für eine überzeugende Animation in einer Szene behandelt. Solch eine Szene könnte zum Beispiel in einem Computerspiel auftreten.

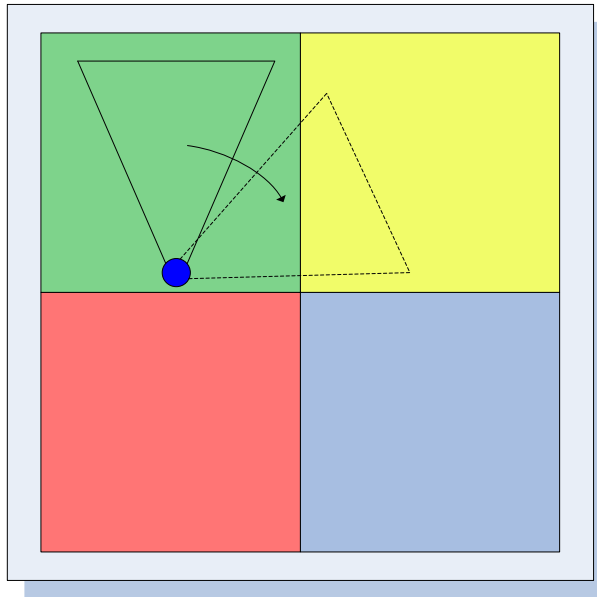


Abbildung 100: Anordnung der Texturen, Betrachter (blauer Kreis)

Dafür werden 6-teilige Billboards benötigt, um ein qualitativ hochwertiges Ergebnis zu erzielen. Die Texturen sind dafür am besten 128x128 Pixel groß. Da nur der Ausschnitt aus der Sicht des Betrachters relevant ist, muss auch nur dieser Ausschnitt animiert werden. Das Sichtfeld sollte dabei nicht größer sein als eine verwendete Textur. Ändert sich nun das Sichtfeld (siehe Abbildung 100), können die Texturen gespiegelt aneinandergereiht oder weitere Texturen geladen werden. Im Falle von mehreren Texturen sollten diese dynamisch nachgeladen werden. Durch dieses Verfahren müssen immer nur maximal 4-fach so viele Texturen im Speicher sein wie bei der gespiegelten Version. Damit beläuft sich der Speicheraufwand auf 3 MB bei der einfachen Variante und 12 MB beim dynamischen Nachladen.

Der Geschwindigkeitsverlust durch die Animation ist dabei gering (siehe Abbildung 89). Durch Verringerung der Billboardanzahl im Sichtfeld des Betrachters kann dieser Verlust weiter minimiert werden. Die Übertragung der Animation sollte allerdings immer über einen Vertexshader passieren, da ein Shader das derzeit effizientest mögliche Vorgehen darstellt.

Die Darstellung der Animation beschränkt sich aufgrund der berechneten Daten auf richtungsgebundene Winde. Es können daher nur Winde mit einer festgelegten Richtung und eventuellen Turbulenzen dargestellt werden. Komplexere, strömungsbasierte Winde (Wirbelwinde, etc.) können mit dem Verfahren nicht simuliert werden. Sollten solche Winde gebraucht werden, müsste die Definition der Kräfte durch ein komplexeres, strömungsbasiertes Verfahren ersetzt werden.

Während der Errechnung der Daten werden keinerlei Kollisionsereignisse zwischen Gräsern beachtet. Dies beruht auf der Tatsache, dass die Errechnung der Daten unabhängig vom gesamten Ausmaß der tatsächlich zu animierenden Geometrie ist, da lediglich das Animationsskelett dieser Geometrie zur Errechnung der Daten verwendet wird. Aufgrund dessen kann eine Kollisionsberechnung nur während der tatsächlichen Verwendung realistisch miteinbezogen werden.

12 Zusammenfassung

Das Konzept stellt grundlegend eine Möglichkeit vor, Vegetation (Gräser, Sträucher, Bäume) durch Wind oder andere windähnlichen Kräfte erzeugte Bewegungen zu simulieren. Diese Bewegungen versuchen die natürlichen Bewegungen möglichst exakt zu kopieren. Die Simulation selbst ist dabei von der tatsächlichen Anwendung der erhaltenen Daten entkoppelt. Die Simulation speichert die errechneten Daten in eine 3D-Texture, welche dann in dem tatsächlichen Programm zur Anwendung verwendet wird. Durch diese Vorberechnung ist nur mehr ein Bruchteil der Zeit zur Animation nötig als zur Berechnung.

Der Aufbau ist modular gehalten, um eventuelle Änderungen in unterschiedlichen Bereichen zu erleichtern und neue Aspekte besser integrierbar zu machen. Das Konzept unterteilt sich in die Definition von vegetativen Strukturen, der Definition von Kräften und dem dynamischen System. Das dynamische System nimmt jene Definitionen auf und verarbeitet sie. Zusätzlich werden alle Definitionen einzeln gespeichert und können durch ihre Abgeschlossenheit gegenüber dem System immer wieder zu einem neuen System kombiniert werden.

Die Definition der Strukturen der Vegetation erfolgt über einen grammatikalischen Weg, im Speziellen parametrisierte 3D0-L-Systeme. Diese L-Systeme sind für die Definition und auch für Festlegung von Parametern gut geeignet. Durch das System werden sowohl die Form der vegetativen Struktur, deren Masse, Steifheit und Winkelsteifheit festgelegt. Durch die Parameter wird das Verhalten der Struktur in der späteren Simulation und deren Reaktion auf die verschiedenen wirkenden Kräfte gesteuert. Um die Strukturen der Vegetation in einem dynamischen System verwenden zu können, werden sie vor der eigentlichen Simulation in ein Partikelsystem übergeführt. Das Partikelsystem enthält die Eigenschaften der Struktur.

Die Definition der Winde wird mit Hilfe der Festlegung von Parametern für eine „Perlin Noise“-basierende Kraft vollzogen. Die Kraft trägt dabei eine Hauptrichtung, die in beide orthogonale Richtungen Turbulenzen aufweisen kann. Für den „Perlin Noise“ wird kein reiner „White Noise“ zur Festlegung verwendet, sondern normalverteilte Zufallszahlen. Die Zufallszahlen können individuell angepasst werden und sind für die Hauptrichtung und die jeweiligen orthogonalen Richtungen getrennt festgelegt. Der Wind wird dem dynamischen System ohne zusätzliche Verarbeitungsschritte direkt übergeben.

Die Definitionen aus beiden Bereichen werden in einem dynamischen System vereinigt. In diesem System werden die Auswirkungen der Kräfte auf die Vegetation beobachtet und aufgezeichnet. Diese Auswirkungen verursachen bei den vegetativen Strukturen unterschiedlich starke Schwingungen, die eine natürliche Bewegung simulieren sollen. Das Interesse gilt dabei allen Abweichungen einer Struktur zu ihrer Ausgangsposition. Diese Abweichungen werden für die Visualisierung in einem 3D-Bildformat einzeln für

jedes Partikel in der Struktur gespeichert. Das Bildformat verwendet dabei die drei Farbkanäle als ein 3D-Koordinatensystem.

Bei der Visualisierung werden die gespeicherten Daten verwendet, um die errechneten Positionsänderungen einer vegetativen Struktur wiederzugeben. Durch die Vorberechnung braucht diese Animation nur mehr sehr wenig Zeit und beinhaltet immer noch die korrekte Schwingung der Vegetation. Da die Daten für jedes Partikel getrennt vorliegen, können diverse Techniken (LOD, Billboards, etc) immer noch Verwendung finden.

Das Konzept bietet damit eine ausgezeichnete Möglichkeit Vegetation zu animieren, da sowohl die Effizienz als auch der Realismus erreicht werden. Dabei ist das System flexibel im Bezug auf dessen Anwendung und die Verbindung mit anderen Techniken der Computergrafik.

13 Future Work

Der Großteil der Folgearbeiten liegt im Bereich der Datenakquirierung. Es stellt sich als nahezu unmöglich dar, notwendige Daten über Vegetation von Bäumen oder Gräsern zu erhalten. Diese Daten wären jedoch für die Simulation ihrer Bewegung und deren Validierung notwendig. Selbst in Zusammenarbeit mit der Universität für Bodenkultur in Wien war es nicht möglich, derartige Daten zu finden. Die Daten waren schlichtweg nicht vorhanden, auch bei gut erforschten und oft verwendeten Pflanzen. Lediglich die Form der jeweiligen Pflanzen ist leicht zu erhalten. Deren Eigenschaften, wie Masse der einzelnen Teile oder gar deren Flexibilität, waren nirgends verzeichnet.

Dieser Mangel an echten Daten spielt aber auch in der Wahl der Parameter für die vegetativen Strukturen eine große Rolle. Die Parameter könnten damit durch einige zusätzliche Parameter ergänzt werden, um noch genauer auf die jeweiligen Strukturen der Vegetation und deren Verhalten eingehen zu können. Bei diesen Parametern könnte es sich um völlig neue Aspekte oder auch um eine Aufteilung eines bestehenden Parameters handeln. Im Vordergrund sollte dabei immer der Gewinn an Realismus stehen.

Ein weiterer Punkt wäre die Erweiterung der möglichen Winde auf lokale Winde, wodurch jedoch eine Strömungsberechnung für diese Art der Winde notwendig wäre. Derartige Berechnungen sind weitaus umfangreicher und zeitintensiver als die derzeit verwendete Berechnung. Durch Strömungsberechnungen könnten jedoch Winde, wie zum Beispiel Windhosen, Fallwinde, etc., in die Simulation miteinbezogen werden. Das würde die Möglichkeiten für die Erstellung von Vegetationsbewegungen noch zusätzlich vergrößern.

Die Speicherung der Daten ist ebenfalls ein Bereich, dem Aufmerksamkeit geschenkt werden kann. Die Speicherung erfolgt derzeit in einem Bildformat, das aufgrund seiner Beschaffenheit nur begrenzt viele Information aufnehmen kann und darüber hinaus abgeschnitten werden muss. Eine Datenstruktur mit mehr als 255 Bewegungseinheiten pro Achse (8Bit-RGB) wäre dabei von Vorteil. Zu denken wäre dabei an ein zukünftiges Bildformat, das im Bereich der Farben eine höhere Auflösung besitzt. Aber auch jedes andere Format könnte verwendet werden, solange es durch Shader verwendet werden kann.

Weiters sind die erstellten Datenstrukturen sehr groß und werden ohne weitere Verarbeitung verwendet. Durch eine entsprechende Kompression könnten die Datenstrukturen in ihrer Größe verringert werden, wodurch weniger Platz im Speicher aufgewendet werden müsste.

Zu guter Letzt wäre wohl der Wunsch nach einer Simulation all dieser Dinge in Echtzeit wünschenswert. Optimalere Algorithmen bei der Berechnung des dynamischen Systems könnte dies vielleicht bewerkstelligen, selbst wenn es sich um eine große Anzahl an

Partikel im System handelte. Derzeit ist nur der Weg über die Simulation von Systemen mit geringer Anzahl von Partikeln möglich, die vorab durchgeführt werden. Durch eine Echtzeitberechnung wäre es einfacher, die Simulation in ihrer Entwicklung zu beobachten und direkt interaktive Änderungen vorzunehmen. Dadurch wäre es einfacher, zu dem gewünschten Ergebnis zu gelangen.

14 Danksagung

Zu allererst geht mein Dank an Dr. Brigitte Klug, die mich bei meiner Suche nach Informationen über Gras zu Beginn meiner Arbeit unterstützt hat, obwohl es aussichtslos schien und sich später auch als aussichtslos herausstellte. Danke für die Mühe!

Mein zweiter Dank ergeht an Christian Forthuber (IT-Journalist), der mir maßgeblich bei der Verfassung meiner Diplomarbeit behilflich war. Die Beantwortung eines „Wie“ war dabei sehr viel hilfreicher als eines „Warum“.

Meiner Familie und Freunden ergeht der dritte Dank, was wäre ich ohne sie.

15 Literaturverzeichnis

[A1] Stochastic Motion - Motion under the Influence of Wind

Mikio Shinya, Alain Fournier

Eurographics Association, Blackwell Publishers 1992

Computer Graphics Forum

Volume 11, Issue 3 (1992)

pp. 119-128

[A2] Randomly Accessible Procedural Animation of Physically Approximate Turbulent Motion

Hui Fang John C. Hart

IEEE Computer Society , 2002

ISBN 0-7695-1594-0

[G1] Quantitative inheritance of some wheat plant traits

D. Novoselovic; Marijana Baric; G. Drezner; J. Gunjaca; A. Lalic

Genetics and Molecular Biology Year 2004

Volume 27, Issue 1

pp. 92-98

[G2] Prairie Grass

Marvin H. Hall, Jerry Jung

Agronomy Facts 39

Issued in furtherance of Cooperative Extension Work

Acts of Congress May 8 and June 30, 1914, in cooperation with the U.S.

Department of Agriculture and the Pennsylvania Legislature. L. F.Hood, Director of

Cooperative Extension, The Pennsylvania State University

[G3] Animated Grass with Pixel and Vertex Shaders

John Isidoro, Drew Card

Direct3D Shaderx: Vertex and Pixel Shader Tips and Tricks

Wordware Publishing Inc., 2002

ISBN 1-556-22041-3

[G4] Measurement of wind-induced motion of crop canopies from digital video images

Charlotte Py, Emmanuel de Langre, Bruno Moulia b, Pascal He´mon

Agricultural and Forest Meteorology 2005

Volume 130

pp. 223–236

- [G5] Modeling plant leaves in marble-patterned colours with particle transportation system
Yodthong Rodkaew, Prabhas Chongstitvatana, Suchada Siripant, Chidchanok Lursinsap
4th International Workshop on Functional-Structural Plant Models, 2004
Volume 7-11
pp. 391-397
- [G6] A Procedural Approach to Animate Interactive Natural Sceneries
Sylvain Guerraz, Frank Perbet, David Raulo, Francois Faure, Marie-Paule Cani
Computer Animation and Social Agents
Institute of Electrical Engineers, 2003
ISBN 0-769-51934-2
- [G7] Animating Prairies in Real-Time
Frank Perbet, Marie-Paule Cani
ACM-SIGGRAPH Symposium on Interactive 3D Graphics, 2001
pp. 103 - 110
ISBN 1-58113-292-1
- [G8] Rendering Realistic Trees and Forests in Real Time
Alberto Candussi, Nicola Candussi, and Tobias Höllerer
Proc. Eurographics, Short Papers 2005
Dublin, Ireland
pp. 73–76
- [G9] Real Time Animated Grass
Brook Bakay
Proceedings of Eurographics 2002
Volume 3
pp. 391–396
- [G10] Realistic modeling and rendering of plant ecosystems
Oliver Deussen, Pat Hanrahan, Bernd Lintermann, Radomir Mech, Matt Pharr,
Przemyslaw Prusinkiewicz
International Conference on Computer Graphics and Interactive Techniques, 1998
ACM Press
pp. 275-286
ISBN 0-89791-999-8
- [G11] Real-Time Rendering of Fur
Gary Sheppard
University of Sheffield, 2004
Undergraduate Dissertations

[G12] Making Grass and Fur Move
Sven Banisch, Charles A. Wuthrich
Journal of WSCG 2006
Volume 14

[G13] Animating and Lighting Grass in Real-Time
Brook M. Bakay
University of British Columbia, 2003
Masters Thesis

[G14] Real-time Rendering of Seasonal Influenced Trees
Michael Braitmaier, Joachim Diepstraten, Thomas Ertl
Visualization and Interactive Systems Group
Theory and Practice of Computer Graphics, 2004
pp. 152-159
ISBN 0-7695-2137-1

[G15] An interactive Forest
Thomas Di Giacomo, Stéphane Capo, Francois Faure
Eurographics Workshop on Computer Animation and Simulation, 2001
pp. 65-74
ISBN 3-211-83711-6

[G16] Interactive Vegetation Rendering
Stephan Mantler
Technische Universität Wien, 2007
Inauguraldissertation

[L1] The Artificial Life of Plants
Przemyslaw Prusinkiewicz, Mark Hammel, Radomir Mech
Artificial life for graphics, animation, and virtual reality SIGGRAPH, 1995
Volume 7
pp. 1-1 - 1-38

[L2] The algorithmic beauty of plants
Premyslaw Prusinkiewicz, Aristid Lindenmayer
Springer Verlag, New York 1990
ISBN 0-387-97297-8

[L3] L-Systeme und andere künstliche Pflanzen.
Florian Breier
Universität Ulm, Insitut für Medien Informatik 2002
pp. 26

[L4] Wachstumssimulationen
Peter Jossen, Daniel Eyer
Interdisziplinärer Essay, 2001

[L5] Animation of Plant Development
Przemyslaw Prusinkiewicz, Mark Hammel, Eric Mjolsness
Computer Graphics Proceedings, Annual Conference Series ACM SIGGRAPH, 1993
pp. 351-360

[L6] Computer-generierte Pflanzen
Oliver Deussen
Springer Verlag, Berlin Heidelberg 2003
ISBN 3-540-43606-5

[L7] L-Systems: From the Theory to visual Models of plants
Przemyslaw Prusinkiewicz, Mark Hammel, Jim Hanan, Radomir Mech
Proceedings of the 2nd CSIRO Symposium on Computational Challenges in Life
Sciences, CSIRO Publishing, 1996
pp. 1-27

[L8] Pflanzen in Interaktion mit ihrer Umwelt
Ilja Radusch, Raimo Ihle, Jens Unger, Thomas Schüppel
Projekt Scientific Visualisation, 1999
Theoriepapier

[L9] Lindenmayer Systems, Fractals and Plants
P. Prusinkiewicz, J. Hanan
Springer Verlag, 1989
ISBN 0-387-97092-4

[L10] Automata, Languages, Development
A. Lindenmayer, G. Rozenberg
North Holland Publishing Co., 1976
ISBN 0-720-40474-6

[L11] Applications of L-Systems to Computer Imagery
Przemyslaw Prusinkiewicz
Springer Verlag, 1986
ISBN 3-540-18771-5

[L12] Handbook of Formal Languages Vol.1
G. Rozenberg, A. Salomaa
Springer Verlag 1997
ISBN 3-540-60420-0

[L13] Handbook of Formal Languages Vol.2
G. Rozenberg, A. Salomaa
Springer Verlag 1997
ISBN 3-540-60648-3

[L14] Handbook of Formal Languages Vol.3
G. Rozenberg, A. Salomaa
Springer Verlag 1997
ISBN 3-540-60649-1

[L15] Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie
John E. Hopcroft, Rajcev Motwani, Jeffrey D. Ullman
Pearson Studium, 2002
ISBN 3-8273-7020-5

[L16] An Introduction to Formal Languages and Automata
Peter Linz
Jones and Bartlett Publishers Inc., 2001
ISBN 0-7637-1422-4

[P1] CRC Concise Encyclopedia of Mathematics
Eric W. Weisstein
CRC Press LLC, 1999
ISBN 0-8493-9640-9

[P2] Texturing and Modeling, A Procedural Approach
David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, Steven Worley
Academie Press Inc., 1994
ISBN 0-12-228760-6

[P3] Mathematik für Naturwissenschaftler
Dr. Hugo Sirk, Dr. Max Draeger
Dr. Dietrich Steinkopf Verlag, 1972
ISBN 3-7985-0338-9

[P4] Introduction to Mathematics for Life Scientists
Edward Batschelet
Springer Verlag, 1979
ISBN 3-540-09662-0

[P5] Improving Noise

Ken Perlin

Proceedings of ACM SIGGRAPH 2002

Volume 21, Issue 3

pp. 681-682

ISBN 1-58113-521-1

[P6] Perlin Noise and Turbulence

Paul Bourke

Northeastern University, Boston, 2000

Computer Graphics Paper

[P7] Probability, Random Variables and Stochastic Processes

Athansios Papoulis, S. Unnikrishna Pillai

McGraw Hill, 2002

ISBN 0-07-112256-7

[P8] Perlin noise pixel shaders

John C. Hart

SIGGRAPH/EUROGRAPHICS Conference On Graphics Hardware archive

Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware table of contents, 2001

pp. 87-94

ISBN 1-58113-407-X

[P9] Texturing & Modeling, A Procedural Approach, Third Edition

David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, Steven Worley

Elsevier Science, USA 2003

ISBN 1-55860-848-6

[P10] Noise in Real-time 3D Graphics

Erik Eriksson

Royal Institute of Technology, Stockholm, 2004

Master Thesis

[M1] Cloth Modeling and Animation

Donald H. House, David E. Breen

A.K. Peters Ltd., 2000

ISBN 1-56881-090-3

[M2] Dynamische Systeme

D.K. Arrowsmith, C.M. Place

Spektrum Akademischer Verlag GmbH, 1994

ISBN 3-86025-308-5

[M3] Modeling Complex Systems
Nino Boccara
Springer Verlag, 2004
ISBN 0-387-40462-7

[M4] Mass-Spring Simulation using Adaptive Non-Active Points
P. Howlett
Computer Graphics Forum, 1998
Volume 17, Issue 3
pp. 345--354

[M5] Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior
Xavier Provot
Proceedings of Graphic Interface 1995
pp. 147-154

[M6] Modern Theory of Dynamical Systems
Anatole Katok, Boris Hasselblatt
Cambridge University Press, 1995
ISBN 0-521-34187-6

[M7] Dynamische Systeme und Fraktale
Karl Heinz Becker, Michael Dörfler
Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, 1989
ISBN 3-528-24461-5

[M8] Differential Equations: A Dynamical System Approach
J.H. Hubbard, B.H. West
Springer Verlag, 1991
ISBN 0-387-97286-2

[M9] Chaos in Discrete Dynamical Systems
Ralph H. Abraham, Laura Gardini, Christian Mira
Springer Verlag, 1997
ISBN 0-387-94300-5

[M10] The Statistical Theory of Linear Systems
E.J. Hannan, M. Deistler
John Wiley & Sons Inc., 1988
ISBN 0-471-80777-X

- [M11] A Practical Model for Hair Mutual Interactions
Johnny Chang, Jingyi Jin, Yizhou Yu
Symposium of Computer Animation, ACM SIGGRAPH, 2002
Volume 12
pp. 73-80
- [M12] Nonlinear Dynamics and Chaos
Steven H. Strogatz
Addison-Wesley Publishing Company 1994
ISBN 0-201-54344-3
- [M13] Dynamics: Numerical Exploration
Helena E. Nusse, James A. Yorke
Springer Verlag New York Berlin Heidelberg 1994, 1998
ISBN 0-387-98264-7
- [M14] Enhancing the Visualization of Characteristic Structures in Dynamical Systems
H. Löffelmann, E. Gröller
EUROGRAPHICS Workshop on Visualization in Scientific Computing, 1998
pp. 35-46

16 Abbildungsverzeichnis

Abbildung 1: Vielfalt der Natur.....	4
Abbildung 2: Bewegung von Gräsern.....	5
Abbildung 3: Animation von Vegetation.....	7
Abbildung 4: Präprozess der Animation von Vegetation	9
Abbildung 5: Laufzeitsystem der Animation von Vegetation.....	10
Abbildung 6: Aufbau des Präprozesses	11
Abbildung 7: Speicherung der Daten des dynamischen Systems.....	12
Abbildung 8: Aufbau des Laufzeitsystems	12
Abbildung 9: Algenwachstumssimulation mit L-System	26
Abbildung 10: Ableitung eines L-System.....	28
Abbildung 11: Beispiel für die Bewegungen einer Turtle	30
Abbildung 12: Stochastisches L-System.....	31
Abbildung 13: Visualisierung von Rauschen.....	35
Abbildung 14: Amplitudenverteilung von Pink Noise	37
Abbildung 15: Amplitudenverteilung von White Noise	38
Abbildung 16: Amplitudenverteilung von Brown Noise	39
Abbildung 17: Aufbau von Perlin Noise.....	40
Abbildung 18: Kurve einer einfachen linearen Interpolation.....	41
Abbildung 19: Kurve einer Kosinusinterpolation	42
Abbildung 20: Kurve einer kubischen Interpolation	42
Abbildung 21: Vergleich „Gradient Noise“ und „Value Noise“	44
Abbildung 22: Vergleich „Sparse Convolution Noise“, „Lattice Convolution Noise“ ..	45
Abbildung 23: Dynamisches System, Pendel.....	47
Abbildung 24: Wetteraufnahme	48
Abbildung 25: 2D-Phasenraums eines Pendels.....	49
Abbildung 26: Bifurkationsdiagramm, Feigenbaumdiagramm.....	51
Abbildung 27: Bewegung von Gräsern.....	55
Abbildung 28: Beispiel eines richtig gedämpften Systems.....	56
Abbildung 29: Grundlegendes Feder-Dämpfer-Masse-System	57
Abbildung 30: Annäherung einer Kurve mittels Euler Step-Verfahren	58
Abbildung 31: Grafische Darstellung des „Runge-Kutta“ Verfahrens.....	59
Abbildung 32: Animation von Bäumen	63
Abbildung 33: Aufbau des Präprozesses	64
Abbildung 34: Speicherung der Daten des dynamischen Systems.....	64
Abbildung 35: Iterationsstufen eines L-Systems.....	66
Abbildung 36: Unterschiedliche Eigenschaften eines L-Systems	68
Abbildung 37: Lokale Parametrisierung des L-Systems.....	69
Abbildung 38: Parameterverwendung in L-System und Partikelsystem	70
Abbildung 39: Beispiel eines L-Systems	71
Abbildung 40: Beispiel eines L-Systems (2).....	71
Abbildung 41: Beispiel eines L-Systems (3).....	72
Abbildung 42: Beispiel eines L-Systems (4).....	73

Abbildung 43: Parametervererbung.....	73
Abbildung 44: Überführung des L-System in Partikelsystem.....	74
Abbildung 45: Trägheit der Masse	75
Abbildung 46: Steifheit der Verbindung.....	75
Abbildung 47: Winkelsteifheit der Verbindung	76
Abbildung 48: Auswirkungen der Schwerkraft.....	78
Abbildung 49: 3D „Perlin Noise“ Gitter	79
Abbildung 50: 4D-Gitter, „Perlin Noise“	80
Abbildung 51: Platzierung des Kräftegitters	82
Abbildung 52: Speicherung der Daten des dynamischen Systems.....	83
Abbildung 53: Verteilung der Vegetation und Kräfte	83
Abbildung 54: Visualisierung zu Adaptionszwecken.....	84
Abbildung 55: Umsetzung des L-Systems in Partikelsystem,	85
Abbildung 56: Feder-Dämpfer-Verbindung als Vektor.....	86
Abbildung 57: Kräftewirkung auf Partikel.....	87
Abbildung 58: Speicherung der Bewegungsdaten.....	87
Abbildung 59: Simulationsfläche mit Kräftegitter und Vegetation.....	88
Abbildung 60: Kraftberechnung	88
Abbildung 61: Implizite Positionsdaten der Textur.....	90
Abbildung 62: Speicherung der Daten pro Partikel.....	90
Abbildung 63: Aufbau des Laufzeitsystems	91
Abbildung 64: Anpassung der Animation an die Szene	92
Abbildung 65: Skalierung der Bewegung	92
Abbildung 66: Zeitanpassung der Bewegung	93
Abbildung 67: Unterteilung der Billboards.....	93
Abbildung 68: Implementierung des Präprozesses.....	95
Abbildung 69: Interface, „New Plant“ und „Edit Plants“	96
Abbildung 70: Interface, Erstellen einer neuen vegetativen Struktur.....	96
Abbildung 71: Interface, Editieren bereits erstellter vegetativen Struktur	98
Abbildung 72: Komplexeres L-System mit zugehörigem Partikelsystem.....	99
Abbildung 73: Interface, „New Wind“ und „Edit Winds“.....	100
Abbildung 74: Interface, Icons „Show Winds“ und „Show Forces“.....	100
Abbildung 75: Interface, Visualisierung „Show Winds“	101
Abbildung 76: Interface, Visualisierung „Show Forces“	101
Abbildung 77: Interface, „New“, „Open“ und „Save“	103
Abbildung 78: Interface, Erstellen eines neuen Dynamischen Systems.....	104
Abbildung 79: Interface, Hauptfenster.....	105
Abbildung 80: Interface, Hauptfenster, Parametereinstellung Pflanzen.....	105
Abbildung 81: Interface, Hauptfenster, Parametereinstellung Winde	106
Abbildung 82: Interface, Einstellung der Krafthauptrichtungen	107
Abbildung 83: Durchführung der Simulation	108
Abbildung 84: Implementierung des Laufzeitsystems	109
Abbildung 85: Interface, Parametereinstellung,	110
Abbildung 86: Billboards, Verteilung.....	111
Abbildung 87: Billboards, einfach, 2-teilig, 6-teilig.....	111

Abbildung 88: Echtzeitvisualisierung der Daten.....	114
Abbildung 89: Performance der Echtzeitvisualisierung	115
Abbildung 90: Zugehörige Tabelle der Performance	115
Abbildung 91: Performancevergleich, ohne Animation,	116
Abbildung 92: Zugehörige Tabelle der Performancevergleich	116
Abbildung 93: Speicherbedarf der Echtzeitvisualisierung.....	117
Abbildung 94: Zugehörige Tabelle des Speicherbedarfs.....	117
Abbildung 95: Vergleich Unterteilung Billboards,	118
Abbildung 96: Einschwingen eines dynamischen Systems	119
Abbildung 97: Vergleich von Winden	120
Abbildung 98: Bewegung von vegetativen Strukturen.....	120
Abbildung 99: Auswirkungen auf die verschiedenen Ebenen	121
Abbildung 100: Anordnung der Texturen.....	122