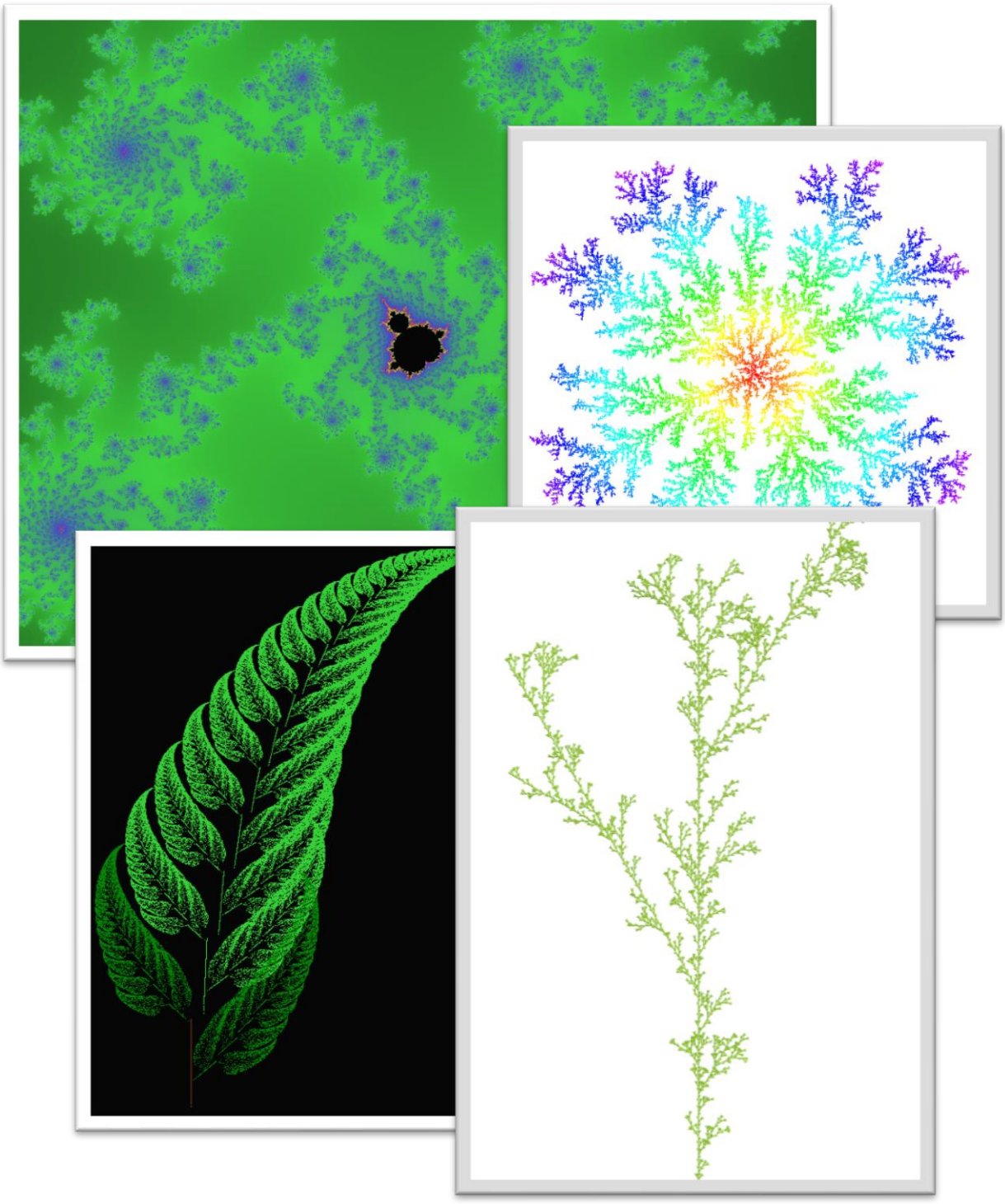
TU Wien, Institute of Visual Computing and Human-Centered Technology

Fraktale UE

Beschreibung und Aufgaben

Dokument Version: 2018-03-28
Dokument Autoren: Dr. Christoph Traxler



Inhaltsverzeichnis

1. Einführung

In der Übung zur Vorlesung Fraktale können die Konzepte der fraktalen Geometrie in die Praxis umgesetzt werden. Die vorgegebenen Standardaufgaben decken die folgenden Bereiche der fraktalen Geometrie ab:

- Iterierte Funktionen Systeme (IFS)
- Julia-Mengen und Mandelbrotmenge
- Stochastische Fraktale
- L-Systeme
- Dynamische Systeme (Chaostheorie)

Die Standardaufgaben haben unterschiedliche Schwierigkeitsgrade und daher unterscheidet sich auch die Anzahl der maximalen Punkte, die durch die Umsetzung erzielt werden können. Was z.B. die Computergrafik betrifft gibt es sowohl 2D als auch 3D Programmierbeispiele. Manche der Aufgaben verlangen nach einem komplexeren GUI, andere wiederum nach Interaktion im 3D Raum. Damit sollte eine Aufgabenstellung zu finden sein, die für die Fähigkeiten der unterschiedlichen Studienzweige angemessen ist.

Sie wählen also eine oder mehrere der Standardaufgaben aus, um die notwendige Punkteanzahl der gewünschten Note zu erreichen (siehe Kapitel 3 für das Notenschema). Die folgende Matrix zeigt, wie die Beispiele kombiniert werden können, um 100 Punkte zu erzielen, falls Sie ein Sehr Gut anstreben (Beispiele mit 100 Punkten referenzieren auf sich selbst). Bitte beachten Sie, dass mit Kombinationen, die mehr als 100 Punkte ermöglichen trotzdem nur ein Sehr Gut erzielt werden kann. Besonders gelungene Aufgaben werden das Jahr darauf in der VO demonstriert.

Aufgabe/ Punkte	1.1 40	1.2 40	1.3 100	1.4 100	2.1 60	2.2 60	2.3 60	3.1 40	3.2 60	4.1 40	4.2 100	5.1 60	5.2 40	5.3 40	5.4 60
1.1/40					•	•	•		•			•			•
1.2/40					•	•	•		•			•			•
1.3/100			•												
1.4/100				•											
2.1/60								•		•			•	•	
2.2/60								•		•			•	•	
2.3/60								•		•			•	•	
3.1/40									•			•			•
3.2/60										•			•	•	
4.1/40												•			•
4.2/100											•				
5.1/60													•	•	
5.2/40															•
5.3/40															•
5.4/60															

Verwirklichung einer eigenen Idee

Nach Absprache mit dem Betreuer kann auch eine eigene Idee aus dem Bereich der Fraktale oder einem Randgebiet implementiert werden. Auch Teamwork ist bei entsprechendem Umfang der eigenen Aufgabenstellung möglich und willkommen. Bei den vorgegebenen Standardaufgaben wird Teamwork aber mit Punkteabzug geahndet!

Bei dieser Variante bitte vor der Implementierung unbedingt die Idee durch den Betreuer genehmigen lassen! Es kann nicht garantiert werden, dass eine bereits umgesetzte Idee für die Fraktale UE zulässig ist.

2. Ablauf

Die Übung findet am eigenen PC statt und die Abgabe erfolgt elektronisch per Email (Attachment oder Link zum Download). Wenn kein eigener PC vorhanden ist, dann muss selbständig ein Arbeitsplatz am Institut organisiert werden (bitte die Techniker fragen).

Alle gängigen Programmiersprachen werden akzeptiert. Die Programme sind entweder kompatibel mit Windows (x86 oder x64) oder Mac OS X oder sie sind plattformunabhängig. Auch Web-Applikationen (HTML5, Javascript) sind möglich. Für plattformabhängige Linux-Lösungen muss eine persönliche Abgabe auf dem eigenen Laptop vereinbart werden.

Das abgegebene ZIP-File muss folgende Dateien enthalten:

- Das lauffähige Programm
- Den kommentierten Source-Code
- Daten-Files zum Testen (falls erforderlich)
- Eine kurze Dokumentation

Die Doku bitte als einfaches Textfile, PDF oder im HTML-Format abgeben. Sie sollte den folgenden Inhalt haben:

- Name, Matrikelnummer, Kennzahl, Aufgabennummer(n)
- Kurze Beschreibung der verwendeten Algorithmen
- Benutzeranleitung

Die Abgabe ist während des gesamten Studienjahres möglich (auch in den Ferialzeiten). Die Beurteilung erfolgt aber nur zweimal pro Semester. Weitere Infos gibt es auch am Web. Und hier noch die Daten des Betreuers:

Dr. Christoph Traxler

VRVis Zentrum für Virtual Reality und Visualisierung Forschungs-GmbH

Donaucity-Straße 1 / OG3

Tel.: 20501 30202

Email: traxler@cg.tuwien.ac.at

Sprechstunde: nach der VO oder nach Vereinbarung.

3. Beurteilung

Die abgegebenen Programmierbeispiele werden immer am Anfang und Ende jedes Semesters beurteilt. Dabei werden die Punkte der abgegebenen Beispiele addiert und nach folgendem Schema benotet:

Gesamte Punkteanzahl	**Note**
95-100	Sehr Gut (S1)
85-94	Gut (G2)
70-84	Befriedigend (B3)
50-69	Genügend (U4)
< 50	Nicht Genügend (N5)

Bei einer genehmigten Spezialaufgabe wird überprüft, ob die vereinbarten Ziele erreicht wurden. Allgemein werden bei der Beurteilung folgende Kriterien berücksichtigt:

- Funktionalität (korrekte Implementierung des Algorithmus)
- Performance (Optimierungen sind bei Fraktalen erforderlich)
- Kommentare im Source-Code (Verständlichkeit)
- Umfang an Testdaten (wo erforderlich, z.B. L-Systeme)
- Benutzerfreundlichkeit (des GUI und/oder der direkten Interaktionen)

Sollte ein abgegebenes Beispiel nicht lauffähig sein (z.B. fehlende DLL), dann wirkt sich das **nicht** auf die Beurteilung aus. Es erfolgt eine Benachrichtigung per Email mit der Fehlermeldung. Das Beispiel kann dann korrigiert und nochmals abgebeben werden. Im Notfall kann auch eine persönliche Abgabe vereinbart werden.

4. Standardaufgaben

4.1 Iterierte Funktionen Systeme (IFS)

Sämtliche Visualisierungsmethoden für IFS aus R^2 und R^3 (Stochastische Methode, Adaptive Cut, Ray Tracing) sind in den folgenden VO-Unterlagen beschrieben:

www.cg.tuwien.ac.at/courses/Fraktale/PDF/fractals3.pdf

Der Sierpinski-Tetraeder ist in den folgenden VO-Unterlagen beschrieben:

www.cg.tuwien.ac.at/courses/Fraktale/PDF/fractals1.pdf

Aufgabe 1.1 Stochastische Methode **Punkte: 40**

Visualisieren Sie Attraktoren von IFS der Form $\{R^2; f_1, \ldots, f_n\}$ mit der Stochastischen Methode, wobei $f_i: R^2 \rightarrow R^2$ eine affine, kontrahierende Transformation ist: $f_i(x) = A_i x + b_i$ mit

$$A_i = \begin{pmatrix} a_{i1,1} & a_{i1,2} \\ a_{i2,1} & a_{i2,2} \end{pmatrix}, \qquad b_i = \begin{pmatrix} b_{i1} \\ b_{i2} \end{pmatrix}$$

Es soll auch die Optimierung der Stochastischen Methode implementiert werden, bei der jeder Funktion f_i eine Wahrscheinlichkeit p_i zugeordnet wird. Die Wahrscheinlichkeiten hängen dabei vom Abdeckungsbereich der Funktionen ab, der durch die Determinante der Transformationsmatrix bestimmt wird:

$$p_i = \frac{|Det\ A_i|}{\sum_{k=1}^{n} |Det\ A_k|}$$

Die Aufgabe soll außerdem die folgenden Anforderungen erfüllen:

- Über ein GUI sind die einzelnen Funktionen f_i des IFS aus einer Liste auswählbar, sodass die Elemente der Transformationsmatrix A_i und des Translationsvektors b_i der selektierten Funktion mittels Textfelder editiert werden können.
- Das IFS kann als Text-File oder XML-File abgespeichert und wieder eingelesen werden. Stellen Sie zum Testen zumindest 5 verschiedene IFS-Definitionen zur Verfügung.
- Die Stochastische Methode wird mittels eines Buttons gestartet und gestoppt. Die ersten 100-200 Punkte sollen nicht gezeichnet werden, da sie den Weg vom Startpunkt in den Attraktor zeigen. Der Startpunkt kann zufällig gewählt werden, liegt aber idealerweise nicht zu weit außerhalb des Attraktors.

Aufgabe 1.2 Adaptive Cut — Punkte: 40

Visualisieren Sie Attraktoren von IFS der Form $\{R^2; f_1, \ldots, f_n\}$ mittels Adaptive Cut, wobei f_i: $R^2 \rightarrow R^2$ eine affine, kontrahierende Transformation ist: $f_i(x) = A_i x + b_i$ mit

$$A_i = \begin{pmatrix} a_{i1,1} & a_{i1,2} \\ a_{i2,1} & a_{i2,2} \end{pmatrix}, \qquad b_i = \begin{pmatrix} b_{i1} \\ b_{i2} \end{pmatrix}$$

Hinweis: Beim ersten Schritt des Adaptive Cut wird einfach der Hutchinson-Operator auf das Ausgansobjekt angewendet. In allen folgenden Iterationen wird die Bounding Box jedes Objektes, das in der vorherigen Iteration durch den Hutchinson-Operator entstanden ist betrachtet. Ist die Bounding Box kleiner als ein Pixel, dann wird dieses Objekt in der nächsten Iteration nicht mehr durch den Hutchinson Operator abgebildet. Es wird jedoch in allen weiteren Iterationen gezeichnet. Alle anderen Objekte werden nochmals durch den Hutchinson Operator abgebildet, um die nächste Iterationsstufe zu erhalten.

Die Aufgabe soll außerdem die folgenden Anforderungen erfüllen:

- Über ein GUI sind die einzelnen Funktionen f_i des IFS aus einer Liste auswählbar, sodass die Elemente der Transformationsmatrix A_i und des Translationsvektors b_i der selektierten Funktion mittels Textfelder editiert werden können.
- Das IFS kann als Text-File oder XML-File abgespeichert und wieder eingelesen werden. Stellen Sie zum Testen zumindest 5 verschiedene IFS-Definitionen zur Verfügung.
- Es soll mittels zweier Pfeiltasten möglich sein die Iterationstiefe der Adaptive Cut Methode zu inkrementieren, bzw. zu dekrementieren. Dabei muss das Resultat jeder Iterationsstufe neu gezeichnet werden.
- Mittels GUI soll es möglich sein eines der drei folgenden Startobjekte zu wählen: Dreieck, Quadrat, Kreis.

Aufgabe 1.3 Ray-Tracing von IFS — Punkte: 100

Visualisieren Sie Attraktoren von IFS der Form $\{R^3; f_1, \ldots, f_n\}$ mittels der Ray-Tracing Methode von Hart et al. [1] , wobei f_i: $R^3 \rightarrow R^3$ eine affine, kontrahierende Transformation ist, repräsentiert als homogene Transformationsmatrix:

$$A_i = \begin{pmatrix} a_{i1,1} & a_{i1,2} & a_{i1,3} & a_{i1,4} \\ a_{i2,1} & a_{i2,2} & a_{i2,3} & a_{i2,4} \\ a_{i3,1} & a_{i3,2} & a_{i3,3} & a_{i3,4} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Hinweis: Verwenden Sie immer Kugeln als Bounding Volume. Es reicht daher aus, nur die Schnittberechnung eines Strahls mit einer Kugel zu implementieren (quadratische Gleichung). Sekundärstrahlen für Reflexionen und Brechungen müssen nicht berücksichtigt werden. Geben Sie die maximale Iterationstiefe vor anstatt zu überprüfen, ob die Projektion jeder Umgebungskugel schon kleiner als ein Pixel ist. Dies ermöglicht die Visualisierung unterschiedlicher Approximationsstufen.

Die Aufgabe soll außerdem die folgenden Anforderungen erfüllen:

- Das Phong-Modell soll für die Schattierung verwendet werden [2].
- Es wird ein Szenen-File geladen, das neben der Definition des IFS auch das Material für die Approximation des Attraktors und die Position einer Punktlichtquelle enthält. Stellen Sie zum Testen zumindest 5 Szenen-Files mit verschiedenen IFS zur Verfügung.
- Ein Schattenfühler vom Schnittpunkt zu der vorgegebenen Punktlichtquelle soll gelegt werden, um Schlagschatten darzustellen. Die Schatten sollen nicht komplett schwarz werden, sondern nur die Schattierung an dem Schnittpunkt abdunkeln.
- Die von Hart et al. [1] vorgeschlagene Methode zum Anti-Aliasing soll auch implementiert werden. Dabei werden die Normalvektoren an den Schnittpunkten der letzten 5 Iterationen gemittelt.

Aufgabe 1.4 Animation Faltung des Sierpinski Tetraeders Punkte: 100

Visualisieren Sie die Faltung des Sierpinski Tetraeders aus einem Dreieck in der Ebene als Animation im 3D Raum. Der Sierpinki Tetraeder hat eine Selbstähnlichkeitsdimension von zwei. Daraus folgt, dass er in die Ebene (R^2) eingebettet werden kann. Dort bildet er dann eine flächenfüllende Verzweigungsstruktur in Form eines gleichseitigen Dreiecks. Somit lässt sich der Sierpinski-Tetraeder auch aus diesem Dreieck rekursiv in den Raum auffalten. Bild 1 zeigt dieses Faltungsschema.

Die Aufgabe soll außerdem die folgenden Anforderungen erfüllen:

- Der Tetraeder soll während der Animation mittels Maus um die z- und x-Achse rotiert werden können.
- Die Animation soll in beide Richtungen abspielbar sein.
- Über ein GUI kann die Animation gesteuert werden. Es soll Buttons geben für Vorlauf, Rücklauf, Pause und Stopp, wobei Stopp zum Anfang zurückspringt, wo das Dreieck in der Ebene zu sehen ist.
- Achten Sie darauf, dass die Animation nicht zu schnell abläuft. Im Idealfall wird die Geschwindigkeit über ein GUI geregelt.
- Es wäre schön, wenn die Tetraeder-Seiten unterschiedliche Farben haben ähnlich wie in Bild 1 dargestellt. Die Farbwahl bleibt Ihnen überlassen.

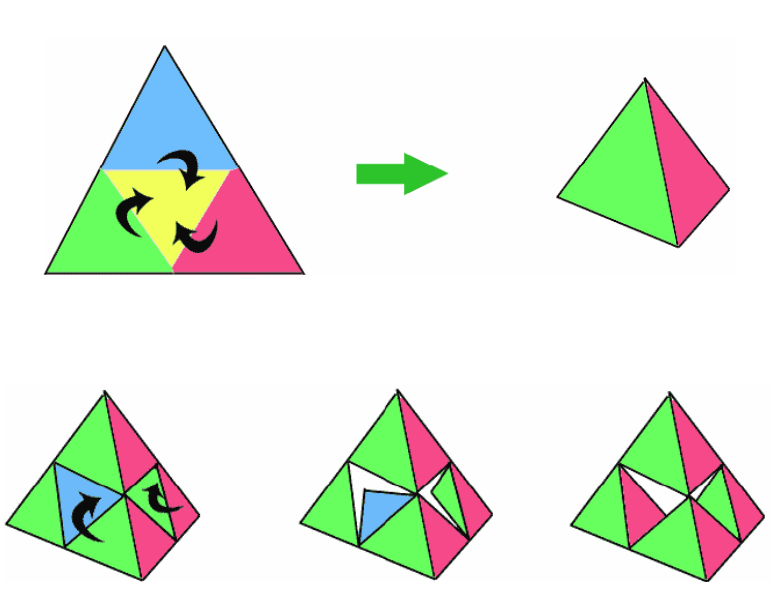


Bild 1 : Auffaltung des Sierpinski-Tetraeders aus einem Dreieck in der Ebene.

4.2 Julia-Mengen und Mandelbrot-Menge

Julia-Mengen, die Mandelbrotmenge und das Pixel Game sind in den folgenden VO-Unterlagen beschrieben:

www.cg.tuwien.ac.at/courses/Fraktale/PDF/fractals5.pdf

www.cg.tuwien.ac.at/courses/Fraktale/PDF/fractals6.pdf

Aufgabe 2.1 Julia-Mengen via Mandelbrotmenge **Punkte: 60**

Die Mandelbrotmenge soll als Landkarte für Julia-Mengen genutzt werden. In einer Darstellung der Mandelbrotmenge kann ein Punkt c der Gaußschen Zahlenebene mittels Maus gewählt werden. Die zugehörige Julia-Menge J_c wird daraufhin unmittelbar in einem eigenen Bereich dargestellt. Durch Bewegung der Maus mit gedrückter Taste im Bereich der Mandelbrotmenge wird kontinuierlich der entsprechende Punkt c berechnet und die Darstellung der Julia-Menge in Echtzeit aktualisiert.

Hinweis: Eine Echtzeit-Visualisierung kann durch den Einsatz eines Fragment-Shaders erzielt werden. Dabei wird eine Szene mit einem Quad generiert und eine orthographische Projektion verwendet, sodass das Quad den Darstellungsbereich komplett ausfüllt. Der Fragment-Shader berechnet dann die Farbe jedes Pixels mittels des Pixel-Games.

Die Aufgabe soll außerdem die folgenden Anforderungen erfüllen:

- Achten Sie beim Pixel Game darauf, dass alle Arten von Julia-Mengen immer gut sichtbar sind. Dies wird durch entsprechende Farbverläufe erzielt. Sie können sowohl kontinuierliche Farbverläufe implementieren als auch diskrete.
- Implementieren Sie einen Zoom-Schritt in die Mandelbrotmenge um den Faktor 10, um den Punkt c genauer auswählen zu können.

Aufgabe 2.2 Mandelbrot Animation Punkte: 60

Implementieren Sie eine Applikation zur Generierung von Zoom-Sequenzen in die Mandelbrotmenge. Dabei werden verschiedene Ansichten der Mandelbrotmenge als Keyframes gesetzt und dann mittels Interpolation die Zwischenbilder berechnet. Jede Ansicht ist dabei durch ein Rechteck auf der Gaußschen Zahlenebene definiert, dessen Seitenverhältnis dem Darstellungsbereich entspricht. Die Bilder werden mit fortlaufender Nummerierung als PNG-Files abgespeichert und können dann von einem Videoschnittprogramm zu einem Film verarbeitet werden. Die Mandelbrotmenge wird dabei mittels Pixel Game visualisiert. Es ist möglich die Farbverläufe über ein GUI zu editieren.

Hinweis: Ändern sie die Berechnungsgenauigkeit (N_{max} im Pixel Game) in Abhängigkeit vom Zoom-Faktor. Es ist hilfreich, wenn dieser über das GUI gesetzt werden kann und als Parameter der Keyframes mitabgespeichert wird. Verwenden Sie immer Double-Precision oder sogar Quadrupel-Precision, falls Sie sehr tief in die Menge zoomen wollen. Im Falle einer GPU-Implementierung muss die höhere Genauigkeit emuliert werden.

Die Aufgabe soll außerdem die folgenden Anforderungen erfüllen:

- Um Keyframes zu bestimmen kann mittels Maus in die Mandelbrotmenge gezoomt werden und diese auch verschoben werden (Zoom & Pan).
- Keyframes erscheinen im GUI in einer Liste. Mittels eines Buttons kann die aktuelle Ansicht als Keyframe hinzugefügt werden. Keyframes können auch wieder gelöscht werden.
- Mittels einer Spin-Box oder eines numerischen Textfelds kann für den in der Liste selektierten Keyframe die Anzahl der Zwischenbilder bestimmt werden, die bis zum nächsten Keyframe berechnet werden sollen (für den letzten Keyframe ist das immer null)
- Wird in der Liste ein Keyframe doppelgeklickt dann wird diese Ansicht der Mandelbrotmenge dargestellt.

Aufgabe 2.3 Mandelbrot Terrains Punkte: 60

Visualisieren Sie die Mandelbrotmenge als Höhenfeld. Die Farben des Pixel Games werden dabei in Höhenwerte umgesetzt. Als Input soll ein rechteckiger Bereich der Mandelbrotmenge angegeben werden können, der durch zwei Koordinaten (linker oberer und rechter unterer Eckpunkt) der Gaußschen Zahlenebene definiert wird. Es ist erforderlich besonders nahe am Rand der Mandelbrotmenge einen Tiefpassfilter (z.B. Gausian Blur [3]) anzuwenden, da sonst wegen der hohen Frequenzen unschöne Artefakte in der Geometrie auftreten können.

Hinweis: Für den Tiefpassfilter können Sie einen vordefinierten Filter-Kernel verwenden, also ein Raster geringer Auflösung wobei jedem Rasterelement ein Gewichtungsfaktor zugeordnet ist, der der Gaußschen Glockenkurve in x- und y-Richtung entspricht:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$, wobei σ die Standardabweichung ist.

Der Filter-Kernel wird dann in der Nachbearbeitung über jeden Wert des Höhenfeldes gelegt und die entsprechenden Höhenwerte gewichtet aufaddiert. Am Rand wird für Werte, die außerhalb des Höhenfeldes liegen null angenommen. Achtung: Übertragen Sie die gefilterten Werte in ein neues Höhenfeld. Als Input für den Filter dürfen nur die ursprünglichen, ungefilterten Werte verwendet werden.

Die Aufgabe soll außerdem die folgenden Anforderungen erfüllen:

- Es soll möglich sein, über das Terrain mittels Fly-Through zu navigieren (Standard Game Navigation).
- Texturieren Sie das Höhenfeld indem Sie die Farbe verwenden, die sich aus dem Pixel-Game ergibt. Das Farbschema bleibt dabei Ihnen überlassen.
- Höhenpunkte, die innerhalb der Mandelbrotmenge liegen, werden auf eine konstante Höhe gesetzt, z.B. null und mit einer fixen Farbe schattiert.
- Mit dem Ausschnitt der Mandelbrotmenge soll auch die Richtung einer direktionalen Lichtquelle als Input mitangegeben werden können. Dies kann über ein Datenfile oder über ein GUI geschehen.

4.3 Stochastische Fraktale

Die Diffusion Limited Aggregation (DLA) und fraktale Terrains werden in den folgenden VO-Unterlagen beschrieben:

www.cg.tuwien.ac.at/courses/Fraktale/PDF/fractals7.pdf

Aufgabe 3.1 Diffusion Limited Aggregation (DLA) Punkte: 40

Implementieren Sie einen einfachen DLA-Simulator. Wie in der VO vorgestellt, basiert dieser auf einem Pixelraster und Zufallswegen. Bestimmte Pixel sind dabei als Zielregion (Kathode) definiert. Ausgehend von einem zufällig gewählten Punkt einer Startregion wird ein Zufallsweg berechnet. Sobald dabei ein Pixel erreicht wird, das an die Zielregion grenzt, wird dieses Pixel selbst zu einem Teil der Zielregion. Dieses Verfahren wird iteriert bis eine Kristallstruktur entstanden ist.

Es ist erforderlich das Verfahren zu optimieren. Dazu wird mit einer kleinen Startregion begonnen, die dann sukzessive vergrößert wird, bis diese an die Grenzen des Darstellungsbereichs stößt.

Hinweis: Für den Zufallsweg reicht es, einfach immer ein Pixel weiter zu gehen. Dabei wird eines der Nachbarpixel zufällig ausgewählt. Sie können mit einer 4er- oder 8er-Nachbarschaft arbeiten. Die Geschwindigkeit der Vergrößerung der Startregion muss empirisch bestimmt werden. Sie hängt von der Art der Start- und Zielregion ab, sowie von der Auflösung des Pixelrasters.

Die Aufgabe soll außerdem die folgenden Anforderungen erfüllen:

- Es soll zumindest die folgenden Kombinationen aus Ziel- und Startregion geben, die über ein GUI auswählbar sind:
    - Punkt mit wachsendem Kreis
    - Zwei horizontale Linien, wobei eine die Startregion definiert und sich nach oben bewegt
- Färben Sie die Pixel in Abhängigkeit ihres Anlagerungszeitpunktes ein. Das Farbschema bleibt dabei Ihnen überlassen.
- Über ein GUI sollen die Wahrscheinlichkeiten der Richtungen des Zufallsweges gesetzt werden können. Diese bringt mehr Asymmetrie in die Simulation. Der Summe der Wahrscheinlichkeitswerte sollte dabei eins ergeben.

Aufgabe 3.2 Fraktale Terrains Punkte: 60

Implementieren Sie eine Applikation zur Generierung fraktaler Terrains. Dabei soll Midpoint Displacement und das Unterteilungsverfahren „Diamond & Squares“ verwendet werden. Die Unterteilung des Höhenfeldes erfolgt dabei in 2 Schritten, wie in Bild 2 dargestellt. Zuerst werden die Mittelpunkte jedes Rasterelements erzeugt und danach die Kanten unterteilt.

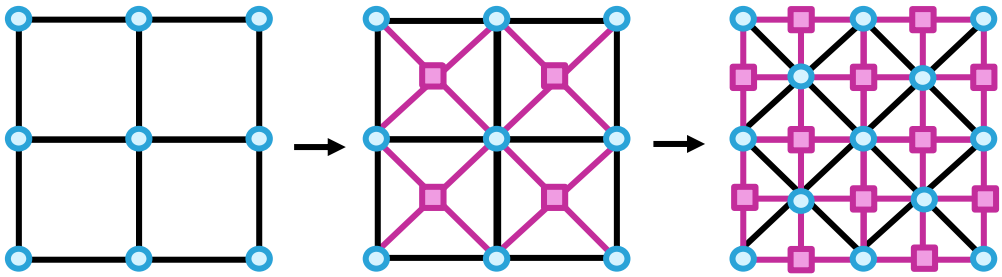


Bild 2 : Die zweistufige Unterteilung des Diamond & Square Algorithmus (neue Punkte in Magenta).

Neue Punkte der Unterteilungstiefe n (Magenta in Bild 2) werden dabei aus den entsprechenden alten Punkten gemittelt und dann um einen Zufallswert D_n verschoben. Die Varianz der Zufallszahlen nimmt dabei nach jeder vollständigen Unterteilung (also nach den 2 Schritten) ab. Der Faktor dieser Abnahme wird durch den Hurst-Exponenten H bestimmt und regelt die Rauigkeit des Terrains und damit dessen fraktale Dimension.

$D_n \cong N\left(0, \frac{\sigma^2}{2^{nH}}\right)$, $H \in [0,1]$, wobei N die Normalverteilung ist und σ die Ausgangsvarianz.

Hinweis: Sie können normalverteilte Zufallszahlen durch die Bates-Verteilung [4] approximieren, die auf gleichverteilten Zufallszahlen basiert (n=10 ist ausreichend):

$$X = \frac{1}{n}\sum_{k=1}^{n} U_k$$, wobei $U_k \in [0,1]$ gleichverteilte Zufallszahlen sind.

Die Aufgabe soll außerdem die folgenden Anforderungen erfüllen:

- Das Terrain soll mittels Echtzeitvisualisierung dargestellt werden. Setzen Sie dabei eine geeignete direktionale Lichtquelle, die für ausreichend Kontrast sorgt.
- Über ein GUI kann der Hurst-Exponent gesetzt werden und die Höhe verändert werden (Schieberegler). Die Darstellung des Terrains wird möglichst in Echtzeit aktualisiert.
- Es soll möglich sein das Terrain um die z-Achse zu drehen und über die x-Achse zu kippen (Trackball-Modus) sowie hinein zu zoomen.
- Es soll möglich sein, über das Terrain mittels Fly-Through zu navigieren (Standard Game Navigation).
- Legen Sie eine transparente Ebene in die Szenen, die so groß ist wie das Höhenfeld und dessen z-Position durch das GUI verändert werden kann. Damit lassen sich Seen simulieren.

4.4 L-Systeme

Die verschiedenen Klassen von L-Systemen, sowie Kanten- und Knotenersetzung und die Turtle-Graphics werden in den folgenden VO-Unterlagen beschrieben:

www.cg.tuwien.ac.at/courses/Fraktale/PDF/fractals8.pdf

Aufgabe 4.1 Deterministische L-Systeme **Punkte: 40**

Implementieren Sie eine Applikation zur graphischen Interpretation von deterministischen L-Systemen (D0L-Systeme) mittels Turtle Graphics. Ein D0L-System ist durch das geordnete Triplet <A, ω, P> definiert, wobei A ein Alphabet aus Symbolen ist, $\omega \in A^+$ ein nicht leerer Start-String aus Symbolen von A und $P \subset A \times A^*$ eine Menge von Produktionen. Eine Produktion wird in der Form $a \rightarrow v$ angegeben und bedeutet, dass alle Vorkommen des Symbols a im aktuellen String durch den String v zu ersetzen sind. Bei L-Systemen werden in jedem Ableitungsschritt immer alle Symbole des aktuellen Strings ersetzt. Falls es für ein Symbol a keine Produktion gibt, dann gilt implizit die identische Ersetzung $a \rightarrow a$.

Die Visualisierung der abgeleiteten Strings soll mittels Turtle-Graphics erfolgen. Der String wird dabei von links nach rechts gelesen und wenn ein Symbol einem Turtle-Kommando entspricht, wird dieses ausgeführt. Alle anderen Symbole werden ignoriert. Die Turtle ändert dabei ihren Zustand (x, y, α), definiert durch ihre Position in der Ebene und ihre Orientierung. Die folgenden Turtle-Kommandos sollen implementiert werden, wobei die Parameter d und δ zuvor definiert wurden.

Turtle Kommandos			
F	Zeichne eine Linie der Länge d, der Zustand ändert sich zu $(x+d\cdot\cos\alpha, y+d\cdot\cos\alpha, \alpha)$.	-	Drehung im Uhrzeigersinn um den Winkel δ, der Zustand ändert sich zu $(x, y, \alpha-\delta)$.
f	Bewegung um die Länge d ohne zu zeichnen, der Zustand ändert sich zu $(x+d\cdot\cos\alpha, y+d\cdot\cos\alpha, \alpha)$.	[	Der aktuelle Zustand der Turtle wird auf einen Stack gelegt.
+	Drehung im Gegenuhrzeigersinn um den Winkel δ, der Zustand ändert sich zu $(x, y, \alpha+\delta)$.	]	Der letzte Zustand wird vom Stack entfernt und die Turtle in diesen Zustand versetzt.

Das Prinzip der Knotenersetzung soll unterstützt werden. Dabei werden zusätzliche Symbole zur Steuerung der Ableitung verwendet, die keinen Turtle-Kommandos entsprechen.

Hinweis: Durch L-Systeme generierte Objekte wachsen. Um sie komplett im Darstellungsbereich zu sehen, müssen sie entsprechend skaliert werden. Deshalb ist es zu empfehlen, während der Ableitung eine Bounding-Box zu berechnen aus deren Größe sich dann der Skalierungsfaktor ergibt.

Die Aufgabe soll außerdem die folgenden Anforderungen erfüllen:

- Die L-Systeme sollen von einem File eingelesen werden. Stellen Sie zum Testen zumindest 5 verschiedene L-Systeme zur Verfügung. Darunter sollen zumindest 2

pflanzenähnliche Verzweigungsstrukturen sein und zumindest eine flächenfüllende Kurve, die mittels Knotenersetzung definiert wird.

- Über das GUI soll die Anzahl der Ableitungen kontrolliert werden können. Änderungen wirken sich unmittelbar auf die Visualisierung aus. Achten Sie dabei auf Speicherlimits.
- Die Parameter d und δ für die Turtle-Kommandos sollen über ein GUI einstellbar sein. Änderungen wirken sich unmittelbar auf die Visualisierung aus.

Aufgabe 4.2 Parametrische L-Systeme in 3D — Punkte: 100

Implementieren Sie eine Echtzeit-Visualisierung von Objekten, die prozedural mittels parametrischer L-Systeme (pL-Systeme) generiert werden. Nach dem Einlesen des pL-Systems erfolgt die Ableitung, danach die Interpretation des resultierenden Strings mittels 3D-Turtle, was zu einem 3D Objekt führt, das in einem Viewer beliebig betrachtet werden kann.

PL-Systeme sind in den VO-Unterlagen beschrieben und im Buch von Prusinkiewicz und Lindenmayer [5]. Eine der wichtigsten Aspekte ist die Verbindung von Symbolen mit reellen Werten, die dann bei den Produktionen für die Auswertung einer Kondition und für Berechnungen verwendet werden können. Dafür ist es notwendig einen kleinen Parser zu implementieren. Für Konditionen sollen die logischen Operatoren (=, ≠, <, >, ≤, ≥, ∩, ∪, ¬) verwendbar sein und für Berechnungen die Grundrechenarten.

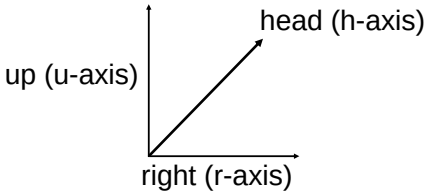


Bild 3 : Lokales Koordinatensystem der 3D-Turtle.

Der Zustand der 3D-Turtle ist durch ein lokales Koordinatensystem gegeben, dass aus den Achsen h (head), r (right) und u (up) besteht (siehe Bild 3). Eine Transformationsmatrix definiert dessen Lage relativ zum Weltkoordinatensystem. Turtle-Kommandos werden die reellen Zahlen der Module als Parameter übergeben, woraus sich eine neue Transformation ergibt, die das lokale Koordinatensystem ändert. Die 3D-Turtle soll zumindest die folgenden Kommandos ausführen können.

Turtle Kommandos			
F(d, r)	Positioniere einen Zylinder der Länge d mit Radius r entlang der h-Achse. Translation entlang der h-Achse um die Länge d (an den Endpunkt des Zylinders).	/(d)	Rotation um die h-Achse um den Winkel δ (Roll).
f(d)	Translation entlang der H-Achse um die Länge d ohne Konstruktion.	|	Drehung um 180° um die u-Achse (Shortcut für +(180))
+(δ)	Rotation um die u-Achse um den Winkel δ (Yaw).	[	Der aktuelle Zustand der Turtle wird auf einen Stack gelegt
&(δ)	Rotation um die r-Achse um den Winkel δ (Pitch).	]	Der letzte Zustand wird vom Stack entfernt und die Turtle in diesen Zustand versetzt

Hinweis: Schönere Ergebnisse werden erzielt, wenn die Translation der Turtle bei F(d, r) etwas kürzer ist als die Zylinderlänge, da sonst zwischen den Zylindern ein Spalt entsteht. Wer einiges an Erfahrung mit dem Bau von Meshes hat, verbindet die Meshes zweier aufeinanderfolgender Zylinder durch separate Polygone (Seams).

Die Aufgabe soll außerdem die folgenden Anforderungen erfüllen:

- Die pL-Systeme sollen von einem File eingelesen werden. Stellen Sie zum Testen zumindest 3 verschiedene pL-Systeme zur Verfügung. Achten Sie darauf, dass auch wirklich Berechnungen stattfinden und Konditionen ausgewertet werden.
- Das File mit den pL-Systemen soll auch die Definition von zwei Punktlichtquellen enthalten zur Beleuchtung des 3D Objekts.
- Es soll möglich sein das generierte Objekt um die z-Achse zu drehen und über die x-Achse zu kippen (Trackball-Modus) sowie hinein zu zoomen.

4.5 Dynamische Systeme (Chaostheorie)

Die Dimensionsberechnung mittels Box Counting wird in den folgenden VO-Unterlagen beschrieben:

www.cg.tuwien.ac.at/courses/Fraktale/PDF/fractals3.pdf

Dynamische Systeme und Methoden zur Analyse sowie das Differential Box Counting werden in den folgenden VO-Unterlagen beschrieben:

www.cg.tuwien.ac.at/courses/Fraktale/PDF/fractals9.pdf

Aufgabe 5.1 Box Counting & Differential Box Counting Punkte: 60

Implementieren Sie eine Applikation zur Berechnung der fraktalen Dimension von Strukturen in der Ebene mittels Box Counting und Differential Box Counting. Die Applikation soll nur Binärbilder analysieren können. Weiße Pixel gelten dabei als Hintergrund, schwarze als Teil der Struktur. Es wird ein Raster mit vorgegebener Anfangsauflösung über das Bild gelegt, der dann sukzessive verfeinert wird. Beim klassischen Box Counting werden bei jeder Iteration die nichtleeren Rasterelemente gezählt. Beim Differential Box Counting wird berücksichtigt wie viele nichtleere Pixel in jedem Rasterelement liegen. Das führt zu einer gewichteten Aufsummierung, wobei die Gewichte normalisiert werden müssen. Die Anzahl der nichtleeren Pixel in jedem Iterationsschritt wird dem Reziprokwert der Rasterauflösung in einem log/log-Diagramm gegenübergestellt. Dann wird eine Regressionsgerade durch die Punkte gelegt, z.B. mittels der Methode der kleinsten Quadrate [6]. Der Anstieg der Geraden ist dann die Dimension.

Hinweis: Testen Sie die Applikation mit strikt selbstähnlichen Fraktalen (z.B. Sierpinski-Dreieck oder Koch-Kurve), um den resultierenden Wert mit der Selbstähnlichkeitsdimension zu vergleichen. Es macht auch Sinn die Methode an euklidischen Formen zu testen, um zu überprüfen, wie groß die Abweichung zum Wert 2 ist.

Die Aufgabe soll außerdem die folgenden Anforderungen erfüllen:

- Es sollen Binärbilder in Form von GIF- oder PNG-Files geladen werden können. Geben Sie zumindest 5 Bilder zum Testen ab.
- Die Anzahl der Verfeinerungsstufen soll von der Auflösung des geladenen Bildes abhängen. Es sollen zumindest 5 Stufen sein.
- Das Verfahren soll visualisiert werden. Die Struktur erscheint in einem Darstellungsbereich und es kann in Einzelschritten mitverfolgt werden, wie die Rasterauflösung verfeinert wird. Nichtleere Rasterelemente werden dann farblich hervorgehoben.
- Das log/log-Diagramm ist in einem anderen Darstellungsbereich zu sehen. Dort wird am Ende auch die Regressionsgerade eingezeichnet.
- Über ein GUI kann zwischen Box Counting und Differential Box Counting gewählt werden. Auch der berechnete Wert der Dimension wird dort in einem Textfeld ausgegeben.

Aufgabe 5.2 Bifurkationsdiagramme Punkte: 40

Visualisieren Sie die Dynamik einer Funktion $f_a(x)$ mittels Bifurkationsdiagramm. Das dynamische System $x_{n+1} = f_a(x_n)$, $n \to \infty$, pendelt sich in Abhängigkeit des Parameters a entweder auf einen Fixpunkt einer bestimmten Periodenlänge ein oder verhält sich chaotisch (es werden immer neue Punkte erzeugt). Bei einem Bifurkationsdiagramm wird auf der horizontalen Achse der Parameter a und auf der vertikalen Achse der Grenzwert $x_\infty = \lim_{n\to\infty} f_a(x_n)$ eingetragen, der durch die endliche Folge $\{x_{min}, x_{min+1}, \ldots, x_{max-1}, x_{max}\}$ approximiert wird. Die Indizes (min, max) der Grenzen der Folge sind dabei empirisch zu bestimmen.

Hinweis: Für jeden diskreten Wert des Parameters a auf der horizontalen Achse wird also die Zeitserie des dynamischen Systems berechnet mit einem beliebigen Startwert x_0. Die ersten min Werte werden dabei nicht in das Diagramm eingetragen, da sie erst den Weg zum Fixpunkt x_∞ beschreiben. Die obere Schranke max hängt von der Auflösung ab, da besonders bei chaotischen Verhalten ausreichend viele Punkte erzeugt werden sollen. Bei einem Zoom in das Diagramm sollte max inkrementiert werden, um die Punktedichte zu erhöhen.

Die Aufgabe soll außerdem die folgenden Anforderungen erfüllen:

- Es sollen zumindest Bifurkationsdiagramme für die folgenden Funktionen und entsprechenden Wertebereiche erstellt werden können:
  - $f_a(x) = ax\cdot(1-x)$, $x \in [0, 1]$, $a \in [0, 4]$
  - $f_a(x) = a\cdot\sin(\pi\cdot x)$, $x \in [0, 1]$, $a \in [0, 1]$
  - $f_a(x) = x(1+a\cdot(1-x))$, $x \in [0, (1+a)/a]$, $a \in [0, 1]$
  - $f_a(x) = x\cdot e^{a\cdot(1-x)}$, $x \in [0, \infty]$, $a \in [0, \infty]$
- Die Funktionen sollen über ein GUI auswählbar sein.
- Die Grenzindizes (min, max) der Folge zur Approximation von x_∞ sollen über das GUI justierbar sein.
- Es soll möglich sein in das Diagramm zu zoomen bis zu einem Faktor von 100. Idealerweise ist das mittels Mausrad möglich, wobei die Position des Mauszeigers das Zentrum des Zooms bestimmt.

Aufgabe 5.3 Graphische Iteration Punkte: 40

Visualisieren Sie die Dynamik einer Funktion $f_a(x)$ mittels Graphischer Iteration. Das dynamische System $x_{n+1} = f_a(x_n)$, $n \to \infty$, pendelt sich in Abhängigkeit des Parameters a entweder auf einen Fixpunkt einer bestimmten Periodenlänge ein oder verhält sich chaotisch (es werden immer neue Wege erzeugt). Bei der Visualisierung wird der Graph der Funktion, das Koordinatensystem und der erste Median (x=y-Gerade) dargestellt. Auf der x-Achse wir ein beliebiger Startpunt x_0 oder ein Startintervall der Länge d definiert $[x_0, x_0+d]$. Danach kann die graphische Iteration gestartet werden, die fortlaufend neue Zyklen des dynamischen Systems darstellt.

Hinweis: Beim Start wird nun zunächst eine vertikale Linie vom Startpunkt (oder den Intervallsgrenzen) zum Graphen der Funktion gelegt. Dies entspricht der Abbildung $f_a(x_0)$, weil der Wert der Funktion durch die y-Koordinate dieses Schnittpunktes gegeben ist. Danach wird eine horizontale Linie zum Median gelegt, was einem Zyklus des dynamischen Systems entspricht (der Funktionswert wird zum neuen Parameter im nächsten Durchlauf). Das Verfahren wird nun iteriert, wobei nun immer von den Schnittpunkten des Medians aus die vertikale Linie zum Funktionsgraphen gelegt wird (siehe Bild 4).

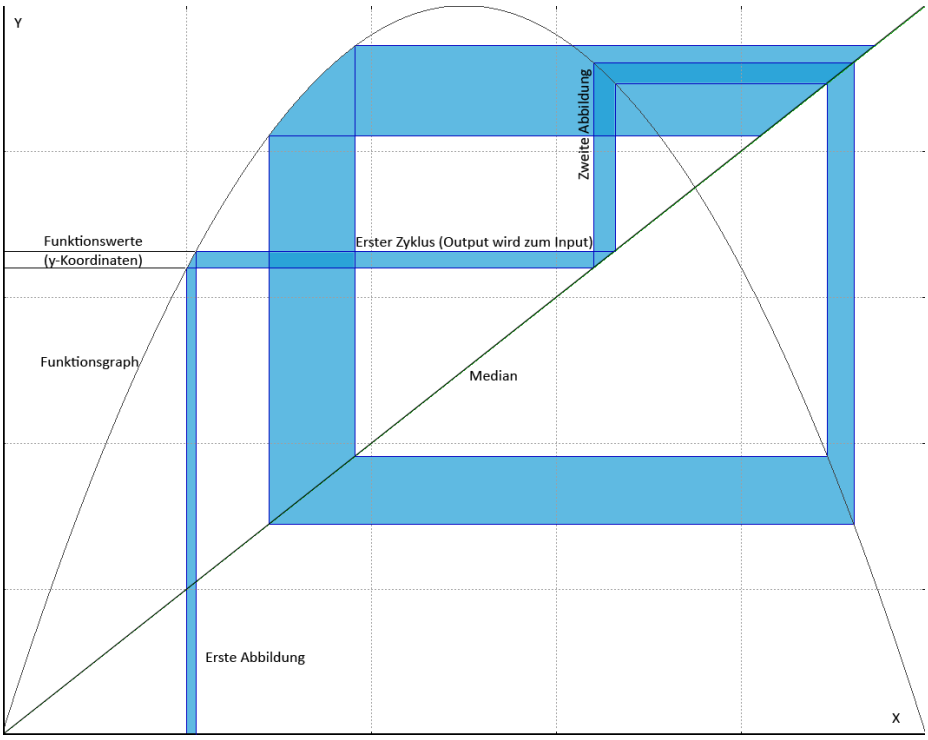


Bild 4 : Erste Schritte der Graphische Iteration.

Die Aufgabe soll außerdem die folgenden Anforderungen erfüllen:

- Es sollen zumindest die folgenden Funktionen und entsprechenden Wertebereiche graphisch iteriert werden können:
  - $f_a(x) = ax\cdot(1-x)$, $x \in [0, 1]$, $a \in [0, 4]$
  - $f_a(x) = a\cdot\sin(\pi\cdot x)$, $x \in [0, 1]$, $a \in [0, 1]$
  - $f_a(x) = x(1+a\cdot(1-x))$, $x \in [0, (1+a)/a]$, $a \in [0, 1]$
  - $f_a(x) = x\cdot e^{a\cdot(1-x)}$, $x \in [0, \infty]$, $a \in [0, \infty]$
- Die Funktionen sollen über ein GUI auswählbar sein.
- Der Parameter a soll mittels GUI (Slider) innerhalb des angegebenen Intervalls variiert werden können. Die Darstellung des Funktionsgraphen wird dabei unmittelbar aktualisiert. Linien vorheriger graphischer Iterationen werden gelöscht.
- Ein Startwert oder Startintervall sollen direkt an der x-Achse mit der Maus gesetzt werden können (z.B. Klick im Bereich der x-Achse für einzelnen Punkt oder Drag & Drop für ein Intervall).
- Über Buttons im GUI soll die Graphische Iteration gestartet, gestoppt und zurückgesetzt werden können. Weiters soll es mittels Button möglich sein in Einzelschritten fortzufahren.
- Es wäre schön, wenn immer nur eine gewisse Anzahl an Zyklen zu sehen ist. Dazu müssten Linien von Zyklen gelöscht werden, die zeitlich zu weit zurück liegen. Damit kann besonders das chaotische Verhalten gut verdeutlicht werden, weil zu sehen ist, dass immer neue Wege erzeugt werden (optionale Anforderung für Interessierte).

Aufgabe 5.4 Seltsame Attraktoren Punkte: 60

Der Rössler- und Lorenz-Attraktor sollen visualisiert werden. Sie beschreiben das langfristige Verhalten der folgenden dynamischen Systeme.

	Differentialgleichungssystem	**Parameterwerte (für chaotisches Verhalten)**
Rössler-Attraktor	x'=–y-z y'=x+ay z'=b+z(x-c)	b = 2 c = 4 a = 0.375
Lorenz-Attraktor	x' = σ(y-x) y' = rx-y-xz z' = xy-bz	σ = 10 B = 8/3 R = 28

Hinweis: Ein seltsamer Attraktor wird dabei durch ein Trajektorie eines beliebigen Startpunktes x_0 approximiert. Nach der Euler-Methode wird eine solche Trajektorie $\{x_0, x_1, ..., x_{nmax}\}$ z.B. für den Lorenz-Attraktor folgendermaßen numerisch berechnet:

$$x_{n+1} = x_n + \Delta t \cdot x'_n \qquad \text{wobei } x'_n = \sigma(y_n - x_n)$$

$$y_{n+1} = y_n + \Delta t \cdot y'_n \qquad \text{wobei } y'_n = rx_n - y_n - x_n z_n$$

$$z_{n+1} = z_n + \Delta t \cdot z'_n \qquad \text{wobei } z'_n = x_n y_n - bz_n$$

Der Faktor Δt bestimmt die Distanz zwischen den diskreten Euler-Schritten und darf nicht zu groß gewählt werden (sonst wird die Visualisierung zu kantig). Die Iterationstiefe nmax hängt von der Auflösung des Darstellungsbereichs und Δt ab und muss empirisch bestimmt werden. Die ersten k Iterationen sollen nicht dargestellt werden, weil sie den Weg vom Startwert x_0 in den Attraktor beschreiben, wobei k ebenfalls empirisch bestimmt werden muss.

Die Aufgabe soll außerdem die folgenden Anforderungen erfüllen:

- Zusätzlich zum Attraktor sollen die Koordinatenachsen dargestellt werden.
- Es soll möglich sein den Attraktor zu drehen, um ihn von allen Seiten betrachten zu können (Trackball-Modus). Weiters soll es möglich sein in die Ansicht zu zoomen.
- Über ein GUI soll zwischen Rössler- und Lorenz-Attraktor umgeschaltet werden können.
- Die Parameter für die Differentialgleichungssysteme der Attraktoren sollen über das GUI variierbar sein (z.B. über Textfelder mit Spin-Boxen). Danach muss der Attraktor neu gezeichnet werden. Damit kann untersucht werden, für welche Parameterwerte chaotisches Verhalten auftritt.

- Es wäre schön, wenn immer nur eine gewisse Anzahl an Euler-Schritten zu sehen ist. Dazu müssten Linien von Euler-Schritten gelöscht werden, die zeitlich zu weit zurück liegen. Damit kann besonders das chaotische Verhalten gut verdeutlicht werden, weil zu sehen ist, dass immer neue Pfade erzeugt werden (optionale Anforderung für Interessierte).

5. References

[1] J. C. Hart und T. A. DeFanti, „Efficient Antialiased Rendering of 3-D Linear Fractals," in SIGGRAPH 91, 1991.

[2] Wikipedia Foundation, „Phong-Beleuchtungsmodell," 2014. [Online]. Available: http://de.wikipedia.org/wiki/Phong-Beleuchtungsmodell. [Zugriff am 03 10 2104].

[3] Wikimedia Foundation, „Gaussian blur," 2014. [Online]. Available: http://en.wikipedia.org/wiki/Gaussian_blur. [Zugriff am 03 10 2014].

[4] Wikimedia Foundation, „Bates distribution," 2014. [Online]. Available: http://en.wikipedia.org/wiki/Bates_distribution. [Zugriff am 03 10 2014].

[5] P. Prusinkiewicz und A. Lindenmayer, The Algorithmic Beauty of Plants, Springer Verlag, 1996.

[6] Wikimedia Foundation, „Methode der kleinsten Quadrate," 2014. [Online]. Available: http://de.wikipedia.org/wiki/Methode_der_kleinsten_Quadrate. [Zugriff am 03 10 2014].