



BERTOLT-BRECHT-GYMNASIUM  
SCHWARZENBERG

KOMPLEXE LEISTUNG

# Fraktale

KEVIN HOANG

Betreuer  
Herr HELLER

20. Dezember 2023

# Inhaltsverzeichnis

<b>Einleitung</b>	<b>2</b>
<b>1 Theoretischer Teil</b>	<b>3</b>
1.1 Was sind Fraktale? . . . . .	3
1.2 Komplexe Zahlen . . . . .	4
1.3 Mandelbrot-Menge . . . . .	5
1.4 Julia-Mengen . . . . .	6
<b>2 Praktischer Teil</b>	<b>7</b>
2.1 Benutzeroberfläche . . . . .	7
2.2 Programmaufbau . . . . .	8
2.3 Implementierung der Algorithmen . . . . .	8
<b>3 Fazit</b>	<b>10</b>
3.1 Zusammenfassung der Ergebnisse . . . . .	10
3.2 Anwendungsmöglichkeiten von Fraktalen . . . . .	10
<b>4 Anhang</b>	<b>11</b>
4.1 Quellcode . . . . .	11
4.2 Bibliotheken . . . . .	11
4.3 Literaturverzeichnis . . . . .	11

# Einleitung

**F**RAKTALE sind faszinierende Strukturen, die uns allgegenwärtig umgeben. Von der verzweigten Struktur eines Baumes bis hin zu den Wolken am Himmel, die sich in immer kleiner werdende Strukturen aufteilen, sind Fraktale ein fester Bestandteil unserer natürlichen Umgebung. Aber auch in der Mathematik sind Fraktale zu finden.

Der Begriff *Fraktal* wurde um 1975 vom Mathematiker Benoît Mandelbrot geprägt und beschreibt natürliche und künstliche Formen, die bestimmte geometrische Eigenschaften aufweisen.

Fraktale zeichnen sich durch ihre *Selbstähnlichkeit* auf unterschiedlichen Größenskalen aus. Das bedeutet, dass Teile der Struktur in verschiedenen Vergrößerungen ähnliche Formen aufweisen.

Im theoretischen Teil dieser komplexen Leistung werde ich mich mit der mathematischen Theorie hinter Fraktalen beschäftigen, insbesondere mit der *Mandelbrot-Menge* und den *Julia-Mengen*.

Der praktische Teil beinhaltet die Algorithmen sowie deren Implementierung zur Darstellung von Fraktalen mithilfe von Computergrafik und der Programmiersprache *C++*, wobei insbesondere die *OpenGL*-Bibliothek genutzt wird.

Ziel dieser komplexen Leistung ist es, nicht nur die mathematischen Grundlagen von Fraktalen zu verstehen, sondern auch die Ästhetik von Fraktalen zu verdeutlichen, sowie auf die Anwendung von Fraktalen in der realen Welt einzugehen.

# 1 | Theoretischer Teil

In diesem Abschnitt werden die theoretischen Grundlagen erschlossen, um im praktischen Teil ein Programm zur Darstellung von Fraktalen erstellen zu können.

## 1.1 Was sind Fraktale?

Fraktale sind Strukturen, die durch die wiederholte Anwendung einfacher Schritte aus sich selbst entstehen. Dabei weisen Fraktale auf unterschiedlichen Größenskalen betrachtet ähnliche oder identische Formen auf, was als Selbstähnlichkeit bezeichnet wird. Durch *Iteration* werden bestimmte Anweisungen auf immer kleiner werdenden Maßstab wiederholt, um das gewünschte Fraktal darzustellen.

Ein anschauliches Beispiel für diese Konzepte ist die *Koch-Kurve*. Diese wird durch die iterative Umwandlung einer geraden Linie gebildet, wobei der Mittelpunkt zu einer Spitze umgeformt wird. Dieser Prozess wird auf den durch die Spitze neu entstandenen Linien wiederholt, wodurch mit jeder Iteration eine immer komplexer werdende, selbstähnliche Struktur entsteht.



Abbildung 1.1:  
1. Iteration der Koch-Kurve

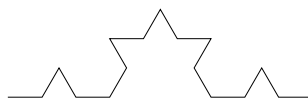


Abbildung 1.2:  
2. Iteration der Koch-Kurve

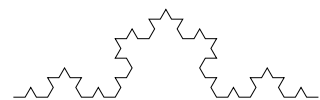


Abbildung 1.3:  
3. Iteration der Koch-Kurve

## 1.2 Komplexe Zahlen

Um die Berechnung der Mandelbrot-Menge und der Julia-Mengen im nächsten Abschnitt zu verstehen, werden zunächst die grundlegenden Konzepte zu *komplexen Zahlen* erläutert.

Komplexe Zahlen sind eine Erweiterung der reellen Zahlen  $\mathbb{R}$ , mit dem Ziel, Gleichungen wie  $x^2 = -1$  lösbar zu machen.

Eine komplexe Zahl  $\mathbb{C}$  besteht aus der Summe eines *Realteils*  $a$  und eines *Imaginärteils*  $b \cdot i$ , wobei  $b$  mit einer *imaginären Einheit*  $i = \sqrt{-1}$  erweitert wird und zu folgender Schreibweise führt.

$$z = a + b \cdot i$$

Anders als reelle Zahlen, welche auf einem 1-dimensionalen Zahlenstrahl dargestellt werden, verwendet man für die komplexen Zahlen die *Gauß'sche Zahlenebene*, welche einem 2-dimensionalen Koordinatensystem gleicht. Dabei wird die  $x$ -Achse in den Realteil und die  $y$ -Achse in den Imaginärteil eingeteilt.

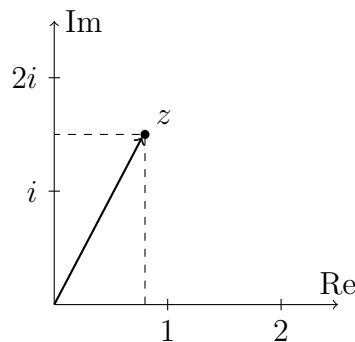


Abbildung 1.4:  $z = 0,8 + 1,5i$  in der Gaußschen Zahlenebene

Um mit zwei komplexen Zahlen  $z = a + bi$  und  $w = c + di$  rechnen zu können, gelten die folgenden Rechenregeln.

- **Addition und Subtraktion:**  $z \pm w = (a \pm c) + i \cdot (b \pm d)$
- **Multiplikation:**  $z \cdot w = (ac - bd) + i \cdot (ad + bc)$

Der Betrag  $|z| = \sqrt{a^2 + b^2}$  gibt die quantitative Menge der komplexen Zahl  $z$  an.

### 1.3 Mandelbrot-Menge

Die Mandelbrot-Menge gilt als das wohl bekannteste Fraktal und wurde im Jahr 1980 vom Mathematiker Benoît Mandelbrot entdeckt. Sie beschreibt die Menge aller komplexen Zahlen, deren Betrag bei Anwendung folgender iterativen Vorschrift gegen einen festgelegten Grenzwert konvergiert.

$$z_{n+1} = z_n^2 + c$$

Man beginnt mit  $z_0 = 0$  als Startwert und einer komplexen Zahl  $c$  als Konstante. Der Grenzwert, gegen den der Betrag der komplexen Zahl  $z$  konvergieren soll, wird auf 2 gesetzt. Zahlen, welche diesen Grenzwert nicht überschreiten, gehören zur Mandelbrot-Menge.

Zahlen, die diesen Grenzwert überschreiten, divergieren höchstwahrscheinlich ins Unendliche und gehören somit nicht zur Mandelbrot-Menge.

Wenn eine komplexe Zahl  $c$  zur Mandelbrot-Menge gehört, zeichnet man diese farbig in die Gauß'sche Zahlenebene ein. Diese Farbe wählt man anhand der Anzahl der Iterationen, die benötigt werden, um den Grenzwert zu erreichen. Auf die Farbgebung wird im praktischen Teil nochmal genauer eingegangen.

Die grafische Darstellung der Mandelbrot-Menge zeigt vielfältige Strukturen, wie das charakteristische *Apfelmännchen* in der Grundform mit zahlreichen Verzweigungen am Rand. Diese Strukturen setzen sich in selbstähnlicher Weise auf immer kleiner werdenden Maßstab fort.

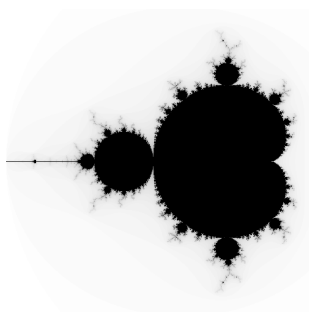


Abbildung 1.5:  
Apfelmännchen

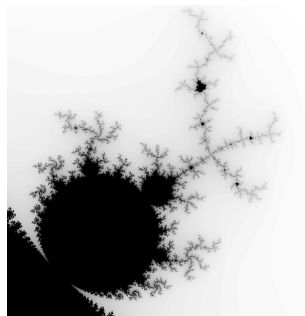


Abbildung 1.6:  
Vergrößerung einer  
Abzweigung

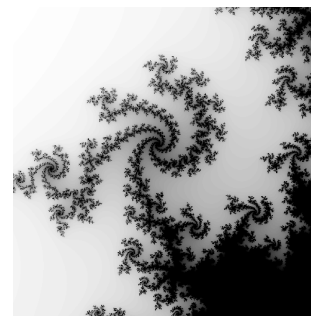


Abbildung 1.7:  
Spiralförmige  
Verzweigungen

## 1.4 Julia-Mengen

Ein weiteres Fraktal, welches eng mit der Mandelbrot-Menge zusammenhängt, sind die Julia-Mengen. Jeder komplexen Zahl  $c$  ist eine Julia-Menge zugeordnet, wobei die komplexen Zahlen, die auch zur Mandelbrot-Menge gehören, Julia-Mengen erzeugen, welche ähnliche Strukturen der Mandelbrot-Menge aufweisen.

Sie wurden erstmals 1918 von dem französischen Mathematiker Gaston Julia und dem französischen Physiker Pierre Fatou beschrieben.

Julia-Mengen werden ebenfalls durch die iterative Zahlenfolge  $z_{n+1} = z_n^2 + c$  gebildet. Anders als bei der Mandelbrot-Menge wird jedoch der Startwert  $z_0$  nicht auf 0 gesetzt, sondern auf die jeweilige komplexe Zahl  $z$ , für die berechnet wird, ob sie zur jeweiligen Julia-Menge der komplexen Zahl  $c$  gehört.

In der grafischen Darstellung weisen Julia-Mengen vergleichbare Eigenschaften auf wie die Mandelbrot-Menge. Ihre Strukturen sind jedoch weniger komplex, gleichen aber den jeweiligen Strukturen der Mandelbrot-Menge.

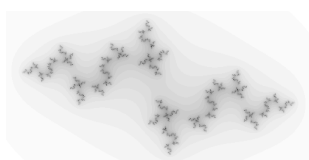


Abbildung 1.8:  
Julia-Menge für  
 $c = -1.0 + 4.0i$

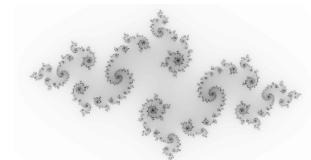


Abbildung 1.9:  
Julia-Menge für  
 $c = -0.8 + 0.2i$

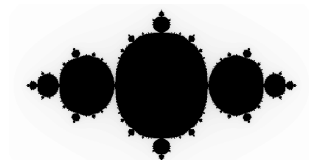


Abbildung 1.10:  
Julia-Menge für  
 $c = -0.8 + 0.0i$

## 2 | Praktischer Teil

Im praktischen Teil dieser komplexen Leistung werden die in der Theorie erläuterten Konzepte angewendet und in ein Programm umgesetzt.

Da die Erklärung des gesamten Programms den Rahmen dieser Arbeit sprengen würde, wird hier nur auf die wichtigsten Aspekte eingegangen. Der Quellcode des Programms ist im Anhang zu finden.

### 2.1 Benutzeroberfläche

Das Programm *Fraktale* ist ein interaktives Programm, um Fraktale zu visualisieren und zu erkunden.

Es besteht aus zwei Teilen: Einer Zeichenfläche auf die Fraktale gerendert werden und einem Einstellungsfenster, in dem man das aktuelle Fraktal und dessen Parameter, wie die Anzahl der Iterationen und den Farbgebungsalgorithmus einstellen kann. Das Fraktal kann man ganz einfach mit der Maus verschieben und mit dem Mausrad zoomen.

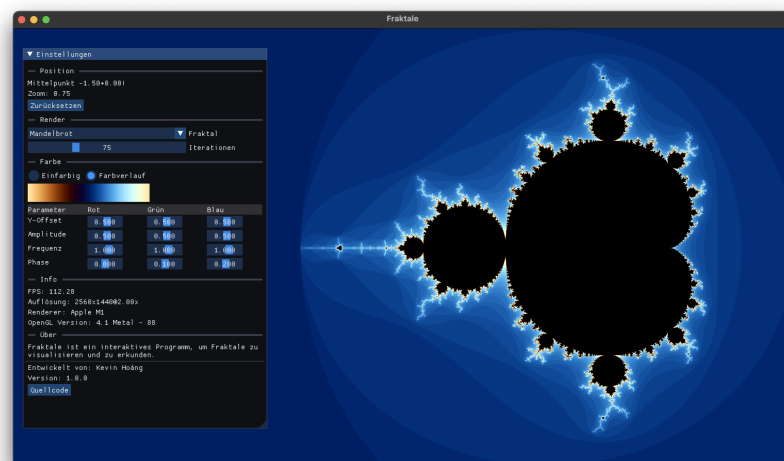


Abbildung 2.1: Das Programm *Fraktale*



## 2.2 Programmaufbau

Das Programm wurde in der Programmiersprache *C++* geschrieben und nutzt die Programmierschnittstelle *OpenGL* um eine effiziente und leistungsstarke Umsetzung der Algorithmen zu ermöglichen. Die Algorithmen wurden in Form von *Shadern* implementiert, die auf der Grafikkarte ausgeführt werden. Diese werden in der Programmiersprache *GLSL* geschrieben.

C++ und GLSL sind bezüglich Syntax sehr ähnlich, was das Deklarieren von Variablen und Funktionen, sowie Berechnungen betrifft.

Zusätzlich wurden Bibliotheken verwendet, deren genauere Beschreibung und Funktion sich im Anhang befindet.

Das Programm besteht aus folgenden Dateien:

- `main.cpp` - Die Hauptdatei, in der das Programm initialisiert wird.
- `window.h` - In dieser Datei befinden sich Funktionen, die das Fenster erstellen und verwalten.
- `dearimgui.h` - In dieser Datei befinden sich Funktionen, um die Benutzeroberfläche zu erstellen und zu verwalten.
- `callbacks.h` - In dieser Datei befinden sich Funktionen, um die Fraktale mit der Maus zu verschieben und zu zoomen.
- `shader.h` / `shader.cpp` - Eine Hilfsdatei, die das Laden und Kompilieren von Shadern ermöglicht.
- `misc.h` - Eine Hilfsdatei, die Funktionen enthält, die in anderen Dateien verwendet werden.
- `VertexShader.vert` - Der Vertex Shader, welcher für die Positionierung der *Vertices* zuständig ist.
- `FragmentShader.frag` - Der Fragment Shader, wo die eigentliche Berechnung des Fraktals stattfindet.

## 2.3 Implementierung der Algorithmen

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist

das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

## 3 | Fazit

### 3.1 Zusammenfassung der Ergebnisse

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

### 3.2 Anwendungsmöglichkeiten von Fraktalen

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

## 4 | Anhang

### 4.1 Quellcode

Der Quellcode und das Programm sind über den beigelegten USB-Stick oder über folgenden GitHub Link verfügbar:

<https://www.github.com/kevinoverflow/Fraktale>

### 4.2 Bibliotheken

#### GLFW

*Eine plattformübergreifende Bibliothek zur Erstellung von Fenstern mit OpenGL-Kontexten.*

#### GLEW

*Die OpenGL Extension Wrangler Library für eine einfachere Handhabung von OpenGL-Erweiterungen.*

#### GLM

*OpenGL Mathematics (GLM) ist eine rein Header basierte C++-Mathematikbibliothek für Grafiksoftware, basierend auf den Spezifikationen der OpenGL Shading Language (GLSL).*

#### Dear ImGui

*Eine einfache Immediate Mode-Grafikbenutzeroberfläche für C++.*

#### CMake

*Ein plattformübergreifendes Open-Source-Build-System.*

#### CMakeRC

*Ein Ressourcencompiler in einem einzigen CMake-Skript.*

### 4.3 Literaturverzeichnis

Literaturverzeichnis