Heinrich-Böll-Gymnasium Troisdorf

Schuljahr 2022/2023

Die Mandelbrot-Menge

Mathematische Grundlagen und die visuelle Darstellung

verfasst von

Christoph Derszteler

Leistungskurs Mathematik

Betreuerin: Frau Dammers

Abgabetermin: 23.02.2023 12:00 Uhr CET

Inhaltsverzeichnis

1 Einleitung

Die Mandelbrot-Menge ist durch ihre hübschen, ansehnlichen Darstellungen, verglichen mit anderen mathematischen Phänomenen, recht bekannt. Dies liegt nicht allein an ihrer visuellen Attraktivität, sondern vielmehr auch an Benoît Mandelbrot [A.1], dem Entdecker dieser Menge. Dieser sorgte mit seinen häufigen Vorträgen und Büchern dafür, dass sich Fraktale, also selbstähnliche[1], geometrische Figuren mit gebrochener Dimension[2], vornehmlich die Mandelbrot-Menge, in der Bevölkerung weit verbreiteten [IBM11, Vgl. letzten Absatz].

Obwohl die Natur mit ihren fraktal-ähnlichen Formationen wie dem Aufbau einer Schneeflocke, dem Verlauf eines Flusses oder die Verteilung von Baumästen [nna] die Inspiration für Mandelbrot war [ZK14], so liegt der Ursprung dieser Arbeit in den für manchen simpler erscheinenden, viel moderneren aber dennoch genauso spannenden, computer-generierten Videos[3], die man im Internet finden kann. Mit unter anderem der Frage, wie diese Videos in Ansätzen generiert werden können, und vielem weiteren beschäftigt sich diese Arbeit.

Dafür und zum vollen Verständnis der Mandelbrot-Menge ist Grundlagenwissen gewisser Themengebiete erforderlich, das in Kapitel 2 näher erörtert wird. Kapitel 3 beschäftigt sich daraufhin mit der mathematischen Betrachtung der Mandelbrot-Menge und insbesondere mit der Analyse visueller Darstellungen dieser. Abschließend befasst sich diese Arbeit in Kapitel 4 mit der praktischen Anwendung der Mandelbrot-Menge in Form von Bildgenerierungen mithilfe von Computern.

[1] Das heißt, sich selbst wiederholend oder in ähnlicher Form erneut aufkommend.

[2] Im Vergleich zu zum Beispiel einem zwei-dimensionalen Viereck.

[3] Vgl. bspw. [Tow17].

2 Theoretische Grundlage

Dieses Kapitel befasst sich mit den benötigten theoretischen Grundlagen, um der restlichen Arbeit folgen zu können. Dafür wird zunächst das Konzept der komplexen Zahlen als auch der für den weiteren Verlauf benötigter Umgang mit diesen erörtert. Darauffolgend wird das Prinzip und die Eigenschaften von Iterationen grob anhand eines Beispiels skizziert.

2.1 Komplexe Zahlen

Unter den komplexen Zahlen $\mathbb{C}$ versteht man die nächst größere Zahlenmenge nach den reellen Zahlen $\mathbb{R}$, die zusätzlich zu einem Realteil auch einen sogenannten Imaginärteil besitzen. Sie werden im weiteren Verlauf in der kartesischen Form $z = a + bi$ dargestellt, wobei a der Realteil und bi der Imaginärteil ist. Der Buchstabe i steht hierbei für die imaginäre Einheit und ist definiert durch die Gleichung $i^2 = -1$.

2.1.1 Multiplikation & Addition von komplexen Zahlen

Viele Rechenoperationen mit komplexen Zahlen funktionieren anders, als man sie von den reellen oder natürlichen Zahlen gewohnt ist. Im Folgenden werden zwei dieser unterschiedlich funktionierenden Operationen vorgestellt:

Zur Addition zwei komplexer Zahlen addiert man den Realteil und den Imaginärteil getrennt voneinander und fügt diesen danach wieder zusammen [Lic02, S. 2]: $(a_1 + b_1 i) + (a_2 + b_2 i) = a_1 + a_2 + (b_1 + b_2)i$.

Um komplexe Zahlen zu multiplizieren, wendet man das Distributivgesetz an, indem man den zweiten Faktor ebenfalls in seinen Realteil und seinen Imaginärteil unterteilt und diese jeweils einzeln mit dem ersten Faktor multipliziert [Lic02, S. 2f.]. Die zwei entstehenden Produkte lassen sich dann wie oben beschrieben addieren. Bei der Multiplikation mit dem Imaginärteil multipliziert man unter anderem zwei imaginäre Elemente miteinander. Da $i^2 = -1$ gilt, entsteht durch

diese Multiplikation ein negatives, aber reales Produkt. Wie in A.2 gezeigt, gilt somit: $(a+bi)(c+di) = ac - bd + (bc + ad)i$

Das in der Mandelbrot-Menge häufig angewandte Quadrieren von komplexen Zahlen lässt sich mit der kartesischen Form ebenfalls herleiten [A.3]. Für eine gegebene, zu quadrierende, komplexe Zahl $a+bi$ gilt somit: $(a+bi)^2 = a^2-b^2+2abi$.

Ein illustriertes Beispiel soll beide Rechenoperationen veranschaulichen:

$$\begin{aligned} & (-3+6i)^2 + (7+(-4i)) \\ = & ((-3\cdot(-3)) - (6\cdot 6) + ((6\cdot(-3)) + (-3\cdot 6))i) + (7+(-4i)) \\ = & (-27 + (-36i)) + (7 + (-4i)) \\ = & -20 + (-40i) \end{aligned} \tag{2.1}$$

2.1.2 Graphische Darstellung komplexer Zahlen

Komplexe Zahlen können wie Zahlen anderer Zahlenmengen grafisch dargestellt werden. Da komplexe Zahlen sowohl aus einem Realteil als auch aus einem Imaginärteil bestehen, reicht eine Achse nicht aus, um diese darzustellen; stattdessen braucht man eine Ebene[4]. Diese komplexe Zahlenebene teilt den Realteil auf die waagerechte Achse und den Imaginärteil auf die horizontale Achse auf. Eine komplexe Zahl $z = a + bi$ besitzt somit die Koordinaten $P(a|b)$.

Zusätzlich lässt sich eine komplexe Zahl wie eine reelle Zahl absolut betrachten, wobei dieser absolute Wert ebenfalls als der Abstand zum Ursprung zu betrachten ist [Lic02, S. 3]. Aufgrund dessen, dass eine komplexe Zahl aus zwei Komponenten besteht, lässt sich der Abstand über den Satz des Pythagoras berechnen:

$$|z|^2 = a^2 + b^2 \quad \text{beziehungsweise} \quad |z| = \sqrt{a^2 + b^2} \tag{2.2}$$

[4]Ebenfalls unter komplexer Zahlenebene und gaußsche Zahlenebene zu finden.

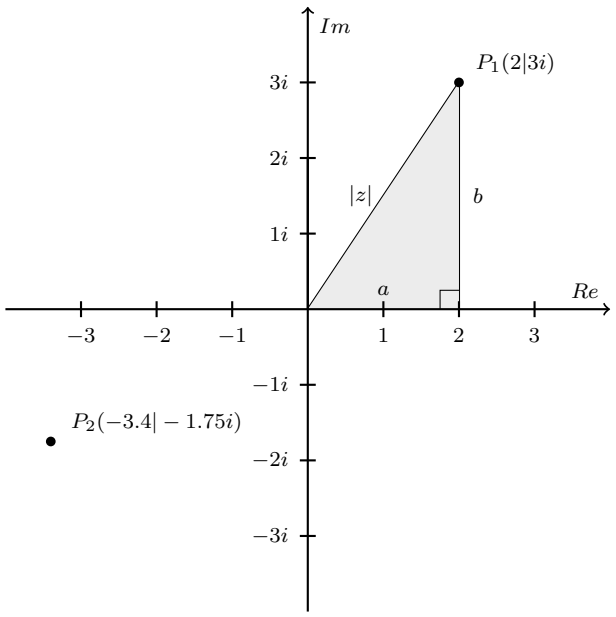

Abbildung 2.1: Komplexe Ebene mit den Punkten P_1, und P_2 und dem absoluten Wert $|z|$ vom Punkt P_1 [Eigene Darstellung].

2.2 Iterationen

Iterationen beziehen sich in der Mathematik auf das Wiederholen einer bestimmten Prozedur beziehungsweise in diesem Fall einer Berechnung. Bei Funktionsiterationen iteriert (also wiederholt) man die Berechnung eines Funktionswerts mit dem Funktionsargument des vorherigen Funktionswerts: $z_1 = f(z_0)$, $z_2 = f(z_1)$, $z_3 = f(z_2)$, $\cdots$, $z_n = f(z_{n-1})$.

Eine wichtige Eigenschaft von Iterationen ist die Entwicklung von z für $z \to \infty$. Dabei wird unterschieden, ob die Iteration divergent ist, das heißt gegen Unendlich verläuft („explodiert“), oder sich einem bestimmten Punkt annähert. Letzteres bezeichnet man als einen beschränkten Verlauf.

Letzter Verlauf ist bei Iterationen schwer vorauszusagen, denn ähnliche wirkende Funktionen können dennoch sehr unterschiedliche Entwicklungen aufweisen. Die Funktionen $f_c(z) = z^2 + c$ mit $z_0 = 0$ stellen beispielhaft die unterschiedlichen Verlaufsformen für verschiedene Parameter c mithilfe von $c_1 = 1$, $c_2 =$

-1 und $c_3 = 0.5$ dar:

	$f_c(z)$ für unterschiedliche Parameter c		
Iteration	$c_1 = 1$	$c_2 = -1$	$c_3 = 0.5$
1.	1	-1	$0,5$
2.	2	0	$0,75$
3.	**5**	-1	$1,0625$
4.	26	0	$\approx 1,6289$
5.	667	-1	$\approx$ **3, 1533**
6.	$\approx 1{,}9 \cdot 10^{11}$	0	$\approx 10,4433$
7.	$\approx 3{,}9 \cdot 10^{22}$	-1	$\approx 109,5625$

Tabelle 2.1: $f_c(z)$ verläuft mit c_1 und c_3 divergent, higegen ist der Verlauf von $f_c(z)$ mit c_2 beschränkt. Die dick markierten Zahlen sind für eine spätere Erwähnung dieser Tabelle relevant [Eigene Darstellung].

3 Mathematische Betrachtung

Nachdem im vorherigen Kapitel die Grundlagen für die Mandelbrot-Menge erklärt wurden, befasst sich dieses Kapitel mit der mathematischen Betrachtung dieser Menge, indem diese zunächst fachlich korrekt definiert und im Anschluss grafisch analysiert wird.

3.1 Definition

Die Mandelbrot-Menge $\mathbb{M}$ wird mit der bereits im vorherigen Kapitel vorgestellen, komplexen Iteration $z_{n+1} = z_n^2 + c$ mit $z_0 = 0$ und einem variablen Wert für c [Wei18, S.477ff.] definiert. Dabei enthält die Menge alle komplexen Werte für c, mit denen die oben angegebene Iteration beschränkt ist. Mathematisch ist die Menge iterativ wie folgt definiert:

$$\mathbb{M} = \{c \in \mathbb{C} \mid \forall n \in \mathbb{N} : \ |f_c^n(z)| \ \leqslant 2\} \quad \text{mit} \quad f_c(z) = z^2 + c;\ z \in \mathbb{C} \tag{3.1}$$

Wie in der Definition zu sehen, wird der Funktionswert der gegen Unendlich strebenden n-ten Iteration, ausgedrückt durch $f^n(z)$, absolut betrachtet, was bedeutet, dass die Funktion symmetrisch zur reellen Achse ist.

Ebenfalls zu betrachten ist die Einschränkung auf Funktionswerte $\leqslant 2$, denn für alle Funktionswerte, die sich in der oben genannten Iteration ergeben und > 2 sind, lässt sich das jeweilige c aus der Mandelbrot-Menge ausschließen. Obwohl der gesamte Beweis dessen über den Rahmen dieser Arbeit hinausginge, so soll dennoch angemerkt werden, dass mithilfe der Dreiecksungleichung und vollständiger Induktion unter der Vorausnahme von $|z_n| > 2$ und $|z_n| > |c|$ folgende Ungleichung, die eine divergente Entwicklung repräsentiert, $\frac{|z_{n+1}|}{|z_n|} > 1$ bewiesen werden kann [MH97], wobei sich zusätzlich zeigen lässt, dass für alle Werte von $|c| > 2$ nach spätestens 2 Iterationen gilt: $z_2 = |c^2 + c| \geqslant |c|^2 - |c| > 2$.

Es befinden sich deshalb alle Werte für c als auch somit die grafische Darstellung der Mandelbrot-Menge in einem Kreis, dessen Mittelpunkt im Ursprung liegt, mit dem Radius 2 [Vgl. A.4].

3.2 Grafische Analyse

Im Folgenden sollen der grundlegende Aufbau der in einem kartesischen Diagramm entstehenden Formation der Mandelbrot-Menge erörtert als auch eine Erklärung zur Farbbedeutung gegeben werden. Zusätzlich zeigt dieses Unterkapitel mit einer exemplarischen Kartografierung verschiedene, sich wiederholende Bereiche der visuellen Darstellung.

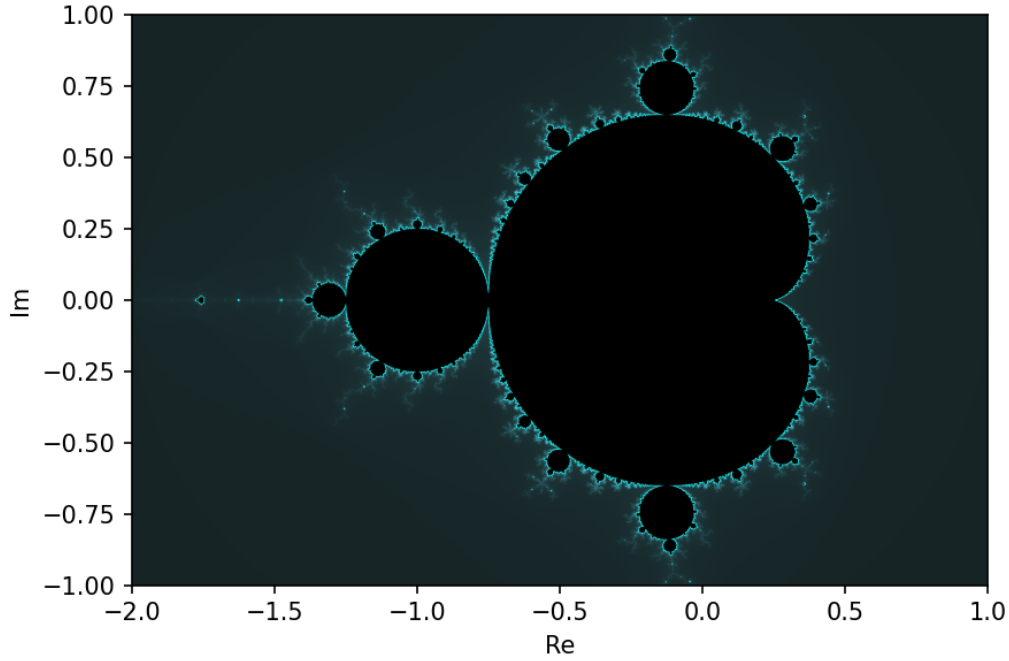

Abbildung 3.1: Exemplarische Darstellung der Mandelbrot-Menge [Eigene Darstellung][5].

[5]Generiert durch den Code A.10 mit einer „RESOLUTION“ von 2000 und einer „MAX_ITERATIONS“ von 500.

Die hier zu sehende Grafik entspricht der Darstellung der Mandelbrot-Menge in einer komplexen Zahlenebene und wird aufgrund ihrer Form „Apfelmännchen“ genannt. Die zu sehenden, schwarz gefärbten Pixel repräsentieren einen jeweiligen Wert für c, der sich in der Mandelbrot-Menge befindet.

Auffällig ist bei erster Betrachtung, dass sich neben der großen, einheitlichen Struktur in der Mitte deutlich kleinere, ähnlich aussende Formationen um den eigentlichen Hauptkörper, dem Apfelmänchen, zum Beispiel im negativen Teil der reellen Achse, erkennen lassen. Diese werden Satelliten genannt und existieren aufgrund der Selbstähnlichkeit der Mandelbrot-Menge in unendlicher Stückzahl – und zwar nicht nur für den Hauptkörper, sondern ebenfalls für jeden Satelliten selbst [LP18].

3.2.1 Farbbedeutung

Im Gegensatz zu den ersten grafischen Darstellungen der Mandelbrot-Menge, auf denen, aufgrund ihrer deutlich geringeren Auflösung, kleine Satelliten als Druckfehler gewertet wurden[6] und die mit ihrem schwarz-weißen Druck lediglich zwischen Werte für c in und außerhalb der Mandelbrot-Menge unterschieden, besitzt die oben dargestellte Figur (3.1) einen Farbverlauf. Dieser, in diesem Fall türkisblaue, Farbgradient gibt an, wie viele Iterationen es benötigte, um festzustellen, ob das jeweilige c außerhalb der Mandelbrot-Menge liegt. Dabei gilt, dass je heller der Pixel ist, desto mehr Iterationen hat es benötigt, um das jeweilige c aus

[6]Dies ist eine recht amüsante Anekdote: Während den frühsten Forschungen, die Benoît Mandelbrot in den 1970er bei IBM anstellte, war das Drucken deutlich mühseliger und aufwendiger, als es heutzutage ist. Deshalb existierte eine ganze Abteilung nur für die Herstellung und Bearbeitung von Drucks, die - da ähnliche Druckfehler damals häufig vorkam - kleine Satelliten am Rande der ersten Darstellungen [A.5] gutgemeint wegretuschierten. Die ersten Bilder, die Herr Mandelbrot also erhielt, verwunderten ihn sehr und er war äußerst aufgebracht, als er von der tolpatschigen Wahrheit erfur [Num19].

$\mathbb{M}$ auszuschließen[7]. Deshalb existiert ein hell erscheinender Rand um die schwarzen Formationen, da für diese Werte, die nicht in $\mathbb{M}$, dafür jedoch sehr nah an tatsächlichen Werten für $\mathbb{M}$ liegen, viele Iterationen benötigt werden, um diese auszuschließen.

Neben dieser einen, verhältnismäßig simplen und dementsprechend auf den ersten Blick aussagekräftigeren Farbkodierung existiert eine Vielzahl an teils deutlich umfangreicheren Algorithmen, mit denen sich eine für das menschliche Auge ansprechendere Farbgestaltung erzielen lässt. Diese werden in der Sektion **??** genauer beschrieben. Eine Reihe an solchen komplexere Farbverläufe benutzenden Beispielen, auf die im nächsten Unterkapitel Bezug genommen wird, lässt sich in A.6 betrachten.

3.2.2 Exemplarische Kartografierung

Die hingegen von der Farbe unabhängigen, entstehenden Formationen der Mandelbrot-Menge, die teilweise erst bei sehr kleinen Ausschnitten erkennbar sind, sind kartografiert und teilweise, wegen einer gewissen Ähnlichkeit, nach Objekten aus der realen Welt benannt. So bezeichnet man die größte kreisförmige Kardioide oder auch „Knospe“ als „Körper“ (wobei dieser genauer unterteilt werden kann) und die daran angrenzende Kardioide als „Kopf“[8].

Obwohl man jeden Punkt beziehungsweise jeden Ausschnitt beliebig detailliert analysieren kann, werden aufgrund der Selbstähnlichkeit Elemente mit ähnlichem oder gleichen Aufbau erneut hervorkehren und dementsprechend gleich benannt. Im Folgenden soll beispielhaft ein Ausschnitt des in diesem Video [Bey17] gezeigten „Tal der Seepferdchen“ analysiert werden:

Die Spalte zwischen Kopf und Körper wird „Tal der Seepferdchen“ genannt[9]

[7] Vgl. z.B. dick markierten Werte für $c_1 = 1$ und $c_3 = 0.5$ in Tabelle 2.1. c_1 ließ sich nach der 3. Iteration aus der Mandelbrot-Menge ausschließen, hingegen war dies bei c_3 erst nach der 5. Iteration der Fall.

[8] Vgl. A.7.

[9] Vgl. A.6.1.

[Rob10], da bei Vergrößerung dieses Ausschnitts sich unter anschaulicher Farbkodierung auf der rechten Seite Seepferdchen-ähnliche Formationen erkennen lassen[10]. Vergrößert man die Sicht auf das Seepferdchen-Tal stark, so lässt sich, neben weiteren (teils deformierten) Satelliten[11], bei genauerer Betrachtung des „Seepferdchenschwanzes“ ein Misiurewicz-Punkt erkennen[12]. Dieser Misiurewicz-Punkt zeigt ebenfalls die Selbstähnlichkeit der Mandelbrot-Menge auf, da sich dieser Punkt neben einer Drehung kaum von der eigentlichen Mandelbrot-Menge unterscheidet [Lei89]. Vergrößert man diesen Punkt weiter, so findet man erneut einen im Vergleich zum Apfelmännchen sehr ähnlichen aussehenden Satelliten[13].

[10] Vgl. A.6.2.
[11] Vgl. A.6.3.
[12] Vgl. A.6.4.
[13] Vgl. A.6.5.

4 Praktische Anwendung

Im Folgenden werden die im letzten Kapitel untersuchten mathematischen Betrachtungen unter Realbedingungen angewandt, indem die dadurch entstehenden Grenzen spezifiziert als auch die konkreten, informationstechnischen Umsetzungen exemplarisch dargestellt werden. Dafür werden zunächst die Funktionsweise als auch die dabei durch die Unterschiede zur reinen Mathematik entstehenden Probleme bei der Bildgenerierung der Mandelbrot-Menge erläutert. Im Anschluss werden verschiedene Herangehensweisen und Algorithmen zur Farbkodierung beispielhaft vorgestellt.

4.1 Korrelation zwischen Informationstechnik und Mathematik

Die im letzten Kapitel analysierte Abbildung 3.1 wurde, wie in der Beschriftung beschrieben, mit dem Code aus A.10 generiert. Im Vergleich zu den mathematischen Überlegungen ist die informationstechnische Herangehensweise dabei sehr ähnlich:

Grundlegend wird jeder Pixel mit seiner x- und y-Koordinate des zu generierenden Ausschnitts einer komplexen Zahl zugeordnet. Dabei entspricht (in der regulären Darstellung) die x-Koordinate dem Realteil a und die y-Koordinate dem Imaginärteil b, sodass eine komplexe Zahl $z = a + bi$ in einem Pixel P durch $x = a$ und $y = b$ ausgedrückt werden kann.

Das naheliegendste Problem, neben vielen ausschließlich informationstechnischen Optimierungsaufgaben, ergibt sich bei der Berechnung, ob es sich beim jeweiligen Pixel beziehungsweise dem jeweiligen c, um ein Element in $\mathbb{M}$ handelt. Denn im Gegensatz zu der mathematischen Betrachtung ist es in der konkreten Umsetzung nicht möglich, die Iterationsanzahl n bei $f^n(z)$ gegen Unendlich konvergieren zu lassen. Deshalb setzt man einen Grenzwert m an Iterationsdurchläufen, ab dem, sofern das zu überprüfende c noch nicht aus der Mandelbrot-Menge

ausgeschlossen wurde, dieses als Element von $\mathbb{M}$ angenommen wird.

Folglich bestimmt dieser Wert indirekt die Auflösung beziehungsweise Genauigkeit des zu generierenden Bilds und muss deshalb bei kleineren Ausschnitten besonders hoch sein, da dabei ein Unterschied zwischen komplexen Zahlen ausgemacht werden muss, dessen Werte sich lediglich um geringe Nachkommastellen unterscheiden. Die Auswirkungen dieser Iterationsgrenze sind durch die Bildergalerie A.8, die generierte Bilder der Mandelbrot-Menge mit unterschiedlich (niedrigen) Grenzen darstellt, anschaulich visualisiert.

4.2 Farbkodierung

Um die in der Sektion 3.2.1 erklärten Farben zu kodieren, eignet sich zunächst das HSV (Hue, Saturation, Value) Farbmodel, womit sich grundlegend mithilfe von prozentualen Werten Farben, vor allem jedoch Farbveränderungen hinsichtlich sowohl der Helligkeit als auch der Farbsättigung, erzielen lassen. Die jeweilige Farbe wird anhand des Verhältnisses der benötigten Iterationen n zu der Iterationsgrenze m für jeden zu überprüfenden komplexen Wert c für $\mathbb{M}$ berechnet [Rob22]. Als Beispiel sei Hue mit 186° und Value mit 100% gegeben, woraus sich für jedes c folgendes HSV ergibt: $HSV(186°, \frac{n}{m} \cdot 100\%, 100\%)$.

Diese Farbkodierung ist aufgrund ihrer Simplizität in ihrer beschriebenen Form direkt informationstechnisch umsetzbar, jedoch eignet sich solch eine Herangehensweise aus Performance- beziehungsweise Optimierungsgründen nicht für jede zu überprüfende komplexe Zahl c, wenn es sich bei der Farbzusammenstellung um einen komplexeren Zusammenhang wie in den Elementen der Bildergalerie A.6 handelt. Um dieses Problem zu lösen, erstellt man sogenannte Colormaps (engl.: Farbpaletten), wie zum Beispiel A.9.1 aus den Werten des oberen Beispiels. Für ein jeweiliges c wird mithilfe solcher Farbpaletten eine Farbe erneut anhand dessen Verhältnisses von benötigten Iterationen n zu der Iterantionsgrenze m ermittelt; dabei existieren jedoch bereits alle vorkommenden Farben, was

somit die Berechnung des aufwendigeren Farbgenerierungsprozesses[14] von dem Berechnungsprozess der eigentlichen Mandelbrot-Menge entkoppelt. In A.9.2 ist die für alle für die Arbeit generierten Abbildungen benutzte Colormap zu sehen, die im Vergleich zur vorherigen dargestellten Palette mehr als zwei (insgesamt 4) Ankerpunkte[15] besitzt, wobei diese, im Gegensatz zu einer komplexen[16] Colormap [z.B. A.9.3], mit der man visuell anschaulichere Bilder generieren kann, weiterhin recht minimal ist.

[14] Also die Auswahl und korrekte Zusammenstellung der Farben einer Farbpalette.

[15] Hier: Eindeutige HSV-Farben, die in gewissen Abständen auf einer Palette platziert werden und zwischen denen ein Gradient ensteht.

[16] Eine Farbpalette mit vielen Ankerpunkten.

5 Fazit

Zusammenfassend lässt sich festhalten, dass durch die verhältnismäßig simple Gleichung $z_{n+1} = z_n^2 + c$ sich eine riesige Welt der Fraktale eröffnet hat, die durch ihre visuellen Darstellungen mit ihrer atemberaubenden Schönheit nicht nur eine Nische von Mathematikern, sondern seit langem erneut auch die breitere Gesellschaft erreichen und erstaunen konnte. Diese Arbeit hat nicht nur einen grundlegenden Überblick über die Mathematik hinter der Mandelbrot-Menge geliefert, sondern auch eine Erklärung und die damit einhergehenden Komplikationen zur Generierung solcher faszinierenden Bildern dargelegt.

Nichtsdestotrotz ist der abgedeckte Bereich dieser Abhandlung begrenzt; es existieren schließlich unzählige weitere Zusammenhänge zwischen der Mandelbrot-Menge und anderen mathematischen Phänomenen wie Pi oder der Fibonacci-Folge. Es würde sich ebenfalls nicht um eine vollständige Arbeit über die Mandelbrot-Menge handeln, wenn nicht die Julia-Mengen und ihre enge Verknüpfung zu der als übergeordnet zu betrachtenden Mandelbrot-Menge erwähnt werden würde.

Abschließend lässt sich mit Sicherheit sagen, dass man gespannt abwarten kann, welche interessanten Entdeckungen in diesem jungen Gebiet der Mathematik folgen werden.

6 Literatur und Quellen

[Bey05a] Wolfgang Beyer. *Partial view of the Mandelbrot set. Step 1 of a zoom sequence: Gap between the "headänd the "bodyälso called the ßeahorse valley".* 12. Sep. 2005. URL: https://de.wikipedia.org/wiki/Datei:Mandel_zoom_01_head_and_shoulder.jpg (besucht am 30. 01. 2023).

[Bey05b] Wolfgang Beyer. *Partial view of the Mandelbrot set. Step 2 of a zoom sequence: On the left double-spirals, on the right ßeahorses".* 12. Sep. 2005. URL: https://de.wikipedia.org/wiki/Datei:Mandel_zoom_02_seehorse_valley.jpg (besucht am 30. 01. 2023).

[Bey05c] Wolfgang Beyer. *Partial view of the Mandelbrot set. Step 3 of a zoom sequence: SSeahorseüpside down. Its "bodyïs composed by 25 ßpokes"consisting of 2 groups of 12 ßpokesëach and one ßpoke"connecting to the main cardioid. These 2 groups can be attributed by some kind of metamorphosis to the 2 "fingersöf the üpper handöf the Mandelbrot set. Therefore the number of ßpokesïncreases from one ßeahorse"to the next by 2. The "hubïs a so called Misiurewicz point. Between the üpper part of the bodyänd the "tailä distorted satellite can be recognized.* 12. Sep. 2005. URL: https://de.wikipedia.org/wiki/Datei:Mandel_zoom_03_seehorse.jpg (besucht am 30. 01. 2023).

[Bey05d] Wolfgang Beyer. *Partial view of the Mandelbrot set. Step 4 of a zoom sequence: The central endpoint of the ßeahorse tailïs also a Misiurewicz point.* 12. Sep. 2005. URL: https://de.wikipedia.org/wiki/Datei:Mandel_zoom_04_seehorse_tail.jpg (besucht am 30. 01. 2023).

[Bey05e] Wolfgang Beyer. *Partial view of the Mandelbrot set. Step 8 of a zoom sequence: Äntennaöf the satellite. Several satellites of second order can be recognized.* 12. Sep. 2005. URL: https://de.wikipedia.org/

wiki/Datei:Mandel_zoom_08_satellite_antenna.jpg (besucht am 30. 01. 2023).

[Bey17] Wolfgang Beyer. *Zoomfahrt in die Mandelbrotmenge*. 16. Juni 2017. URL: https://www.wolfgangbeyer.de/chaos/mandelzoom.htm (besucht am 29. 01. 2023).

[Elp07] Elphaba. *Command-line depiction of the Mandelbrot set, just like the picture Brooks and Matelski included in their article of 1978 on Kleinian groups.* 2. Feb. 2007. URL: https://en.wikipedia.org/wiki/File:Mandel.png (besucht am 30. 01. 2023).

[Gai97] Raphael Gaillarde. *Benoit Mandelbrot, mathematician, inventor of fractals*. 9. Feb. 1997. URL: https://www.gettyimages.co.uk/detail/news-photo/benoit-mandelbrot-mathematician-inventor-of-fractals-in-news-photo/110137025 (besucht am 22. 01. 2023).

[IBM11] IBM. *Fractal Geometry*. 21. Mai 2011. URL: https://www.ibm.com/ibm/history/ibm100/us/en/icons/fractal/ (besucht am 16. 01. 2023).

[Lei89] TAN Lei. "Similarity Between the Mandelbrot Set and Julia Sets". In: *Communications in Mathematical Physics*. 10. Juli 1989, S. 587–617. URL: https://arxiv.org/pdf/1410.6729v2.pdf (besucht am 29. 01. 2023).

[Lic02] Klaus Lichtenegger. *Komplexe Analysis*. Mai 2002. URL: https://www.math.tugraz.at/~lichtenegger/kompan.pdf (besucht am 16. 01. 2023).

[LP18] Luna Lomonaco und Carsten Lunde Petersen. "On quasi-conformal (in-) compatibility of satellite copies of the Mandelbrot set". In: (10. Okt. 2018). URL: https://arxiv.org/pdf/1505.05422.pdf (besucht am 29. 01. 2023).

[MH97] Robert P. Munafo und Mike Hurley. “Escape Radius, Mu-Ency at MROB”. In: (19. Sep. 1997). URL: http://mrob.com/pub/muency/escaperadius.html (besucht am 24. 01. 2023).

[MSC16] Arun Mahanta, Hemanta Sarmah und Gautam Choudhury. “MANDELBROT SET, THE MESMERIZING FRACTAL WITH INTEGER DIMENSION”. In: *International Journal of Applied Mathematics and Statistical Sciences* 6 (Dez. 2016), S. 1–18. URL: https://www.researchgate.net/figure/The-Body-B-and-Head-H-of-the-Mandelbrot-Set-The-surface-of-these-two-parts-are-covered_fig6_310798765 (besucht am 29. 01. 2023).

[nna] Mike (nnart). *Fractals in Nature*. How Do Fractals Appear in Nature? 10 Outstanding Examples. URL: https://nnart.org/fractals-in-nature/ (besucht am 16. 01. 2023).

[Num19] Numberphile. *What's so special about the Mandelbrot Set? - Numberphile*. 18. Apr. 2019. URL: https://youtu.be/FFftmWSzgmk?t=602 (besucht am 30. 01. 2023).

[Rob10] Munfao Robert P. “Seahorse Valley, Mu-Ency at MROB”. In: (7. Sep. 2010). URL: http://www.mrob.com/pub/muency/seahorsevalley.html (besucht am 30. 01. 2023).

[Rob22] Munfao Robert P. “Color, Mu-Ency at MROB”. In: (12. Dez. 2022). URL: http://www.mrob.com/pub/muency/color.html.

[Tow17] Maths Town. *Eye of the Universe*. Eye of the Universe - Mandelbrot Fractal Zoom (e1091) (4k 60fps). 28. Aug. 2017. URL: https://www.youtube.com/watch?v=pCpLWbHVNhk (besucht am 16. 01. 2023).

[Tro] Heinrich Böll Gymnasium Troisdorf. *HBG Logo*. In der Titelseite zu finden. URL: https://www.hbgtroisdorf.de/images/hbg_logo_web.png (besucht am 18. 12. 2022).

[Wei18] Edmund Weitz. *Konkrete Mathematik (nicht nur) für Informatiker: Mit vielen Grafiken und Algorithmen in Python*. 1. Aufl. Springer Spektrum, 8. Aug. 2018. 942 S. ISBN: 978-3-658-21565-1.

[ZK14] Iris Zink und Hanna Kotarba. *Der kosmische Code*. Der kosmische Code. 28. Sep. 2014. URL: `https://www.zdf.de/dokumentation/terra-x/faszination-universum-der-kosmische-code-mit-harald-lesch-100.html` (besucht am 16. 01. 2023).

7 Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die Facharbeit selbstständig und ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benutzt habe. Alle verwendeten Materialien habe ich im Anhang angegeben. Ich erkläre ferner verbindlich, dass ich alle Zitate kenntlich gemacht habe und ihre Herkunft im Anhang angeben habe. Mir ist klar, dass ein Zuwiderhandeln gegen diese Bestimmungen zu einer 0-Punkte Bewertung der Arbeit führt.

Niederkassel, den _____

Unterschrift des Verfassers: _____

A Anhang

A.1:

Abbildung A.1: Benoît Mandelbrot 1997 in Frankreich [Gai97].

A.2:

$$\begin{aligned} & z_1 \cdot z_2 \\ &= (a+bi) \cdot (c+di) \\ &= c(a+bi) + di(a+bi) \\ &= ac + bci + adi + bdi^2 \\ &= ac + bci + adi - bd \\ &= ac - bd + (bc+ad)i \end{aligned} \tag{A.2}$$

A.3:

$$\begin{aligned} z_1^2 &= z_1 \cdot z_1 \\ &= (a+bi) \cdot (a+bi) \\ &= a \cdot (a+bi) + bi \cdot (a+bi) \\ &= a^2 + abi + abi - b^2 \\ &= a^2 - b^2 + 2abi \end{aligned} \tag{A.3}$$

A.4:

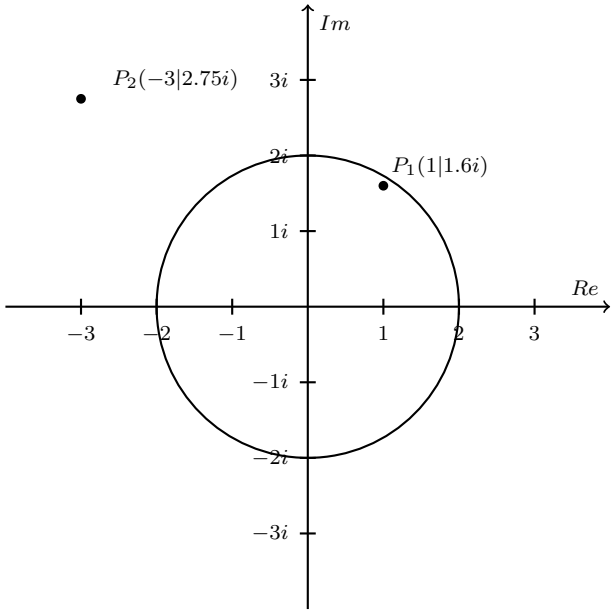

Abbildung A.4: Einheitskreis mit dem Radius 2. Zu sehen ist der Punkt P_1, der im Einheitskreis liegt und Punkt P_2, der außerhalb des Einheitskreises liegt [Eigene Darstellung].

A.5:

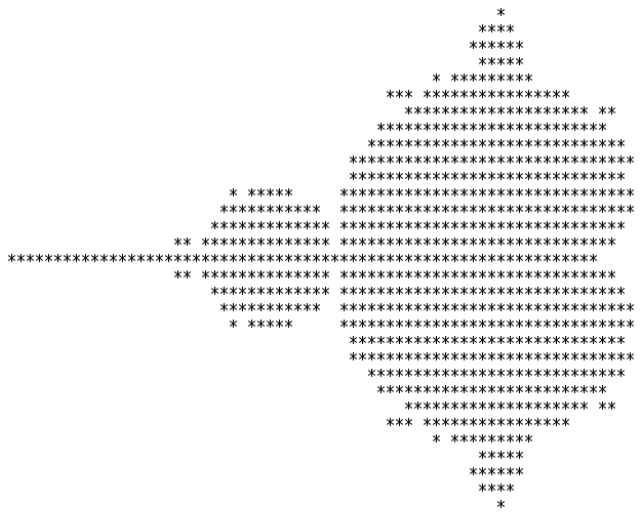

Abbildung A.5: Darstellung der ersten Drucks der Mandelbrot-Menge [Elp07].

A.6:

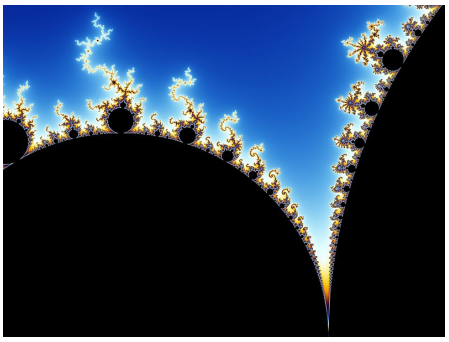

Abbildung A.6.1: Spalte zwischen Kopf und Körper [Bey05a].

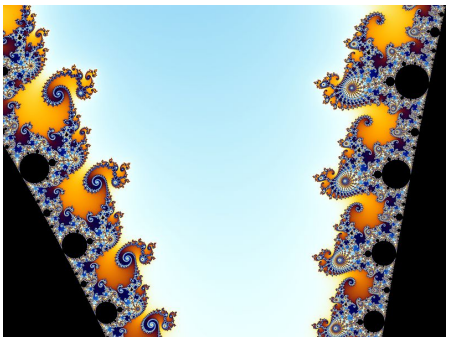

Abbildung A.6.2: “Tal der Seepferdchen„ [Bey05b].

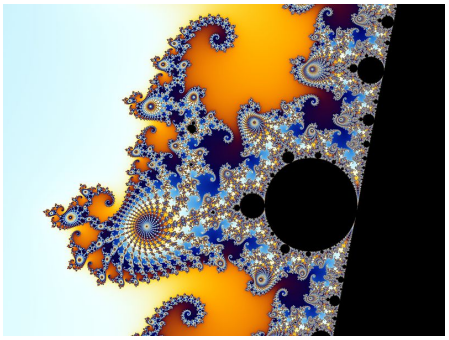

Abbildung A.6.3: Rechts ein deformierter Satellit und links Misiurewicz-Punkt [Bey05c].

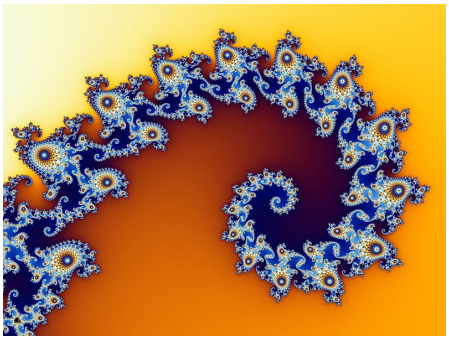

Abbildung A.6.4: Misiurewicz-Punkt [Bey05d].

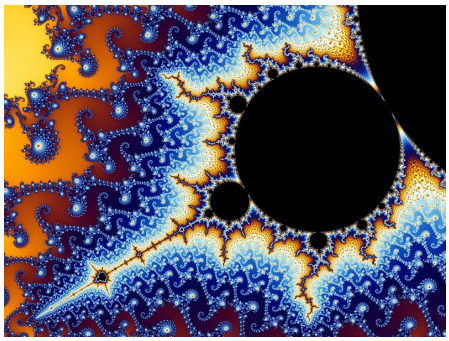

Abbildung A.6.5: Satellit mit ähnlicher Struktur wie das Apfelmännchen [Bey05e].

A.7:

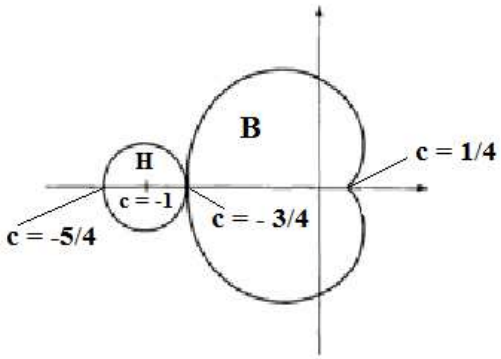

Abbildung A.7: Körper (B) und Kopf (H) der Mandelbrot-Menge [MSC16].

A.8:

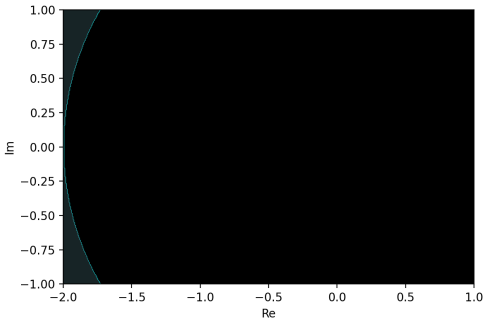

Abbildung A.8.1: Generiertes Bild mit einer Iterationsgrenze von 1 [Eigene Darstellung].

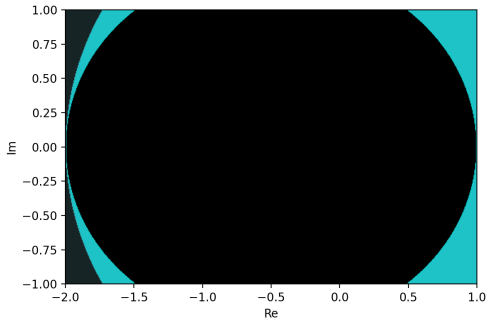

Abbildung A.8.2: Generiertes Bild mit einer Iterationsgrenze von 2 [Eigene Darstellung].

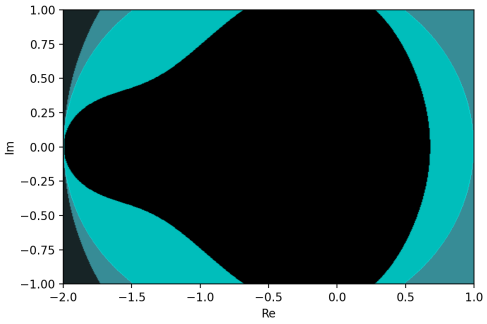

Abbildung A.8.3: Generiertes Bild mit einer Iterationsgrenze von 3 [Eigene Darstellung].

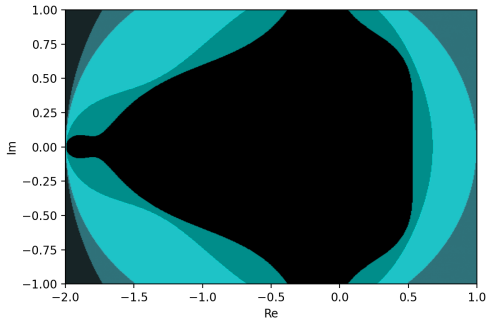

Abbildung A.8.4: Generiertes Bild mit einer Iterationsgrenze von 4 [Eigene Darstellung].

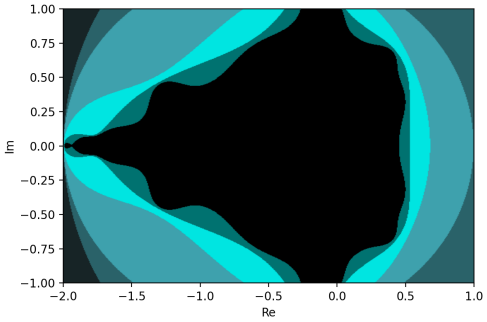

Abbildung A.8.5: Generiertes Bild mit einer Iterationsgrenze von 5 [Eigene Darstellung].

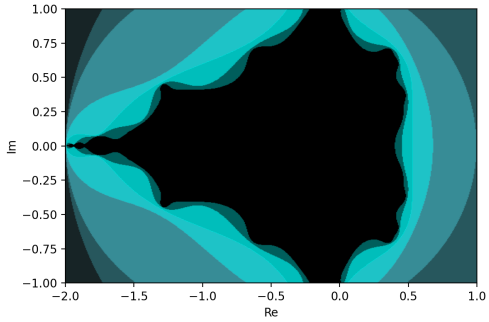

Abbildung A.8.6: Generiertes Bild mit einer Iterationsgrenze von 6 [Eigene Darstellung].

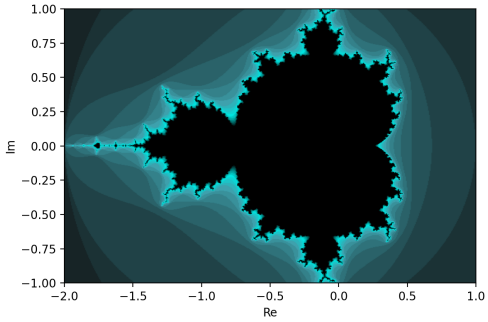

Abbildung A.8.7: Generiertes Bild mit einer Iterationsgrenze von 20 [Eigene Darstellung].

A.9:

Abbildung A.9.1: Simple HSV-Colormap mit einem linearen Farbverlauf [Eigene Darstellung].

Abbildung A.9.2: Komplexe HSV-Colormap mit vier verschiedenen Ankerpunkten [Eigene Darstellung].

Abbildung A.9.3: Komplexe HSV-Colormap mit vielen unterschiedlichen Ankerpunkten [Eigene Darstellung].

A.10:

```
import matplotlib.pyplot as plot
import numpy

def divergent_iterations(real, imaginary, max_iterations):
    c = complex(real, imaginary)
    z = 0.0j

    for iteration in range(max_iterations):
        z = z ** 2 + c
        if abs(z) > 2:
            return iteration

    return max_iterations

def generate_pixel_iterations_map(resolution, max_iterations):
    xs = resolution
    ys = resolution

    pixels = numpy.zeros([xs, ys])

    for x, real in enumerate(numpy.linspace(SCOPE[0], SCOPE[1], num=
        xs)):
        for y, imaginary in enumerate(numpy.linspace(SCOPE[2], SCOPE
            [3], num=ys)):
            pixels[x, y] = divergent_iterations(real, imaginary,
                max_iterations)
    return pixels

def draw_plot(pixels, color_map, size):
    plot.figure(dpi=size)
    plot.imshow(pixels.T, cmap=color_map, interpolation="bilinear",
        extent=SCOPE)
    plot.xlabel("Re")
```

```
    plot.ylabel("Im")

# This value defines the quality/resolution of the image. For quick
    generations,
# 500 is recommended. For higher quality images, use a value between
# 2000-5000
RESOLUTION = 500
# This value defines the image size. It should be updated accordingly
    with the
# resolution. An image size above 300 should often not be necessary.
IMAGE_SIZE = min(RESOLUTION * 1.25, 300)
# This value defines the scope/detail of the image. Default: [-2, 1,
    -1, 1]
SCOPE = [-2, 1, -1, 1]
# This value should be changed if the scope is not the default.
    Default: 100
MAX_ITERATIONS = 100
# This value changes the color mapping of the generated image. For a
    small
# scope, "twilight" might be interesting. The default value uses
# a custom generated color map
COLOR_MAP = {
    'red': (
        (0.0, 0.09019607843137255, 0.09019607843137255),
        (0.4, 0.24313725490196078, 0.24313725490196078),
        (0.6,  0.0, 0.0),
        (1.0, 0.0, 0.0)
    ),
    'green': (
        (0.0, 0.1411764705882353, 0.1411764705882353),
        (0.4, 0.6313725490196078, 0.6313725490196078),
        (0.6, 0.8980392156862745, 0.8980392156862745),
        (1.0, 0.0, 0.0)
    ),
    'blue': (
        (0.0, 0.14901960784313725, 0.14901960784313725),
        (0.4, 0.6784313725490196, 0.6784313725490196),
        (0.6, 0.8823529411764706, 0.8823529411764706),
        (1.0, 0.0, 0.0)
    )
}

pixels = generate_pixel_iterations_map(RESOLUTION, MAX_ITERATIONS)
draw_plot(pixels, COLOR_MAP, IMAGE_SIZE)
plot.show()
```

Code A.1: Python-Code zur Generierung unter anderem der Abbildung 3.1 [Eigene Darstellung].

Weitere Anhänge und verwendete Webseiten befinden sich in der digitalen Version.