# Mandelbrot- und Julia-Menge

*Vorwissenschaftliche Arbeit verfasst von*

Jakob Ausserer
Klasse 8B

*Betreuer: Mag. Werner Jägersberger*

Februar 2022

---

GRG XIII, Wenzgasse 7, A-1130 Wien

# Abstract

Im Zuge dieser Arbeit werden die Mandelbrot-Menge und Julia-Mengen mithilfe eines JavaScript-Programms berechnet und dargestellt. Für beide Fraktale gibt es je eine quadratische Iterationsformel, welche für eine komplexe Zahl entweder divergiert oder beschränkt bleibt. Bleibt die Zahlenfolge beschränkt, so gehört die untersuchte komplexe Zahl zur Mandelbrot- bzw. Julia-Menge. Zur Erstellung von Bildern der beiden Mengen wird der Escape-Time Algorithmus verwendet, wobei Optimierungen zur Verkürzung der Rechenzeit zum Einsatz kommen. Der Algorithmus stellt fest, ob eine komplexe Zahl innerhalb von $n_{max}$ Iterationen divergiert und färbt diesen Punkt entsprechend ein. $n_{max}$ muss abhängig von der untersuchten Stelle in der komplexen Zahlenebene und dem Zoomlevel in geeigneter Größe gewählt werden. Da JavaScript keine komplexen Zahlen kennt, wird im Programmcode mit Real- und Imaginärteil gerechnet. Aus mathematischer Sicht ist es möglich, unendlich tief in die Mandelbrot- und Julia-Menge hineinzublicken, wobei immer wieder ähnliche Strukturen zum Vorschein kommen. Jedoch stößt jedes Berechnungsprogramm aufgrund seiner endlichen Rechengenauigkeit ab einer bestimmten Vergrößerung an seine Grenzen. Die Arbeit gibt eine Näherungsformel für die notwendige Rechengenauigkeit in Abhängigkeit vom Zoomlevel an. Abschließend wird der Zusammenhang zwischen der Mandelbrot- und der Julia-Menge aufgrund der nahe verwandten Iterationsformeln besprochen.

# Inhaltsverzeichnis

# 1 Einleitung

In dieser Arbeit geht es um die Darstellung der Mandelbrot- und der Juliamenge. Dabei handelt es sich um Zahlenmengen im Raum der komplexen Zahlen. Bei der Darstellung von den Mengen im zweidimensionalen Zahlenraum kommen interessant aussehende Bilder zu Stande. Egal wie klein der Bildausschnitt ist, der betrachtet wird, zeigt sich eine interessante Figur. Bei dem Erkunden der Fraktale tauchen immer wieder ähnliche Strukturen auf unterschiedlichen Zoomlevels auf, was als Selbstähnlichkeit bezeichnet wird. Die Mengen werden durch rekursive mathematische Formeln berechnet. Dies ist praktisch nur durch einen Computer möglich, da enorme Zahlenmengen verarbeitet werden müssen. Im Laufe der Arbeit wird genauer erklärt, was ein Fraktal ausmacht. Die Fraktale werden anhand vieler Bilder veranschaulicht und die Algorithmen zu deren Berechnung mit Programmcode erklärt. Es wird auch der Frage nachgegangen, wieso die beiden Mengen einander ähnlich sind.

# 2 Die Mandelbrot-Menge

## 2.1 Benoît Mandelbrot

Der in Polen geborene Franzose Benoît Mandelbrot ist der Schöpfer des Mandelbrot-Fraktals, das wegen seines Aussehens auch als „Apfelmännchen“ bezeichnet wird. Er entdeckte sein Interesse an Fraktalen erstmalig bei dem Durchlesen eines Textes von L. F. Richardson über die Messung von Küstenlängen. Das Problem bei solch einer Vermessung ist, dass die Messeinheiten ausschlaggebend für das Endergebnis sind. Je genauer die Küste vermessen wird, desto größer wird die gemessene Länge, da immer mehr Details erfasst werden. Benoît Mandelbrot verstand, dass bei der Küstenvermessung eigentlich eine beliebig große Strecke gemessen werden kann, solang der Maßstab klein genug ist. Jahre später untersuchte er unter anderem die Hilbert-Kurve, welche durch Iterationsschritte generiert wird. Dabei wird die Kurve immer feiner und raumfüllender, und so fiel ihm auf, dass ihre Länge nicht definierbar ist, sondern mit zunehmender Tiefe immer länger wird. Daraufhin erschuf er den Begriff Fraktal. Ein

weiteres seiner Erkenntnisse war, dass Objekte auf verschiedenen Skalen sich ähnlich sind. So zum Beispiel sehen die immer wieder sich verzweigenden Äste eines Baumes fast ident aus. Nachdem er sich weiterhin mit dem Thema beschäftigt hatte, gelang es ihm am 1. April 1980 das erste Bild des Mandelbrot-Fraktals zu zeichnen. In weiterer Folge schrieb er sein Lebenswerk „The fractal Geometry of Nature“, welches bis heute eines der erfolgreichsten mathematischen Bücher ist [1].

## 2.2 Definition Fraktal

Fraktale sind geometrische Muster mathematischen Ursprungs, welche auf unendlich viele Skalen aus ähnlichen Strukturen bestehen. Diese Eigenschaft wird als „Selbstähnlichkeit“ bezeichnet. Dabei wird zwischen linearen und nicht-linearen Systemen unterschieden, worüber die mathematische Ursprungsformel entscheidet. Lineare Fraktale sind exakt selbstähnlich, während nicht-lineare Fraktale nur ähnliche Abbilder ihrer selbst beinhalten. Jedes Fraktal besitzt eine bestimmte räumliche Dimension, welche nicht unbedingt ganzzahlig sein muss, die sogenannte Hausdorff-Dimension, oder auch fraktale Dimension. Eine weitere Eigenschaft von Fraktalen ist die nie endende innere Komplexität, welche beispielsweise bei der Cantor-Menge durch die immer feiner werdenden Linien sichtbar wird [2].

### 2.2.1 Die Hausdorff-Dimension

In der nebenstehenden Grafik ist die Cantor-Menge dargestellt, eine Linie, bei welcher in jedem Iterationsschritt aus allen Teilen das mittlere Drittel entfernt wird. Dabei werden die einzelnen Segmente immer ähnlicher einem Punkt. Eine Linie besitzt die Dimension 1, ein Punkt jedoch nur die Dimension 0. Das bedeutet, dass die Cantor-Menge weder die Dimension 1 noch die Dimension 0 besitzt, tatsächlich besitzt

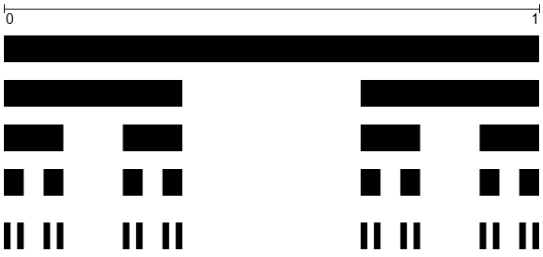


Abb. 1: Die Cantor-Menge (Eigene Darstellung)

[1] (Herrmann, 1996, S. 26)

[2] (Bräuner, 2002, S. 47)

die Menge eine Hausdorff-Dimension von 0.6309 [3]. Sowohl die Randlinie der Mandelbrot-Menge als auch bestimmte Julia-Mengen haben die fraktale Dimension 2 [4]. Obwohl es Sinn macht, dass die Mandelbrot-Menge als Fläche die Dimension 2 annimmt, ist es etwas anders, wenn man nur die Randlinie betrachtet. Wie in der Mathematik bereits bewiesen, sind Linien normalerweise eindimensional. Das bedeutet, es wird genau eine Koordinate benötigt, um jeden beliebigen Punkt auf der Linie zu beschreiben. Auch eine gebogene Linie ist eindimensional, zum Beispiel der Einheitskreis. Abermals kann jeder Punkt mit nur einer Koordinate beschrieben werden, beispielsweise mit dem Winkel ausgehend vom Mittelpunkt, oder dem Abstand ausgehend von einem Fixpunkt auf der Umlaufbahn. Allerdings ist dies bei der Randlinie des Mandelbrotes nicht möglich, da die Linie eine unendliche Länge besitzt, und bei genauerer Betrachtung immer länger wird. Das bedeutet, dass die Randlinie der Mandelbrot-Menge mit zwei Koordinaten beschrieben werden muss, sie hat die Hausdorff-Dimension zwei.

Aus mathematischer Sicht kann die Hausdorff-Dimension durch Erforschen von zwei Eigenschaften eines Fraktals berechnet werden.

$$D = \frac{\log a}{\log s}$$

Hierbei wird in dem Fraktal die nächst-tiefere Ebene gesucht. Bei der Cantor-Menge ist dies eine weitere Iteration. Danach wird gesucht, aus wieviel Teilen die Ebene besteht, welche ident mit der vorherigen Ebene sind. Dieser Wert wird für $a$ eingesetzt. Für $s$ wird der Wert des Skalierungsfaktors zwischen den soeben untersuchten Ebenen verwendet. Bei der Cantor-Menge entstehen nach jeder Iteration zwei weitere Teile, welche mit der darüberstehenden Figur ident sind, daher beträgt $a = 2$. Ihre Größe beträgt je ein Drittel, der Skalierungsfaktor („Vergrößerungsfaktor“) ist somit $s = 3$. Die fraktale Dimension berechnet sich mit $\frac{\log 2}{\log 3} \approx 0{,}6309$. Diese Formel kann auch bei Objekten aus der echten Welt angewendet werden. Eine rechteckige Tischfläche kann in

[3] (Herrmann, 1996, S. 143)

[4] (Shishikura, 1998)

vier Teilstücke zerlegt werden, welche je die Hälfte der ursprünglichen Tischmaße messen. Daher hat die Tischfläche die Hausdorff-Dimension $\frac{\log 4}{\log 2} \approx 2$ [5].

## 2.3 Fraktale und Chaos

Chaotische Systeme sind dadurch gekennzeichnet, dass minimale Abweichungen der Ausgangszustände zu sehr unterschiedlichen Endzuständen führen. Ein Beispiel für ein chaotisches System ist die Erdatmosphäre. Der sogenannte Schmetterlingseffekt besagt, dass der Flügelschlag eines Schmetterlings einen Tornado auslösen oder auch verhindern kann. Beide Fälle werden von denselben physikalischen Gleichungen beschrieben. Es handelt sich daher um ein deterministisches System, also ein System mit definierten Gesetzen, welche theoretisch eine genaue Vorhersage der Zukunft ermöglichen.

In der Wettervorhersage können wir mit gemessenen physikalischen Ausgangswerten, wie Temperatur oder Windgeschwindigkeit, welche die Atmosphäre zu einem bestimmten Zeitpunkt beschreiben, das Wetter für die nächste Woche berechnen. Nach dieser Zeit sind die Vorhersagen jedoch nicht mehr genau, was auf das chaotische Verhalten der Gleichungen zurückzuführen ist. Das bedeutet, dass nur minimale Abweichungen der Ausgangsdaten mit der Realität, aufgrund begrenzter Messgenauigkeit, zu einem komplett anderen Resultat führen.

Die Chaostheorie beschäftigt sich mit eben solchen Systemen, deren Entwicklung praktisch unvorhersehbar ist, obwohl sie durch deterministische Gleichungen beschrieben werden. Diese Theorie hat in vielen Bereichen Relevanz, beispielsweise auch bei Computer-Algorithmen. Der SHA-256 Algorithmus verarbeitet Ausgangsdaten, in Form von Text oder einer Datei, und berechnet daraus eine Folge von Ziffern und Buchstaben. Es ist zu beachten, dass die Eingabe von beliebiger Länge sein kann, aber die Ausgabe immer 64 Zeichen lang ist. Schon bei der Veränderung eines einzelnen Zeichens der Ausgangsdaten erhält man ein vollkommen anderes Ergebnis. Dieser Algorithmus findet zum Beispiel beim Zuweisen eines kurzen Namens für eine Datei oder bei der Berechnung von Kryptowährungen Anwendung.

---

[5] (Bräuner, 2002, S. 48)

Auch bei dem Mandelbrot-Fraktal spielt die Chaostheorie eine Rolle. An vielen Stellen des Randes der Mandelbrot-Menge sind auf kleinem Raum Punkte, die zur Mandelbrot-Menge gehören und Punkte, welche nicht dazugehören, scheinbar ohne jegliche Ordnung vermischt. Der Rand des Fraktals ist vergleichbar mit einer Küstenlinie, welche bei genauem Betrachten eine sehr komplexe Form annimmt.

## 2.4 Mathematik

Die Mandelbrot-Menge ist jene Menge der komplexen Zahlen $c$, für welche die folgende Zahlenfolge $z_n$ mit $n \in \mathbb{N}$ beschränkt bleibt:

$$z_0 = 0$$

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

Die komplexe Zahl $c$ ist definiert als

$$c = c_r + c_i \cdot i$$

Dabei ist $i$ die imaginäre Einheit, für welche gilt:

$$i^2 = -1$$

$c_r$ und $c_i$ sind Real- und Imaginärteil der Zahl $c$, für welche berechnet wird, ob sie zur Mandelbrot-Menge gehört oder nicht. Bei der grafischen Darstellung werden $c_r$ und $c_i$ dementsprechend auf der x- und y-Achse als Koordinaten aufgetragen. Sämtliche Zahlen $c$ mit einem Betrag

$$|c| = \sqrt{c_r^2 + c_i^2} > 2$$

sind mit Sicherheit kein Teil der Mandelbrot-Menge, da sich diese Punkte bei Anwendung obiger Iterationsvorschrift immer weiter vom Ursprung entfernen.

Stellt man die Punkte der Mandelbrot-Menge grafisch dar, so entsteht die nebenstehende Abbildung. Bei der Berechnung der Mandelbrot-Menge wird für alle $c$ in einem bestimmten Bereich der komplexen Zahlenebene untersucht, ob die Zahlenfolge $z_n$ beschränkt bleibt. Dabei wird die Iteration abgebrochen, sobald

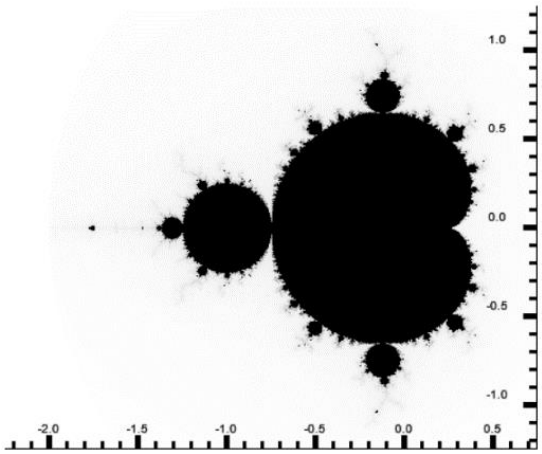


Abb. 2: Die Mandelbrot-Menge (Eigene Darstellung)

$$|z_n| = \sqrt{z_{n,r}^2 + z_{n,i}^2} > 2$$

$$\Longrightarrow z_{n,r}^2 + z_{n,i}^2 > 4$$

mit dem Ergebnis, dass die untersuchte Zahl $c$ nicht zur Mandelbrot-Menge gehört. Praktisch wird die Iteration immer nach $n_{max}$ Schritten abgebrochen, wobei üblicherweise für $n_{max}$ Werte im Bereich zwischen 50 und 500 bereits für eine klare grafische Darstellung ausreichen (Siehe Abschnitt 4.4). Falls bis $n = n_{max}$ der Betrag stets $|z_n| \leq 2$ bleibt, wird angenommen, dass die Zahlenfolge $z_n$ beschränkt bleibt und dass folglich $c$ zur Mandelbrot-Menge gehört.

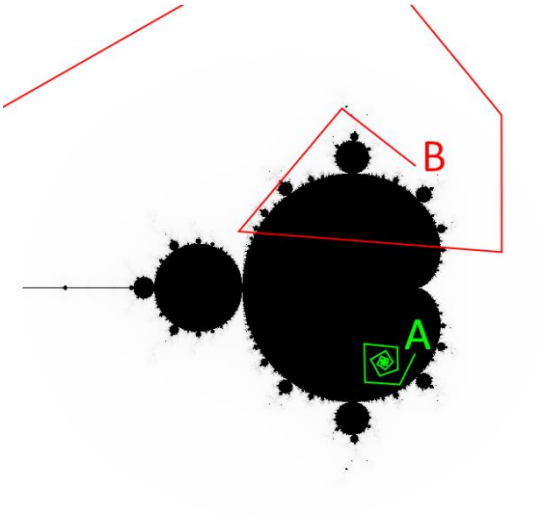


Abb. 3: Die Zahlenfolge zweier Punkte (Eigene Darstellung)

In der folgenden Abbildung sind die Punkte, welche Teil der Mandelbrot-Menge sind, schwarz gekennzeichnet, jene außerhalb weiß. Beispielhaft sind zu zwei Punkten A, B die Zahlenfolgen $z_n$ eingezeichnet, wobei jede Ecke der Linien einen berechneten Wert darstellt. Man sieht, dass die eine Zahlenfolge konvergiert und die andere divergiert. Da bei dem Punkt A die Zahlenfolge beschränkt bleibt, gehört er zur Mandelbrot-Menge. Das erkennt man daran, dass die grüne Linie nie den sichtbaren Zahlenraum verlässt. Daher ist die Zahlenfolge, welche aus der Rechnung mit $c = A$ entsteht, konvergent. Außerdem nähert sich die Zahlenfolge von Punkt A immer näher an einen Punkt

an, jedoch gilt dies nicht für jeden Punkt der Mandelbrot-Menge. Der Punkt B liegt aufgrund der gegen unendlich strebende Zahlenfolge außerhalb der Mandelbrot-Menge, wie durch die rote Linie gekennzeichnet ist. Daher ist die Zahlenfolge, welche aus der Rechnung mit $c = B$ entsteht, divergent.

### 2.4.1 Sonderfall reelles c ($c_i = 0$)

Für alle Punkte der Mandelbrot-Menge, welche sich auf der x-Achse befinden, ist der Rechenvorgang viel anschaulicher. Da der imaginäre Teil der Rechnung wegfällt kann die Iterationsformel auf

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

vereinfacht werden. Abermals wird die Iteration abgebrochen, sobald $|z_n|$ den Wert 2 überschreitet. Mit der neuen Formel ist es um einiges einfacher, konvergierende von divergierenden Zahlen zu unterscheiden. Beispielsweise ist es offensichtlich, dass für $c = -2 + 0i$ die Zahlenfolge beschränkt bleibt.

$$z_0 = \mathbf{0}$$

$$z_1 = 0^2 - 2 = \mathbf{-2}$$

$$z_2 = (-2)^2 - 2 = \mathbf{2}$$

$$z_3 = 2^2 - 2 = \mathbf{2}$$

$$z_4 = 2^2 - 2 = \mathbf{2}$$

Da der Ausgangswert der dritten Iteration ebenfalls dem Ergebnis der dritten Iteration entspricht, ist dies auch für alle kommenden Iterationen der Fall. Ein Beispiel mit einem chaotischen Verhalten ist $c = -1{,}7 + 0i$.

$$z_0 = \mathbf{0}$$

$$z_1 = 0^2 - 1{,}7 = \mathbf{-1{,}7}$$

$$z_2 = (-1{,}7)^2 - 1{,}7 = \mathbf{1{,}19}$$

$$z_3 = 1{,}19^2 - 1{,}7 = \mathbf{-0{,}2839}$$

$$z_4 = (-0{,}2839)^2 - 1{,}7 \approx \mathbf{-1.61940}$$

$$z_5 = (-1.61940)^2 - 1{,}7 \approx \mathbf{0.92245}$$

Bei der soeben gezeigten Zahlenfolge kann noch keine Aussage über das Verhalten von $c = -1{,}7 + 0i$ gemacht werden. Um zu bestimmen, ob dieses $c$ beschränkt bleibt, oder nicht, müssen mehr Rechenschritte gemacht werden. Allerdings wird in praktischen Berechnungen ein $n_{max}$ festgelegt, und nur bei sehr wenigen Zahlen im Randbereich der Mandelbrot-Menge, so wie bei $c = -2 + 0i$ kann je eine eindeutige Aussage über die Zugehörigkeit zur Mandelbrot-Menge gemacht werden. Selbst wenn ein $c$ im Randbereich nach 1000 Iterationen beschränkt bleibt, ist es möglich, dass es bereits im nächsten Iterationsschritt den Schwellenwert 2 überschreitet.

Für $c = -3 + 0i$, kann nach Betrachten der Zahlenfolge eine Aussage getroffen werden.

$$z_0 = \mathbf{0}$$

$$z_1 = 0^2 - 3 = \mathbf{-3}$$

$$z_2 = (-3)^2 - 3 = \mathbf{6}$$

$$z_3 = 6^2 - 3 = \mathbf{33}$$

$$z_4 = 33^2 - 3 = \mathbf{1086}$$

In diesem Fall wurde der Schwellenwert bereits nach der zweiten Iteration überschritten und die Rechenoperation könnte abgebrochen werden. Da das System außer Kontrolle geraten ist, und die divergierende Zahlenfolge immer größer wird, ist der Punkt $c = -3 + 0i$ kein Element der Mandelbrot-Menge.

### 2.4.2 Eigenschaften

Eine wichtige Eigenschaft des Mandelbrot-Fraktals ist, dass sämtliche Punkte der Menge miteinander verbunden sind. Das bedeutet, dass das Mandelbrot-Fraktal eine einzige zusammenhängende Fläche ohne isolierte Inseln ist, welche einen Umfang in Form einer einzelnen Linie hat. Dieser Umfang ist jedoch nicht berechenbar, da er durch seine fraktalen Eigenschaften eine unendliche Länge annimmt. Die Fläche jedoch ist beschränkt, da sich das Fraktal innerhalb eines Kreises mit dem Radius 2 befindet. Momentane numerische Schätzungen liefern das Ergebnis 1,5065918849 [6]. Außerdem

[6] (Numerical estimation of the area of the Mandelbrot set (2012), 2015, Tabelle 1)

ergibt sich aufgrund der Formel eine Symmetrie bezüglich der x-Achse. Diese Symmetrie ist auf die quadratische Iterationsformel zurückzuführen. Das Mandelbrot-Fraktal ist auf mehreren Ebenen zu sich selbst ähnlich, nimmt jedoch nie exakt dieselbe Form an. Zu guter Letzt befindet sich die Mandelbrot-Menge in einem bestimmten Zahlenraum, und erstreckt sich nicht ins Unendliche. Außerdem besitzt die Menge einen definierten Rand. Besitzt eine Menge diese beiden Eigenschaften, so wird sie in der Mathematik kompakt genannt. Zu guter Letzt belegt die Mandelbrot-Menge alle Punkte auf der x-Achse im Intervall $\left[-2, \frac{1}{4}\right]$ [7].

#### 2.4.2.1 Erklärung, weshalb $\left[-2, \frac{1}{4}\right] \epsilon\, Mandelbrot - Menge$ gilt

Ein mathematischer Beweis kann in (Zeitler & Neidhardt, 1993) auf Seite 180 nachgelesen werden. Für die Zwecke dieser Arbeit erfolgt eine grafische Erklärung. Da sich diese Punkte auf der x-Achse befinden, ist der Imaginärteil stets 0. Nun können wir den Rest der Formel als Funktion $f(x) = x^2 + c_r$ anschreiben. Der erste Schritt der Berechnung ist es, den Funktionswert des Ausgangswertes zu finden. Da das Ergebnis $f(x)$ im nächsten Iterationsschritt als Argument $x$ wiederverwendet wird, kann eine lineare Funktion $g(x) = x$ aufgestellt werden, um bei der Veranschaulichung zu helfen. Zeichnet man nun für $x_0 = c = 0{,}25$ eine Treppe zwischen $g(x)$ und $f(x)$ mit $x_2 = (x_1, f(x_1))$ bzw. $x_{n+1} = (x_n, f(x_n))$ so erhält man Abbildung 4. Wie erkennbar ist, konvergiert die Zahlenfolge $x_n$ gegen 0,5 und bleibt daher beschränkt. Da $g$ tangential

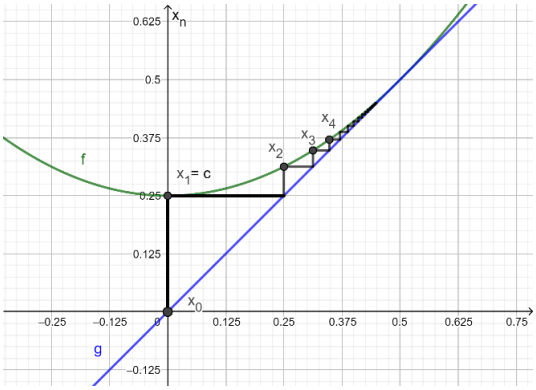


Abb. 4: Grafische Untersuchung des Punktes c = 0,25 (Eigene Darstellung)

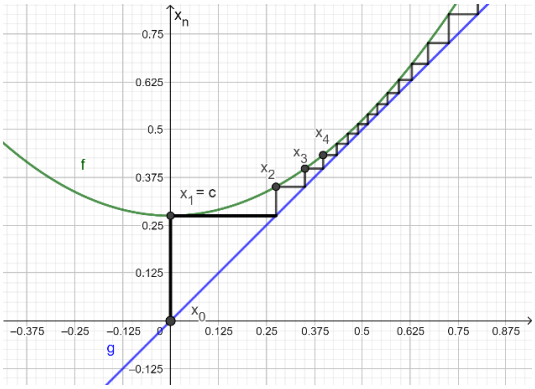


Abb. 5: Grafische Untersuchung des Punktes c = 0,275 (Eigene Darstellung)

[7] (Herrmann, 1996, S. 229)

zu $f$ verläuft haben die beiden Funktionen einen Berührungspunkt bei $(0{,}5|0{,}5)$. Wird das $c$ auch nur um einen minimalen Wert erhöht, so verläuft $f$ vollständig oberhalb von $g$, daher strebt die Zahlenfolge $x$ gegen unendlich. In Abbildung 5 wurde $c = 0.275$ gewählt. Daher sind alle Punkte über $c = 0.25$ keine Elemente der Mandelbrot-Menge.

Setzt man nun $c = -2$ ein, so ist die Zahlenfolge konstant bei $x_2 = 2$. Bei der rechten oberen Grafik wurde dieser Wert gewählt. Alle folgenden Punkte ab $x_3$ fallen zusammen, allerdings ist der Funktionswert bereits eine Iteration davor konstant bei 2, vergleiche Abschnitt 2.4.1. Im zweiten Bild wurde $c = -2{,}01$ gewählt. Obwohl dieser Wert nur geringfügig unter -2 liegt, verlässt die Zahlenfolge den Zahlenraum $|x_n| < 2$ nach wenigen Iterationsschritten. Daher sind alle Werte $c < -2$ ebenfalls keine Elemente der Mandelbrot-Menge.

In der letzten Grafik wurde $c = -1{,}7$ gewählt. Wie man sieht, handelt es sich dabei um eine chaotische Zahlenfolge, welche jedoch beschränkt bleibt. Das Bild zeigt etwa 20 Iterationsschritte.

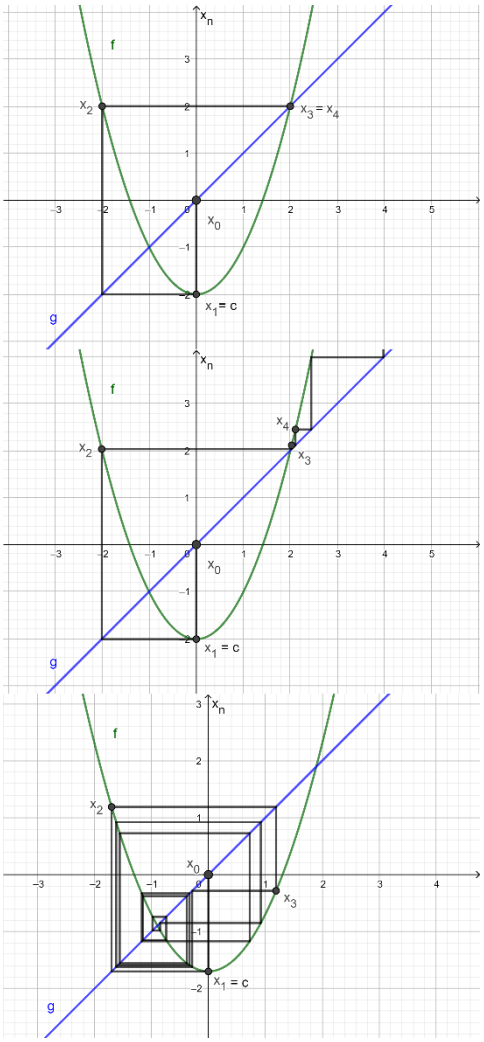


Abb. 6: Grafische Darstellung der Zahlenfolgen verschiedener Punkte (Eigene Darstellung)

### 2.4.3 Häufungspunkte und die logistische Gleichung

Die logistische Gleichung ist ein Sonderfall einer Gleichung zur Beschreibung des Populationswachstums in einem abgeschlossenen System, welche ursprünglich von dem

Mathematiker Pierre-François Verhulst 1845 erstellt wurde. Erst viel später fiel weiteren Mathematikern das chaotische Verhalten der Gleichung auf. Die mathematische Schreibweise der logistischen Gleichung ist

$$x_{n+1} = r \cdot x_n \cdot (1 - x_n) = r \cdot x_n - r \cdot x_n^2$$

Dabei nimmt $x_0$ einen beliebigen Wert im Interaval $[0; 1]$ an. Der Wert $r$ wird am Anfang der Berechnung festgelegt, und ist ein Element der reellen Zahlen. Für alle $0 < r \leq 3$ konvergiert die Gleichung zu genau einem Wert, entweder die Population stirbt aus ($x = 0$) oder bleibt konstant ($x = 1 - \frac{1}{r}$). Jene Zahlenwerte, welchen sich die Zahlenfolge nach vielen Iterationsschritten annähert, werden Fixpunkte genannt. Im Intervall $3 < r \leq 1 + \sqrt{6}$ wechselt die Zahlenfolge nach jeder Iteration zwischen zwei Fixpunkten hin und her. Für $r > 1 + \sqrt{6}$ wechselt die Zahlenfolge zwischen vier Werten und für noch größere Werte für $r$ verhält sich die Zahlenfolge chaotisch. Um dieses Chaos zu veranschaulichen, wird ein sogenanntes Feigenbaum-Diagramm der logistischen Gleichung erstellt. Dabei werden zu den Werten für $r$ auf der x-Achse die dazugehörigen Fixpunkte auf der y-Achse eingezeichnet [8].

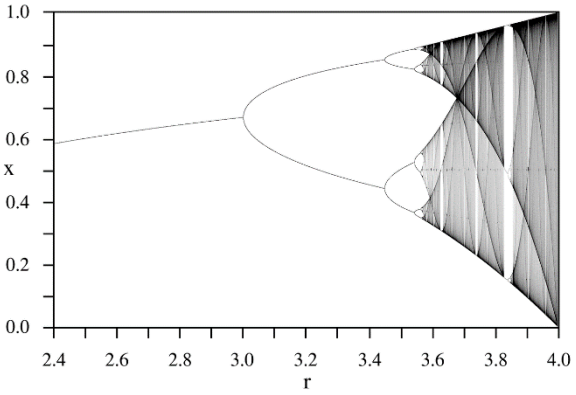


Abb. 7: Die logistische Gleichung als Feigenbaumdiagramm (PAR: A bifurcation diagram for the Logistic map, 2005. Wikipedia)

Ähnlich wie die logistische Gleichung, ist auch die Formel der Mandelbrot-Menge quadratisch. Feigenbaumdiagramme können zu beliebigen Gleichungen erstellt werden, welche ein Iterationsverfahren verwenden. In der Nebenstehenden Grafik ist über

[8] (Herrmann, 1996, S. 30-32)

der Mandelbrot-Menge das Feigenbaum-Diagramm der Mandelbrot-Menge eingezeichnet. Sieht man sich das Verhalten der Punkte der Mandelbrot-Menge auf der x-Achse an, so kann man folgende Aussage über die ersten Intervalle machen

- $-0{,}75 < c_r < 0{,}25$ konvergiert gegen einen Wert
- $-1{,}25 < c_r < -0{,}75$ konvergiert gegen zwei Werte

Für noch kleinere $c_r$, also $c_r < -1{,}25$, erfolgt zunächst eine weitere Periodenverdoppelung (Verdoppelung der Fixpunkte) und schließlich der Übergang zum chaotischen Verhalten.

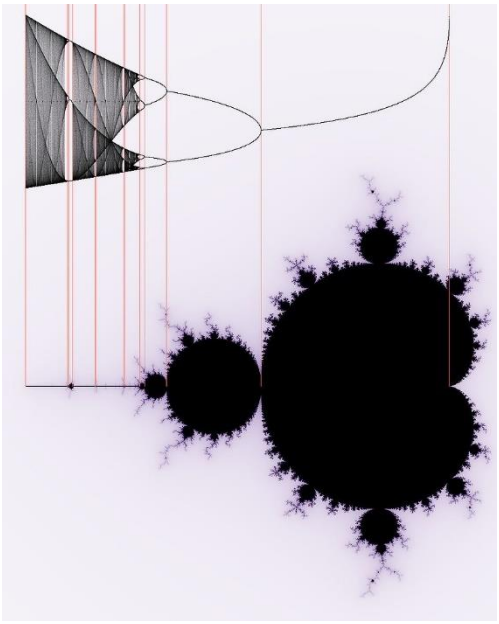

Abb. 8: Das Feigenbaumdiagramm der Mandelbrot-Menge (Georg-Johann Lay: Diagram showing the connexion between Verhulst dynamic and Mandelbrot set, 2008. Wikipedia)

### 2.4.4 Unbestimmbare Randpunkte

Obwohl die Mandelbrot-Menge abgeschlossen ist, das bedeutet sie hat einen eindeutigen Rand, ist es unmöglich, bis auf einzelne Ausnahmen, Randpunkte der Mandelbrot-Menge exakt anzugeben. Grund dafür ist die unendliche Komplexität. Es scheint zwar eindeutig, die Randpunkte auf einem stehenden Bild auszumachen, jedoch wandert, sobald der Bildausschnitt vergrößert wird, der zuvor ausgewählte Punkt weg vom Rand. Dies ist vergleichbar mit einer Küste: wählt man auf einer Satellitenaufnahme einen Punkt auf der Küste aus, scheint der Punkt genau auf dem Rand zwischen Wasser und Land zu liegen. Sobald man jedoch diesen Punkt zu Fuß aufsucht, erkennt man, dass man, wenn auch nur wenige Meter, daneben lag. Selbst wenn man auf dem Strand steht, kann man den Rand nicht genau bestimmen, da die Grenze zwischen Landmasse und Ozean bis auf die atomare Ebene nicht genau ausmachbar ist. Eine Ausnahme bildet der Punkt $c = -2 + 0i$. Wie bereits bewiesen, ist der Punkt ein Teil der Mandelbrot-Menge. Außerdem ist dieses $c$ eindeutig ein Randpunkt, da daneben-liegende Punkte mit einem kleineren $c_r$ den Schwellenwert $|c| > 2$ bereits in der ersten Iteration überschreiten würden.

# 3 Die Julia-Menge

## 3.1 Gaston Julia

Gaston Julia hat als Erster die Mathematik von Fraktalen beschrieben, wie wir sie heute kennen. Sein Werk ist die Grundlage für etliche weitere Arbeiten, und auch für Benoît Mandelbrots Erfolg. Bereits in jungen Jahren zeigte der Franzose Interesse für Mathematik. Nach seinem Fronteinsatz im 1. Weltkrieg, widmete er sich intensiv seinen Forschungen. Schlussendlich publizierte er 1918 einen 200 Seiten langen Artikel „*Mémoire sur l'itération des fonctions rationnelles*“ („*Diplomarbeit über die Iteration rationaler Funktionen*“), welcher in einem Journal für angewandte Mathematik publiziert wurde. Dieses Werk machte ihn für kurze Zeit berühmt, und verschaffte ihm sogar einen Preis, den „*Grand Prix des Sciences Mathématiques*“. Sein Werk war jedoch nach kurzer Zeit wieder vergessen, bis Benoît Mandelbrot ihn in seinem Lebenswerk „The fractal Geometry of Nature“ erwähnte. Gaston Julia schaffte es ohne jegliche Hilfe von Computern, die Komplexität der Julia-Menge zu verstehen.

## 3.2 Mathematik

Bei der Julia-Menge gilt dieselbe Iterationsformel, wie bei der Mandelbrot Menge.

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

Allerdings fängt die Rechnung nicht mit $z_0 = 0$ an, sondern mit dem Wert des untersuchten Punktes in der Ebene, das bedeutet $z_0 \in \mathbb{C}$. Der Wert $c$ wird für die Berechnung einer Julia-Menge konstant gehalten und vor der Rechnung festgelegt. Nun wird durch Variieren des Startwertes $z_0$ untersucht, für welche Startwerte $z_0$ die Zahlenfolge gemäß obiger Iterationsformel beschränkt bleibt. Die Julia-Menge für ein bestimmtes $c$ ist der Rand der Menge aller Startwerte $z_0$, für welche die Zahlenfolge gemäß obiger Iterationsformel beschränkt bleibt. Durch das Variieren des Wertes $c$ erhält man verschiedene Julia-Mengen [9].

[9] (Bräuner, 2002, S. 60)

### 3.2.1 Eigenschaften

Obwohl die Iterationsformel der Julia-Menge sehr ähnlich der Mandelbrot-Menge ist, haben die beiden Fraktale andere Eigenschaften. Als wichtigste Eigenschaft gilt, dass die Julia-Menge ein Fraktal ist, außer bei zwei bestimmten Werten für $c$. Grund dafür ist, dass die Julia-Menge für $c = 0 + 0 \cdot i$ der Kreis mit dem Radius eins und dem Mittelpunkt $(0|0)$, also der Einheitskreis, ist. Für $c = -2 + 0 \cdot i$, wiederum, erhält man als Julia-Menge die Strecke mit dem Startpunkt $(-2|0)$ und dem Endpunkt $(2|0)$. Bei manchen Julia-Mengen erscheint nicht eine abgeschlossene Kurve, sondern eine Wolke von unendlich vielen Inseln, der sogenannte Fatou-Staub. Diese Inseln sehen ident aus, was auf die Selbstähnlichkeit zurückzuführen ist. Untersucht man nun, ob die Julia-Menge für ein bestimmtes $c$ eine einzige Kurve ist, oder nicht, so kann man auch eine Aussage über den Punkt in der Mandelbrot-Menge mit den Koordinaten von $c$ treffen.

> „Die Mandelbrotmenge $\boldsymbol{M}$ ist die Menge aller Punkte $c \in \mathbb{C}$, für die die Julia-Menge zusammenhängend ist.“ (Bräuner, 2002, S. 62)

Anders als das Mandelbrot-Fraktal, ist die Julia-Menge punktsymmetrisch bezüglich des Ursprungs $(0|0)$ [10]. So wie die Mandelbrot-Menge ist auch jede Julia-Menge kompakt. Das bedeutet, dass sich die Menge in einem bestimmten Zahlenraum befindet und einen eindeutigen Rand hat. Allerdings ist der Zahlenraum, in welchem sich die Julia-Menge befindet, für jedes $c$ individuell [11].

# 4 Programmcode

Der folgende Teil der Arbeit widmet sich der Beschreibung eines Programmes, mit welchem es möglich ist, die Mandelbrot-Menge anhand des „Escape-Time Algorithmus“ zu visualisieren. Es gibt weit effizientere Algorithmen, welche eine Grafik der Mandelbrot-Menge erstellen [12]. Der „Escape-Time Algorithmus“ wurde gewählt, da er sich für JavaScript gut eignet. Als Programmsprache wird JavaScript verwendet, aufgrund der

---

[10] (Zeitler & Neidhardt, 1993, S. 200)

[11] (Herrmann, 1996, S. 208)

[12] (Herrmann, 1996, S. 214,229)

einfachen Programmierweise. Des Weiteren ist es mithilfe von HTML am Ende sehr leicht das berechnete Bild abzurufen. Allerdings gibt es andere Programmiersprachen, welche JavaScript in diesem mathematischen Gebiet überlegen sind. JavaScript ist langsamer in generellen mathematischen Berechnungen als zum Beispiel Java. Beim Mandelbrot-Fraktal müssen mehrere Millionen Rechenschritte pro Bild gemacht werden, daher wird bei dem Programmcode auf einen optimierten Algorithmus geachtet. Ein weiteres Problem ist die Rechengenauigkeit. JavaScript verwendet für das Speichern und Rechnen von Zahlen 64-Bit. Das bedeutet, dass es eine größte, sowie eine kleinste Zahl, und auch einen kleinsten Schritt zwischen zwei Zahlen gibt. Daher wird beim Rechnen ab einer bestimmten Anzahl an Dezimalstellen gerundet, was das Ergebnis minimal verfälscht. Allerdings ist dieser Rundungsfehler nicht groß genug, um in der Grafik einen bemerkbaren Unterschied zu machen. Dennoch bedeutet es, dass das Bild ab einer bestimmten Vergrößerung der Mandelbrot-Menge nicht weiter vergrößert werden kann. Dies tritt etwa ab der $5 \cdot 10^{14}$-fachen Vergrößerung auf. Zu guter Letzt kann JavaScript nicht mit imaginären Zahlen rechnen. Eine Alternative wäre, das Programm mit Python zu erstellen, da Python das Rechnen mit imaginären Zahlen unterstützt. Dies ist jedoch kein großes Problem, da die Iterationsformel umgeformt werden kann, um die Rechenschritte mit reellen Zahlen durchzuführen.

## 4.1 Die Iterationsformel

Der erste Schritt der Umformung von $z_{n+1} = z_n^2 + c$ ist es $z_n^2$ auszurechnen. Da $z_n = z_r + i \cdot z_i$ gilt, kann die binomische Formel

$$(z_r + i \cdot z_i)^2 = z_r^2 + 2 \cdot i \cdot z_r \cdot z_i + i^2 \cdot z_i^2$$

aufgestellt werden. Da $i^2 = -1$ bekannt ist, kann die Formel weiter auf

$$z_r^2 - z_i^2 + 2 \cdot i \cdot z_r \cdot z_i$$

vereinfacht werden. Anschließend wird $c = c_r + i \cdot c_i$ addiert, mit dem Ergebnis

$$z_{n+1} = z_r^2 - z_i^2 + 2 \cdot i \cdot z_r \cdot z_i + c_r + i \cdot c_i$$

Nach dem Vereinfachen auf

$$z_{n+1} = z_r^2 - z_i^2 + c_r + i \cdot (2 \cdot z_r \cdot z_i + c_i)$$

ist beim Ergebnis wieder klar der Realteil ($z_{n+1_r} = z_r^2 - z_i^2 + c_r$) und der Imaginärteil ($z_{n+1_i} = 2 \cdot z_r \cdot z_i + c_i$) von $z_{n+1}$ erkennbar. Diese beiden Teile können problemlos mit reellen Zahlenwerten im Programm berechnet werden. Nach der Berechnung muss überprüft werden, ob $|z_{n+1}|$ größer als 2 ist, da die Zahlenfolge dann mit Sicherheit gegen unendlich geht. Dies wird überprüft mithilfe der zuvor ausgerechneten Teile $z_{n+1_r}$ und $z_{n+1_i}$ durch die Formel $|z_{n+1}| = \sqrt{{z_{n+1_r}}^2 + {z_{n+1_i}}^2}$. Um sich die enorme Rechenleistung einer Quadratwurzel, im Vergleich zu Multiplikationen und anderen Rechenoperationen, zu sparen, wird sie im Programmcode weggelassen. Daher muss überprüft werden, ob ${z_{n+1_r}}^2 + {z_{n+1_i}}^2 > 4$. Im Programmcode wird für den ganzen Vorgang die folgende Funktion mit den Parametern $c_r$, $c_i$ und $n_{max}$ verwendet:

```
function calcPixel(cr, ci, nmax) {
    let zr = 0;
    let zi = 0;
    for (let n = 0; n < nmax; i++) {
        let zold = zr;
        zr = zold * zold - zi * zi + cr;
        zi = 2.0 * zold * zi + ci;
        if (zr * zr + zi * zi > 4.0) {
            return n;
        }
    }
    return nmax;
}
```

Das Ziel der Funktion ist es, für $c = c_r + i \cdot c_i$ festzustellen, ob es zur Mandelbrot-Menge gehört, oder nicht. Nach dem Belegen der Werte $z_r$ und $z_i$ mit dem Wert 0 wird eine Schleife gestartet, welche sich selbst höchstens $n_{max}$-Mal wiederholt. Die Begrenzung durch $n_{max}$ verhindert den Absturz des Programmes durch eine Endlosschleife. In der Schleife wird die oben ausgerechnete Iterationsformel verwendet. Da $z_r$ in der Berechnung von beiden Teilen von $z_{n+1}$ verwendet wird, und das Programm nicht zwei Rechnungen gleichzeitig machen kann, wird der Wert von $z_r$ in der Zwischenvariable $z_{old}$ gespeichert. Am Ende jeder Iteration wird überprüft, ob bereits der Schwellenwert von 4 überschritten wurde. Ist dies der Fall, wird die Funktion abgebrochen, und man erhält die Anzahl $n$ der bis zum Abbruch durchgeführten Iterationen für das $c$, welches untersucht wurde. Wird der Schwellenwert nicht erreicht, erhält man ebenfalls die Anzahl der durchgeführten Iterationen, in diesem Fall ist das $n_{max}$. Wenn

der Rückgabewert der maximalen Anzahl der Iterationen $n_{max}$ entspricht, ist das untersuchte $c$ ein Element der Mandelbrot-Menge. Allerdings kann sich dieses Urteil bei Erhöhen von $n_{max}$ verändern. Falls $n_{max}$ nicht erreicht wird, ist die Anzahl der Iterationen $n$ aussagekräftig darüber, nach wie vielen Iterationen der Punkt $c$ divergierte und daher eindeutig kein Element der Mandelbrot-Menge ist.

## 4.2 Darstellung eines Bildes

Um mithilfe der oben erklärten Funktion $calcPixel$ ein Bild zu erstellen, muss sie für jeden Pixel mit den richtigen Werten für $c_r$ und $c_i$ aufgerufen werden. Um diese Werte zu erhalten, sind einige Parameter nötig. Vor allem ist es wichtig, den abzubildenden Ausschnitt der Mandelbrot-Menge anzugeben. Dabei gibt es zwei Möglichkeiten. Erstens kann man zwei Eckpunkte des Bildausschnitts definieren, zum Beispiel die obere linke und die untere rechte Ecke. Damit ist von jedem Pixel die Koordinate eindeutig berechenbar.

Die andere Möglichkeit besteht darin, nur einen Punkt im Bild anzugeben, wobei sich dafür der Mittelpunkt des Bildes eignet. Allerdings muss in diesem Fall auch noch eine Skalierung vorgegeben werden, um die Koordinaten der zu berechnenden Punkte zu definieren.

Zu guter Letzt ist die Größe des Bildes bei beiden Methoden beliebig. Bei der ersten Methode bleibt der Bildausschnitt gleich, wenn die Auflösung erhöht wird. Bei der zweiten Methode vergrößert sich der sichtbare Bildausschnitt mit steigender Auflösung. Da die zweite Methode leichter lesbar ist, wird sie in dem Programm verwendet.

```
let midCr = -1;
let midCi = 0;
let scale = 1;
let nmax = 250;
```

In obigem Quelltext speichern die Variablen $midCr$ und $midCi$ den Bildmittelpunkt $c = -1 + i \cdot 0$. Für die Skalierung wird die Formel

$$Abstand\ zwischen\ benachbarten\ Pixeln\ in\ der\ Zahlenebene = \frac{0{,}01}{Skalierungsfaktor}$$

verwendet. Im Programmcode wird für den Skalierungsfaktor die Variable $scale$ verwendet. Bei einem Skalierungsfaktor von 1 entsprechen 100 Pixel einer Einheit in der Zahlenebene, bei einem Skalierungsfaktor von 10 wird eine Einheit in der Zahlenebene mit 1000 Pixeln aufgelöst. Eine weitere wichtige Variable ist die schon weiter oben definierte Anzahl der maximalen Iterationsschritte $n_{max}$, welche beispielsweise mit dem Wert 250 belegt wurde. Bei Erhöhung des Skalierungsfaktors $scale$ wird $n_{max}$im Normalfall ebenfalls erhöht, da sich bei den Punkten am Rand des Mandelbrot-Fraktals bei genauerem Betrachten erst später im Iterationsvorgang herausstellt, ob sie divergieren oder konvergieren. Bevor der Rechenvorgang beginnen kann, müssen noch drei Werte berechnet werden.

```
let dstBetweenPixels = 0.01 / scale;
let startCr = midCr - (canvas.width / 2) * dstBetweenPixels;
let startCi = midCi + (canvas.height / 2) * dstBetweenPixels;
```

Mit der zuvor aufgestellten Definition für den Skalierungsfaktor $scale$ wird nun die Distanz zwischen zwei Pixeln in $dstBetweenPixels$ gespeichert. Um den Rechenvorgang leichter zu machen, wird mithilfe der Bildbreite in Pixeln ($canvas.width$) und der Bildhöhe in Pixeln ($canvas.height$) berechnet, welche Koordinate die obere linke Ecke des Bildes hat. Die obere linke Ecke des Bildes hat die Koordinaten $c = startC_r + i \cdot startC_i$. Bei einer Auflösung mit $scale = 1$ und einer Bildgröße von 300x300 Pixeln ist es in diesem Fall der Punkt $c = -2{,}5 + i \cdot 1{,}5$.

Nun wird eine Schleife gestartet, welche oben links anfangend, jeden Punkt des Bildes überprüft.

```
for (let y = 0; y < canvas.height; y++) {
    for (let x = 0; x < canvas.width; x++) {
        let cr = startCr + x * dstBetweenPixels;
        let ci = startCi - y * dstBetweenPixels;
        let iter = calcPixel(cr, ci, nmax);
        if (iter == nmax) {
            ctx.fillStyle = "#000000";
        } else {
            ctx.fillStyle = "#ffffff";
        }
        ctx.fillRect(x, y, 1, 1);
}   }
```

In dieser Schleife werden $x$ und $y$ die jeweiligen Koordinaten des Punktes, welcher untersucht wird, in Pixel zugewiesen. Mithilfe der zuvor ausgerechneten Werte können nun durch die Formeln

$$c_r = Relle\ Koordinate\ des\ Eckpunktes\ (startC_r) + Abstand\ zum\ linken\ Bildrand\ in\ Pixel\ (x) \cdot Distanz\ zwischen\ zwei\ Pixeln\ (dstBetweenPixels)$$

und

$$c_i = Imaginäre\ Koordinate\ des\ Eckpunktes\ (startC_i) - Abstand\ zum\ oberen\ Bildrand\ in\ Pixel\ (y) \cdot Distanz\ zwischen\ zwei\ Pixeln\ (dstBetweenPixels)$$

Real- und Imaginärteil des momentan untersuchten $c$ berechnet werden. Anschließend werden diese Koordinaten in der Funktion $calcPixel$ verwendet. Der Rückgabewert $iter$, welcher zwischen 0 und $n_{max}$ liegt, ist aussagekräftig über die Zugehörigkeit von $c$ zur Mandelbrot-Menge. Um dies darzustellen, wird überprüft, ob der Rückgabewert $n_{max}$ entspricht. Ist dies so, wird der Pixel schwarz gefärbt, andernfalls wird er weiß gefärbt. Lässt man dieses Programm ablaufen, so entsteht die nebenstehende Grafik. Durch Verändern der Variablen $midCr, midCi, scale$ und $canvas.width$, sowie $canvas.height$ kann leicht ein anderer Bildausschnitt aus der Mandelbrot-Menge erzeugt werden.

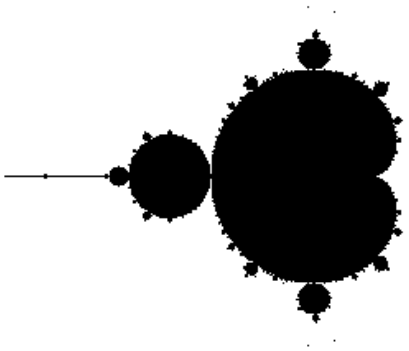

Abb. 9: Ausgabe des beschriebenen Programmes, die Mandelbrot-Menge (Eigene Darstellung)

## 4.3 Einfärbung

Bisher kann das Programm nur Schwarz-Weiß Bilder erstellen. Um die Grafik ansprechender zu gestalten, kann mithilfe des Rückgabewertes $iter$ eine Einfärbung geschehen. Vor allem bei näherem Ansehen des Mandelbrot-Fraktals ist die zweifärbige Darstellung nicht mehr ausreichend, um den fraktalen Rand zu veranschaulichen. Eine einfache Methode dieses Problem zu verbessern, ist es den Rückgabewert, also die Anzahl

der durchgeführten Iterationsschritte, in den drei Farbkanälen, Rot, Grün und Blau, als Farbwert zu verwenden. Da die Farbkanäle Zahlenwerte von 0-255 verwenden, muss zuerst der Rückgabewert umgerechnet werden. Dies wird berechnet mithilfe der Formel

$$Farbwert = \left(1 - \frac{Rückgabewert}{n_{max}}\right) \cdot 255$$

Der berechnete Farbwert liegt im Intervall $[0{,}255]$. Dabei gilt, dass je früher ein $c$ den Schwellenwert überschreitet, desto heller wird der Pixel gezeichnet. Es ergibt sich folgender Programmcode.

```
    let iter = calcPixel(cr, ci, nmax);
    let color = (1 - iter / nmax) * 255;
    ctx.fillStyle = `rgb(${color},${color},${color})`;
```

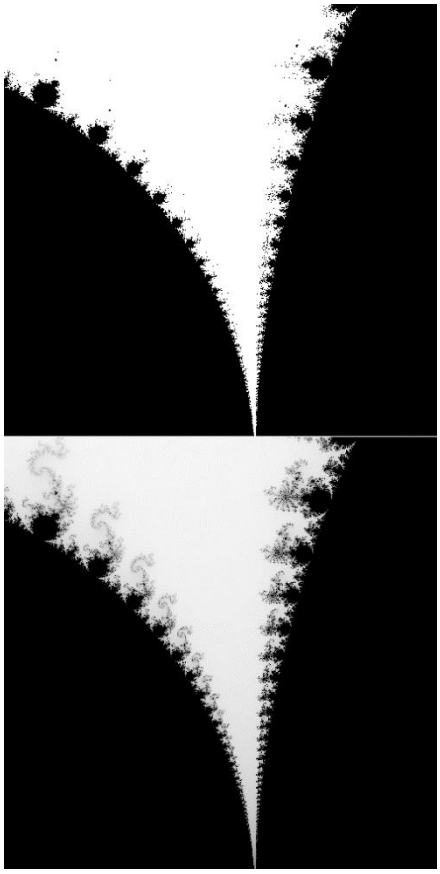

Abb. 10: Vergleich zweier Farbalgorithmen (Eigene Darstellung)

Die Variable $color$ speichert den ausgerechneten Farbwert. Dieser wird gleichgesetzt für alle Farbkanäle des Pixels, resultierend in einem Grauton. Die folgenden beiden Grafiken zeigen den Unterschied der beiden Einfärbungsmethoden an der Stelle $c = -0{,}77 + 0.15 \cdot i$ mit dem Skalierungsfaktor 20.

Es ist deutlich erkennbar, dass die untere Grafik, welche den neuen Farbalgorithmus verwendet, mehr Information enthält. Bei der oberen reinen Schwarz-Weiß-Grafik kann man nicht-verbundene Inseln erkennen. Da die Mandelbrot-Menge zusammenhängend ist, gibt es diese nicht. In Grafik 2 werden diese Inseln durch Arme verbunden, was durch das Hinzufügen der Graustufen ermöglicht wurde. Dies ist eine Verbesserung der Darstellung, allerdings war die Grafik bisher nur Monochron. Durch eine Aufspaltung der Farbkanäle in Rot, Grün und Blau, können noch komplexere Farbpaletten erzeugt werden. Beispielsweise setzt folgender Algorithmus den grünen Farbwert immer auf 0 und färbt jedes $c$, welches Teil

der Mandelbrot-Menge ist schwarz ein. Außerdem werden die Zahlenwerte des roten und blauen Farbkanals invertiert.

```
    let iter = calcPixel(cr, ci, nmax);
    let r = (iter / nmax) * 255;
    let g = 0;
    let b = (iter / nmax) * 255;
    ctx.fillStyle = `rgb(${r},${g},${b})`;
    if (iter == nmax) ctx.fillStyle = "#000000";
```

Dadurch entsteht das nebenstehende Bild. Die Einfärbung des Mandelbrot-Fraktales ist nicht universell. Meist wird eine Farbpalette mit vielen verschiedenen Farben verwendet, um die verschiedenen Rückgabewerte, also die Anzahl der durchgeführten Iterationsschritte, leichter sichtbar zu machen. Im Rahmen dieser Arbeit werden die meisten Grafiken mit dem Graustufen-Algorithmus gezeichnet, um den Unterschied zwischen Punkten, welche Teil der Mandelbrot-Menge sind, und jenen, welche außerhalb der Mandelbrot-Menge liegen, eindeutiger zu machen.

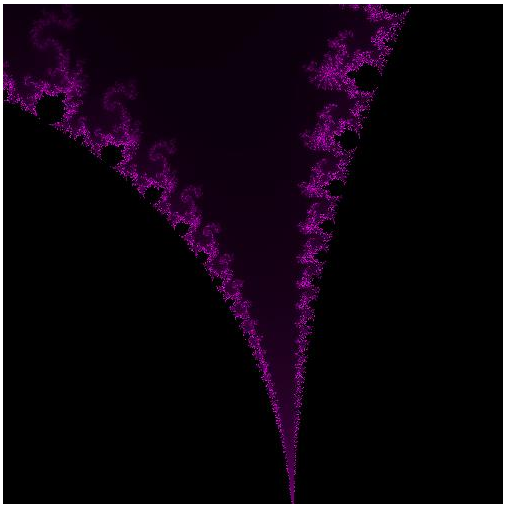

Abb. 11: Violette Einfärbung der Mandelbrot-Menge (Eigene Darstellung)

## 4.4 Bedeutung von $n_{max}$

Bei der Darstellung eines Ausschnittes der Mandelbrot-Menge ist es wichtig $n_{max}$ entsprechend zu wählen. Allerdings ist es nicht immer vor der Berechnung eindeutig, welches $n_{max}$ ausreichend ist. Um dies zu demonstrieren, wurde die untenstehende Abbildung erstellt. Dabei wurde ein alternativer Einfärbungsalgorithmus gewählt, welcher die Bildpunkte unabhängig von $n_{max}$ in Graustufen darstellt. Für drei verschiedene Skalierungsfaktoren wurden jeweils drei Werte für $n_{max}$ gewählt. Dabei wurde für die Bilder der ersten Zeile der Bildmittelpunkt $c = -0{,}75 + i \cdot 0$ gewählt, $canvas.width$ und $canvas.height$ betrugen immer 300 Pixel. Es ist die Grundform des Mandelbrots bereits mit nur maximal 16 Iterationsschritten erkennbar. Trotzdem ist der Umriss des

Fraktals erst ungenau dargestellt. Bei einer Erhöhung auf maximal 256 Iterationsschritte verbessert sich die Qualität merklich. Darüber hinaus kommt es zu keiner weiteren Verbesserung. In der zweiten Zeile ist der Skalierungsfaktor wesentlich höher, der Bildmittelpunkt befindet sich bei $c = -0{,}70189 + i \cdot 0{,}26823$. Hier ist $n_{max} = 16$ nicht mehr ausreichend für einen Bildaufbau, da kein Punkt im Bild vor 16 Iterationsschritten divergiert. Bei einem Maximum von 256 Iterationsschritten ist bereits eine Struktur erkennbar. Um eine genauere Darstellung zu erhalten, muss $n_{max}$ weiter erhöht werden. Wird $n_{max}$ auf 4096 erhöht, so kann man erkennen, dass alle Punkte, welche davor noch nicht divergierten, nun hellgrau dargestellt und daher tatsächlich kein Element der Mandelbrot-Menge sind. Tatsächlich gibt es dünne spiralförmige Linien, deren Punkte Teil der Mandelbrot-Menge sind. Diese sind zu dünn, um in der Grafik als schwarze Punkte sichtbar zu sein. Ihre Position kann aufgrund der Graustufen angenommen werden. In der letzten Zeile wiederholen sich die Eigenschaften der Bilder aus der zweiten Zeile. Dafür wurde der Bildmittelpunkt $c = -0.7663958119083875 + i \cdot 0.10108912048774574$ gewählt. Obwohl der Skalierungsfaktor um sieben Größenordnungen größer ist, wird mit $n_{max} = 256$ bereits eine zufriedenstellende Bildqualität erreicht.

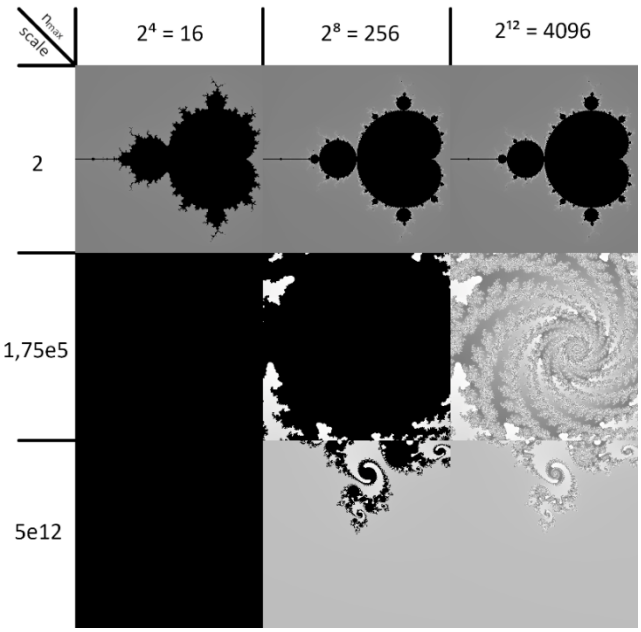


Abb. 12: Zusammenhang zwischen dem Skalierungsfaktor und $n_{max}$ (Eigene Darstellung)

Aufgrund der soeben festgestellten Eigenschaften kann erschlossen werden, dass tendenziell mit steigendem Skalierungsfaktor ein größeres $n_{max}$ benötigt wird. Allerdings ist $n_{max}$ nicht nur vom Skalierungsfaktor abhängig, sondern auch vom gewählten Bildmittelpunkt. Sind die schwarzfärbigen Bereiche (Zahlenfolge divergiert nicht bis $n_{max}$) scharfrandig begrenzt, so wurde für den betrachteten Bereich $n_{max}$ zu niedrig gewählt. $n_{max}$ ist meist erst dann hoch genug, wenn in den Randbereichen chaotisches Verhalten durch starken Farbwechsel bei geringem Abstand der Punkte erkennbar ist.

## 4.5 Beliebige Genauigkeit

Wie bereits erwähnt wurde, kann aufgrund der Rechengenauigkeit in JavaScript der Bildausschnitt nicht beliebig vergrößert werden. Mit dem bisherigen Programmcode wird an der Stelle $c = -0.7663958119083875 + i \cdot 0.10108912048774574$ als Bildmittelpunkt, mit dem Skalierungsfaktor $5 \cdot 10^{14}$, $n_{max} = 256$ und einer Bildauflösung von 200x200, das nebenstehende Bild berechnet. Wie deutlich erkennbar ist, sind die einzelnen Pixel gestreckt. Dies liegt an der begrenzten Rechengenauigkeit von Zahlenvariablen in JavaScript. Daher hat es keinen Sinn den Skalierungsfaktor über $5 \cdot 10^{14}$ zu erhöhen.

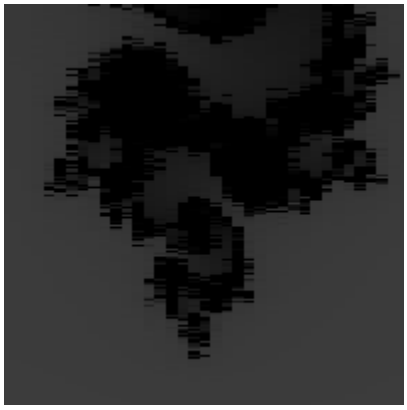

Abb. 13: Die Limitierungen JavaScripts bezüglich der Rechengenauigkeit (Eigene Darstellung)

Mithilfe der Bibliothek $decimal.js$ kann dieses Problem auf Kosten von Rechenzeit gelöst werden. Dabei muss der gesamte JavaScript-Programmcode auf das Format der Bibliothek umgeschrieben werden. Da die Bibliothek eine Rechengenauigkeit verlangt, muss diese als Variable festgelegt werden. Die Rechengenauigkeit gibt an, auf wie viele signifikante Stellen eine Rechenoperation durchgeführt wird.

Bei demselben Bildausschnitt im Mandelbrot-Fraktal wie oben erhält man bei einer Rechengenauigkeit von $20$ signifikanten Stellen bereits ein viel klareres Bild, wobei der Bildmittelpunkt, der Skalierungsfaktor und $n_{max}$ unverändert geblieben sind. Allerdings ist diese zusätzliche Genauigkeit nur bei sehr großen Skalierungsfaktoren (ab etwa $10^{14}$) notwendig. Ähnlich wie bei $n_{max}$ muss die Anzahl der signifikanten Stellen bei Vergrößern des Skalierungsfaktors erhöht werden. Allerdings hat diese Methode ein großes Problem: die Rechendauer. Während das Programm ohne Verwen-

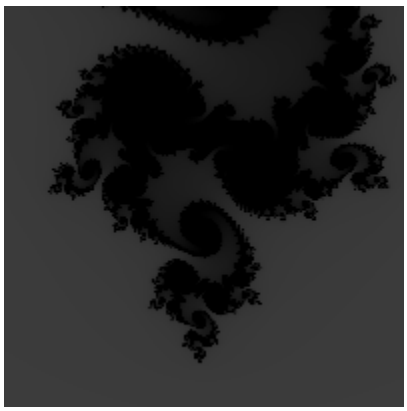

Abb. 14: Verbesserte Rechengenauigkeit (Eigene Darstellung)

dung der Bibliothek $decimal.js$ etwa $0{,}15$ Sekunden für die Berechnung der Grafik benötigt, dauert der Vorgang mit erweiterter Rechengenauigkeit bei 20 signifikanten Stellen $71{,}7$ Sekunden. Da die Rechendauer fast das 500-Fache beträgt, wird dieser Genauigkeitsmodus nur verwendet, sobald JavaScript ohne $decimal.js$ seine Grenzen erreicht. Durch Ausprobieren hat sich herausgestellt, dass bei einer 10-Fachen Vergrößerung des Skalierungsfaktors die Anzahl der signifikanten Stellen um 1 erhöht werden muss. Es ergibt sich die Näherungsformel

$$Mindestens\ notwendige\ Anzahl\ signifikanter\ Stellen = \log_{10}(scale) + 4$$

## 4.6 Algorithmus zum progressiven Bildaufbau

Um das generierte Bild der Mandelbrot-Menge leicht erkundbar zu machen, kann das Programm mithilfe der Maus interaktiv gemacht werden. Nimmt man die Bewegung der Maus auf, so kann man den Bildmittelpunkt entsprechend verschieben. Durch erneutes Zeichnen des Bildes mit dem neuen Bildmittelpunkt ermöglicht dies dem Benutzer, den Bildausschnitt in der Ebene Stück für Stück zu verschieben. Des Weiteren kann das Mausrad verwendet werden, um den Skalierungsfaktor zu verändern. Mithilfe dieser beiden Navigationsmöglichkeiten ist es möglich, das gesamte Mandelbrot-Fraktal mit der Maus zu erkunden. Dabei ist es wichtig, dass der Bildaufbau eine sehr kurze Zeit dauert, da sonst das Navigieren erschwert wird.

Um die Rechendauer zu verringern, wird das Zeichnen der Grafik in mehrere Schritte zerlegt. Anfangs werden der Skalierungsfaktor und die Bildauflösung sehr klein gewählt, sodass der Bildausschnitt gleichbleibt. Dadurch erhält man ein verpixeltes Bild. Da der Rechenaufwand nur ein Bruchteil des Rechenaufwandes für das ganze Bild ist, kann der Benutzer so problemlos das Mandelbrot-Fraktal erkunden. Entschließt sich der Benutzer auf einem Punkt stehen zu bleiben, so wird die Bildauflösung, sowie der

Skalierungsfaktor verdoppelt. Anschließend wird das Bild neu berechnet, wobei alle Punkte, welche bereits berechnet wurden, übersprungen werden. Dieser Verdopplungsprozess wird so lange fortgesetzt, bis die gewünschte Bildauflösung erreicht ist. In den nebenstehenden Grafiken ist dieser Prozess dargestellt. Dabei wurden folgende Werte verwendet: $c = -0{,}8125 + i \cdot 0{,}1875$, $n_{max} = 256$, mit einer Auflösung der Reihe nach von 1x1, 2x2, 4x4, …, 128x128, 256x256 Pixel. Der Skalierungsfaktor des letzten Bildes ist 50. In der Praxis werden die ersten 5 Bilder der Grafik berechnet, bevor die erste Darstellung am Bildschirm erfolgt. Entscheidet sich der Benutzer zu diesem Zeitpunkt dazu, den Bildausschnitt zu verändern, so bricht der Bildaufbau ab, und eine erneute Berechnung startet für den neuen Bildausschnitt.

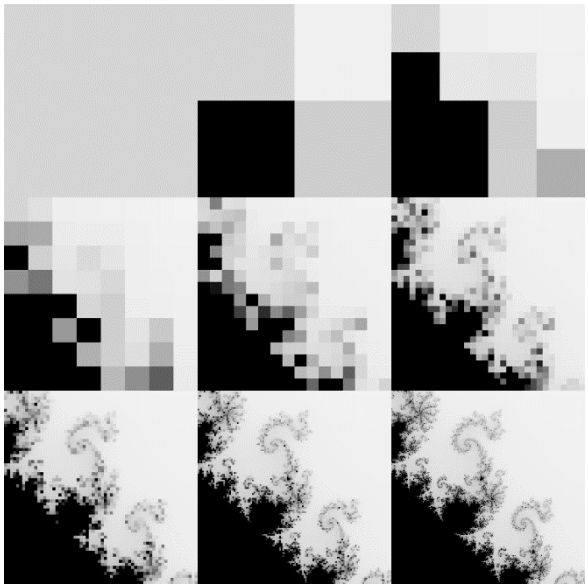

Abb. 15: Der progressive Bildaufbau (Eigene Darstellung)

## 4.7 Iterationsformel der Julia-Menge

Um eine Grafik der Julia-Menge zu erzeugen, muss das Programm erst umgeschrieben werden. Da die Iterationsformel nicht dieselbe ist, muss die Funktion $calcPixel$ verändert werden. Das abgewandelte Programm sieht folgendermaßen aus:

```
function calcPixel(z0r, z0i, cr, ci, nmax) {
    let zr = z0r;
    let zi = z0i;
    for (let i = 0; i < nmax; i++) {
        const zold = zr;
        zr = zold * zold - zi * zi + cr;
        zi = 2 * zold * zi + ci;
        if (zr * zr + zi * zi > 4) {
            return i;
        }
    }
    return nmax;
}
```

Für das Berechnen einer Julia-Menge muss vor dem Programmablauf ein konstanter Wert $c = c_r + i \cdot c_i$ gewählt werden. Dieser verändert sich weder in der Iterationsformel noch im Hauptprogramm. Im Gegensatz zur Mandelbrot-Menge wird der Startwert $z_0$ variiert, und durchläuft alle Bildpunkte. Daher benötigt die Funktion zwei weitere Parameter, $z_{0_r}$ und $z_{0_i}$. Diese belegen $z_0 = z_{0_r} + i \cdot z_{0_i}$ am Beginn der Funktion. Wird die Funktion mit den neuen Parametern für jeden Bildpunkt aufgerufen, so entsteht folgendes Bild. Wichtig bei der Julia-Menge ist es anzugeben, für welchen konstanten Wert $c$ die Julia-Menge gezeichnet wurde. Im nebenstehenden Bild wurde $c = -0{,}8 + i \cdot 0{,}2$ gewählt. Außerdem wurde $0 + i \cdot 0$ als Bildmittelpunkt, $Skalierungsfaktor = 2$, und $n_{max} = 100$, gewählt. Auf den ersten Blick ist erkennbar, dass sich die Julia-Menge im Aussehen stark vom Mandelbrot-Fraktal unterscheidet. Da die Julia-Menge in diesem Beispiel in mehrere Inseln zerfällt, kann mit Sicherheit gesagt werden, dass der Punkt $c = -0{,}8 + i \cdot 0{,}2$ kein Element der Mandelbrot-Menge ist.

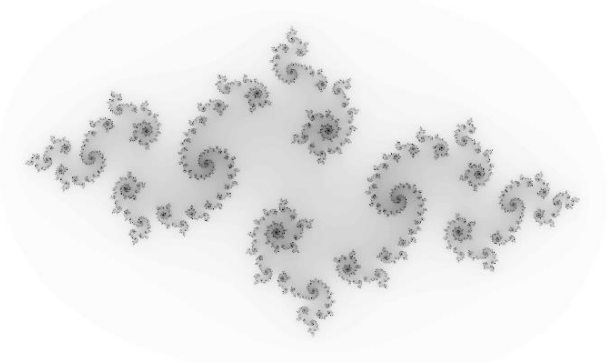

Abb. 16: Ausgabe des beschriebenen Programmes, eine Julia-Menge (Eigene Darstellung)

## 4.8 Optimierung des Algorithmus

Obwohl bereits einige Mittel verwendet wurden, um die Iterationsformel so effizient wie möglich anzuwenden, kann im Programmcode noch eine weitere Methode angewendet werden, um die Rechenzeit weiter zu kürzen. Diese Methode überprüft in der Schleife, ob ein Wert $z_n$ in der Iterationsfolge bereits einmal vorgekommen ist. Ist dies der Fall, so kann die Berechnung abgebrochen werden, da sich die Zahlenreihenfolge immer wieder wiederholen wird. Daher kann der Punkt als Element der Mandelbrot- oder Julia-Menge markiert werden. Um die Werte zu speichern, werden am Anfang der Funktion $calcPixel$ die beiden Variablen $temp_r$ und $temp_i$ reserviert. Dabei gilt $temp = temp_r + i \cdot temp_i$.

```
    let tempr = zr;
    let tempi = zi;
```

Die Variablen werden am Anfang mit den Werten von $z_0$ belegt. In der Schleife, welche die Zahlenreihenfolge berechnet, wird anschließend der folgende Programmcode ergänzt:

```
if (Math.abs(tempr - zr) < threshold && Math.abs(tempi - zi) < threshold) {
    return nmax;
}
if (i % 20 == 0) {
    memr = zr;
    memi = zi;
}
```

Nach der Berechnung von $z_n$ wird überprüft, ob das jetzige $z_n$ bereits berechnet wurde. Dafür wird mit den Werten $temp_r$ und $temp_i$ verglichen. Da sich eine Zahlenfolge fast nie exakt selbst wiederholt, wird zusätzlich am Beginn des Programms ein Schwellenwert $threshold$ festgelegt. Ist die Differenz zwischen $z_n$ und $temp$ kleiner als der Schwellenwert $threshold$, so wird die Berechnung abgebrochen, und der Punkt wird durch Rückgabe von $n_{max}$ als Element der Mandelbrot- oder Julia-Menge festgelegt. Ist dies nicht der Fall, so wird die Berechnung fortgesetzt. Zusätzlich wird $temp$ alle $20$ Iterationsschritte mit dem momentanen $z_n$ belegt.

Da die Differenz bei einem größeren Skalierungsfaktor geringer ist, muss der Schwellenwert $threshold$ vom Skalierungsfaktor abhängig gemacht werden. Es gilt: je größer

der Schwellenwert $threshold$, desto geringer ist die Rechenzeit, und desto geringer ist die Bildqualität. Mit dem Schwellenwert

$$threshold = \frac{0{,}01}{Skalierungsfaktor}$$

ist die Bildqualität praktisch unverändert. Die obere nebenstehende Grafik der Mandelbrot-Menge wurde mithilfe des neuen Algorithmus berechnet. Die nebenstehende Grafik wurde mit verschiedenen Werten für $threshold$ gezeichnet. Dabei wurde $c = -0.766 + i \cdot 0.101$ als Bildmittelpunkt festgelegt, mit einem Skalierungsfaktor von $20$, und $n_{max} = 256$. Die oberste Grafik wurde mit dem nicht optimierten Algorithmus gezeichnet, mit einer Rechendauer von etwa $640ms$. In der zweiten Grafik wurde der oben vorgeschlagene Wert für den Schwellenwert $threshold$ gewählt. Die Rechenzeit betrug nur $550ms$. Die letzte Grafik verwendet einen Schwellenwert $threshold = \frac{0{,}25}{Skalierungsfaktor}$. Dies verkürzte die Rechenzeit auf nur $475ms$, jedoch weicht das Bild deutlich von der genauen Berechnung ab.

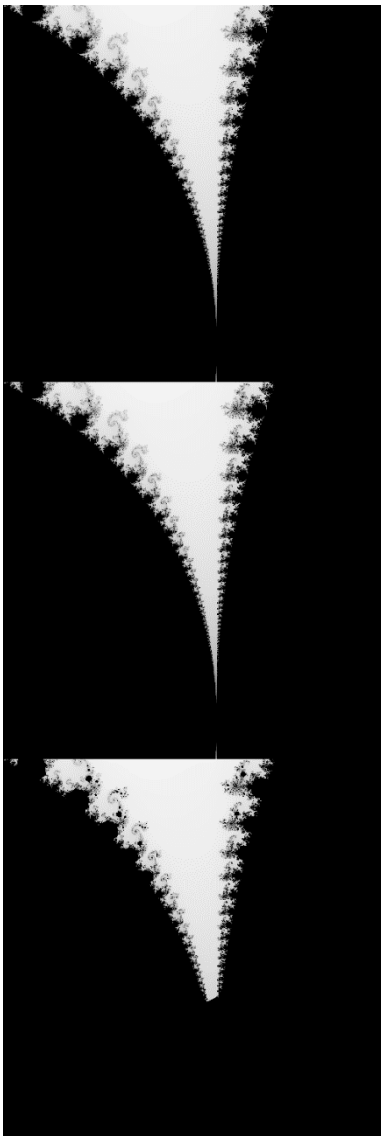

Abb. 17: Unterschiedliche Werte für $threshold$. (Eigene Darstellung)

Obwohl dieser veränderte Algorithmus effizienter erscheint, ist dies nur der Fall, wenn viele Punkte, welche eindeutig Element der Mandelbrot- oder Julia-Menge und damit schwarz eingefärbt sind, im Bild sichtbar sind. Dennoch ist es sinnvoll, den Algorithmus immer zu verwenden, da vor allem jene Punkte, welche Teil der Mandelbrot- oder Julia-Menge sind, viel Rechenaufwand beanspruchen.

# 5 Zusammenhang

Wie bereits erwähnt, ist die Julia-Menge eng mit der Mandelbrot-Menge verwandt. Aus der Form der Julia-Menge für den Punkt $c$ lässt sich ableiten, ob dieser Punkt zur Mandelbrot-Menge gehört oder nicht. Bei Julia-Mengen gibt es zwei Möglichkeiten: entweder alle Punkte der Julia-Menge sind zusammenhängend, oder es gibt unendlich viele nicht-zusammenhängende Punktgruppen („Inseln“). Ist eine Julia-Menge für ein festgelegtes $c$ zusammenhängend, so ist auch jener Punkt $c$ ein Element der Mandelbrot-Menge. Diese Aussage gilt ebenfalls in der umgekehrten Richtung. Ist ein Punkt $c$ ein Element der Mandelbrot-Menge, so ist die Julia-Menge dieses Punktes $c$ zusammenhängend.

Um diese Eigenschaft zu verdeutlichen, wurde die nebenstehende Grafik von zwei Julia-Mengen erstellt. Ihre konstanten Werte für $c$ unterscheiden sich nur minimal. Das obere Bild verwendet $c = -0{,}76 + i \cdot 0$, während das untere Bild $c = -0{,}75 + i \cdot 0{,}2$ verwendet. Der erste genannte Punkt $c = -0{,}76$ ist ein Element der Mandelbrot-Menge, während der andere Punkt $c = -0{,}75 + i \cdot 0{,}2$ kein Element der Mandelbrot-Menge ist. Dementsprechend besteht die obere Grafik aus einer einzigen Fläche, während die untere Grafik in unendlich viele Inseln zerfällt.

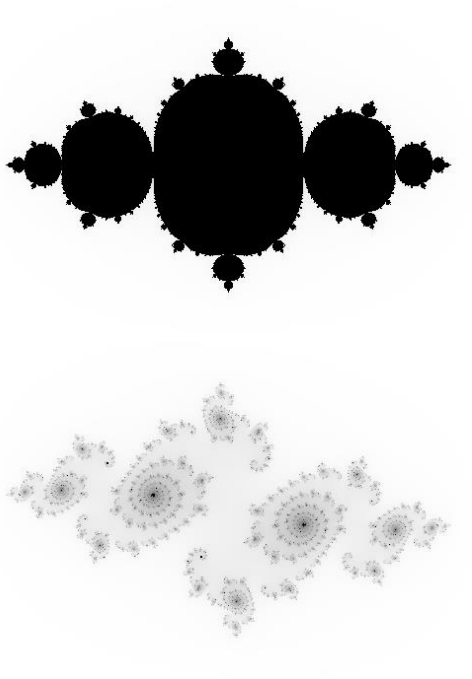

Abb. 18: Zwei Julia-Mengen, die obige zusammenhängend, die untere nicht (Eigene Darstellung)

## 5.1 Zyklisches Verhalten

Allerdings verbindet die beiden Mengen mehr als nur das. Punkte in der Ebene der Julia-Menge, welche konvergieren, haben nach wenigen Iterationsschritten eine sehr ähnliche Zahlenfolge, wie jener Punkt in der Mandelbrot-Menge, welcher die Julia-Menge bildet. In der rechten Abbildung ist dieses Verhalten wiederzuerkennen. Der erste Teil der Grafik zeigt das Iterationsverhalten des Punktes $c = -0{,}62 + i \cdot 0{,}4275$, welcher zur Mandelbrot-Menge gehört und durch einen grünen Kreis gekennzeichnet ist. Die rote Linie zeigt die Zahlenfolge in der komplexen Zahlenebene, ausgehend von dem Startwert $c$. Sie durchläuft zyklisch immer in etwa dieselben sieben Werte. Die Zahlenfolge divergiert also nicht, der Punkt $c = -0{,}62 + i \cdot 0{,}4275$ ist folglich ein Element der Mandelbrot-Menge.

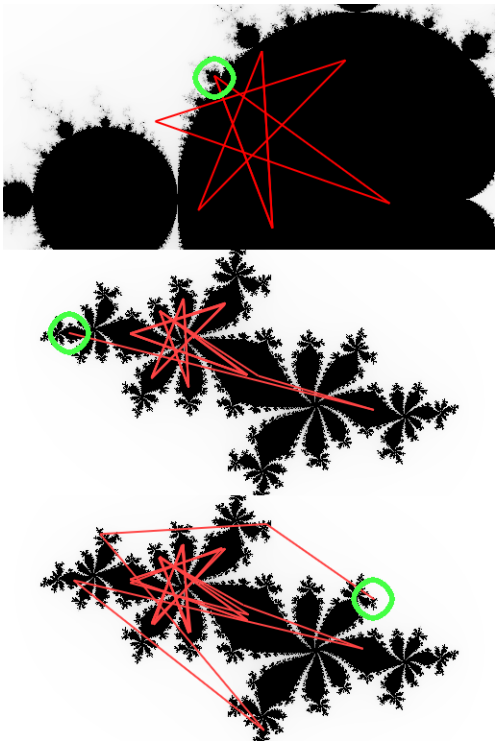

Abb. 19: Zyklisches Verhalten mehrerer Punkte in der Mandelbrot- und Julia-Menge (Eigene Darstellung)

Im zweiten und dritten Teil der Grafik ist die Julia-Menge für $c = -0{,}62 + i \cdot 0{,}4275$ gezeichnet. Sie ist zusammenhängend, da $c$ Element der Mandelbrot-Menge ist. Diesmal wurden zwei zufällige Punkte gewählt, welche zu dieser Julia-Menge gehören, und ebenfalls mit einem grünen Kreis gekennzeichnet sind. Es ist leicht zu erkennen, dass sich beide Male, obwohl weit entfernte Punkte gewählt wurden, die entstehenden Zahlenfolgen ähnlich sind. Bereits nach wenigen Iterationen erhält man wieder den Zyklus derselben sieben Werte. Diese Werte sind ident mit denen des Punktes in der Mandelbrot-Menge.

Aus mathematischer Sicht ist dieses Verhalten nicht schwer zu begründen. Bei der Julia-Menge werden alle Punkte nach jeder Iteration um den Wert $c$ erhöht. Wählt man nun einen Punkt $c$ in der Mandelbrot-Menge, so wird in der Berechnung seiner Zahlen-

folge ebenfalls in jedem Schritt um dieses $c$ erhöht. Das bedeutet der einzige Unterschied in den oben dargestellten Zahlenfolgen ist der Startwert der Zahlenfolgen. Man sieht daher, dass die Zahlenfolge, unabhängig vom Startwert, nach ein paar Iterationsschritten immer im Wesentlichen auf denselben Zyklus „einschwenkt“.

Eine weitere Besonderheit ist, dass die Iterationsformel für den Mittelpunkt jeder Julia-Menge ident ist, mit der Iterationsfolge für den entsprechenden Punkt in der Mandelbrot-Menge, da der Startwert beide Male 0 ist. Daher gilt: falls der Mittelpunkt der Julia-Menge konvergiert (in der Abbildung schwarz ist), so konvergiert der Punkt in der Mandelbrot-Menge ebenfalls, und die Julia-Menge ist zusammenhängend. Ist der Mittelpunkt $(0, 0)$ der Julia-Menge weiß, weist also eine divergente Zahlenfolge auf, so gehört $c$ nicht zur Mandelbrot-Menge und die Julia-Menge zerfällt in Inseln.

## 5.2 Ähnliche Formen

Die nebenstehende Abbildung zeigt oben einen Ausschnitt der Mandelbrot-Menge bei $c = -0.747915234375 + i \cdot -0.148673046875$, und unten die Julia-Menge für dasselbe $c$. Der Skalierungsfaktor des oberen Bildes ist 2048, der des unteren beträgt nur 1,5. Es ist sofort erkennbar, dass sich die beiden Grafiken ähnlich sehen. Sieht man sich den Mittelpunkt von der Spirale in der Mandelbrot-Menge an, so kann man durch Abzählen 21 vom Mittelpunkt ausgehende strahlenförmige Arme erkennen. Auch in der Julia-Menge haben die Spiralen je 21 Arme. Auch andere strukturelle Ähnlichkeiten fallen auf, wie zum Beispiel die charakteristischen Endspiralen, welche auch als Seepferdchen bezeichnet werden. Sowohl in der Mandelbrot- als auch in der Julia-Menge wiederholen sich diese Spiralen und Seepferdchen unendlich oft in sich selbst, mit verschiedenen Skalierungsfaktoren.

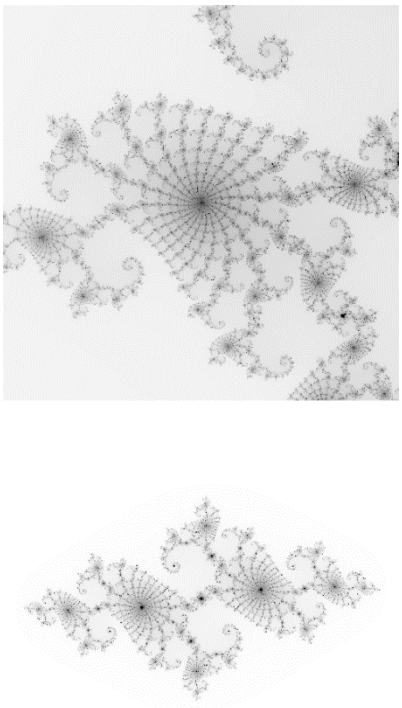

Abb. 20: Selbstähnlichkeit in einer Julia-Menge (Eigene Darstellung)

Aus dem schwarzen Mittelpunkt der Julia-Menge ergibt sich, dass $c$ ein Element der Mandelbrot-Menge ist und die Julia-Menge zusammenhängend ist.

Sieht man sich den Punkt $c = -0{,}62 + i \cdot 0{,}4275$ aus Abschnitt 5.1 genauer an, so kann man 7 Arme erkennen. Diese Zahl findet man auch in der Zahlenfolge wieder, welche einen Zyklus von 7 verschiedenen Werten hat.

# 6 Persönliche Erkenntnisse

Im Laufe dieser Arbeit veränderten sich einige meiner Sichtweisen auf die Mandelbrot- und Julia-Mengen bezogen. Meine größte Erkenntnis ist, dass mit der einfachen Formel $z_{n+1} = z_n^2 + c$ eine chaotische Form entsteht, welche nicht zur Gänze dargestellt werden kann. Nicht einmal ein einziger Punkt auf der Randlinie, bis auf wenige Ausnahmen, lässt sich bestimmen, da mit fortschreitender Vergrößerung der ausgefranste Rand immer detaillierter wird. Außerdem war ich mir vor dem Schreiben der Arbeit nicht bewusst, was die Hausdorff-Dimension ist, und dass sich eine Linie auch in die zweite Dimension erstrecken kann. Des Weiteren fiel mir durch das Entwickeln des Programmes die Wichtigkeit der Recheneffizienz in der Programmierung auf. Vor allem gilt dies bei der Mandel- und Julia-Menge, da bei dem Berechnen der Mengen viele Rechenschritte für nur wenige Punkte anfallen. Der langsame Ablauf des Programmes motivierte mich beispielsweise den progressiven Bildaufbau zu entwickeln.

# Literaturverzeichnis

Bräuner, Kurt: Chaos, Attraktoren und Fraktale. mathematische und physikalische Grundlagen nichtlinearer Phänomene mit Anwendungen in Physik, Biologie und Medizin. Berlin: Logos Verlag, 2002

Herrmann, Dietar: Algorithmen für Chaos und Fraktale. Bonn, Paris: Addison-Wesley (Deutschland) GmbH, 1996

Shishikura, Mitsuhiro: The Hausdorff dimension of the boundary of the Mandelbrot set and Julia sets. In: Mathematics Department, Princeton University: Annals of Mathematics. Band 147. Princeton, New Jersey: Princeton University and the Institute for Advanced Study, 1998, 225-267

Zeitler, Herbert; Neidhardt, Wolfgang: Fraktale und Chaos. Darmstadt: Wissenschaftliche Buchgesellschaft

# Verzeichnis der Onlinequellen

Förstemann, Thorsten: Numerical estimation of the area of the Mandelbrot set (2012). 2015. https://www.foerstemann.name/dokuwiki/doku.php?id=numerical_estimation_of_the_area_of_the_mandelbrot_set_2012 [Zugriff: 24. Mai 2021].

# Abbildungsverzeichnis

Abbildungen 1-6, 9-20 sind eigene Darstellungen welche mithilfe der Programme GeoGebra, GIMP und eigener JavaScript-Programme erstellt wurden.

# Anhang

Der folgende Text ist der in der Arbeit beschriebene Programmcode, zum Zeichnen einer Mandelbrot- oder Julia-Menge.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Mandelbrot</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">

    <script src="https://code.jquery.com/jquery-3.6.0.min.js" integ-
rity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" cros-
sorigin="anonymous"></script>

    <script>
        let threshold = 0.01;
        $(document).ready(function() {
            const canvas = $("#canvas0")[0];
            canvas.width = 500;
            canvas.height = 500;
            const ctx = canvas.getContext("2d");
            let midCr = -0.70189,
                midCi = 0.26823,
                scale = 17,
                nmax = 256;
            renderMandelbrot(midCr, midCi, scale, nmax, canvas, ctx);
        });

        function renderMandelbrot(midCr, midCi, scale, nmax, canvas, ctx)
{
            let dstBetweenPixels = 0.01 / scale;
            let startCr = midCr - (canvas.width / 2) * dstBetweenPixels;
            let startCi = midCi + (canvas.height / 2) * dstBetweenPixels;
            for (let y = 0; y < canvas.height; y++) {
                for (let x = 0; x < canvas.width; x++) {
                    let cr = startCr + x * dstBetweenPixels;
                    let ci = startCi - y * dstBetweenPixels;
                    let iter = calcPixel(0, 0, cr, ci, nmax, threshold /
scale);

                    // Black and White coloring
                    if (iter == nmax) {
                        ctx.fillStyle = "#000000";
                    } else {
```

```
                    //ctx.fillStyle = "#ffffff";
                    iter = iter % 128 + 127;
                    //Grayscale
                    ctx.fillStyle = `rgb(${iter},${iter},${iter})`;
                }

                /* Purple coloring
                iter = calcPixel(cr, ci, nmax);
                let r = (iter / nmax) * 255;
                let g = 0;
                let b = (iter / nmax) * 255;
                ctx.fillStyle = `rgb(${r},${g},${b})`;
                if (iter == nmax) ctx.fillStyle = "#000000";
                */
                ctx.fillRect(x, y, 1, 1);
            }
        }
    }

    //Alternative function to be called to render the Julia-Set
    function renderJulia(midCr, midCi, scale, nmax, canvas, ctx, c_r,
c_i) {
        let dstBetweenPixels = 0.01 / scale;
        let startCr = midCr - (canvas.width / 2) * dstBetweenPixels;
        let startCi = midCi + (canvas.height / 2) * dstBetweenPixels;
        for (let y = 0; y < canvas.height; y++) {
            for (let x = 0; x < canvas.width; x++) {
                let cr = startCr + x * dstBetweenPixels;
                let ci = startCi - y * dstBetweenPixels;
                let iter = (1 - calcPixel(cr, ci, c_r, c_i, nmax,
threshold / scale) / nmax) * 255;
                ctx.fillStyle = `rgb(${iter},${iter},${iter})`;
                ctx.fillRect(x, y, 1, 1);
            }
        }
    }

    function calcPixel(startCr, startCi, cr, ci, nmax, threshold) {
        let zr = startCr;
        let zi = startCi;
        let memr = zr;
        let memi = zi;
        for (let i = 0; i < nmax; i++) {
            const zold = zr;
            zr = zold * zold - zi * zi + cr;
            zi = 2 * zold * zi + ci;
            if (zr * zr + zi * zi > 4) {
                return i;
            }
```

```
                if (Math.abs(memr - zr) < threshold && Math.abs(memi -
zi) < threshold) {
                    return nmax;
                }
                if (i % 20 == 3) {
                    memr = zr;
                    memi = zi;
                }
            }
            return nmax;
        }
    </script>
</head>

<body>
    <header>
    </header>
    <main>
        <div class="container">
            <div class="row">
                <div class="col justify-content-md-center" id="canvash-
ousing">
                    <canvas id="canvas0" class="">Please update your
browser to use this website.</canvas>
                </div>
                <div class="col justify-content-md-center" id="iter">
                </div>
            </div>
        </div>
    </main>
</body>

</html>
```

# Selbständigkeitserklärung

Ich, *Jakob Ausserer*, erkläre hiermit, dass ich diese vorwissenschaftliche Arbeit selbstständig und ohne Hilfe Dritter verfasst habe. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als Zitate kenntlich gemacht und keine anderen als die angegebenen Quellen verwendet habe.

Ich bestätige, dass die aktuelle abgegebene Datei der hochgeladenen entspricht.

☒ Ich gebe mein Einverständnis, dass ein Exemplar meiner vorwissenschaftlichen Arbeit in der Schulbibliothek meiner Schule aufgestellt wird.

Wien, am 20. Februar 2022 …….

*Unterschrift des Schülers / der Schülerin*