# Farbgebungsalgorithmen



# Farbgebungsalgorithmen der Mandelbrot-Menge

## Die Rolle der Farbgebung

Die Mandelbrot-Menge selbst wird traditionell als schwarze Fläche dargestellt. Doch die wahren visuellen Schönheiten entstehen erst, wenn wir die Punkte *außerhalb* der Menge betrachten. Für diese Punkte stellt sich die Frage: Wie schnell entweichen sie ins Unendliche? Die Farbgebung ist ein künstlerisches und informatives Mittel, um diese "Fluchtgeschwindigkeit" zu visualisieren.

Jeder Punkt $c$ außerhalb der Mandelbrot-Menge divergiert, wenn er iteriert wird. Die Anzahl der Iterationen, die benötigt werden, bis der Betrag von $z_n$ einen bestimmten "Fluchtradius" (meist 2) überschreitet, gibt uns Aufschluss über seine Divergenzgeschwindigkeit. Diese Iterationszahl können wir dann einer Farbe zuordnen.

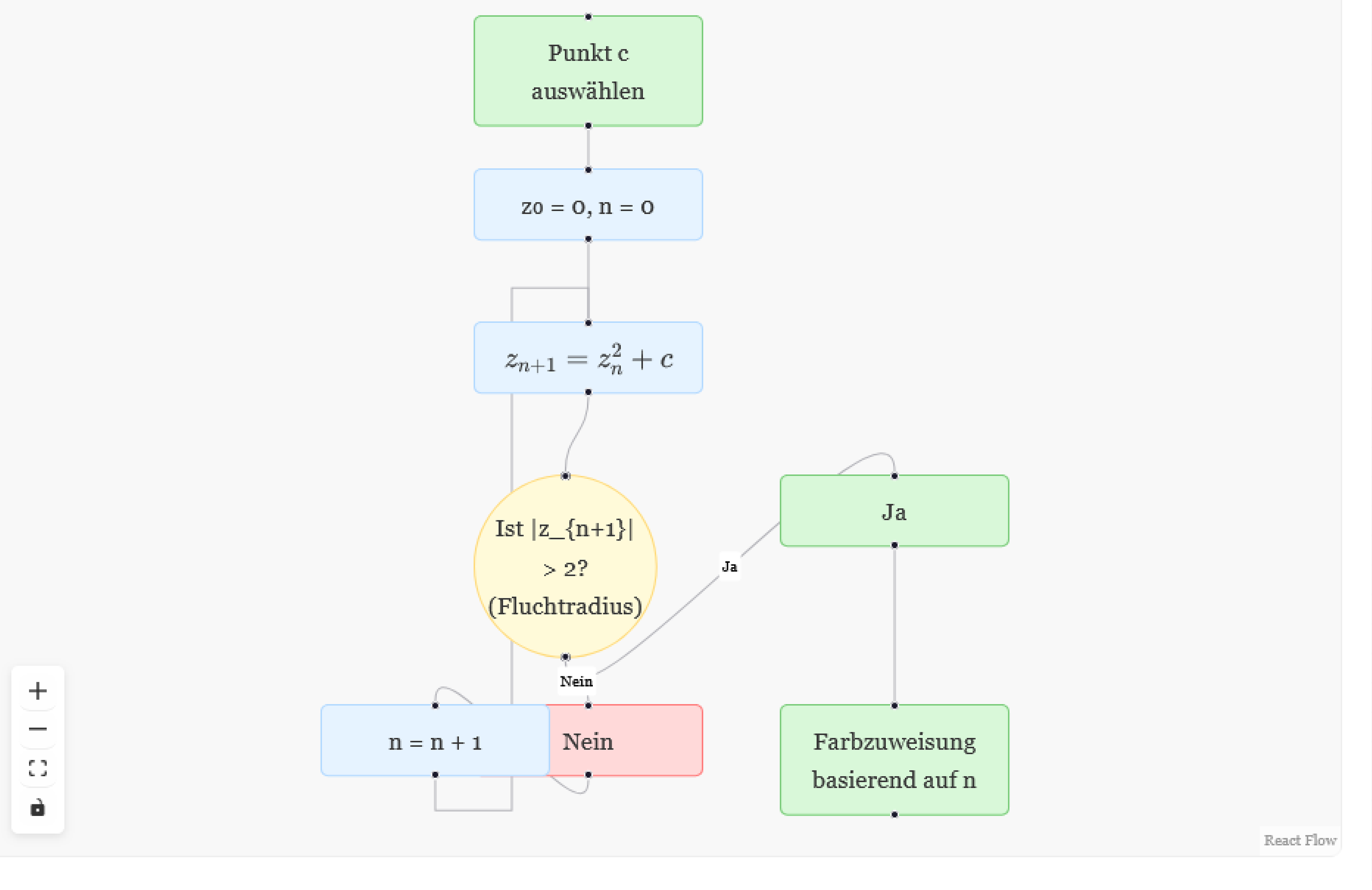
## Der Escape-Time-Algorithmus

Der einfachste und intuitivste Farbgebungsalgorithmus ist der **Escape-Time-Algorithmus (ETA)**. Er basiert direkt auf der Iterationszahl $n$, die ein Punkt benötigt, um den Fluchtradius zu überschreiten.

### Funktionsweise:

1. Für jeden Pixel, der einem komplexen Punkt $c$ entspricht:
2. Starte die Iteration mit $z_0 = 0$ und $n = 0$.
3. Berechne $z_{n+1} = z_n^2 + c$.
4. Überprüfe, ob $|z_{n+1}|$ den Fluchtradius (meist 2) überschreitet.
5. Wenn ja: Der Punkt entweicht. Weise dem Pixel eine Farbe zu, die von $n$ abhängt. Beende die Iteration für diesen Punkt.
6. Wenn nein: Erhöhe $n$ um 1 und wiederhole Schritt 3.
7. Wenn $n$ eine maximale Iterationszahl erreicht, ohne zu entweichen: Der Punkt gehört wahrscheinlich zur Mandelbrot-Menge (oder ist ihr sehr nahe) und wird oft schwarz gefärbt.

### Prozessvisualisierung:



### Pseudo-Code Beispiel:

```
function getColor(c, maxIterations, bailoutRadius = 2) {
    let z = { real: 0, imag: 0 };
    let n = 0;

    while (n < maxIterations) {
        // Berechne z_n+1 = z_n^2 + c
        let zRealSq = z.real * z.real;
        let zImagSq = z.imag * z.imag;
        let newReal = zRealSq - zImagSq + c.real;
        let newImag = 2 * z.real * z.imag + c.imag;
        z.real = newReal;
        z.imag = newImag;

        // Prüfe auf Divergenz (|z_n+1| > bailoutRadius)
        if (z.real * z.real + z.imag * z.imag > bailoutRadius * bailoutRadius) {
            // Punkt entweicht, gebe Farbe basierend auf n zurück
            return mapIterationToColor(n);
        }
        n++;
    }
    // Punkt gehört wahrscheinlich zur Menge, gebe Schwarz zurück
    return 'black';
}

function mapIterationToColor(n) {
    // Hier wird n auf eine Farbe abgebildet.
    // Beispiel: Regenbogenfarben-Palette, Farbtöne basierend auf n % Palette.length
    const palette = ['#0000ff', '#00ff00', '#ffff00', '#ff0000', '#ff00ff']; // Beispiel-Palet
    return palette[n % palette.length];
}
```

## Der "Banding-Effekt"

Obwohl der Escape-Time-Algorithmus einfach ist, hat er einen entscheidenden Nachteil: den sogenannten **"Banding-Effekt"**. Da die Iterationszahl $n$ immer eine ganze Zahl ist, werden benachbarte Pixel, die nur geringfügig unterschiedlich divergieren, derselben Farbe zugeordnet, wenn sie in der gleichen Iterationszahl den Fluchtradius überschreiten.

Dies führt zu sichtbaren, abrupten Farbübergängen oder "Bändern" im Bild, anstatt zu weichen, kontinuierlichen Farbverläufen. Stellen Sie sich vor, Sie malen eine Landschaft mit nur 256 Farben – die Farbübergänge sind deutlich sichtbar. Beim Mandelbrot-Set sind es noch weniger diskrete Iterationsschritte, die zu Farbsprüngen führen.


Die oben gezeigten Beispiele illustrieren den Unterschied: Links sehen Sie harte, diskrete Farbübergänge (Banding), wie sie beim einfachen Escape-Time-Algorithmus auftreten können. Rechts sehen Sie einen weichen, kontinuierlichen Übergang, wie er durch fortgeschrittenere Algorithmen erreicht wird.

## Normalized Iteration Count (Smooth Coloring)

Um den Banding-Effekt zu überwinden und ästhetisch ansprechendere, weiche Farbübergänge zu erzielen, wird der **Normalized Iteration Count** (oft auch "Smooth Coloring" genannt) verwendet. Dieser Algorithmus interpoliert die Iterationszahl zu einem nicht-ganzzahligen Wert, der die "echte" Divergenzgeschwindigkeit genauer widerspiegelt.

### Die Formel:

$$v(n) = n + \frac{\log(\log b) - \log(\log |z_n|)}{\log 2}$$

Hierbei ist:

- $n$: Die kleinste ganze Iterationszahl, bei der $|z_n|$ den Fluchtradius $b$ (meist 2) überschreitet.
- $|z_n|$: Der Betrag der komplexen Zahl $z_n$ in der Iteration $n$.
- $b$: Der Fluchtradius (Bailout-Radius), typischerweise 2.
- $\log$: Der natürliche Logarithmus (ln).

Diese Formel erzeugt einen Gleitkommawert $v(n)$, der typischerweise zwischen $n-1$ und $n$ liegt. Der Term $\frac{\log(\log b) - \log(\log |z_n|)}{\log 2}$ dient als Korrekturfaktor, der den Wert von $n$ basierend darauf anpasst, wie weit $|z_n|$ den Fluchtradius $b$ hinausgeht. Wenn $|z_n|$ nur knapp über $b$ ist, ist der Korrekturterm nahe Null, und $v(n)$ ist nahe $n$. Je weiter $|z_n|$ über $b$ hinausgeht, desto stärker beeinflusst der Term $v(n)$ in Richtung $n-1$. Dieser kontinuierliche Wert kann dann auf eine kontinuierliche Farbpalette abgebildet werden, was zu den "glatten" und hochdetaillierten Mandelbrot-Bildern führt, die wir kennen.

### Interaktive Demonstration der Formel:

#### Berechnung des Normalisierten Iterationszählers

Experimentieren Sie mit den Werten für die Iterationszahl $n$ und den Betrag von $z_n$, um zu sehen, wie sich der normalisierte Iterationszähler $v(n)$ verhält.

**Iterationszahl (n):** 10

**Betrag von z_n ($|z_n|$):** 2,5

(Muss größer als 2 sein)

Fluchtradius $b$: 2

$\log(\log b)$: -0.367

$\log(\log |z_n|)$: -0.087

$\log 2$: 0.693

**Normalisierter Iterationszähler $v(n)$: 9.597**

Dieser Wert liegt zwischen $n-1$ (9) und $n$ (10), was eine stufenlose Interpolation ermöglicht.

## Alternative Ansätze zur Farbgebung

- **Distance Estimation (Abstandsschätzung):** Dieser Algorithmus färbt Punkte nicht nur nach ihrer Fluchtgeschwindigkeit, sondern auch nach ihrer Entfernung zum Rand der Mandelbrot-Menge. Dies ermöglicht eine extrem hohe Detailgenauigkeit und kann auch die Innenbereiche der Menge detaillierter darstellen.
- **Kombinationen und Polarkoordinaten:** Fortschrittliche Renderings nutzen oft eine Kombination verschiedener Algorithmen. Beispielsweise kann die Iterationszahl zur Bestimmung des Farbtons verwendet werden, während der Winkel des Punktes in Polarkoordinaten für die Helligkeit oder Sättigung sorgt. Dies eröffnet unendliche Möglichkeiten für kreative und komplexe Visualisierungen.

## Zusammenfassung

Die Farbgebung verwandelt die abstrakte Mathematik der Mandelbrot-Menge in atemberaubende Kunstwerke. Während der einfache Escape-Time-Algorithmus ein guter Ausgangspunkt ist, revolutioniert der Normalized Iteration Count die Visualisierung durch die Erzeugung weicher, kontinuierlicher Farbübergänge.

Die Erzeugung dieser Bilder ist oft rechenintensiv, insbesondere bei hohen Zoomstufen und Iterationszahlen. Doch die resultierenden, unendlich detaillierten Fraktale belohnen diesen Aufwand mit einer "Welt voller Schönheiten", wie Mandelbrot es selbst ausdrückte.