

Heinrich-Heine-Universität Düsseldorf

Mathematisch-Naturwissenschaftliche Fakultät

Bachelorarbeit Informatik

# **Generierung und Visualisierung künstlicher Landkarten**

Jens Fröschel

[Jens.Froeschel@hhu.de](mailto:Jens.Froeschel@hhu.de)

Düsseldorf, 15.04.2016

eingereicht bei

Prof. Dr. Martin Mauve  
PD Dr. Dr.-Ing. Wilfried Linder

## **Inhaltsverzeichnis**

1. Einleitung.....	1
1.1 Motivation.....	1
1.2 Ziel der Bachelorarbeit.....	1
1.3 Aufbau der Bachelorarbeit.....	1
2. Rauschfunktionen.....	2
2.1 Value Noise.....	2
2.2 Open Simplex Noise.....	5
2.3 Billowy Noise und Ridged Noise.....	5
2.4 Turbulenzen.....	6
3. Klimatologie.....	8
3.1 Höhenfunktion.....	9
3.2 Temperaturfunktion.....	11
3.3 Niederschlagsfunktion.....	12
3.4 Köppensche-Klimaklassifikation.....	12
4. Visualisierung.....	14
4.1 Höhenkarte.....	15
4.2 Temperaturkarte.....	16
4.3 Niederschlagskarte.....	18
4.4 Köppen-Karte.....	19
5. Das Programm.....	21
5.1 Programmkerne.....	22
5.2 Rauschfunktionen.....	24
5.3 Visuelle Karten.....	26
6. Schlussbetrachtung.....	27

## Inhaltsverzeichnis

---

6.1 Vergleich mit Karten der Erde.....	27
6.2 Fazit.....	32
6.3 Ausblick.....	32
7. Literaturverzeichnis.....	34

## 1. Einleitung

### 1.1 Motivation

Die Erstellung einer Karte ist der erste Schritt zur Erschaffung einer Spielewelt. Karten sind dadurch ein wichtiger Aspekt in der Spieleindustrie. Mit der Weiterentwicklung der Computer entwickeln sich auch die Spielewelten. Sie werden größer, realistischer und frei zugänglich. Dies führt bei der Spieleentwicklung zu Problemen, denn ab einer gewissen Größe kann man die Spielewelt nicht mehr von Menschenhand erzeugen und muss sie mithilfe von Algorithmen generieren. Um große, real wirkende Welten zu erschaffen, benötigt man deswegen realitätsnahe Algorithmen.

### 1.2 Ziel der Bachelorarbeit

Ziel der Bachelorarbeit ist es realitätsnahe, geowissenschaftliche Karten künstlich zu generieren. Dafür werden zuerst verschiedene Algorithmen für die Kartengenerierung vorgestellt. Sie werden formal beschrieben und ihre besonderen Eigenschaften hervorgehoben. Die Algorithmen werden außerdem visualisiert um eine Vergleichsmöglichkeit der generierten Karten mit Karten der Erde zu geben. Anhand der visuellen Darstellungen lässt sich der Realitätsgrad von den Algorithmen bewerten.

Eine zusätzliche Anforderung an die Bachelorarbeit ist es das ganze Projekt so zu gestalten, dass sich ohne großen Aufwand neue Algorithmen und Visualisierungen hinzufügen lassen. Es soll dem Anwender ermöglicht werden auch eigene Änderungen an dem Programm vorzunehmen und eigene Tests mit dem Programm durchzuführen.

### 1.3 Aufbau der Bachelorarbeit

Die Bachelorarbeit lässt sich in fünf Abschnitte untergliedern. Die ersten drei Abschnitte orientieren sich an dem Generierungsprozess von Landkarten. Der vierte Abschnitt stellt das entwickelte Programm vor und der fünfte Abschnitt vergleicht die erzeugten mit echten Karten.

Die Generierung der Karten erfolgt in drei Schritten. Zuerst werden Rauschfunktionen verwendet um ein zufällig wirkendes, zweidimensionales Feld zu erzeugen. Man hat so-

zusagen eine Fläche aus Werten zwischen 0 und 1. Unterschiedliche Verfahren für die Generierung solcher Felder werden in Kapitel zwei vorgestellt.

Danach werden in Kapitel drei die erzeugten Felder in realitätsnahe, geowissenschaftliche Daten umgewandelt. Dafür wird Bezug auf Geodaten der Erde genommen. Zusätzlich wird die *Köppensche-Klimaklassifikation* vorgestellt, welche ein gutes Vergleichsmodell der generierten Daten mit den Daten der Erde darstellt.

Das vierte Kapitel handelt von der Darstellung der geowissenschaftlichen Daten. Dabei werden Verfahren zur Visualisierung von Höhen-, Temperatur-, Niederschlagskarten und weiteren Karten vorgestellt.

Im fünften Kapitel wird auf das Programm eingegangen. In diesem Abschnitt werden die wichtigsten Eigenschaften des Programms aufgeführt und anhand von kurzen Codebeispielen erklärt.

Im letzten Kapitel werden die zuvor erläuterten Karten mit Karten der Erde verglichen. Außerdem gibt es ein kurzes Fazit mit Zukunftsaussichten für das Programm.

## 2. Rauschfunktionen

In dem folgenden Kapitel werden Rauschfunktionen vorgestellt. Es werden ausschließlich zweidimensionale Funktionen erläutert, jedoch lassen sich die Verfahren nach demselben Prinzip auf höhere Dimensionen erweitern.

Das Ziel von Rauschfunktionen innerhalb der Kartengenerierung ist es ein zweidimensionales Feld von pseudozufälligen Werten  $m_{x,y}$  mit  $m_{x,y} \in [0,1]$  zu erzeugen, wobei  $x$  und  $y$  auf  $x, y \in [0,1]$  skaliert Kartenkoordinaten sind. Eine zweidimensionale Rauschfunktion ist somit eine Funktion  $\text{noise}: \mathbb{R}^2 \rightarrow \mathbb{R}$ ,  $\text{noise}(x, y) = m_{x,y}$ .

Es gibt sehr viele Variationen solcher Rauschfunktionen. Einige wichtige werden im folgenden erläutert und ihre Vorteile beschrieben.

### 2.1 Value Noise

Der einfachste Weg um die Werte  $m_{x,y}$  zu berechnen wäre es Zufallszahlen zu nehmen  $m_{x,y} = \text{random}()$ . Auf diese Weise würde man jedoch lediglich ein Rauschbild erhalten.

Stattdessen erzeugt man ein kleineres Feld von  $k \times k$  vielen Zufallszahlen und interpoliert zwischen diesen Werten. Man erzeugt beispielsweise ein  $11 \times 11$  Raster. An den Rasterpunkten werden Zufallszahlen verteilt und an allen Koordinaten dazwischen wird zweidimensional interpoliert. Dieser Ansatz wird als *Value Noise*  $\text{value} : \mathbb{R}^2 \rightarrow \mathbb{R}$  bezeichnet.<sup>1</sup>

Für die Interpolation gibt es sehr viele verschiedene Algorithmen. Die einfachste Methode ist die bilineare Interpolation.<sup>2</sup> Bei der bilinearen Interpolation werden für jeden Punkt ausschließlich die vier umliegenden Rasterpunkte betrachtet. Es wird zuerst zwischen den zwei oberen und den zwei unteren Rasterpunkten in x-Richtung interpoliert. Danach werden die beiden berechneten Werte in y-Richtung interpoliert.

---

**Algorithm 1** Value Noise

---

$Z[][] = \text{random}(k, k)$

```

value(x, y){
    //Die Eingabe wird auf den Bereich [0,k] skaliert
    x = x * k;
    y = y * k;

    //Der Abstand von dem Punkt zum oberen / linken Kästchenrand
    xoff = x - ⌊x⌋;
    yoff = y - ⌊y⌋;

    //Die Interpolation in x Richtung am oberen Kästchenrand
    top = (1-xoff) * Z(⌊x⌋, ⌊y⌋) + xoff * Z(⌈x⌉, ⌊y⌋);

    //Die Interpolation in x Richtung am unteren Kästchenrand
    bottom = (1-xoff) * Z(⌊x⌋, ⌈y⌉) + xoff * Z(⌈x⌉, ⌈y⌉);

    //Die Interpolation in y Richtung
    return (1-yoff) * top + yoff * bottom;
}

```

---

Abbildung 2.1: Der *Value Noise* Algorithmus.

---

1 Burger, Wilhelm(2008): „Gradientenbasierte Rauschfunktionen und Perlin Noise“, S.1-2

2 Han, Dianyuan(2013): „Comparison of Commonly Used Image Interpolation Methods“, S.1-2

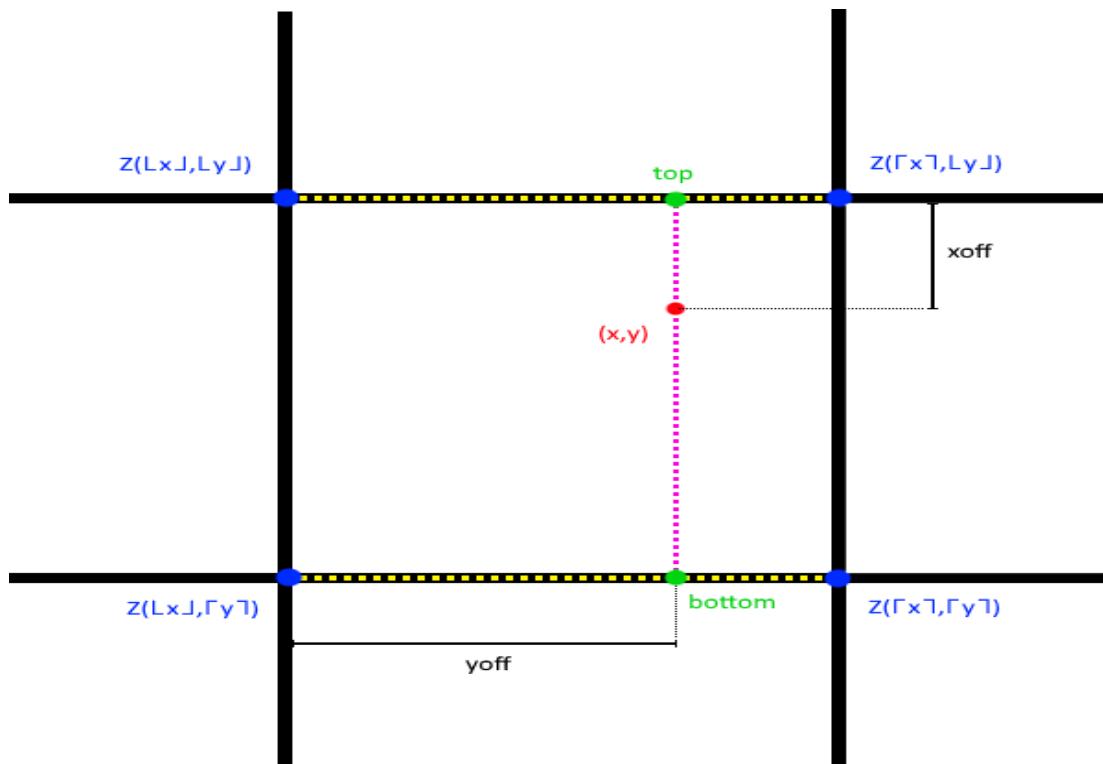


Abbildung 2.2: Zweidimensionale, bilinare Interpolation: Zuerst wird entlang der gelben Linien interpoliert um die Werte bei den grünen Punkten zu berechnen. Danach wird entlang der rosa Linie interpoliert um den Wert bei  $(x,y)$  zu berechnen.

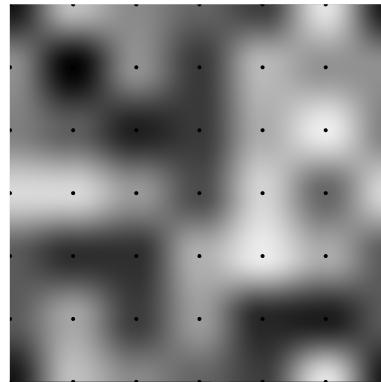


Abbildung 2.3: Screenshot aus dem Programm vom 25.03.2016: *Value Noise*. Die Rasterpunkte sind schwarz markiert, dazwischen wird interpoliert.

Ein Problem mit *Value Noise* ist der deutlich erkennbare rasterartige Aufbau der generierten Karte. Diese sichtbare Rasterung entsteht, weil jede Extremstelle der Funktion einem der generierten Zufallswerte entspricht und somit auf den Rasterpunkten liegt. Eine Möglichkeit dieses Problem zu beheben bietet *Gradient Noise*<sup>3</sup>.

---

3 Burger, Wilhelm(2008): „Gradientenbasierte Rauschfunktionen und Perlin Noise“, S.1 f.

Bei *Gradient Noise* wird an den Rasterpunkten eine zufällige Steigung (engl. gradient) anstelle eines zufälligen Wertes festgelegt und es wird zwischen den Steigungen interpoliert. Dadurch liegen die Extremstellen nicht mehr auf den Rasterpunkten.

## 2.2 Open Simplex Noise

Zu den bekanntesten Rauschfunktionen gehört das von Ken Perlin entwickelte *Simplex Noise*<sup>4</sup>. Bei *Simplex Noise* wird anstelle eines Rasters aus Quadraten, ein Raster aus Simplexen (im zweidimensionalen Raum sind das Dreiecke) verwendet. An den Rasterstellen werden Steigungen festgelegt und dazwischen wird interpoliert. Somit ist *Simplex Noise* ein *Gradient Noise* Algorithmus. Zusätzlich verwendet Simplex Noise eine höherdimensionale Interpolation für fließendere Übergänge. Im folgenden wird aus Lizenzgründen *Open Simplex Noise* verwendet.

*Open Simplex Noise*<sup>5 6</sup>  $\text{simplex} : \mathbb{R}^2 \rightarrow \mathbb{R}$  ist ein von Kurt Spencer veröffentlichtes Verfahren, welches sich an *Simplex Noise* orientiert. *Open Simplex Noise* ist ein *Gradient Noise* Algorithmus, welcher simplektische Honigwarben als Raster verwendet.



Abbildung 2.4: Screenshot aus dem Programm vom 25.03.2016: *Open Simplex Noise*

## 2.3 Billowy Noise und Ridged Noise

Einer der Nachteile von *Open Simplex Noise* ist die gleichmäßige Verteilung von Rauschwerten. Vor allem für die Generierung von Höhenkarten ist diese Verteilung un-

4 Gustavson, Stefan(2005): „Simplex noise demystified“, S.4-7

5 Spencer, Kurt(2014): „Noise!“. <http://uniblock.tumblr.com/>

6 Spencer, Kurt(2014): „Open Simplex Noise in Java“.

<https://gist.github.com/KdotJP/b1270127455a94ac5d19>

zureichend, da diese in der Regel durch flache Berge und tiefe Schluchten charakterisiert sind. Aus diesem Grund werden im folgenden zwei weitere Algorithmen vorgestellt, mit welchen sich die Eigenschaften von Höhenkarten besser erfassen lassen.

Das erste dieser Verfahren ist *Billowy Noise*<sup>7</sup>. Für *Billowy Noise* wird ein von *Open Simplex Noise* berechneter Wert auf den Bereich  $[-1,1]$  skaliert und dann der Betrag genommen. Dadurch kommen die Werte in der Nähe von 0 weitaus häufiger vor als die Werte nahe 1. Außerdem sind durch die Interpolation von *Open Simplex Noise* die Steigungen nahe der 0 steiler als die Steigungen nahe der 1. Dadurch erhält man Höhenkarten, welche aus flachen Hügeln und steilen Schluchten bestehen.

$$\text{billowy}(x, y) := |\text{simplex}(x, y) - 0.5| \cdot 2$$

Einen ähnlichen Ansatz verfolgt *Ridged Noise*<sup>8</sup>. *Ridged Noise* ist die Vertauschung von hohen und tiefen Werten bei *Billowy Noise*. Durch die Vertauschung entsteht bei *Ridged Noise* ein Gelände, welches aus steilen, hohen Bergen und flachen Tälern besteht.

$$\text{ridged}(x, y) := 1 - |\text{simplex}(x, y) - 0.5| \cdot 2 = 1 - \text{billowy}(x, y) .$$

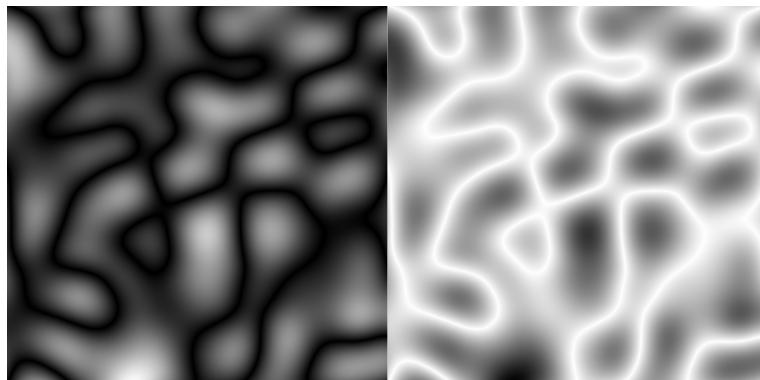


Abbildung 2.5: Screenshot aus dem Programm vom 25.03.2016: *Billowy Noise*(links), *Ridged Noise*(rechts)

## 2.4 Turbulenzen

Das Problem mit den bisherigen Rauschfunktionen ist, dass sie zu grob wirken. In der Realität sind sehr viele Eigenschaften der Erde fraktal aufgebaut. Genau das gilt auch

7 De Carpentier, Giliam J.P.(2008): „Effective GPU-based synthesis and editing of realistic heightfields“, S.72 f.

8 De Carpentier, Giliam J.P.(2008): „Effective GPU-based synthesis and editing of realistic heightfields“, S.72 f.

für Höhen, Temperaturen und Niederschläge. Man hat beispielsweise einen großen Höhenunterschied zwischen Bergen und Tälern. Gleichzeitig gibt es aber auch innerhalb der Täler Höhenunterschiede, welche durch Hügel und Flachland entstehen. Und auch auf den Hügeln gibt es geringere Höhenunterschiede durch Felsen.

Dieses Verhalten soll mit einem einfachen Ansatz simuliert werden. Dafür berechnet man nicht nur eine Rauschfunktion, sondern man berechnet die Rauschfunktion für unterschiedliche Skalierungen der Karte und mit unterschiedlichen Gewichtungen. Die so berechneten Rauschfunktionen werden aufsummiert. Dieses Verfahren wird als Turbulenzen<sup>9</sup> (engl. turbulence) bezeichnet.

Turbulenzen sind so gesehen ein iterativer Aufruf der Rauschfunktion, wobei jeder weitere Aufruf stärker skaliert ist und weniger Auswirkungen auf das Gesamtergebnis hat. Die Anzahl der iterativen Aufrufe wird als Oktaven (engl. octave) bezeichnet und die Gewichtung nennt man Amplitude. Es hat sich als gutes Verfahren erwiesen die Amplitude bei jedem Aufruf zu halbieren und die Skalierung zu verdoppeln. Damit sind Turbulenzen gegeben als:

$$\text{turbulence}(x, y) := \sum_{i=0}^{\text{octave}-1} \left(\frac{1}{2}\right)^i \cdot \text{noise}((x \cdot 2^i) \bmod 1, (y \cdot 2^i) \bmod 1) .$$

Die so entstandenen Muster kommen wegen ihres natürlichen Aussehens vielfach in der Computergrafik zum Einsatz. Zum Beispiel um Himmel, Wolken, Wasser oder auch Berge zu simulieren.

---

<sup>9</sup> De Carpentier, Giliam J.P.(2008): „Effective GPU-based synthesis and editing of realistic heightfields“, S.117 f.

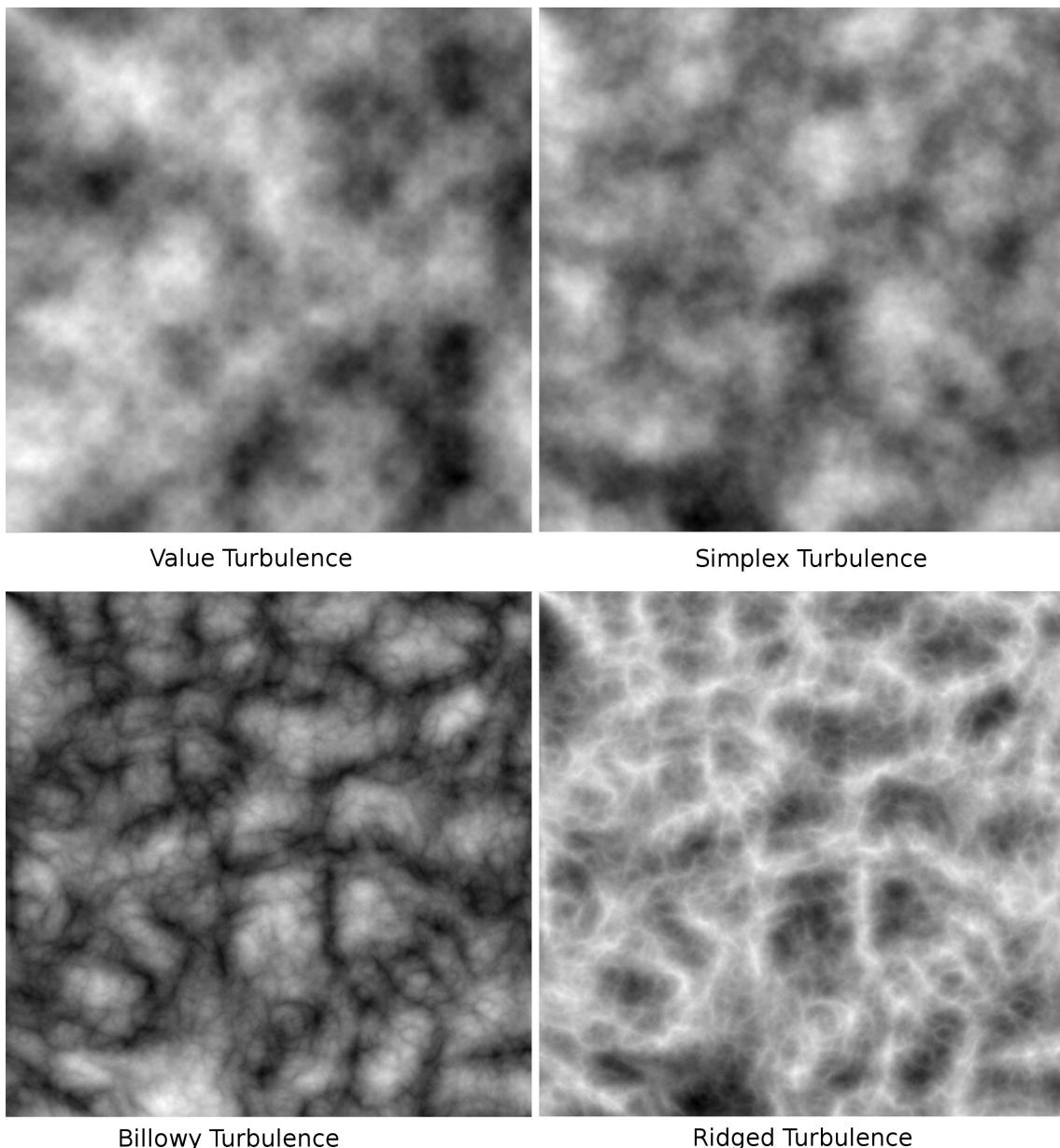


Abbildung 2.6: Screenshot aus dem Programm vom 25.03.2016: Die zuvor vorgestellten Rauschfunktionen als Turbulenzen

### 3. Klimatologie

Im vorherigen Kapitel wurden verschiedene Rauschfunktionen vorgestellt, mit denen sich ein zweidimensionales Feld von Zufallswerten  $m_{x,y} \in [0,1]$  erzeugen lässt. In diesem Kapitel werden die von Rauschfunktionen erzeugten Felder in thematische Karten umgewandelt. Dabei werden zuerst drei Funktionen beschrieben, welche Rauschwerte in geowissenschaftliche Daten umwandeln:

- Eine Höhenfunktion (in Meter)

- Eine Temperaturfunktion (in Grad Celsius)
- Eine Niederschlagsfunktion (in Millimeter)

Anhand dieser drei Eigenschaften werden zwei weitere Funktionen basierend auf der in Kapitel 3.4 beschriebenen *Köppensche-Klimaklassifikation* berechnet.

### 3.1 Höhenfunktion

Die Höhenkarte wird von einer zweidimensionale Funktion  $h: \mathbb{R}^2 \rightarrow \mathbb{R}$  generiert. Diese Funktion wird im folgenden als Höhenfunktion bezeichnet. Die Eingabeparameter sind die Kartenkoordinaten  $x$  und  $y$ .

Die Höhenfunktion verwendet eine Rauschfunktion  $\text{noise}(x, y)$  um pseudozufällige Werte  $m_{x,y} \in [0,1]$  zu berechnen und wandelt diese Werte in Höhenwerte  $h_{x,y} \in [-8848, 8848]$  in der Einheit Meter um. Die Obergrenze der Höhenwerte ist die Höhe des Mount Everest ( 8.848m )<sup>10</sup>. Als untere Grenze wurde –8.848m gewählt, damit der Meeresspiegel genau in der Mitte liegt. Alternativ könnte man als untere Grenze auch die Tiefe des Marianengraben mit ca. –11.000m verwenden.<sup>11</sup>

Der erste Ansatz Rauschwerte in Höhenwerte umzuwandeln ist eine einfache Skalierung, gegeben durch  $h_{x,y} := -8848 + m_{x,y} \cdot 17696$ . Dadurch gilt

$$h_{x,y} = \begin{cases} -8848 & , \text{falls } m_{x,y} = 0 \\ 0 & , \text{falls } m_{x,y} = 0.5 \\ 8848 & , \text{falls } m_{x,y} = 1 \end{cases} .$$

Wie man anhand von Höhenkarten der Erde<sup>12 13</sup> und Programmen zur Bestimmung der Höhe über dem Meeresspiegel<sup>14 15</sup> feststellt, ist nur ein geringer Anteil der Erdoberfläche

10 Quiong, Pu(2004): „Qomolangma Biosphere Reserve Information“. [http://www.unesco.org/mabdb/br\\_brdir/directory/biores.asp?code=CPR+26&mode=all](http://www.unesco.org/mabdb/br_brdir/directory/biores.asp?code=CPR+26&mode=all)

11 Botica, Melania(2011): "Fakten über den tiefsten Abgrund der Erde". [http://www.focus.de/wissen/mensch/tid-24437/marianengraben-fakten-ueber-den-tiefsten-abgrund-der-erde\\_aid\\_692590.html](http://www.focus.de/wissen/mensch/tid-24437/marianengraben-fakten-ueber-den-tiefsten-abgrund-der-erde_aid_692590.html)

12 Tan, Howard(2011): „Global Digital Elevation Map Announcement“. <http://asterweb.jpl.nasa.gov/gdem.asp>

13 „Visible Earth“. <http://visibleearth.nasa.gov/view.php?id=73934>

14 Karte mit Höhenwerten von Google Maps: <http://www.mapcoordinates.net/>

15 Automatische Höhenprofilberechnung: <http://geo.ebp.ch/gelaendeprofil/>

che Gebirge oder Tiefsee. Ein Großteil der Erdoberfläche liegt um die Höhe des Meeresspiegels verteilt. Eine genaue Höhenverteilung der Erde hat das *National Centers for Environmental Information* in seiner Grafik „Hypsographic Curve of Earth's Surface from ETOPO1“<sup>16</sup> visualisiert.

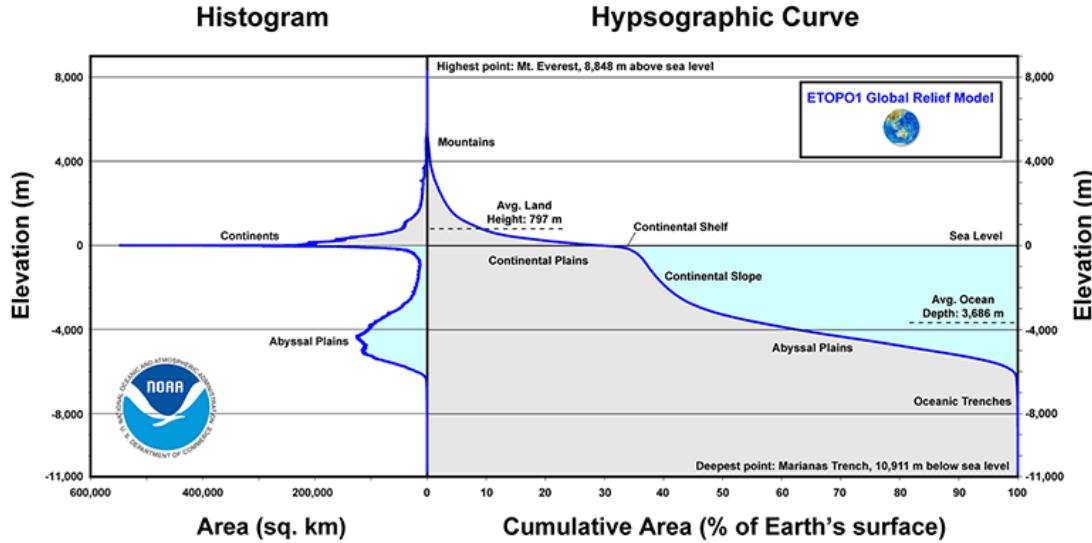


Abbildung 3.1: Hypsographic Curve of Earth's Surface from ETOPO1

Diese Höhenverteilung (Abbildung 3.1, rechts) lässt sich durch eine logistische Funktion  $\text{logit} : \mathbb{R} \rightarrow \mathbb{R}$  annähern. Eine logistische Funktion ist gegeben durch

$$\text{logit}(m) := -\log\left(\frac{1}{m} - 1\right) .$$

Weil die logistische Funktion bei 0 und 1 gegen  $\pm\infty$

konvergiert, wird die Eingabe  $m$  auf  $m \in [0.0475, 0.9525]$  skaliert. Dadurch liegt der berechnete Wert im Bereich  $\text{logit}(m) \in [-3, 3]$ . Dieser Wert wird auf den Wertebereich  $[0, 1]$  zurück skaliert, damit für die logistische Funktion sowohl der Wert für die Eingabe, wie auch der Wert für die Ausgabe im Intervall  $[0, 1]$  liegen. Somit ist die Funktion eine Umverteilung gegeben als

$$\text{logit}(m) := \frac{1}{6} \cdot \left( 3 + \log\left(\frac{1}{0.0475 + m \cdot 0.905} - 1\right) \right) .$$

16 Eakins, B.W. und Sharman, G.F.(2012): „Hypsographic Curve of Earth's Surface from ETOPO1“.

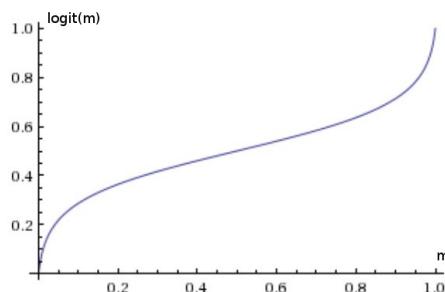


Abbildung 3.2: Die zur Skalierung verwendete logistische Funktion  $\text{logit}(m)$ .

Für die Höhenfunktion gilt schließlich  $h(x, y) := -8850 + \text{logit}(m_{x,y}) \cdot 17700$ .

### 3.2 Temperaturfunktion

Sowohl die Temperatur-, wie auch die Niederschlagsfunktion werden zeitabhängig berechnet. Es wird also für jeden Monat des Jahres eine eigene Temperatur- und Höhenkarte erzeugt. Monatliche Durchschnittstemperaturen und Niederschläge werden für die in Kapitel 3.4 besprochene *Köppensche-Klimaklassifikation* benötigt.

Die Temperaturfunktion ist damit eine dreidimensionale Funktion  $t: \mathbb{R}^3 \rightarrow \mathbb{R}$ . Die Eingabeparameter sind  $x$  und  $y$  als Koordinaten und  $s$  als Zeit und die Ausgabe ist die Temperatur in °C. Die Temperaturfunktion verwendet eine dreidimensionale Rauschfunktion  $\text{noise}: \mathbb{R}^3 \rightarrow \mathbb{R}$  und skaliert die generierten Werte auf den Bereich  $[-48, 57]$ .  $-48^\circ\text{C}$  ist die Jahresschnittstemperatur des Südpols<sup>17</sup> und  $57^\circ\text{C}$  ist die bisher gemessene Höchsttemperatur vom Death Valley in Kalifornien<sup>18</sup>. Eine einfache Variante der Funktion wäre demnach  $t(x, y, s) := -48 + \text{noise}(x, y, s) \cdot 105$ .

Die Erde besitzt eine Art Wärmeäquator, welcher entlang den Breitengraden nahe des Äquators einen horizontalen Wärmestrang bildet. Dieser Wärmestrang kühlt sich zu den Polen hin ab. Um diese Wärmeverteilung zu simulieren wird bei der Temperaturfunktion ein künstlicher Wärmeäquator erzeugt, welcher über das Jahr verteilt senkrecht zu den Breitengraden wandert.<sup>19 20</sup> Die Wärmeverteilung ist wichtig um ein präziseres Gesamtergebnis zu erhalten.

17 Klimadaten Südpool: <http://www.weatherbase.com/weather/weather.php3?s=90098&cityname=South-Pole-Antarctica>

18 „World: Highest Temperature“. <http://wmo.asu.edu/world-highest-temperature>

19 Saisonzyklus der Temperatur: [http://cci-reanalyzer.org/animations/scycle/World ERAI\\_T2\\_scycle.gif](http://cci-reanalyzer.org/animations/scycle/World ERAI_T2_scycle.gif)

Im folgenden wird die Position des Wärmeäquators entlang der y-Koordinatenachse als *equatorLine* bezeichnet. Der Einfluss des Äquators auf einen gegebenen Punkt ist ausschließlich abhängig von der y-Koordinate des Punktes und berechnet sich als  $equ(y):=(0.5 - |y - equatorLine|) \cdot 2$  .

Es hat sich nach eigenen Tests als vielversprechend herausgestellt die Gesamttemperatur zu ungefähr 45% anhand des Wärmeäquators und 55% anhand der Rauschfunktion zu berechnen. Dadurch ändert sich die Temperaturfunktion zu

$$t(x, y, s):=(-48 + noise(x, y, s) \cdot 86 + equ(y) \cdot 0.8) / 1.8 .$$

### 3.3 Niederschlagsfunktion

Die Niederschlagskarte wird ebenfalls zeitabhängig generiert, weswegen die Niederschlagsfunktion eine dreidimensionale Funktion ist  $p: \mathbb{R}^3 \rightarrow \mathbb{R}$  . Die Eingabe sind ebenfalls die Koordinaten  $x$  und  $y$  und die Zeit  $s$  . Die Ausgabe ist der monatliche Niederschlag in mm.

Die Funktion generiert Werte in dem Bereich  $[0,600]$  . Der Durchschnittsniederschlag liegt unter anderem in Wüstenregionen bei 0mm <sup>21</sup>, während er in Regenwäldern bei bis zu 600mm liegt.<sup>22</sup> Als Niederschlagsfunktion wird eine einfache Skalierung  $p(x, y, s):=-250 + noise(x, y, s) \cdot 850$  verwendet. Falls der Niederschlag negativ ist, wird er auf Null gesetzt, wodurch man Wüstenregionen mit einem Monatsniederschlag von 0mm erhält.

### 3.4 Köppensche-Klimaklassifikation

Die *Köppensche-Klimaklassifikation* von Vladimir Köppen und Rudolf Geiger unterteilt die Erde in 31 verschiedene Klimazonen. Diese Klimazonen sind nur abhängig von der Temperatur und dem Niederschlag. Deswegen eignet sich diese Klimaklassifikation sehr gut um die generierten Temperatur- und Niederschlagskarten mit der Erde zu vergleichen.

---

20 „Seasonal Temperature“. <http://climvis.org/content/global/anim/gif/tmp2m.gif>

21 Klimadaten Villa Cisneros(Sahara): <http://www.weatherbase.com/weather/weather.php3?s=600960&cityname=Villa-Cisneros-Oued-Ed-Dahab-Lagouira-Western-Sahara>

22 „Welches Klima herrscht im Regenwald?“. <http://www.faszination-regenwald.de/info-center/allgemein/klima.htm>

Für die Bestimmung der Klimazone werden folgende Eigenschaften benötigt:

- $T_{min}$  die Temperatur des kältesten Monats
- $T_{max}$  die Temperatur des wärmsten Monats
- $T_{ann}$  die Jahresdurchschnittstemperatur
- $P_{min}$  der Niederschlag im trockensten Monat
- $P_{smin} / P_{wmin}$  der Niederschlag im trockensten Sommer/Winter Monat
- $P_{max}$  der Niederschlag im feuchtesten Monat
- $P_{smax} / P_{wmax}$  der Niederschlag im feuchtesten Sommer/Winter Monat
- $P_{ann}$  der gesamt Jahresniederschlag
- $P_{th} = \begin{cases} 2 \cdot T_{ann} & , \text{falls } 2/3 \text{ des Niederlags im Winterhalbjahr auftreten} \\ 2 \cdot T_{ann} + 28 & , \text{falls } 2/3 \text{ des Niederlags im Sommerhalbjahr auftreten} \\ 2 \cdot T_{ann} + 14 & , \text{sonst} \end{cases}$

$P_{th}$  ist ein für die Klimaklassifikation benötigter Schwellenwert.

Anhand dieser Eigenschaften, lassen sich Regionen in fünf Hauptgruppen A bis E einteilen. Diese Hauptgruppen werden dann zu 31 Subgruppen weiter unterteilt. Die Gruppen werden wie folgt bestimmt:

Typ	Beschreibung	Kriterium
<b>A</b>	<b>Äquatoriales Klima</b>	$T_{min} \geq 18^\circ C$
Af	Regenwald	$P_{min} \geq 60\text{mm}$
Am	Monsun	$P_{ann} > 25 \cdot (100 - P_{min})$
As	Savanne mit trockenen Sommern	$P_{min} < 60\text{mm}$ im Sommer
Aw	Savanne mit trockenen Wintern	$P_{min} < 60\text{mm}$ im Winter
<b>B</b>	<b>Trockenes Klima</b>	$P_{ann} < 10 \cdot P_{th}$
Bsh	Heiße Steppe	$P_{ann} > 5 \cdot P_{th}$ und $T_{ann} \geq 18^\circ C$
Bsk	Kalte Steppe	$P_{ann} > 5 \cdot P_{th}$ und $T_{ann} < 18^\circ C$
Bwh	Heiße Wüste	$P_{ann} \leq 5 \cdot P_{th}$ und $T_{ann} \geq 18^\circ C$
Bwk	Kalte Wüste	$P_{ann} \leq 5 \cdot P_{th}$ und $T_{ann} < 18^\circ C$

<b>C</b>	<b>Warm gemäßiges Klima</b>	$-3^{\circ}C < T_{min} < 18^{\circ}C$
<b>D</b>	<b>Kalt gemäßiges Klima</b>	$T_{min} < -3^{\circ}C$
<b>E</b>	<b>Polar Klima</b>	$T_{max} < 10^{\circ}C$
ET	Tundra	$0^{\circ}C \leq T_{max} < 10^{\circ}C$
EF	Frost	$T_{max} < 0^{\circ}C$

Für die Hauptgruppen **C** und **D** gibt es zwei weitere Unterteilungsstufen. Die erste Untergliederung ist

s	Trockene Sommer	$P_{smin} < P_{wmin}$ , $P_{wmax} > 3 \cdot P_{smin}$ und $P_{smin} < 40\text{mm}$
w	Trockene Winter	$P_{wmin} < P_{smin}$ und $P_{smax} > 10 \cdot P_{wmin}$
f	Humid	Weder (s) noch (w)

und die zweite Untergliederung ist

a	Heiße Sommer	$T_{max} \geq 22^{\circ}C$
b	Warm Sommer	Nicht (a) und mindestens 4 monatliche Durchschnittstemperaturen sind $\geq 10^{\circ}C$
c	Kalte Sommer und Winter	Weder (a) noch (b) und $T_{min} > -38^{\circ}C$
d	Sehr kontinental	Weder (a) noch (b) und $T_{min} \leq -38^{\circ}C$

Folglich gibt es beispielsweise die Gruppe *Dsa* ein kalt gemäßiges Klima mit trockenen, heißen Sommern oder *Cfc* ein warm gemäßiges, humides Klima mit kalten Sommern und Wintern.<sup>23</sup>

## 4. Visualisierung

In dem vorherigen Kapitel wurden verschiedene Methoden für die Berechnung von zweidimensionalen, thematischen Karten beschrieben. Dieses Kapitel behandelt die Visualisierung dieser generierten Karten. Dabei ist es vor allem wichtig, dass sie über-

<sup>23</sup> Kottek, Markus et al.(2006): „World Map of the Köppen-Geiger climate classification updated“

sichtlich und intuitiv sind. So werden beispielsweise blaue Farbtöne mit Kälte und Nässe assoziiert, während rote Töne mit Wärme und gelbe mit Trockenheit in Verbindung gebracht werden.<sup>24</sup>

Für die Farbverläufe wird der HSV-Farbraum verwendet, welcher aus einem Farbwert (engl. hue), der Farbsättigung (engl. saturation) und einem Helligkeitswert (engl. value) besteht. Der Farbraum orientiert sich an der Farbwahrnehmung des Menschen und ermöglicht die Erschaffung von gleichmäßigeren, „schöneren“ Farbverläufen als im RGB-Farbraum.<sup>25</sup>

## 4.1 Höhenkarte

Die Höhenkarte basiert auf den von der Höhenfunktion berechneten Werten und wird in Graustufen dargestellt. Dabei ist Schwarz die minimale Höhe und Weiß die maximale Höhe. Diese Färbung wird im Allgemeinen für Höhenkarten verwendet.<sup>26</sup>

---

24 Eckert-Greifendorff, Max(2012): „Kartographie: Ihre Aufgaben und Bedeutung für die Kultur der Gegenwart“, S.42, 164

25 Mascolus, Wilfried(2006): „Farbe in der Computergrafik“, S.6 f.

26 Allen, Jesse(2005): „Visible Earth“. <http://visibleearth.nasa.gov/view.php?id=73934>

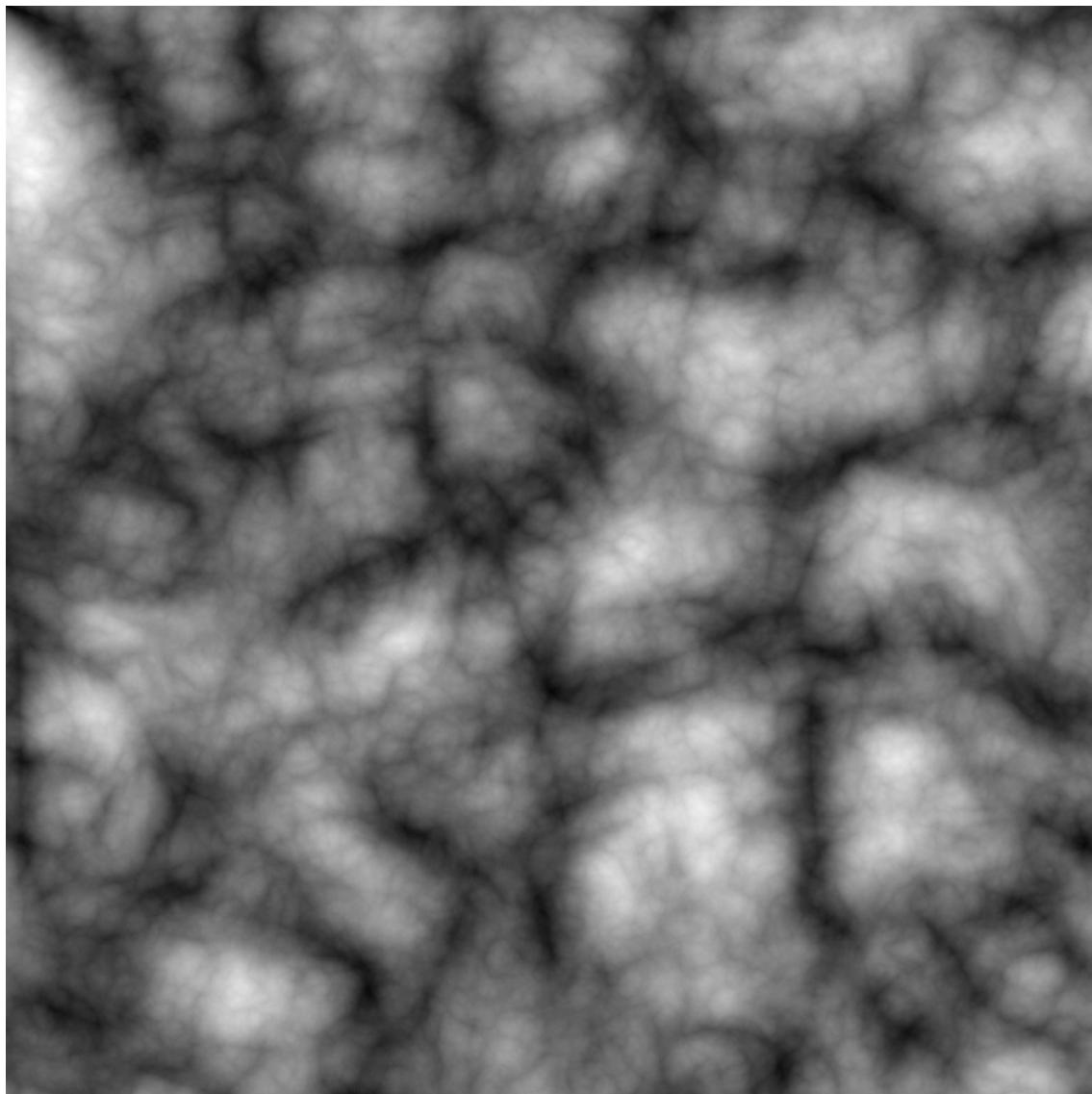


Abbildung 4.1: Screenshot aus dem Programm vom 28.03.2016: Darstellung der Höhenkarte

**Höhe [m]**

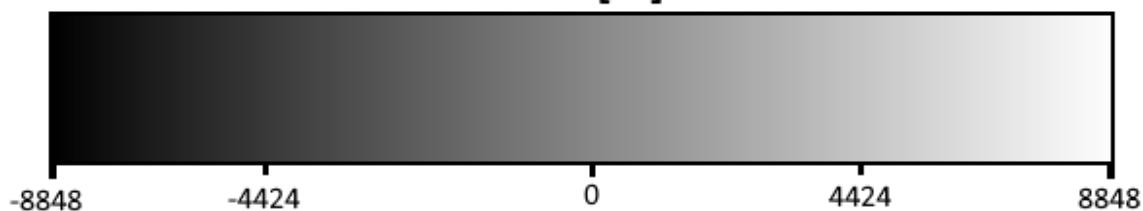


Abbildung 4.2: Legender der generierten Höhenkarten

## 4.2 Temperaturkarte

Die Temperaturkarte basiert auf den von der Temperaturfunktion berechneten Werten. Die Karte wird in einem Farbverlauf von Blau über Grün zu Rot dargestellt. Die Farbgebung Blau für Kälte und Rot für Wärme gilt als intuitiv und wird auch im Alltag ver-

wendet, beispielsweise bei Wasserhähnen. Die gemäßigten Regionen dazwischen werden grün gefärbt.<sup>27</sup>

Wegen des Wärmeäquators sind die Temperaturschwankungen innerhalb eines Breitengrades nur gering. Damit auch minimale Schwankungen gut erkennbar sind, wurde deshalb ein fließender Farbübergang für die Temperaturkarte gewählt.

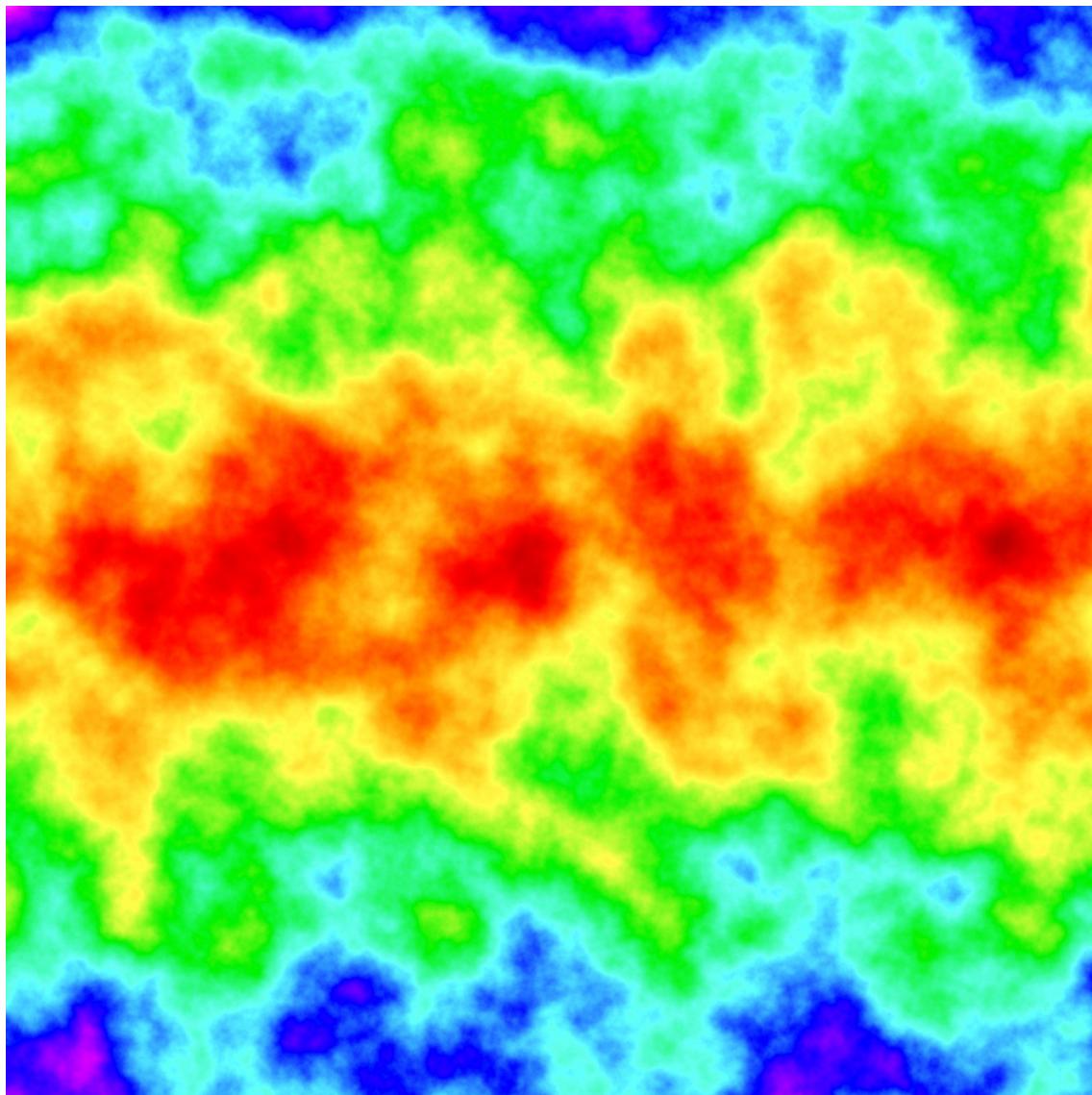


Abbildung 4.3: Screenshot aus dem Programm vom 28.03.2016: Darstellung der Temperaturkarte

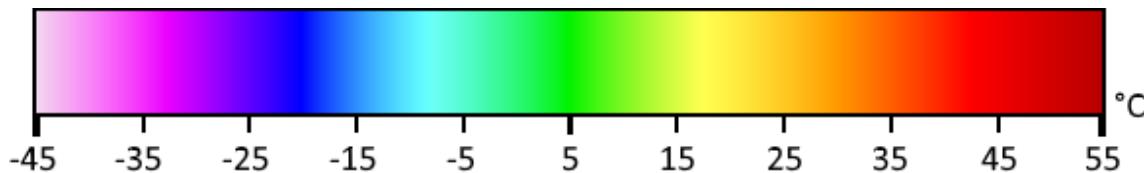


Abbildung 4.4: Legende der generierten Temperaturkarten

27 Eckert-Greifendorff, Max(2012): „Kartographie: Ihre Aufgaben und Bedeutung für die Kultur der Gegenwart“, S.42

### 4.3 Niederschlagskarte

Die Niederschlagskarte basiert auf den von der Niederschlagsfunktion berechneten Werten. Die Karte wird in einem stufenweisen Farbübergang von Blau für Feuchtigkeit zu Gelb für Trockenheit dargestellt. Die Farbe Blau wird mit Wasser und dadurch mit Feuchtigkeit assoziiert. Die trockenen Regionen sind gelb/braun, weil diese Farben mit Wüsten/Steppen in Verbindung gebracht werden. Für die Temperaturkarte wurden stufenweise Farbübergänge gewählt, damit man schneller erkennt in welchen Regionen viel bzw. wenig Niederschlag fällt.<sup>28</sup>

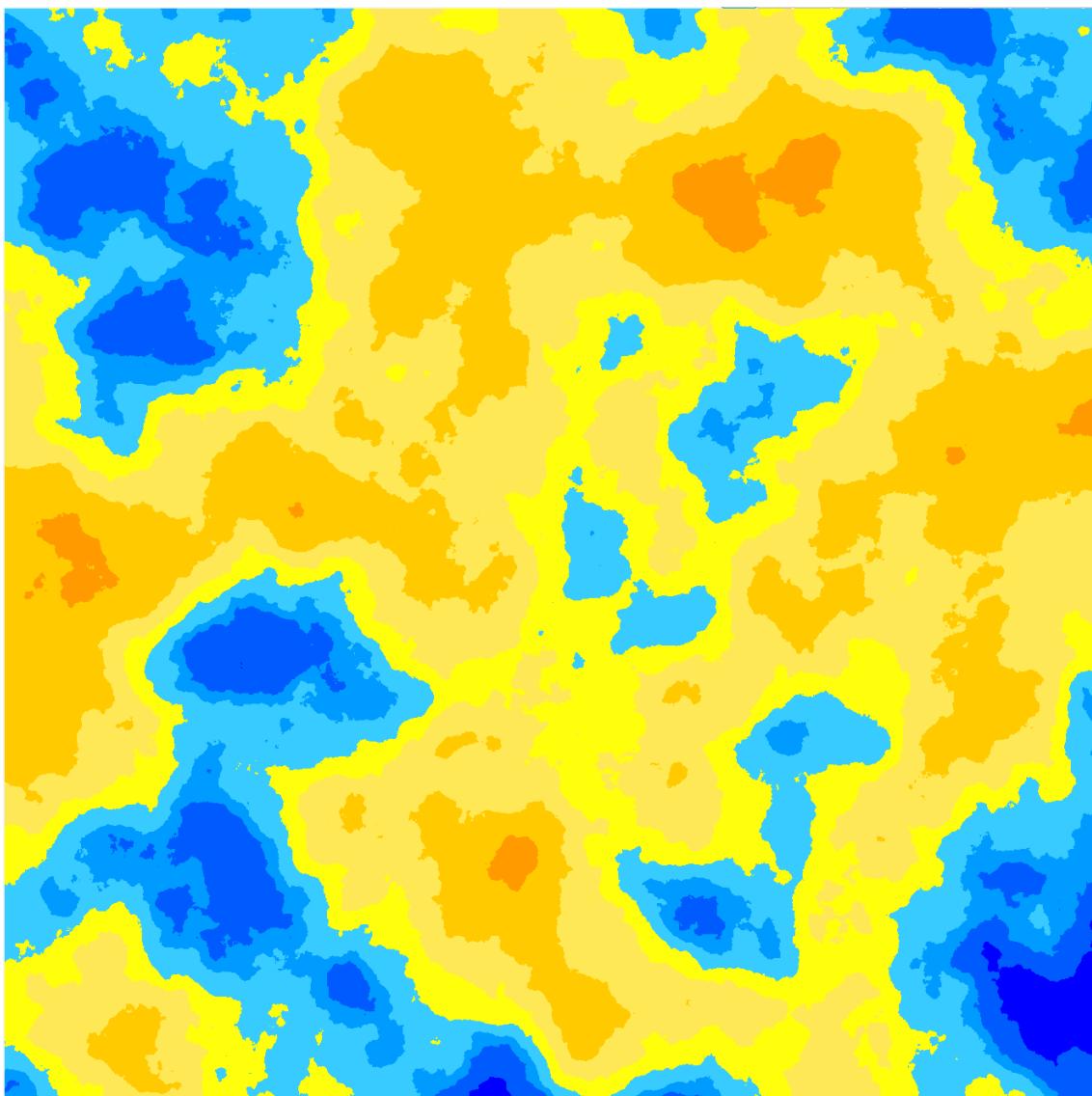


Abbildung 4.5: Screenshot aus dem Programm vom 28.03.2016: Darstellung der Niederschlagskarte

---

<sup>28</sup> Eckert-Greifendorff, Max(2012): „Kartographie: Ihre Aufgaben und Bedeutung für die Kultur der Gegenwart“, S.164



Abbildung 4.6: Legende der generierten Niederschlagskarten

#### 4.4 Köppen-Karte

Die *Köppensche-Klimaklassifikation* wird über zwei unterschiedliche Karten visualisiert. Auf der ersten Karte werden lediglich die fünf Hauptgruppen abgebildet und auf der zweiten Karte werden alle 31 Klimazonen dargestellt.

Die Köppen-Karte wird in Kapitel 6 als Vergleichskriterium für die Realitätsnähe der generierten Höhen- und Temperaturwerte eingesetzt. Deswegen orientiert sich die Färbung der Karte an der Färbung von „World Map of the Köppen-Geiger climate classification updated“<sup>29</sup> einer *Köppensche-Klimaklassifikation* der Erde. Es wurde dieselbe Färbung gewählt, damit sich die beiden Karten ohne großen Aufwand vergleichen lassen.

Die Farbwahl ist Rot für das äquatoriale Klima und Blau für die kalten Polarklimate. Die wüstenartigen Trockenklima sind gelb und die gemäßigte Klima sind grün bzw. lila.

---

<sup>29</sup> Kottek, Markus et al.(2006): „World Map of the Köppen-Geiger climate classification updated“, S.261

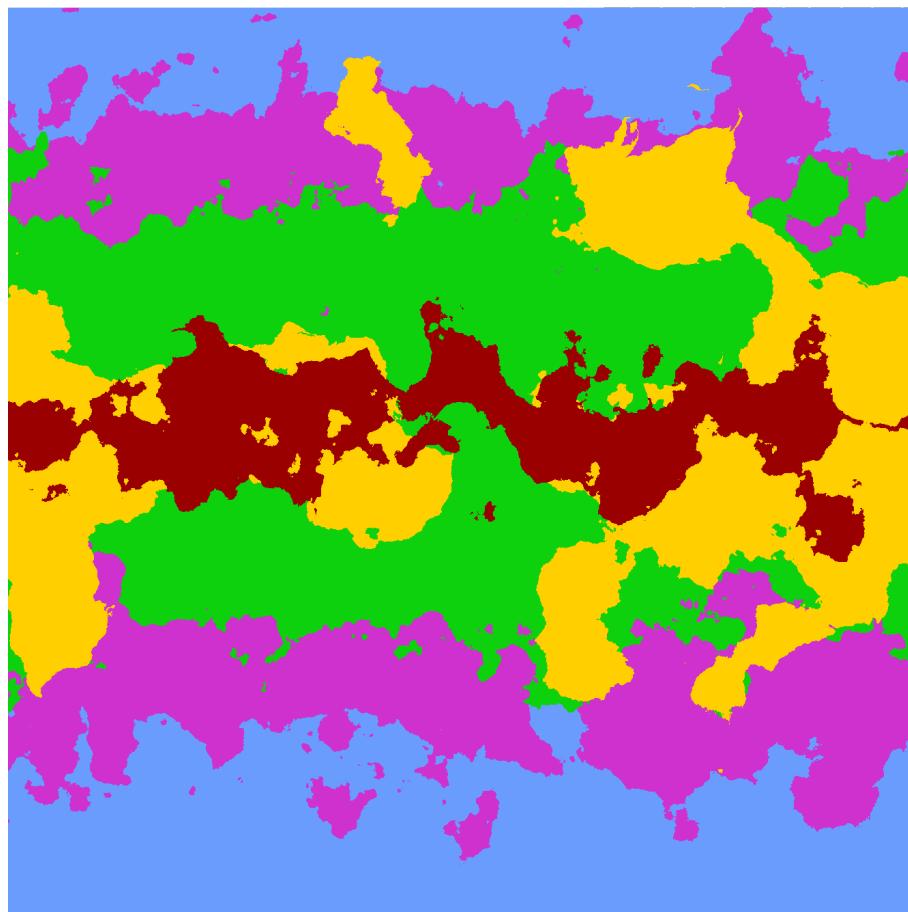


Abbildung 4.7: Screenshot aus dem Programm vom 28.03.2016: Darstellung der fünf Hauptgruppen der *Köppensche-Klimaklassifikation* ohne Meeresspiegel

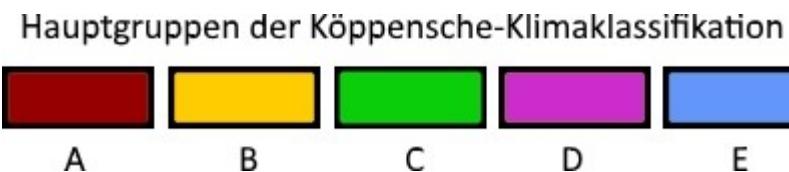


Abbildung 4.8: Legende der Köppen-Karte mit den fünf Hauptgruppen

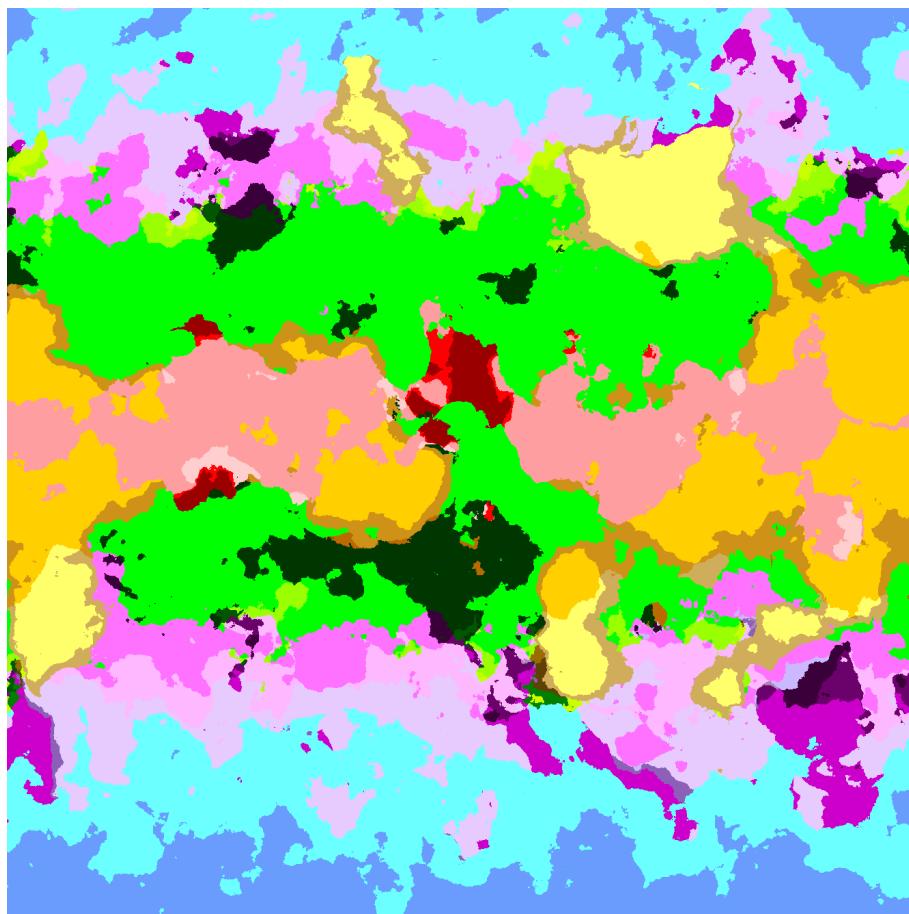


Abbildung 4.9: Screenshot aus dem Programm vom 28.03.2016: Darstellung der *Köppensche-Klimaklassifikation* ohne Meeresspiegel

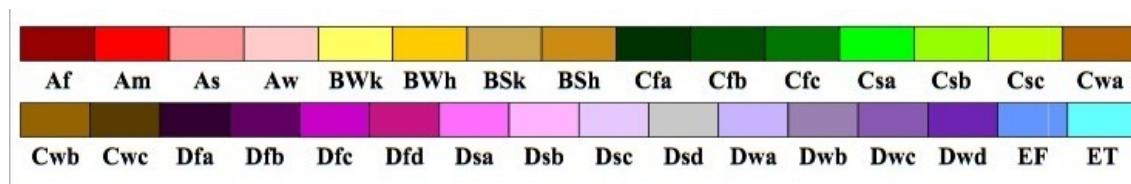


Abbildung 4.10: Legende der Köppen-Karte mit allen Gruppen (übernommen aus „World Map of Köppen-Geiger Climate Classification“)

## 5. Das Programm

In diesem Kapitel wird auf die entwickelte Software Bezug genommen. Es werden wichtige Programmfeatures so wie Eigenschaften des programmierten Codes vorgestellt. Zuerst wird die Struktur des Programms behandelt und danach werden die einzelnen Programmteile genauer beschrieben.

Eines der Ziele der Bachelorarbeit ist es das Projekt offen zu gestalten. Es soll mit wenig Aufwand um neue Funktionen erweitert werden können. Diese Erweiterungen sind vor allem neue Rauschfunktionen und visuelle Karten.

## 5.1 Programmkernel

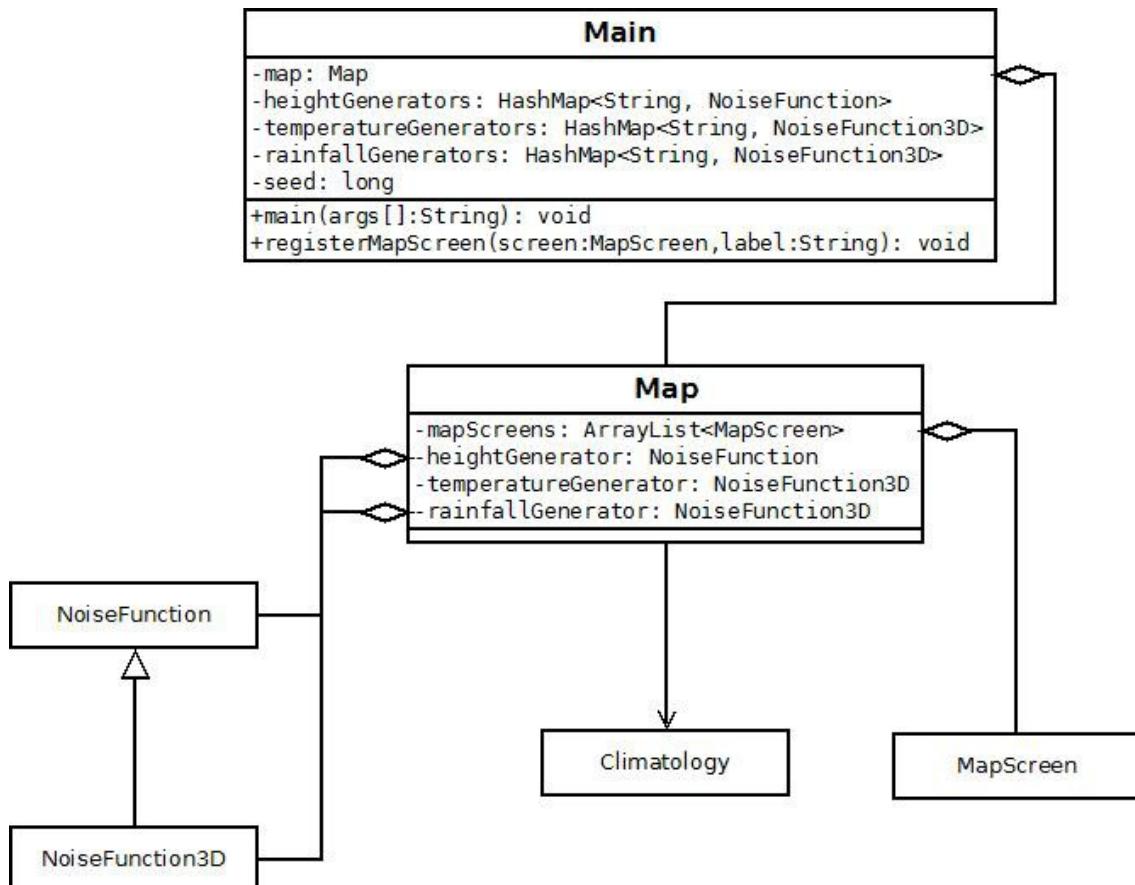


Abbildung 5.1: UML Diagramm mit den wichtigsten Funktionen des Programmkerne

Der Kern des Programms sind die Klassen *Main* und *Map*. Die *Main*-Klasse kümmert sich um die gesamte Verwaltung der Software und um die Kommunikation zwischen dem Programm und dem Anwender.

Eine Aufgabe der *Main*-Klasse ist die Verwaltung aller vorhandenen Rauschfunktionen und visuellen Karten. Wenn eine neue Rauschfunktion oder visuelle Darstellung in das Programm eingebunden werden soll, muss diese in der *Main*-Klasse registriert werden.

Eine weitere Aufgabe der Klasse ist die Kommunikation zwischen dem Programm und dem Anwender. Es wird das Benutzerinterface initialisiert und gestaltet. Außerdem werden alle Benutzereingaben und Grafikausgaben behandelt.

Die Benutzeroberfläche ist wie folgt aufgebaut:

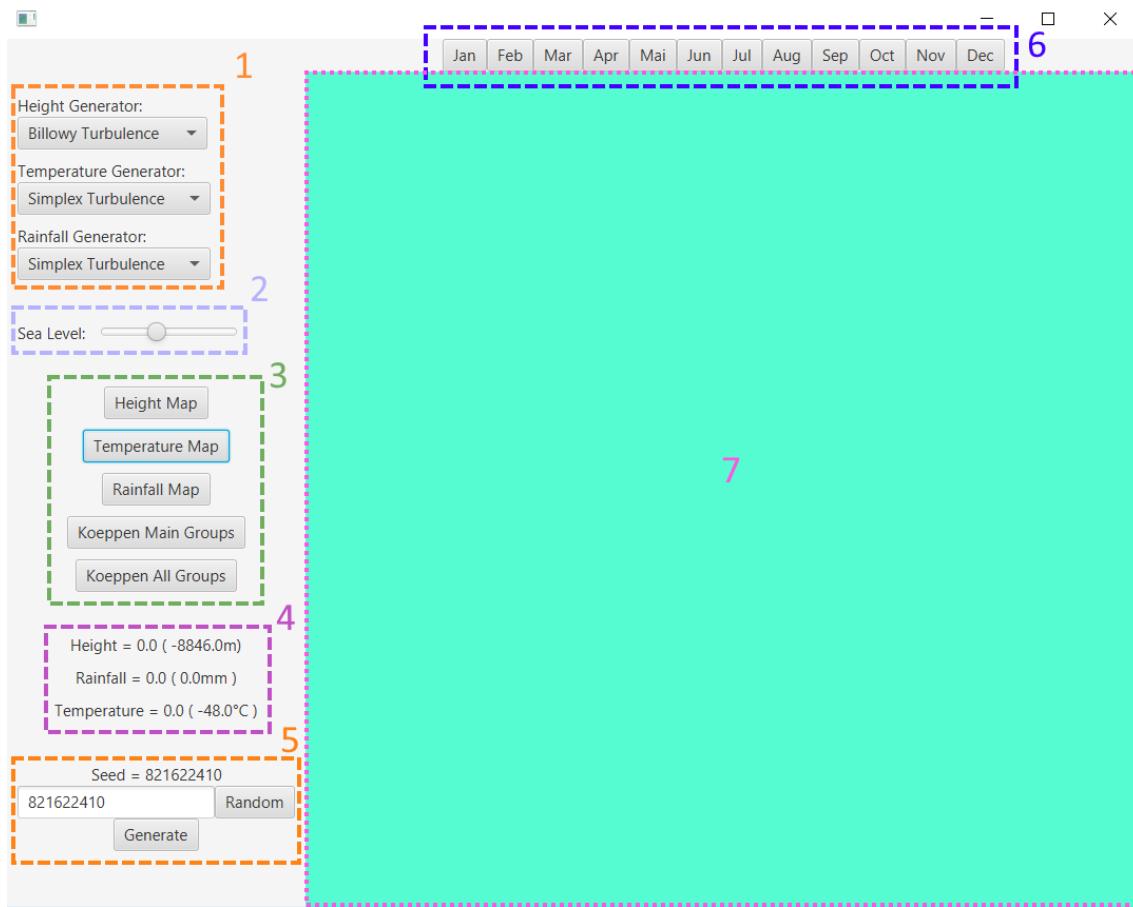


Abbildung 5.2: Screenshot aus dem Programm vom 04.04.2016: Die Benutzeroberfläche mit Markierungen der einzelnen Bereiche

1. Auswahl der zu verwendenden Rauschfunktionen
2. Regler für die Höhe des Meeresspiegels
3. Auswahl der angezeigten Karte
4. Informationen über den zuletzt auf der Karte ausgewählten Punkt
5. Informationen über den Seed
6. Monatsauswahl
7. Fläche für die Darstellung der Karte

Die Klasse *Map* verwaltet und generiert die Karten, welche zuvor in dieser Arbeit beschrieben wurden. Es werden Rauschfunktionen vom Typ *NoiseFunction* bzw. *NoiseFunction3D* verwendet um zufällige Rauschwerte zu generieren. Diese werden mithilfe

der Klasse *Climatology* in geowissenschaftliche Werte umgewandelt und zuletzt als Instanzen von *MapScreen* visuell dargestellt. Alle diese Schritte finden in Hintergrundprozessen statt, damit das Programm einerseits schneller läuft und andererseits nicht den Prozess der Benutzeroberfläche blockiert.

## 5.2 Rauschfunktionen

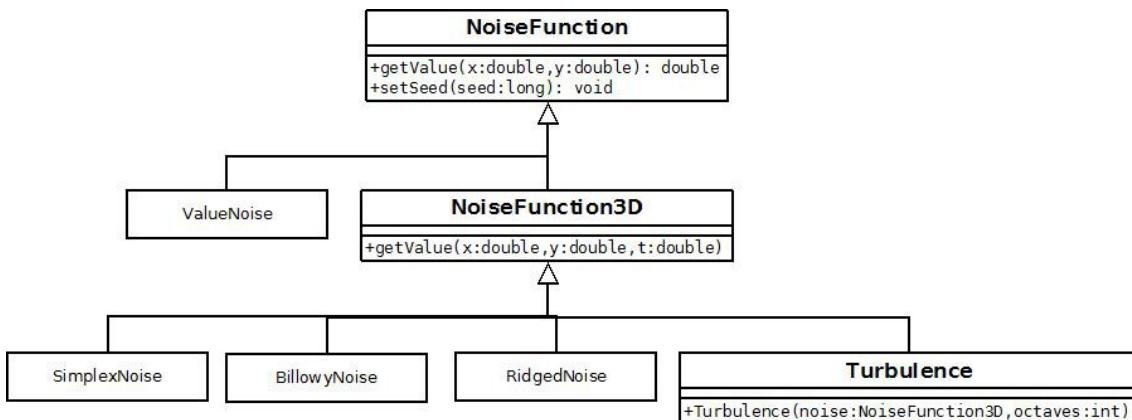


Abbildung 5.3: UML Diagramm mit den wichtigsten Faktoren der Rauschfunktionen

Jede von dem Programm verwendete Rauschfunktion muss das Interface *NoiseFunction* implementieren und die Methoden *double getValue(double x, double y)* und *void setSeed(long seed)* überschreiben.

Die Methode *double getValue(double x, double y)* entspricht der in Kapitel 2 beschriebenen zweidimensionalen Rauschfunktion. Die Eingaben sind also die Kartenkoordinaten  $x, y \in [0,1]$  und die Ausgabe ist ein Rauschwert  $m_{x,y} \in [0,1]$ . *setSeed(long seed)* wird aufgerufen, wenn der verwendete *Seed* geändert wurde und die Funktion neu berechnet werden soll.

Ein *Seed* ist eine einzelne Zahl, welche verwendet wird um weitere Zufallswerte und letztendlich alle Karten eindeutig zu berechnen. Er stellt somit einen Anfangswert für die Rauschfunktionen dar. Bei demselben *Seed* sind alle berechneten Rauschfunktionen und visuelle Karten dieselben.

Falls man die Temperatur oder Niederschlagswerte generieren möchte, benötigt man eine dreidimensionale Rauschfunktion. In diesem Fall muss man das Interface *NoiseFunction3D*, welches *NoiseFunction* erweitert, implementieren und die Funktion *double getValue(double x, double y, double t)* überschreiben. Diese Funktion stellt die dreidi-

dimensionale Rauschfunktion dar, welche für Temperatur- und Niederschlagsberechnungen benötigt wird.

Wenn Turbulenzen generiert werden sollen, muss eine Instanz von *Turbulence(Noise-Function noise, int octaves)* erstellt werden. Die Anzahl der Iterationsschritte wird dabei durch *octaves* festgelegt und *noise* ist die Rauschfunktion, welche iterativ aufgerufen werden soll.

Die Rauschfunktionen müssen dann in der *Main*-Klasse in die Hash-Maps *heightGenerators*, *temperatureGenerators* und *rainfallGenerators* eingefügt werden, damit sie im Programm auswählbar sind. Jede dieser Hashtabellen enthält eine Menge von Rauschfunktionen, welche für die Generierung der jeweiligen Eigenschaft verwendet werden.

Folgende Rauschfunktionen sind bereits in dem Programm implementiert:

- Value Noise (nur zweidimensional implementiert)
- Value Turbulence (nur zweidimensional implementiert)
- Simplex Noise<sup>30</sup>
- Simplex Turbulence
- Billowy Noise
- Billowy Turbulence
- Ridged Noise
- Ridged Turbulence

---

<sup>30</sup> Basiert auf Open Simplex Noise von Kurt Spencer

<https://gist.github.com/KdotJP/b1270127455a94ac5d19>

### 5.3 Visuelle Karten

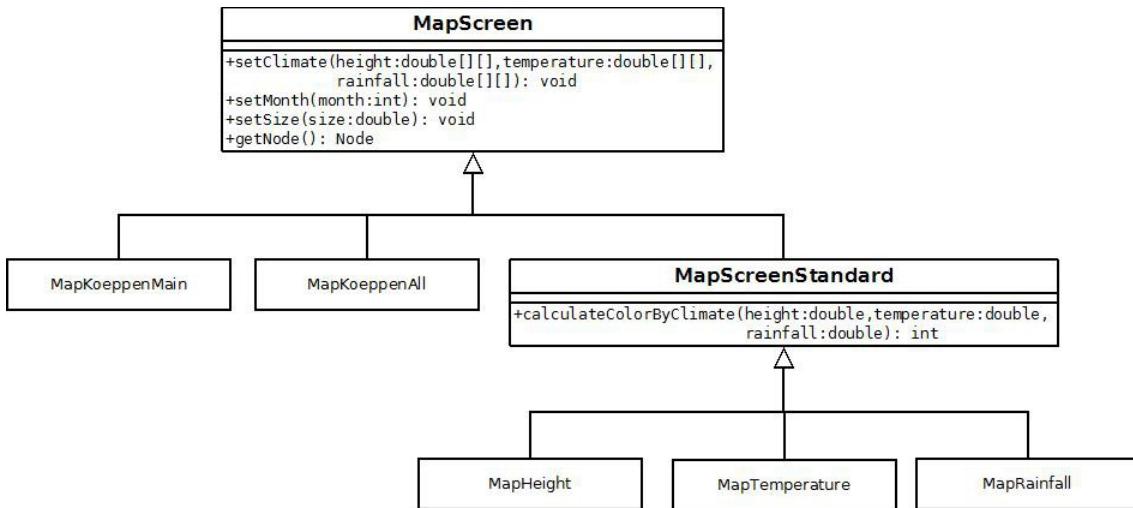


Abbildung 5.4: UML Diagramm mit den wichtigsten Faktoren der Kartenvizualisierung

Jede von dem Programm dargestellte Karte implementiert das Interface *MapScreen*. Für *MapScreen* müssen vier Methoden überschrieben werden. Einerseits die drei Methoden *setSize(double size)*, *setClimate(double[][] height, double[][] rainfall, double[][] temperature)*, *setMonth(int month)* ohne Rückgabewert. Diese Funktionen werden aufgerufen, wenn sich der jeweilige Parameter ändert und die Karte neu gezeichnet werden soll. So wird *setSize* aufgerufen, wenn sich die Größe des Anzeigefensters ändert. *setClimate* wird aufgerufen, wenn sich einer der generierten Geodaten verändert und *setMonth* wird aufgerufen, wenn der angezeigte Monat geändert wurde. Andererseits muss die Methode *getNode* überschrieben werden. Die Methode gibt einen JavaFx Node wieder, welcher an der Stelle der Landkarte (Abbildung 5.1 Fläche 7) in das Programm eingebunden wird.

Für eine einfache und schnelle Implementierung neuer Karten kann von der Klasse *MapScreenStandard* geerbt werden. Die Klasse erweitert das Interface *MapScreen*. In der Klasse *MapScreenStandard* muss lediglich die Methode *calculateColorByClimate* überschrieben werden, welche die Höhe, die Temperatur und den Niederschlag eines Punktes als Eingabe erhält und einen RGB Wert als Ausgabe zurückgibt.

## 6. Schlussbetrachtung

In diesem Kapitel werden die Ergebnisse der Bachelorarbeit zusammengefasst und bewertet. Dafür werden die zuvor erzeugten Daten und Karten mit der Erde verglichen. Anhand des Vergleiches wird erklärt, inwieweit das Programm ein Erfolg ist und an welchen Stellen man das Projekt noch verbessern und erweitern kann. Zuletzt werden noch einige Zukunftsaussichten gegeben.

### 6.1 Vergleich mit Karten der Erde

In diesem Abschnitt werden die zuvor erstellten Karten und Daten mit der Erde verglichen. Auf diese Weise soll ermittelt werden, wie realistisch die generierten Karten sind und welche der vorgestellten Algorithmen die Welt am besten simulieren.

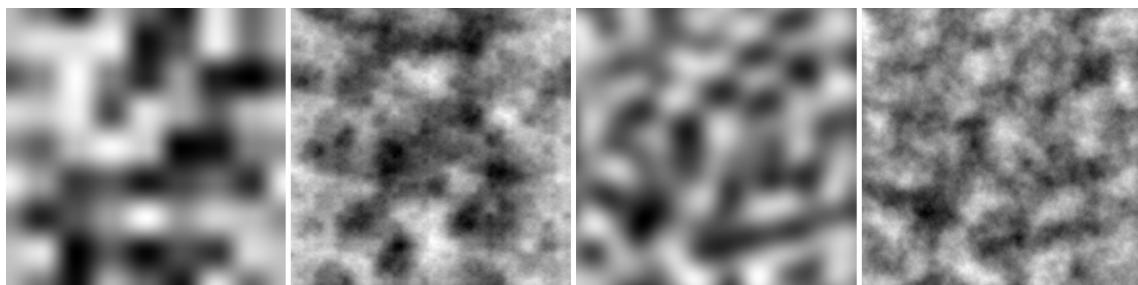
Bei der Generierung von Höhenkarten stellte sich heraus, dass Turbulenzen besser geeignet sind als einfache Rauschfunktionen. Turbulenzen erzeugen unter anderem eine Fraktal ähnliche Küstenlinie wie in der echten Welt.<sup>31</sup>

Weiterhin hat sich *Billowy Turbulence* als der beste Algorithmus herausgestellt. *Billowy Turbulence* generiert Höhenkarten, welche durchzogen sind von verästelten Schluchten. Diese Schluchten entstehen durch Erosion auch auf der Erde.<sup>32</sup> Die Eigenschaften von *Billowy Turbulence* führen aber nicht nur zu realistischen Schluchten und Küstenlinien, sondern auch dazu, dass vereinzelt Inseln im Meer und häufiger in Küstennähe und zwischen Kontinenten entstehen.

---

31 Mandelbrot, Benoît B.(1967): „How long is the coast of Britain?“, S.636-638.

32 Perron, J. Taylor: „Patterns in river networks“. <http://web.mit.edu/perron/www/research.html>

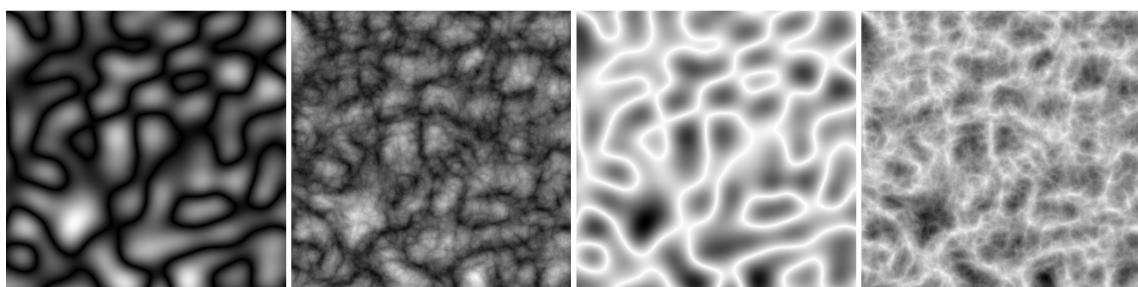


Value Noise

Value Turbulence

Simplex Noise

Simplex Turbulence



Billowy noise

Billowy Turbulence

Ridged Noise

Ridged Turbulence

Abbildung 6.1: Screenshot aus dem Programm vom 10.04.2016: Höhenkarten generiert mit unterschiedlichen Rauschfunktionen

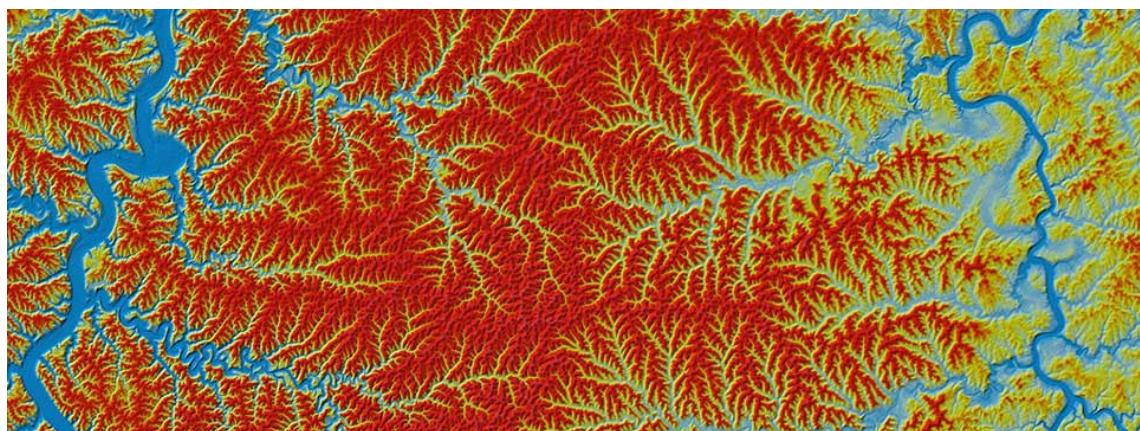


Abbildung 6.2: Veranschaulichung von verästelten Tälern der Erde. <http://web.mit.edu/perron/www/index.html>

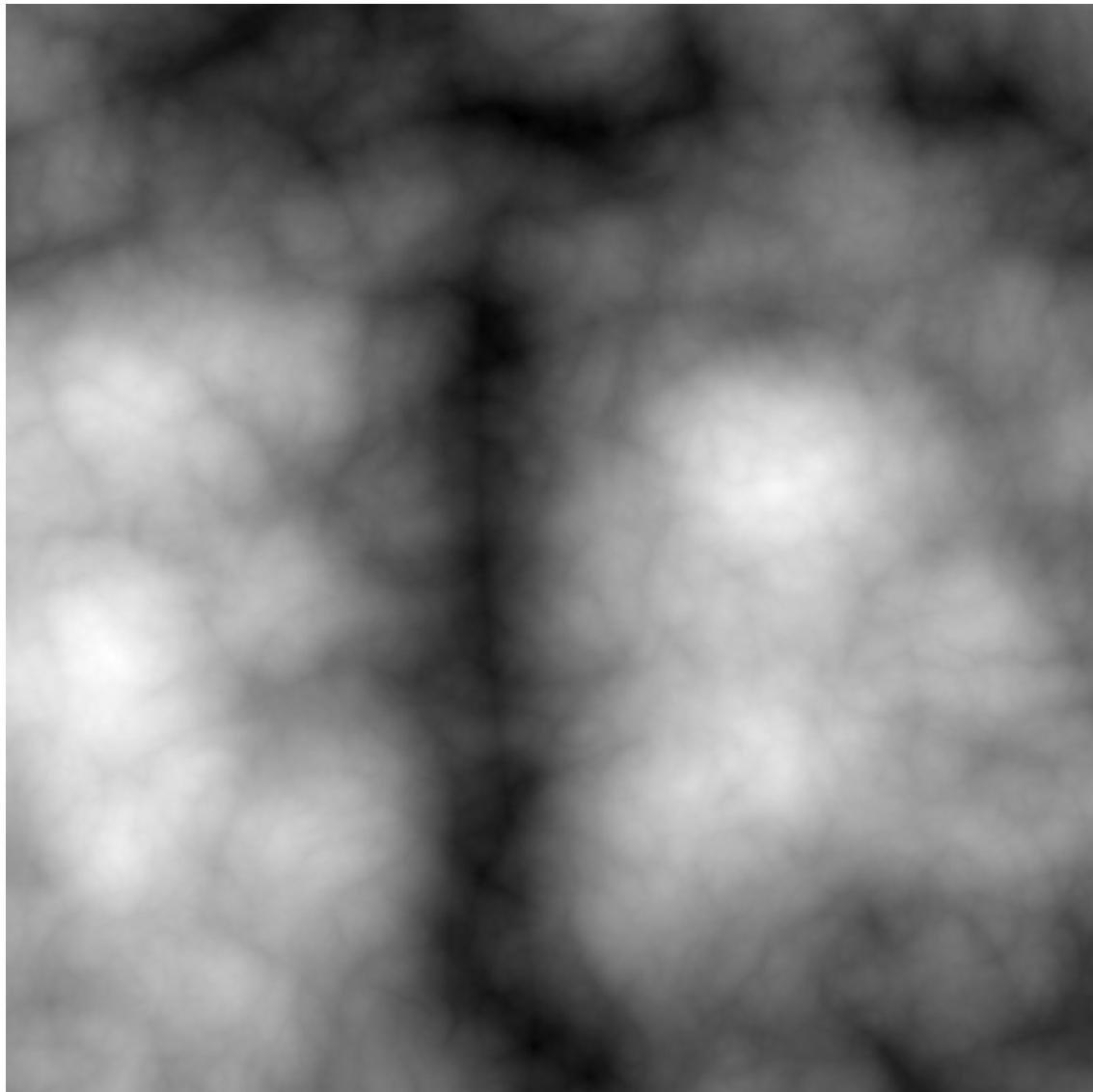


Abbildung 6.3: Screenshot aus dem Programm vom 10.04.2016: Zoom auf eine mit *Billowy Turbulence* generierte, verästelte Schlucht für den Vergleich mit Abbildung 6.2 .

Die Realitätsnähe der Temperatur und Niederschlagsfunktionen wird anhand der *Köppensche-Klimaklassifikation* bewertet. Die Aufteilung der Klimazonen in den generierten Karten ähnelt stark der Verteilung bei der Erde. So liegen die äquatorialen Klimate um den Äquator verteilt. In der Nähe des Äquators kommen Wüstenklimate vor und um die Wüstenklimate verteilt herrscht warm gemäßiges Klima. An den Polen herrscht polares Klima und in der Nähe der Pole gibt es großteils kalt gemäßigte Klimate.

Auch bei Temperatur und Niederschlagskarten hat sich der fraktale Aufbau von Turbulenzen bewährt. Bei nicht turbulenten Rauschfunktionen sind in den erzeugten Karten deutlich unnatürliche Muster erkennbar. Außerdem erzeugen sowohl *Ridged Turbulence* wie auch *Billowy Turbulence* schlauchartige Karten, was weder für Temperatur- noch

für Niederschlagskarten typisch ist. Hingegen generiert *Simplex Turbulence* gute Niederschlags- und Temperaturkarten.

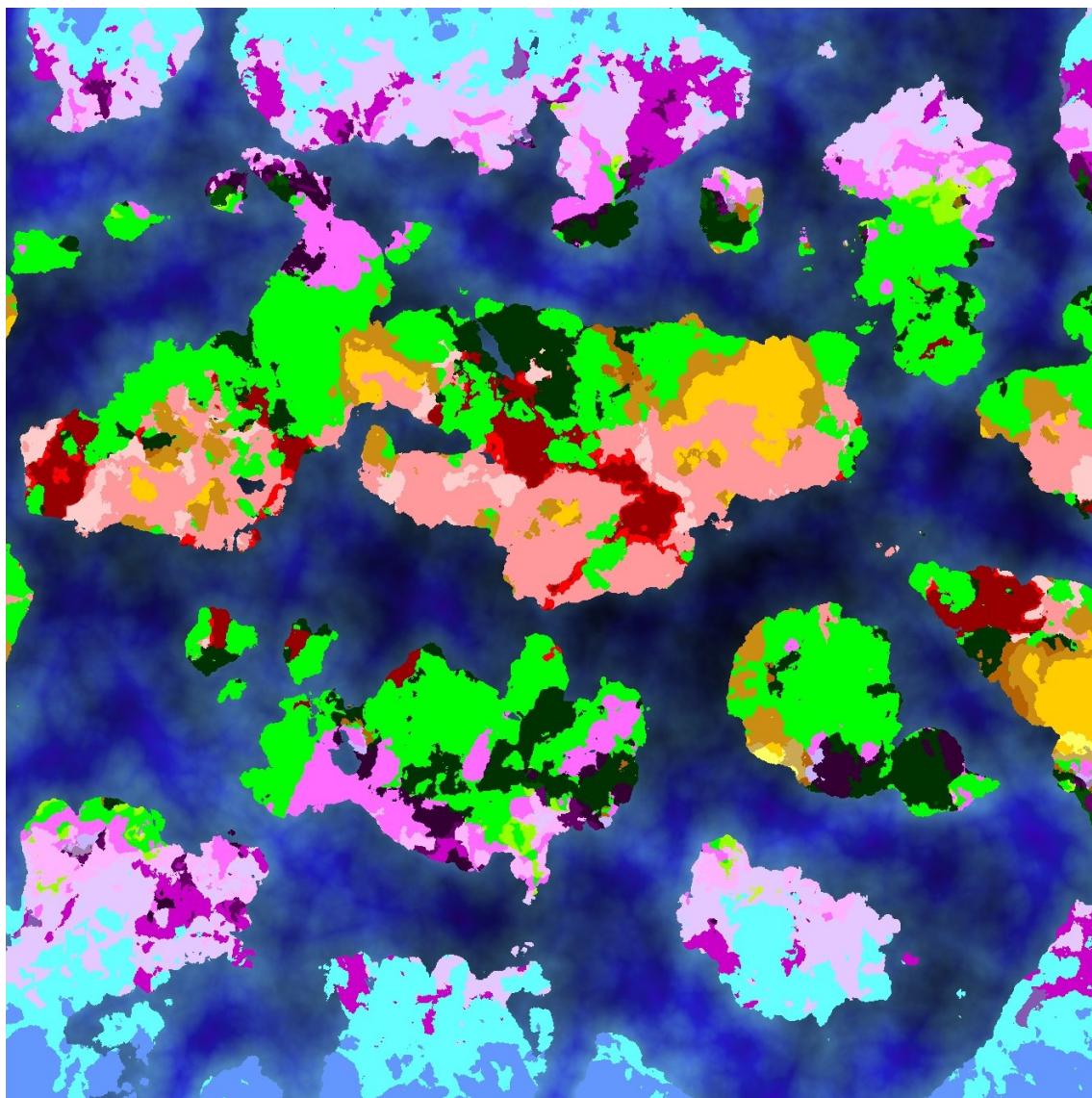


Abbildung 6.4: Screenshot aus dem Programm vom 10.04.2016: Eine generierte Köppen-Karte für den Vergleich mit Abbildung 6.5

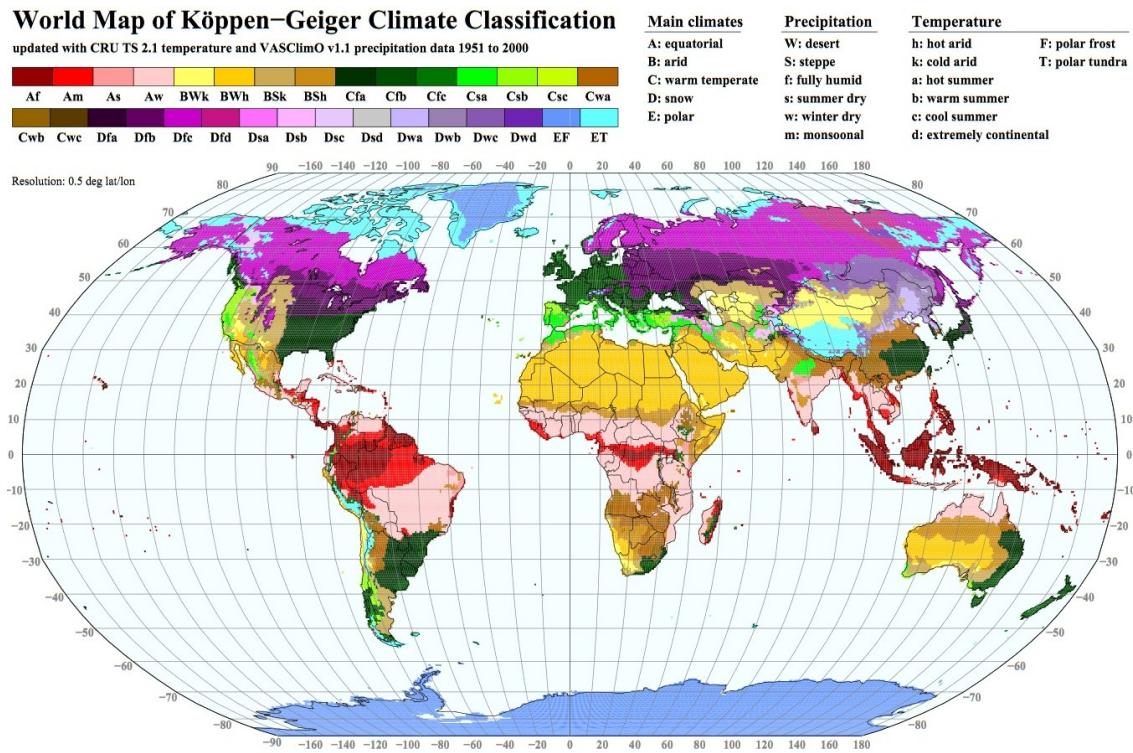


Abbildung 6.5: *Köppensche-Klimaklassifikation* der Erde (übernommen aus „World Map of Köppen-Geiger Climate Classification“)

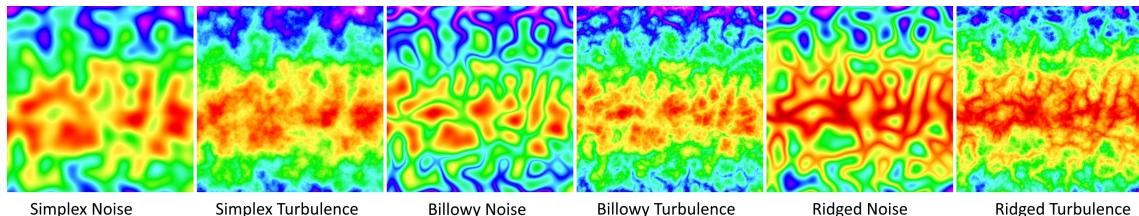


Abbildung 6.6: Screenshots aus dem Programm vom 13.04.2016: Temperaturkarten generiert mit unterschiedlichen Rauschfunktionen

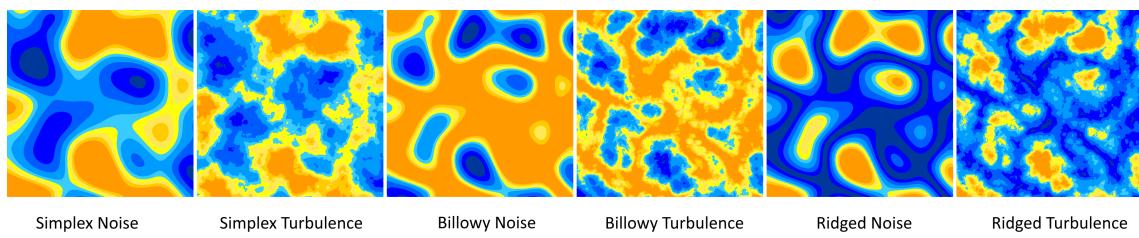


Abbildung 6.7: Screenshots aus dem Programm vom 13.04.2016: Niederschlagskarten generiert mit unterschiedlichen Rauschfunktionen

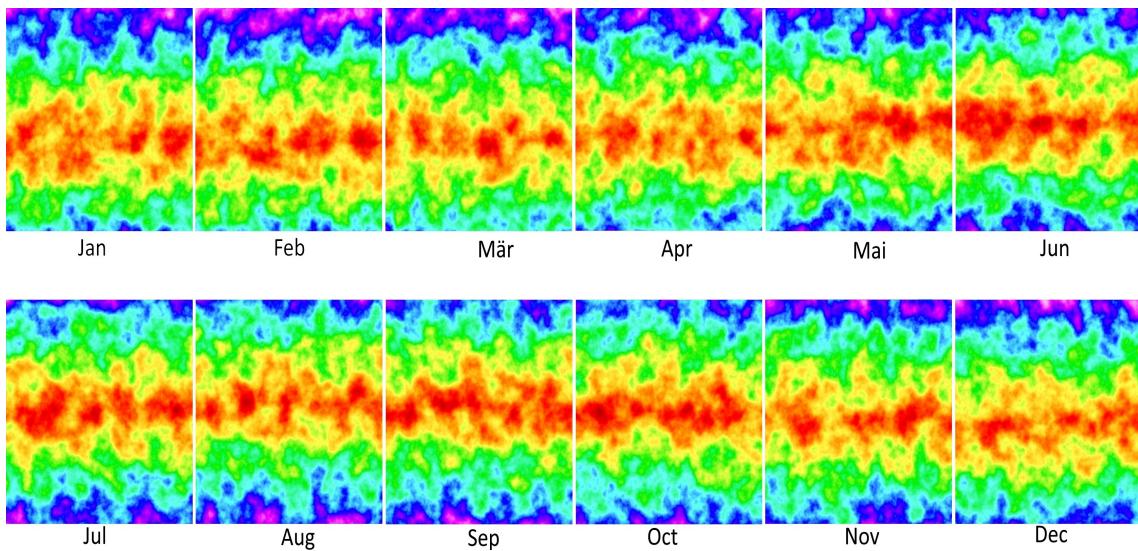


Abbildung 6.8: Screenshots aus dem Programm vom 13.04.2016: Für den Vergleich verwendete monatliche Temperaturkarten. Die Karten wurden verglichen mit „Seasonal Temperature“  
<http://climvis.org/content/global/anim/gif/tmp2m.gif>.

## 6.2 Fazit

Die in dieser Bachelorarbeit erschaffenen Karten und Daten weisen die wichtigsten Merkmale der Erde auf und das entwickelte Programm stellt somit eine solide Basis für die Generierung von realistischen, künstlichen Landkarten dar. Weil es für die Bewertung des Realismus einer Karte keine Skala gibt, entspricht dies jedoch nur den Ansichten des Autors. Falls einem Anwender die Karten allerdings nicht ansprechen, lässt sich das Programm ohne Probleme anpassen und erweitern.

## 6.3 Ausblick

Auf der einen Seite bietet das entwickelte Programm wie zuvor beschrieben viel Potential für Erweiterungen. So können neue Rauschfunktionen und Karten hinzugefügt oder die Umwandlung von Rauschwerten in geowissenschaftliche Daten angepasst werden. Man könnte beispielsweise eine Unterscheidung zwischen maritimem und kontinentalem Klima ergänzen. Eine weitere Idee wäre das Generieren von Flüssen und Seen oder von Menschen gebauter Städte.

Weiterhin können einzelne Teile der Software schnell in Spiele, Simulationen und ähnliches eingebunden werden um beispielsweise eine Vorlage für virtuelle Welten zu bieten. So könnte man leichte Veränderungen an dem Code vornehmen um anstelle einer realis-

tischen Welt, eine Fantasie-Welt zu generieren. Die Erweiterungsmöglichkeiten des Projektes sind vielfältig und müssten auf die jeweiligen Problemstellungen zugeschnitten werden.

## 7. Literaturverzeichnis

- [1] Burger, Wilhelm(2008): „Gradientenbasierte Rauschfunktionen und Perlin Noise“. FH Oberösterreich.
- [2] Han, Dianyuan(2013): „Comparison of Commonly Used Image Interpolation Methods“. In Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE), 2013, Atlantis Press.
- [3] Gustavson, Stefan(2005): „Simplex noise demystified“. Linköping Universität, Schweden.
- [4] Spencer, Kurt(2014): „Noise!“. <http://uniblock.tumblr.com/> (Stand: 19.03.2016)
- [5] Spencer, Kurt(2014): „Open Simplex Noise in Java“. <https://gist.github.com/KdotJPG/b1270127455a94ac5d19> (Stand: 19.03.2016)
- [6] De Carpentier, Giliam J.P.(2008): „Effective GPU-based synthesis and editing of realistic heightfields“. Delft University of Technology, W!Games.
- [7] Pu, Qiong(2004): „Qomolangma Biosphere Reserve Information“. <http://www.unesco.org/mabdb/br/brdir/directory/biores.asp?code=CPR+26&mode=all> (Stand: 19.03.2016), UNESCO.
- [8] Botica, Melania(2011): "Fakten über den tiefsten Abgrund der Erde". [http://www.focus.de/wissen/mensch/tid-24437/marianengraben-fakten-ueber-den-tiefsten-abgrund-der-erde\\_aid\\_692590.html](http://www.focus.de/wissen/mensch/tid-24437/marianengraben-fakten-ueber-den-tiefsten-abgrund-der-erde_aid_692590.html) (Stand: 13.04.2016), Focus Online.
- [9] Tan, Howard(2011): „ASTER Global Digital Elevation Map Announcement “. <http://asterweb.jpl.nasa.gov/gdem.asp> (Stand: 19.03.2016), NASA, METI.
- [10] „Visible Earth“. <http://visibleearth.nasa.gov/view.php?id=73934> (Stand: 19.03.2016)
- [11] Karte mit Höhenwerten von Google Maps: <http://www.mapcoordinates.net/> (Stand: 19.03.2016)
- [12] Automatische Höhenprofilberechnung: <http://geo.ebp.ch/gelaendeprofil/> (Stand: 13.04.2016), Ernst Basler + Partner.
- [13] Eakins, B.W. und Sharman, G.F.(2012): „Hypsographic Curve of Earth's Surface from ETOPO1“. NOAA National Geophysical Data Center.

- [14] Klimadatenbank: <http://www.weatherbase.com/> (Stand: 19.03.2016)
- [15] „World: Highest Temperature“. <http://wmo.asu.edu/world-highest-temperature> (Stand: 13.04.2016), Arizona State University.
- [16] ERA-Interim. European Centre for Medium-Range Weather Forecasts
- [17] „Global climate animations“. [http://geog.uoregon.edu/envchange/clim\\_animations/](http://geog.uoregon.edu/envchange/clim_animations/) (Stand: 11.04.2016), University of Oregon.
- [18] Deutschle, Tom: „Faszination Regenwald - die Regenwald-Initiative von Dr. Tom Deutschle“. <http://www.faszination-regenwald.de> (Stand: 19.03.2016)
- [19] Kottek, Markus et al.(2006): „World Map of the Köppen-Geiger climate classification updated“.  
In Meteorologische Zeitschrift, Vol.15, No. 3, S. 259-263, 2006.
- [20] Eckert-Greifendorff, Max(2012): „Kartographie: Ihre Aufgaben und Bedeutung für die Kultur der Gegenwart“. De Gruyter
- [21] Mascolus, Wilfried(2006): „Farbe in der Computergrafik“. TU Dresden
- [22] Allen, Jesse(2005): „Visible Earth“.  
<http://visibleearth.nasa.gov/view.php?id=73934> (Stand: 19.03.2016)
- [23] Mandelbrot, Benoît B.(1967): „How long is the coast of Britain? Statistical Self-Similarity and Fractional Dimension“.  
In Science Vol. 156, Issue 3775. S. 636-638, 1967.
- [24] Perron, J. Taylor: „Patterns in river networks“.  
<http://web.mit.edu/perron/www/research.html> (Stand: 11.04.2016), MIT.

## **Eidesstattliche Erklärung**

Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Bachelorarbeit selbstständig und ohne Benutzung anderer, als der angegebenen Hilfsmittel, angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Diese Bachelorarbeit hat in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegen.

Düsseldorf, den 15.04.2016

---

Jens Fröschel