Entwicklung eines Biom-Generators für landschaftsähnliche Höhenkarten

Marcus Gagelmann
Hochschule Anhalt & Martin-Luther-Universität Halle-Wittenberg
marcus.gagelmann@student.uni-halle.de

ABSTRACT

Diese Projektarbeit beschäftigt sich mit der Entwicklung einer Software, die es ermöglicht, landschaftsähnlichen Höhenkarten in diskrete Biom-Gebiete einzuteilen. Ziel ist es hierbei, dem Nutzer sowohl eine vollständig automatische Biomgenerierung anzubieten, als auch individuelle interaktive Anpassungsmöglichkeiten zur Verfügung zu stellen. Umgesetzt wird dieser Ansatz, aufbauend auf einer vorhergehenden Projektarbeit, innerhalb der Unity Game Engine.

KEYWORDS

Interaktive Medien Projekt 2, Winter 2022/23, Unity, Biome, Voronoi-Diagramme, Prozedurale Generierung

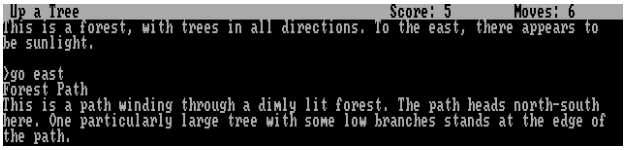

Abbildung 1: Die Beschreibung eines Waldes aus dem Textadventure *Zork* lässt Rückschlüsse auf das Biom zu Quelle: [1]

1 EINLEITUNG

Im folgenden wird eine Einführung in die Thematik der Arbeit gegeben. Hierzu werden zunächst die Motivation (Abschnitt 1.1), die Zielstellung (Abschnitt 1.2) sowie die Gliederung der Arbeit (Abschnitt 1.3) dargestellt.

1.1 Motivation

Die Oberfläche der Erde ist durchzogen von vielen unterschiedlichen Ökosystemen. Durch Erzählungen, gesammeltes Wissen aus Büchern und der Schule oder auch Bilder aus den digitalen Medien ist der moderne Mensch mit einer großen Anzahl dieser Biome schon seit dem Kindesalter vertraut.

Seit den 1970er Jahren, mit der Entwicklung der ersten Textadventures, bilden diese vertrauten Biome die Grundlage der allermeisten Welten in digitalen Spielen (siehe Abbildung 1). Auch heute noch wird bei der Konzeption von Spielwelten häufig auf Biome zurückgegriffen, die der Mensch vom Planeten Erde kennt (siehe Abbildung 11, 10 und 12). Damals wie heute kann auf diese Weise vom Spiel eine intuitive Umgebung präsentiert werden, welche beim Spieler Assoziationen weckt. So verstehen beispielsweise die meisten Nutzer eines Survival-Spiels automatisch, dass ihre Chancen der Nahrungssuche für ihren Charakter in einem Laubwald besser stehen als in der Wüste und dass sie vermutlich warme Kleidung benötigen, wenn sie einen Berg besteigen wollen.

Es ist ersichtlich, dass die Modellierung realistischer Biome einen wichtigen Bestandteil bei der Weltenentwicklung für digitale Spiele ausmacht. Im Rahmen des vorliegenden Projekts werden Konzepte diskutiert, um Biome in digitalen Spielen automatisch zu generieren. Speziell liegt der Fokus hierbei auf der Verteilung und Lage verschiedener Biome zueinander und nicht auf dem Aufbau einzelner Biome. Weiterhin wird ein Prototyp implementiert, der möglichst realistische Biom-Einteilungen für vorgegebene landschaftsähnliche Höhenkarten generieren soll. Mithilfe eines solchen Ansatzes zur automatischen Biom-Verteilung könnten Entwickler potenziell Personalkosten bei der Erstellung ihrer Spielwelten sparen. Außerdem können mithilfe eines Biom-Generators Spielwelten in einer wesentlich höheren Quantität geschaffen werden, als es durch manuelle Modellierung durch Menschen in einem Editor der Fall ist.

1.2 Zielstellung

Das Ziel dieser Projektarbeit ist es, Möglichkeiten zur Realisierung eines Biom-Generators für digitale Spiele zu diskutieren und einen Prototyp zu implementieren, der die besprochenen Konzepte umsetzt. Folgende Punkte sollen dabei besondere Beachtung finden:

- *Automatische Unterteilung* einer vorgegebenen Höhenkarte in möglichst realistische Biome
- *Einstellungsmöglichkeiten des Generators* durch den Nutzer
- *Möglichkeiten zur manuellen Nachbearbeitung* der erstellten Biom-Einteilung

Die Eingabe des Biom-Generators, die landschaftsähnlichen Höhenkarten, stammen von einem Landschaftsgenerator, der im Rahmen eines vorausgegangenen Projektes implementiert wurde. Details zu der zuvor geleisteten Arbeit können im Kapitel 2 eingesehen werden.

1.3 Gliederung

Der Aufbau der Arbeit folgt dem Entwicklungsprozess des Biom-Generator Prototypen und ergibt sich aus den Zielen, die im vorangegangenen Abschnitt skizziert wurden. Als Grundlage für die folgenden Ausführungen wird zuerst in Abschnitt 2 die vorausgegangene Arbeit erläutert, auf der dieses Projekt aufbaut. Im Anschluss werden dann in Abschnitt 3 die Begriffe Biom, Voronoi-Diagramm, morphologische Operation, Flood-Fill und Minkowski-Distanz als Theorie für das Verständnis der weiteren Arbeit erklärt. Im Abschnitt 4 wird zwischen verschiedenen Ansätzen zur Umsetzung eines Biom-Generators abgewogen, um im Anschluss aus diesen Überlegungen ein funktionierendes Konzept abzuleiten. Die Implementierung dieses Konzepts als Prototyp wird in Abschnitt 5 dokumentiert. Darauf folgt eine technische Evaluation des Prototypen in Abschnitt 6, in der die Kriterien *Realismus* und *Gebrauchstauglichkeit* genauer betrachtet werden. Abschließend fasst Abschnitt 7 die wesentlichen Ergebnisse der Arbeit zusammen und gibt einen Ausblick auf weitere Entwicklungsmöglichkeiten in diesem Bereich.

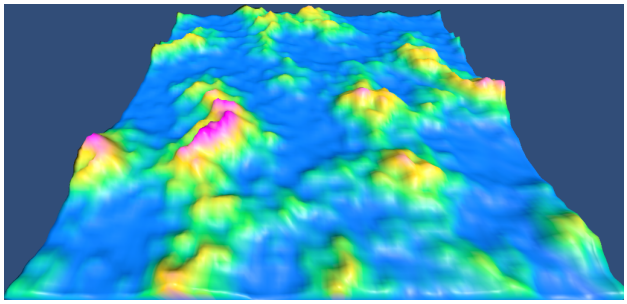

Abbildung 2: Prozedural generierte Höhenkarte
Quelle: [6]

2 AUSGANGSSITUATION

Die vorliegende Projektarbeit baut auf einem vorhergehenden Projekt aus dem Sommer 2022 auf [6]. Im Rahmen dieses Projektes wurden Möglichkeiten zur prozeduralen Generierung von landschaftsähnlichen Höhenkarten diskutiert. Zusätzlich wurde ein Prototyp in der Unity Game Engine implementiert, der die Generierung beinahe beliebiger Höhenkarten unterstützt. Der Nutzer kann zu diesem Zweck im Inspektor der Game Engine eine Vielzahl an Parametern einstellen, wodurch das generierte dreidimensionale Mesh der Höhenkarte in Echtzeit angepasst und visualisiert wird.

Die rechteckigen Höhenkarten, die das Programm generieren kann, basieren auf mehreren Schichten von Perlin Noise unterschiedlicher Frequenz. Diese Rauschfunktionen werden an jedem Knoten des Meshs mit den vom Nutzer eingestellten Parametern zu einem Höhenwert verrechnet. Durch diese Technik ist die Generierung unendlich vieler, realistischer Terrain-Ausschnitte möglich, wie sie beispielsweise in Abbildung 2 zu sehen sind.

Der Biom-Generator, der im Rahmen des vorliegenden Projektes konzipiert wird, soll als weitere Komponente zum Unity Projekt des vorhergehenden Projektes hinzugefügt werden. Das Ziel ist es, die Biome direkt auf den Höhenkarten zu generieren und dadurch Höheneinflüsse mit in die Verteilung der Biome einbeziehen zu können. Das Ergebnis des aktuellen Projektes soll eine Kombination aus dem bereits existierenden Höhenkarten-Generator und dem hier beschriebenen Biom-Generator sein.

3 THEORETISCHE GRUNDLAGEN

In diesem Abschnitt werden die Grundlagen von unterschiedlichen Bereichen und Methoden vermittelt, auf denen die darauf folgenden Abschnitte der Arbeit aufbauen.

3.1 Biome

Mit dem Begriff „Biom“ wird Wissen über die Struktur und die Funktionen der Ökosysteme der Erde organisiert [4]. Biome werden dabei eher durch ihre Physiognomie charakterisiert, als durch die genaue Komposition ihrer einzelnen Lebensformen [11]. Die Physiognomie eines Bioms bezeichnet hierbei das äußere Erscheinungsbild seiner Pflanzen- und Tierkomposition in seiner Gesamtheit.

Im Laufe der Zeit wurden verschiedene Ansätze mit ihren eigenen Vor- und Nachteilen entwickelt, um die Welt in ökologische Zonen zu unterteilen und damit Biome zu klassifizieren. In dieser Arbeit wird das Klassifikationsschema von Robert H. Whittaker von 1975 betrachtet, welches Biome mithilfe der Variablen *Temperatur* und *Feuchtigkeit* in Kategorien einordnet [16]. Aufgrund der simplen Klassifizierung mit lediglich zwei Faktoren,

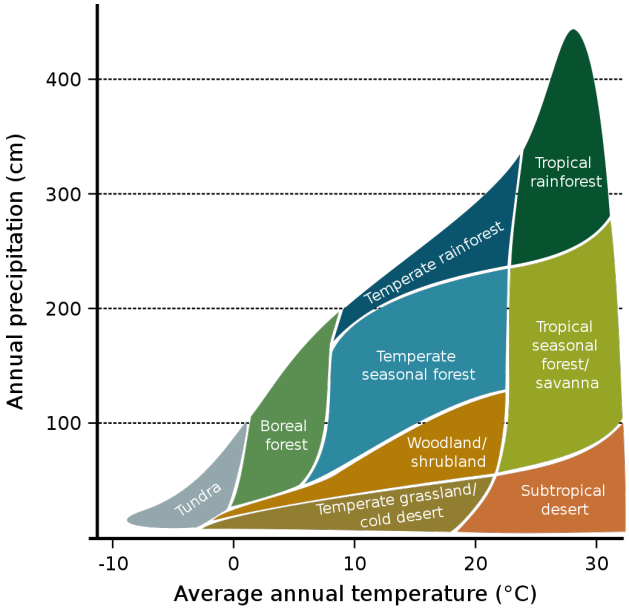

Abbildung 3: Whittakers Klassifikationsschema für Biome
Quelle: [17]

ist Whittakers Ansatz wesentlich zugänglicher als andere Klassifikationsschematas. Damit hat Whittaker einen Kompromiss gefunden, der Einbußen bei der Genauigkeit in Kauf nimmt und dafür die untergeordneten Faktoren der Bodenbeschaffenheit und der Sonneneinstrahlung vernachlässigen kann.

Whittaker beschreibt in seinem Schema sowohl für die *Temperatur* als auch für die *Feuchtigkeit* jeweils zwei Faktoren, die die Ausprägung der jeweiligen Variable an einem beliebigen Ort auf der Erde bestimmen. Die durchschnittliche Jahrestemperatur wird laut Whittaker lokal durch die jeweilige Höhenlage und global durch den zugrundeliegenden Breitengrad und dem damit zusammenhängenden Abstand und Winkel zur Sonne bestimmt. Die Feuchtigkeit eines Ortes wird wiederum von lokalen Gewässern und der Feuchtigkeit des Klimas an diesem Ort festgelegt. Da sich jedoch besonders lokale Gewässer in jedem individuellen Fall stark unterscheiden können, vereinfachte Whittaker sein Schema, indem er für den Feuchtigkeitsgrad eines Ortes lediglich den zugehörigen Jahresniederschlag und keine Gewässer mit einbezieht. Eine Visualisierung von Whittakers Klassifikationsschema für die Biome der Erde ist in Abbildung 3 zu sehen.

3.2 Voronoi-Diagramme

Als Voronoi-Diagramm bezeichnet man eine Menge von Polygonen, die durch vordefinierte Punkte, die sogenannten Centroiden, im zweidimensionalen Raum beschrieben werden. Es gibt in einem Voronoi-Diagramm genau so viele Polygone, wie es Centroide gibt. Um zu bestimmen, in welches Polygon ein beliebiger Punkt der Ebene fällt, werden Nachbarschaftsbeziehungen zwischen dem gewählten Punkt und allen Centroiden ausgewertet. Oft wird hierfür der euklidische Abstand verwendet, wobei der gewählte Punkt dem Polygon zugeordnet wird, zu dessen Centroid der geringste Abstand besteht. Ein vollständiges Voronoi-Diagramm entsteht, wenn jeder Punkte einer Ebene genau einem Centroiden zugeordnet sind [3].

Ein Beispiel für ein solches Voronoi-Diagramm, bei dem der euklidische Abstand als Distanzmaß verwendet wurde, ist in Abbildung 4 zu sehen. Beim Vergleich mit dem Diagramm der

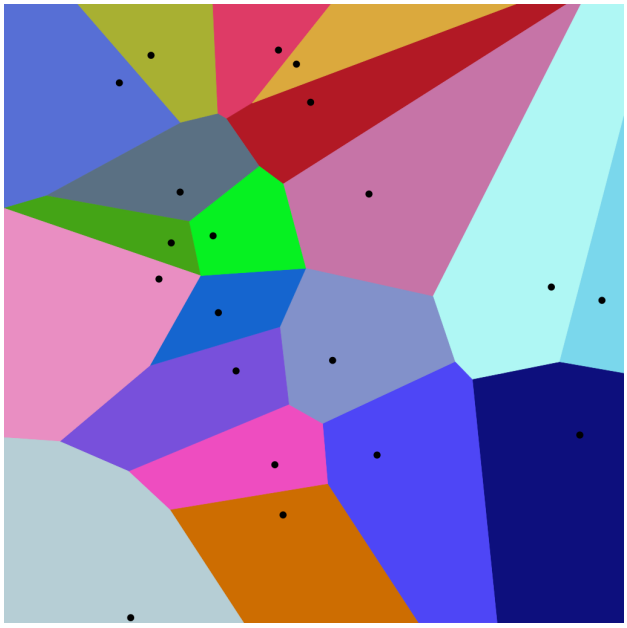

Abbildung 4: Voronoi-Diagramm - euklidische Distanz
Quelle: [18]

Abbildung 5: Voronoi-Diagramm - Manhattan-Distanz
Quelle: [18]

Abbildung 5, das unter Verwendung der gleichen Centroiden-Menge mithilfe des Manhattan-Distanzmaßes erstellt wurde, fallen die Unterschiede in den Zellstrukturen auf, die die Wahl des Distanzmaßes ausmachen kann.

3.3 Morphologische Operationen in der Bildverarbeitung

In der Bildverarbeitung beschäftigt sich unter anderem das Teilgebiet der Morphologie mit der Nachbearbeitung von Bildern. Die zwei wichtigsten Operationen auf binären Bildern sind hierbei die Erosion und die Dilatation. In beiden Fällen wird die jeweilige

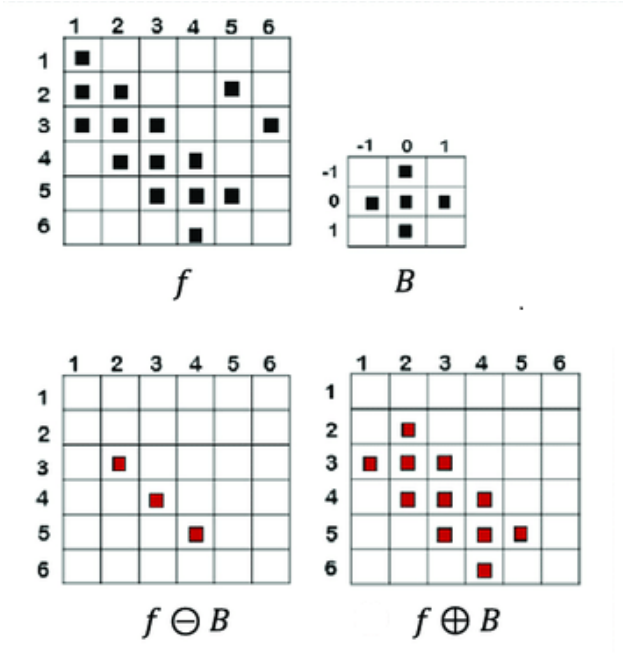

Abbildung 6: Opening-Operation mit f als Input, B als Strukturelement sowie ⊖ und ⊕ als Erosion und Dilatation
Quelle: [13]

Operation mithilfe eines Ankerpunktes sowie eines strukturierenden Elementes ausgeführt. Die Position des Strukturelements ist dabei relativ ausgerichtet zur Position des Ankerpunktes. Der Ankerpunkt hingegen wird pro Operation genau ein Mal auf jedes mögliche Pixel im zu bearbeitenden Bild positioniert. Ist der Ankerpunkt positioniert, so wird die jeweilige Operation mithilfe des Strukturelementes auf dem Bild ausgeführt. Bei der Erosion wird das Pixel unter dem Ankerpunkt auf den Wert 0 gesetzt, falls weitere Pixel mit dem Wert 1 innerhalb des Strukturelementes liegen. Die Dilatation hingegen ändert den Wert des Pixels unter dem Ankerpunkt auf 1, falls weitere Pixel mit dem Wert 1 im Strukturelement liegen. Die Erosion verkleinert somit Strukturen im Bild, wohingegen sie durch die Dilatation vergrößert werden [12].

Durch die Kombination von Erosion und Dilatation entstehen komplexere Operationssequenzen. Eine dieser Operationssequenzen ist die Opening-Operation. Sie ist über die Durchführung einer Erosion gefolgt von einer Dilatation definiert und kann verwendet werden, um kleinere Strukturen zu entfernen und anschließend die ursprüngliche Größe des Objektes wiederherzustellen [12]. Eine Illustration eines Openings mit einem kreuzförmigen Strukturelement zentriert über dem Ankerpunkt ist in Abbildung 6 zu sehen.

3.4 Flood Fill Algorithmus

Flood Fill ist ein Algorithmus zum Erfassen und bearbeiten zusammenhängender Pixelflächen in digitalen Bildern. Beginnend bei einem Pixel, wird dieses betrachtet und bearbeitet. Anschließend werden alle in einer 4er- oder 8er-Nachbarschaft angrenzenden Pixel ebenfalls entweder iterativ oder aber rekursiv betrachtet und bearbeitet, solange diese die gleiche Ursprungsfarbe wie des Startpixel besitzen [9].

Das Konzept der Breitensuche bietet eine Möglichkeit, diesen Algorithmus zu implementieren. Für dieses iterative Verfahren

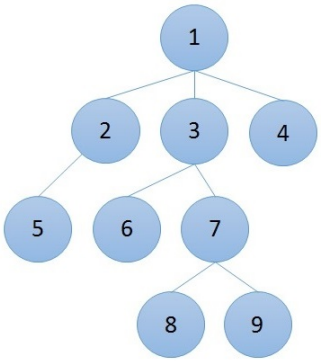

Abbildung 7: Breitensuche in einem Baumgraphen
Quelle: [5]

5	2			
4	1			
	3	6		
9	7			
	8			

Abbildung 8: Breitensuche aus Abbildung 7 als Flood Fill mit 4er-Nachbarschaft
Quelle: Eigene Darstellung

kann die Überprüfungsreihenfolge der Pixel im Bild während der Ausführung als Baum betrachtet werden, der das Startpixel als Wurzel besitzt. Das Ziel der Breitensuche ist es, alle Knoten eines Baumes zu durchlaufen und dabei immer zuerst die Knoten zu betrachten, die am nächsten an der Wurzel liegen.

Das genannte Konzept kann mithilfe einer Queue umgesetzt werden. In jeder Iteration wird das erste Pixel in der Queue betrachtet, als *besucht* markiert, bearbeitet und dessen zulässige Nachbarn hinten in die Queue eingefügt. Sobald die Queue leer ist, terminiert das Programm [7]. Ein Beispiel für die Anwendung von Flood Fill mit Breitensuche in einem Bild ist in Abbildung 7 und 8 zu sehen.

3.5 Minkowski-Distanz

Das euklidische Distanzmaß beschreibt den kürzesten Weg von einem Punkt X zu einem Punkt Y durch den Raum mithilfe folgender Formel [10]:

$$D_e = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Die Manhattan-Distanz hingegen beschreibt den Abstand zwischen zwei Punkten über einen rasterartigen Pfad und wird mithilfe der absoluten Summe der Differenzen zweier Punkte mit folgender Formel berechnet [10]:

$$D_{Ma} = \sum_{i=1}^{n} |x_i - y_i|$$

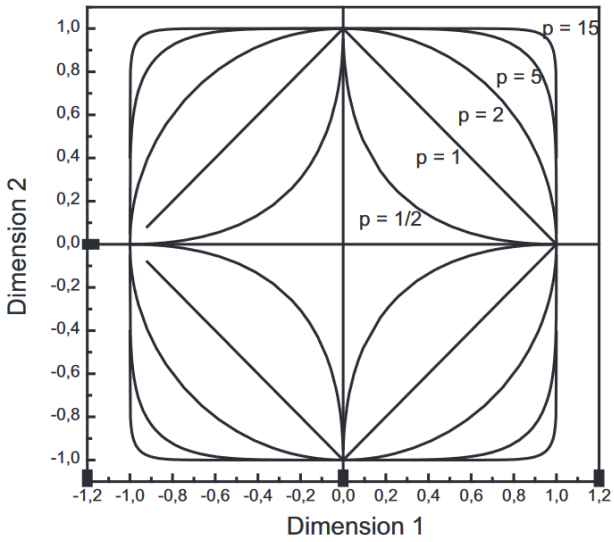

Abbildung 9: Punkte mit einer Minkowski-Distanz von 1 zum Zentrum für verschiedene p-Werte
Quelle: [10]

Mit der Minkowski Distanz existiert die Formel

$$D_{Mi} = \left(\sum_{i=1}^{n} (x_i - y_i)^p \right)^{\frac{1}{p}}$$

für ein generalisiertes Distanzmaß, von dem die euklidische Distanz und die Manhatten-Distanz zwei Spezialfälle sind. Wird der Parameter $p = 1$ oder $p = 2$ gewählt, so erhält man das Manhatten Distanzmaß oder das euklidische Distanzmaß. Da p jedoch ein freier Parameter ist, der lediglich größer als 0 sein muss, können hierfür auch gänzlich andere positive reelle Zahlenwerte gewählt werden [10]. Der gewählte Zahlenwert wirkt sich auf die Höhe des gemessenen Distanzwertes zwischen denselben Punkten im zweidimensionalen Raum aus, wie in Abbildung 9 zu erkennen ist.

4 KONZEPT

Es gibt verschiedene Möglichkeiten für die Umsetzung eines Generators, der automatisch vorgefertigte Höhenkarten in Biome einteilt und dabei dem Nutzer einige Einstellungsmöglichkeiten und Werkzeuge zur Nachbearbeitung überlässt. Im Folgenden werden mögliche Ansätze zur Umsetzung diskutiert. Anschließend wird aus den bestehenden Möglichkeiten der Ansatz für dieses Projekt konzipiert.

4.1 Ansätze für ein Biom-Modell

Zuerst muss festgelegt werden, anhand welcher Merkmale und Faktoren das Programm die Position und den Typ von Biomen bestimmen soll. In der realen Welt entstehen Biome laut Whittaker (siehe 3.1) aus der Kombination folgender Einflüsse:

- Temperatureinfluss des **Breitengrads**
- Temperatureinfluss der **Höhenlage**
- Feuchtigkeitseinfluss **lokaler Gewässer**
- Feuchtigkeitseinfluss des **Niederschlags**

4.1.1 Temperatureinflüsse. Es ist klar, dass die Temperatur des jeweiligen Breitengrads eines Ortes einen großen Einfluss auf das dort vorkommende Biom hat. Außerdem ist deutlich, dass weniger Sonnenwärme die Pole der Erde erreicht, als den Äquator.

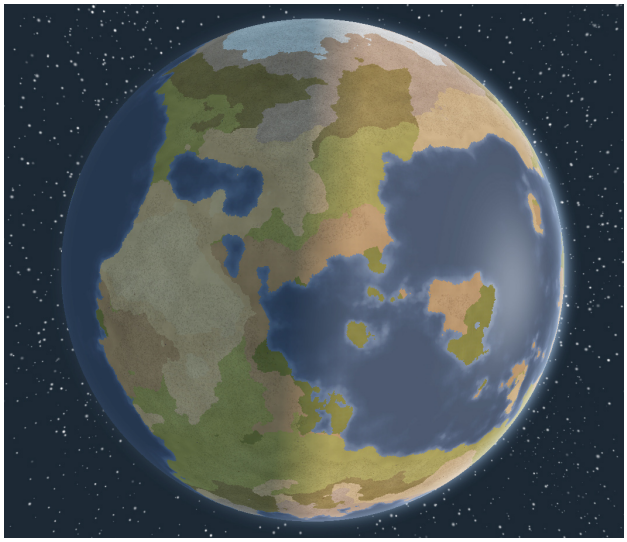

Abbildung 10: Prozedural generierte kugelförmige Welt aus dem Spiel *RimWorld* von 2018
Quelle: [14]

Abbildung 11: Planare Weltkarte des 2022 erschienenen Spiels *Mount & Blade II: Bannerlord*
Quelle: [8]

Der Breitengrad als Faktor für die Bestimmung von Klimazonen wurde bereits in vielen Spielen bei der Erstellung von Weltkarten verwendet. Einige Beispiele sind in den Abbildungen 10 und 11 zu sehen.

Die generierte Karte aus dem Spiel *RimWorld* (Abbildung 10) erreicht dabei den höchsten Grad an Realismus, weil hier eine kugelförmigen Welt verwendet wird. Die Pole und der Äquator des Planeten sind dadurch eindeutig bestimmbar und ein Temperaturabfall in Richtung der Pole kann wie in der realen Welt umgesetzt werden. Betrachtet man die Weltkarte aus dem Spiel *Mount & Blade II: Bannerlord* (Abbildung 11), so fällt auf, dass diese nicht als Kugelwelt, sondern als zweidimensionale Ebene umgesetzt wurde. Um auf dieser Weltkarte trotzdem den von der Erde gewohnten Temperaturabfall einzubringen, wurden hier der gesamte nördliche und südliche Rand der Karte als Pole verwendet. Ein Teil des Südens ist in dieser Karte jedoch abgeschnitten, weshalb man den unteren Pol nur vermuten kann. Die Pole dieser Spielwelt sind also kein Punkt, wie bei *RimWorld*, sondern eine Gerade entlang der kompletten Weltkante in Richtung der x-Achse.

Genauso gibt es aber auch Beispiele für Spiele, die bei der Konzeption ihrer Welt den Breitengrad als Temperatureinfluss

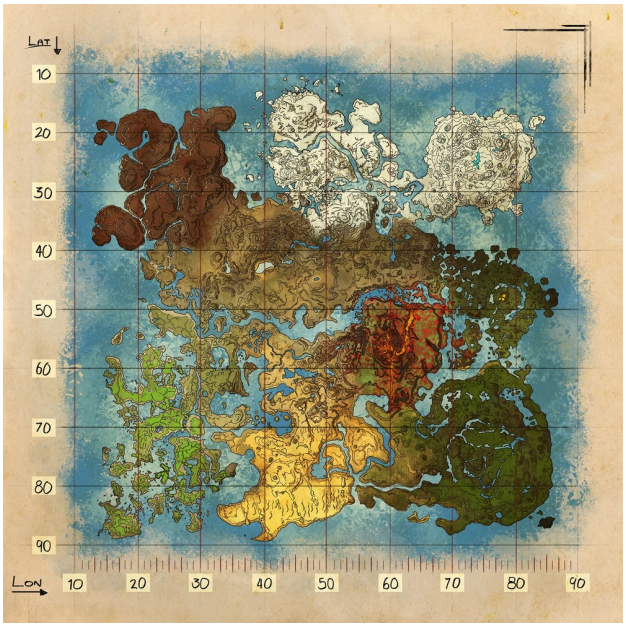

Abbildung 12: Planare Weltkarte aus dem 2017 erschienenen Spiel *ARK: Survival Evolved*
Quelle: [2]

missachten. Ein Beispiel hierfür ist die in Abbildung 12 dargestellte Karte einer Spielwelt aus dem Spiel *ARK: Survival Evolved*. In dieser planaren Karte ist ein Raster zu erkennen, welches den Betrachter die Längen- und Breitengrade verschiedener Orte ablesen lässt. Betrachtet man die Karte im Ganzen, so sind im Norden der Karte tendenziell kältere und im Süden wärmere Biome zu erkennen. Es fällt jedoch auf, dass sich auf der Höhe der allermeisten Breitengrade viele unterschiedliche Biome befinden. So durchzieht der achtzigste Breitengrad beispielsweise sowohl die zentrale subtropische Wüste als auch das gemäßigte Sumpfland im Osten [2]. Die Entwickler konnten beim Design dieser Karte offensichtlich keine realistische Weltkarte mit dem Konzept ihres Spiels vereinen und haben spürbare Abwechslung im Terrain einer besonders glaubwürdigen Temperaturverteilung vorgezogen.

Die Höhenlage als Faktor für die Temperaturbestimmung findet unter den genannten Beispielen eindeutig in Abbildung 11 Anwendung. In der Karte von *Ark: Survival Evolved* (Abbildung 12) sind hingegen keine Biom-Veränderungen in großer Höhe, wie zum Beispiel verschneite Berggipfel, zu erkennen. Innerhalb des Spiels wird dem Spieler jedoch zu jeder Zeit die Umgebungstemperatur angezeigt, die direkte Auswirkungen auf das Wohlbefinden des eigenen Charakters hat. Diese Temperatur der Umgebung fällt in größeren Höhenlagen drastisch ab. Zuletzt kann noch die generierte Spielwelt in Abbildung 10 betrachtet werden. Diese Welt verzichtet auf die Umsetzung von Höhenlagen und lässt somit auch keine Temperaturunterschiede auf dieser Grundlage erkennen.

4.1.2 Feuchtigkeitseinflüsse. Auch den Einfluss der Feuchtigkeit setzen die in Abschnitt 4.1.1 gezeigten Beispiele auf unterschiedliche Art und Weise um. Das Vorhandensein lokaler Gewässer scheint hierbei nur in Abbildung 11 eine Rolle bei der Biomverteilung zu spielen. An den Rändern des südöstlichen Meeres und des zulaufenden Flusses können grüne Flächen inmitten eines trockenen wüstenartigen Bioms erkannt werden. Im

Vergleich dazu sind in Abbildung 10 viele trockene hellbraune Gebiete zu sehen, die direkt an den abgebildeten Gewässern anliegen und im Gegenteil auch viele feuchte dunkelgrüne Biome, die sich weit abseits jeglicher Gewässer befinden. Die Karte in Abbildung 12 ignoriert aus allen Beispielen wohl am meisten das Vorhandensein natürlicher Gewässer. In dieser Welt grenzen ganze Flüsse und Meere direkt an trockene Savannen und sandige Wüsten.

Berücksichtigt man den Niederschlag, der auf die jeweiligen Weltkarten niedergehen sollte, kommt man wieder zu dem Schluss, dass die Karte von *Mount & Blade Bannerlord: II* (Abbildung 11) dem realen Beispiel am nächsten kommt. Hier sind nicht nur klare Abgrenzungen der einzelnen Biome mit glaubwürdigen Übergangszonen zu sehen, sondern der Niederschlag scheint auch vom Meer im Nordwesten her über das Land zu ziehen und die nordwestlichen Regionen der Landmasse besonders zu begrünen. Eine unrealistischere Niederschlagsverteilung existiert in der generierten Welt aus *RimWorld* (Abbildung 10). Die Biome besitzen abrupte Übergänge mit festen Grenzen und an vielen Stellen sind feuchte und trockene Gebiete direkt benachbart. Eine Richtung, aus der der Niederschlag vom Meer aus über die Landmassen zieht, ist nicht zu erkennen. Die Weltkarte die sich von den genannten Beispielen jedoch am weitesten in Bezug auf den Niederschlag von der realen Welt abgrenzt, ist diejenige aus *Ark: Survival Evolved* (Abbildung 12). In dieser Karte grenzt im Süden der niederschlagsreiche tropische Regenwald direkt an die trockene subtropische Wüste, die wiederum in einer anderen Richtung mit durchnässtem Sumpfland benachbart ist. Im Norden grenzt ebenfalls die niederschlagsarme Tundra an die durchgängig verschneite Eislandschaft ohne viel Raum für einen fließenden Übergang. Die meisten anderen, gänzlich verschiedenen Biome werden lediglich von Flüssen voneinander abgegrenzt und besitzen eine relativ kurze Luftlinie zueinander.

4.1.3 Biom-Übergänge. Ein weiterer Punkt, der Beachtung bei Konzeption des Biom-Generators finden sollte, ist die Art der Biom-Abgrenzung. Wie im Vergleich der drei genannten Beispiele auffällt, können Weltkarten in digitalen Spielen mit stetigen Biomen (siehe Abbildung 11) oder aber mit diskreten Biomen (siehe Abbildungen 10 und 12) versehen werden.

Stetige Biome zeichnen sich dadurch aus, dass einzelne Biome fließend ineinander übergehen, ohne dass eine konkrete Grenze erkennbar ist. Trotzdem kann in der Gesamtheit häufig eine endliche Anzahl an Biomen in einer Karte mit dieser Art von Übergängen ausgemacht werden, wobei ein Spielraum für Ungewissheiten bleibt. So sind zum Beispiel in Abbildung 11 trotz der Verwendung stetiger Biome ein Schnee-Biom im Norden, ein gemäßigtes Biom im Zentrum sowie Wüsten-Biome im Süden und Osten erkennbar. Sieht man genauer hin, so kann man sich an den Biom-Übergängen sicherlich auch noch Savannen im Süden des Zentrums und boreale Nadelwälder im Norden des Zentrums denken.

Im Gegensatz zu stetigen Biomen besitzen diskrete Biome feste Grenzen im Form von genau bestimmbaren Pfaden zwischen den Biomen. Die Haupteigenschaft dieser Biome ist, dass sie durch den Betrachter eindeutig identifizierbar sind. Könnte man beispielsweise die kugelförmige Spielwelt in Abbildung 10 von allen Seiten betrachten, so wäre es möglich, die Anzahl unterschiedlicher Biome auf der Weltkarte eindeutig zu zählen.

4.2 Abwägen des Biom-Modells

Ein Kriterium, nach dem ein Ansatz zur Biom-Bestimmung ausgewählt werden könnte, ist die Nähe zur Realität. Sollen die Biome auf den generierten Karten möglichst realistisch verteilt sein, dann sollten diese nach dem Klassifikationsschema von Whittaker anhand von Temperatur- und Feuchtigkeitseigenschaften der jeweiligen Region bestimmt werden. Dazu gehört sowohl die Berücksichtigung des Breitengrads und der Höhenlage bei der Temperaturermittlung als auch die Berücksichtigung des Niederschlags und der lokalen Gewässer bei der Ermittlung der Feuchtigkeit einer Region. Auch die Biom-Grenzen sollten möglichst realistisch gestaltet werden. Zu diesem Zweck wäre die Verwendung von stetigen Biomen für fließende Übergänge in der Landschaft empfehlenswert.

Ein weiteres Kriterium, das bei der Umsetzung des Projekts berücksichtigt werden muss, ist die Kompatibilität der generierten Karten mit digitalen Spielen. Das Landschaftsmodell des Terrain-Generators der vorhergehenden Arbeit aus dem Sommer 2022 beschreibt zur Umsetzung dieses Kriteriums lediglich eine abstrahierte Version davon, wie die Oberfläche der Erde in der Realität strukturiert ist. Da der Biom-Generator auf dem Terrain-Generator aufbaut, müssen zwangsweise auch in diesem Projekt Einbußen im Kriterium *Realismus* hingenommen werden, die sich aus dem abstrahierten Landschaftsmodell ergeben.

Zum einen beschreibt das Landschaftsmodell des Terrain-Generators lediglich planare Weltkarten, wohingegen für die Berücksichtigung der Breitengradtemperaturen, wie in der Realität, eine kugelförmige Welt nötig wäre. Eine Möglichkeit, die Temperaturen der Breitengrade auf planaren Karten zu simulieren, wird im Spiel *Mount & Blade II: Bannerlords* (siehe Abbildung 11) mit den Polen aufgezeigt, die an den Kartenrändern verlaufen. Um eine Terrain-Generierung nach den Wünschen des Anwenders zu gewährleisten, wurde der Höhenkartengenerator jedoch variabel in den Kartenausmaßen konzipiert. Aus diesem Umstand ergibt sich, dass feste Polgrenzen an der Nord- und Südgrenze der Karte unrealistisch wirken würden, weil die Karte in der aktuellen Version in den allermeisten Fällen nicht die Ausmaße eines ganzen Planeten annimmt. Um wiederum die Pole auf zwei Geraden außerhalb des generierten Bereichs festzulegen und aus den Abständen zur generierten Region das dortige Klima zu bestimmen, fehlt ein angemessenes Maß für die Gesamtgröße einer Welt. Die Weltkarten werden für kein konkretes Spielekonzept generiert, wodurch verschiedene Kartengrößen benötigt werden könnten. Eine feste Weltengröße mit immer gleichen Polabständen würde deshalb die Variabilität des Generators durch die Vorgabe von maximalen Kartenausmaßen zu sehr einschränken. Stattdessen werden im Rahmen dieses Projekts die benötigten Temperaturzonen mit einem ähnlichen Ansatz wie beim Spiel *Ark: Survival Evolved* (siehe Abbildung 12) nach Bedarf wechselnd und unabhängig von den Breitengraden erzeugt. Durch diesen Ansatz geht die natürliche, schichtartige Anordnung der Klimazonen der Erde verloren im Austausch für einen höheren Grad an Variabilität bei der Kartengenerierung.

Ein weiterer Punkt, der zu Einbußen im Kriterium *Realismus* führt, ist die Abwesenheit eines Systems zur Umsetzung eines realistischen Wasserkreislaufs. Es wurde lediglich ein Wassergrenze-Parameter aus dem vorhergehenden Projekt übernommen, der verwendet wird, um die Höhe einer Ebene zu bestimmen, die das Programm waagerecht im Terrain platziert und die die Höhe des Wasserstandes von angenommenen Gewässern symbolisiert. Diese Ebene dient bis jetzt jedoch einzig und allein dazu, die

Anschaulichkeit des Terrains zu erhöhen und steht in keinerlei Wechselwirkung zu dessen Umgebung. Die Umsetzung von realitätsnahem Niederschlag unter Einberechnung von Luftdruckgebieten und Höhenlagen ist ebenfalls nicht vorhanden und bietet mit seinen vielen Abhängigkeiten von weiteren natürlichen Faktoren den Umfang für ein gesondertes Projekt. Der Einfluss von Gewässern auf die umgebenden Biome und die unterschiedlichen Niederschlagsgebiete der Welten werden stattdessen in diesem Projekt in einer generalisierenden Art und Weise simuliert. Zusätzlich könnten in Bezug auf die existierende Wassergrenze auch Unterwasserbiome berücksichtigt werden. Das aktuelle Gewässermodell ist jedoch noch zu abstrakt, um diese Art von Biomen sinnvoll umzusetzen. Eine Unterscheidung zwischen Gewässerart (fließend oder stehend), Gewässertemperatur und Gewässertiefe wäre ein möglicher Ansatz für zukünftige Projekte.

Zuletzt muss auch bei der Konzeption der Biom-Grenzen von der Umsetzung der realistischsten Möglichkeit, den stetigen Biomen, abgesehen werden. Grund dafür ist, dass nach diesem Konzept zu einem späteren Zeitpunkt der Einfluss von Feuchtigkeit und Temperatur auf die vielen einzelnen Bestandteile des Ökosystems jeweils für ein ganzes Intervall von Parameterwerten fein abgestimmt werden muss. Ein Beispiel für einen solchen möglichen Parameter wäre, neben vielen anderen, die Veränderung der Nadelbaumdichte bei steigenden oder fallenden Temperaturen. Das aktuelle Biom-Modell ist zu ungenau, um einzelne Bestandteile der Flora und Fauna zu erfassen und die Weiterentwicklung sowie die Feinabstimmung des Modells übersteigt den Rahmen dieses Projekts. Ein Ansatz, der besser zur Umsetzung des Projekts geeignet ist, ist die Verwendung diskreter Biome. Für dieses Konzept müssen bei der späteren Zusammenstellung der Biome für jeden Ökosystem-Parameter lediglich einige abzählbare Feuchtigkeits- und Temperaturabstufungen beachtet werden, anstatt eines ganzen Intervalls von Werten. Diese Art von Biom-Modell wird bereits erfolgreich in digitalen Spielen verwendet, wie am Beispiel von *RimWorld* (siehe Abbildung 10) und *Ark: Survival Evolved* (siehe Abbildung 12) erkennbar ist. Ein Biom-Generator mit diesem Ansatz kann also, trotz der geringeren Realitätsnähe, einen Mehrwert bei der Entwicklung von digitalen Spielen bieten.

4.3 Konzeption des Biom-Generators

Aus den in Abschnitt 4.2 besprochenen Biom-Eigenschaften und den Vorgaben des vorhergehenden Projekts ergibt sich ein Konzept für ein Biom-Modell, das zur weiteren Umsetzung verwendet werden kann. Mithilfe dieses Biom-Modells und unter Berücksichtigung der Zielstellung wird im Folgenden ein Konzept für den Biom-Generator erstellt.

4.3.1 Biom-Modell. Die Terrain-Meshes, die vom Terrain-Generator des vorhergehenden Projektes erzeugt werden, bestehen aus $N \times M$ Knoten, die mithilfe von Kanten gitterartig vernetzt sind. Für die Umsetzung der besprochenen Eigenschaften ist es notwendig, dass jedem Knoten des Meshs sowohl ein eigener Temperaturwert als auch ein eigener Feuchtigkeitswert zugeordnet wird. Ein Biom ist in diesem Model als Gruppierung benachbarter Knoten mit ähnlichen Temperatur- und Feuchtigkeitswerten definiert. Um außerdem das Konzept der diskreten Biome im Modell anzuwenden, muss eine endliche Anzahl an möglichen Temperatur- und Feuchtigkeitsabstufungen für die einzelnen Knoten festgelegt werden. In diesem Biom-Modell wird es die drei Variablenbelegungen *niedrig*, *mittel* und *hoch* geben, von denen für jeden Mesh-Knoten immer genau einer der Werte für die Temperatur und einer der Werte für die Feuchte abgespeichert ist.

4.3.2 Zielumsetzung: Terrainunterteilung in Biome. Die automatische Funktion zur Einteilung von Höhenkarten in Biome wird mithilfe von Vorornoi-Diagrammen, die die Ausmaße der jeweiligen Karte besitzen, realisiert. Hierbei wird die zellenartige Struktur dieser Diagramme zur Erzeugung von separaten Biom-Gebieten ausgenutzt. Für jede Höhenkarte wird dafür sowohl ein Temperatur-Voronoi-Diagramm als auch ein Niederschlags-Voronoi-Diagramm berechnet. Die einzelnen Zellen der Diagramme können dabei, unter Verwendung des Biom-Modells, die Intensitätswerte *niedrig*, *mittel* oder *hoch* bezüglich der Temperatur oder des Niederschlags annehmen und werden zufällig ausgelost. Für eine realistischere Platzierung der Biome dürfen sich dabei benachbarte Zellen in den beiden Diagrammen in maximal einer Intensitätsstufe unterscheiden. So können beispielsweise Gebiete mit hoher Temperatur nur an Gebiete mit mittlerer oder hoher Temperatur angrenzen. Zusätzlich sollen sich die einzelnen Gebiete der Voronoi-Diagramme in der Höhenkarte annähernd über Täler aufspannen und durch Berge begrenzt werden. Zuletzt ist zu beachten, dass große Höhenlagen und Gewässer Einfluss auf die Biome haben sollen. Um diesen Aspekt umzusetzen, wird der Temperatur-Intensitätswert in großer Höhe um eine Stufe abgesenkt und der Feuchtigkeits-Intensitätswert in der Nähe von Gewässern um eine Stufe angehoben, solange die entsprechenden Werte noch nicht die Minimal- bzw. Maximalstufe erreicht haben.

4.3.3 Zielumsetzung: Einstellungsmöglichkeiten. Der Nutzer des Biom-Generators soll die Möglichkeit erhalten, die unterschiedlichen Aspekte des Programms bei der Generierung auf seine Bedürfnisse abzustimmen. Hierfür werden folgende verstellbare Parameter bereitgestellt:

- Durchschnittstemperatur
- Durchschnittsniederschlag
- Minkowski-P-Wert
- Minima-Suchradius
- Temperaturabfallhöhe
- Feuchtigkeitsausbreitung

Die ersten beiden Parameter können dazu verwendet werden, die Temperatur und den Niederschlag über die Karte verteilt eher in höheren oder niedrigeren Intensitätsstufen ausfallen zu lassen. Der *Minkowski-P-Wert* hingegen ist Teil der Formel zur Berechnung der Minkowski-Distanz und wird bei der Erzeugung der Voronoi-Diagramme eingesetzt. Im Abschnitt 3.5 ist die Formel zur Berechnung der Minkowski-Distanz abgebildet. Das Distanzmaß wird für die Abstandmessungen zwischen den Centroiden und Pixeln der Voronoi-Diagramme eingesetzt und hat direkte Auswirkungen auf den Verlauf der Grenzen zwischen den Voronoi-Zellen (siehe Abbildungen 4 und 5). Mithilfe des *Minima-Suchradius* kann der Nutzer bei der Suche nach lokalen Minima im Terrain die Größe der Fläche vorgeben, in der die gefundenen Punkte absolute Minima sein müssen. Da sich dieser Wert direkt auf die Centroiden-Anzahl für die Berechnung der Voronoi-Diagramme auswirkt, kann der Nutzer hiermit indirekt die durchschnittliche Größe der Biome beeinflussen. Zuletzt soll es dem Nutzer ebenfalls möglich sein, mit den letzten beiden Parametern die Höhenlage festzulegen, ab der die Temperaturintensität um eine Stufe abgesenkt wird sowie die Größe der Fläche, um ein Gewässer festzulegen, in der die Feuchtigkeitsintensität um eine Stufe angehoben wird.

4.3.4 Zielumsetzung: Nachbearbeitung. Den Nutzern sollen Funktionen bereitgestellt werden, mit denen unerwünschte Ausprägungen der generierten Biome nachbearbeitet werden können. Zu diesem Zweck soll es automatische Programmroutinen geben, die der Nutzer nach Belieben ausführen kann. Hierzu gehört zum einen eine Routine zur Eliminierung kleiner Regionen, wobei der Nutzer die maximale Größe der zu eliminierenden Regionen über einen veränderbaren Parameter bestimmen kann. Weiterhin wird dem Nutzer eine Routine zur Glättung der Biomgrenzen und zum Entfernen kleinerer Biom-Fragmente bereitgestellt, die mit einer nicht-binären Abwandlung der Opening-Operation (siehe Abbildung 6) arbeitet. Zuletzt soll der Nutzer auch auf die manuelle Nachbearbeitung mithilfe eines Zeichentools zurückgreifen können, um die Biom-Karten ganz auf seine Bedürfnisse anpassen zu können.

5 IMPLEMENTIERUNG

Der Pseudocode in Abbildung 13 zeigt die grundsätzliche Reihenfolge der Generatorberechnungen. Die einzelnen benötigten Funktionen werden hierbei stark abstrahiert dargestellt und dienen lediglich dazu, einen Überblick über das Gesamtkonzept zu vermitteln. Genauere Ausführungen zum Code sind in den folgenden Abschnitten zu finden.

5.1 Generierung der Biom-Karte

5.1.1 Bestimmung der Centroiden. Der erste Schritt bei der Generierung von Biom-Karten besteht in der Berechnung geeigneter Voronoi-Diagramme, die anschließend für die Verteilung der Temperatur- und Niederschlagsgebiete verwendet werden können. Die Voronoi-Diagramme des Biom-Generators sollen lokale Minima der Höhenkarte als Centroiden verwenden. Hierfür müssen jedoch zuerst lokale Minima innerhalb der Höhenkarte ausgemacht werden. Dies geschieht in dem Code-Abschnitt, der in Abbildung 14 zu sehen ist.

Zuerst wird vom Nutzer mithilfe des *searchRadius*-Parameters die Größe der Fläche festgelegt, in der nach einem Minimum gesucht wird. Die abgebildete Funktion *isLocalMinimum* wird anschließend für jeden Knoten des Meshs der Höhenkarte mit diesem Flächenparameter ausgeführt. Gesucht wird nach einem lokalen Minimum innerhalb eines Quadrates, dessen Mittelpunkt der aktuell betrachtete Mesh-Knoten ist. Die Seitenlänge a des Quadrates wird mithilfe der folgenden Formel beschrieben:

$$a = 2 \times searchRadius + 1$$

Grund hierfür sind die zwei verschachtelten Schleifen, deren Laufvariablen den Offset vom Ursprungsknoten zur Findung des nächsten Vergleichsknotens angeben und von $-searchRadius$ bis $searchRadius$ laufen. Ergibt keiner der Vergleiche, dass der Vergleichsknoten einen niedrigeren Höhenwert besitzt als der Ursprungsknoten, dann gibt die Funktion *isLocalMinimum True* zurück und bestätigt damit, dass ein lokales Minimum gefunden wurde. Die Koordinaten der gefundenen Minima werden anschließend in einer Liste gespeichert und als Centroiden für die Berechnung der Voronoi-Diagramme, wie in Abschnitt 3.2 beschrieben, verwendet.

5.1.2 Berechnung der Regionen. Zur Berechnung der Voronoi-Diagramme muss für jeden Knoten des Höhenkarten-Meshs der am nächsten gelegene Centroid gefunden werden. Dabei wird der Abstand zweier Knoten im Mesh mithilfe der Minkowski-Distanz (siehe Abschnitt 3.5) und des vom Nutzer gewählten *Minkowski-P-Wertes* gemessen.

```
generateBiomes() {
    tempSprite = calcTemperatureDiagram();
    tempSprite = addHeightTemperatures(tempSprite);

    humiditySprite = calcPrecipitationDiagram();
    humiditySprite = addWatersHumidity(humiditySprite);

    return merge(tempSprite, humiditySprite);
}

Update() {
    if(generateBiomeMap) {
        biomeSprite = generateBiomes();
    }
    if(doPostProcessing) {
        eliminateSmallRegions();
        smoothBiomeBorders();
    }
    applyDrawingChanges();
}
```

Abbildung 13:
Überblick über das Programm in Pseudocode

```
bool isLocalMinimum(int x, int y, int searchRadius) {
    bool isLocalMinimum = true;
    for(int offsetX = -searchRadius;
    offsetX <= searchRadius;
    offsetX++) {
        for(int offsetY = -searchRadius;
        offsetY <= searchRadius;
        offsetY++) {
            isLocalMinimum = isLocalMinimum &&
                !isPixelLower(
                    x + offsetX, y + offsetY, x, y);
            if(!isLocalMinimum) {
                return false;
            }
        }
    }
    return isLocalMinimum;
}
```

Abbildung 14:
Ermittlung lokaler Minima in der Höhenkarte

```
float smallestDst = float.MaxValue;
int nearestRegionID = 0;

for(int i = 0; i < centroids.Length; i++) {
    float distance = calcMinkowskiDistance(
        pixelPos, centroids[i], minkowskiP);
    if (distance < smallestDst) {
        nearestRegionID = i;
        smallestDst = distance;
    }
}
```

Abbildung 15:
Ermittlung des nächstliegenden Centroiden zu einem Mesh-Knoten

In Code-Abschnitt 15 ist zu sehen, wie das Programm für einzelne Knoten den nächstliegenden Centroiden findet. Die Berechnungen werden hierbei in Mesh-Koordinaten durchgeführt und die Höhenlage der Knoten im Mesh wird vernachlässigt. Abschließend wird die ID des nächstliegenden Centroiden von der Funktion ausgegeben und zum Mesh-Knoten zugehörig abgespeichert.

Nachdem dieser Vorgang für alle Knoten durchgeführt wurde, wird auf die gleiche Weise für jeden Knoten der zweitnächste Centroid ermittelt. Diese Daten werden benötigt, um für jede Region des Voronoi-Diagramms alle seine Nachbarregionen für spätere Berechnungen abzuspeichern. Zu diesem Zweck speichert das Programm für jeden Knoten die ID des zweitnächsten Centroiden in einer Liste, die der Region zugeordnet ist, in der sich der Knoten befindet. Duplikate werden dabei nicht mehrmals abgespeichert.

5.1.3 Bestimmung der Intensitätswerte. Um aus dem berechneten Voronoi-Diagramm die benötigten Temperatur- und Niederschlagskarten zu erzeugen, müssen jetzt Intensitätswerte für die einzelnen Regionen des Diagramms ermittelt werden. Die in Abschnitt 4.3.1 konzipierten Intensitätsabstufungen *niedrig*, *mittel* und *hoch* werden im Programm mit den Intensitätswerten 0, 1 und 2 dargestellt. Bei der Berechnung dieser Intensitätswerte kommen sowohl die beiden benutzerdefinierten Parameter *Durchschnittstemperatur* und *Durchschnittsniederschlag* als auch die im Voraus bestimmten Listen mit Regionsnachbarn zur Anwendung. Der Prozess der Intensitätswertbestimmung wird für alle Regionen des Voronoi-Diagramms durchgeführt und überprüft zuerst, wie groß die Spanne an möglichen Intensitätswerten ist, die die betrachtete Region annehmen kann. Folgende Fälle werden überprüft:

(1) Falls die betrachtete Region keine Nachbarregionen mit den Intensitätswerten 0 und 2 hat, dann reicht die Spanne der möglichen Intensitätswerte von 0 bis 2.
(2) Falls die betrachtete Region eine Nachbarregion mit dem Intensitätswert 0 hat, dann reicht die Spanne der möglichen Intensitätswerte von 0 bis 1.
(3) Falls die betrachtete Region eine Nachbarregion mit dem Intensitätswert 2 hat, dann reicht die Spanne der möglichen Intensitätswerte von 1 bis 2.
(4) Falls für noch keine Nachbarregion ein Intensitätswert bestimmt wurde, dann wähle den Intensitätswert 1 für die betrachtete Region.

Mithilfe dieser Auswertung kann sichergestellt werden, dass sich angrenzende Regionen niemals um mehr als eine Stufe in ihrer Intensität unterscheiden. Der konkrete Intensitätswert der betrachteten Region wird anschließend mithilfe der Intensitätsspanne und dem *Durchschnittstemperatur*- bzw. *Durchschnittsniederschlag*-Parameter in einem Zufallsverfahren ermittelt. Welcher der beiden Parameter verwendet wird, ist davon abhängig, ob mit dem aktuellen Prozess eine Temperaturkarte oder eine Niederschlagskarte erzeugt werden soll.

Innerhalb des Zufallsverfahrens, das in Code-Abschnitt 16 zu sehen ist, wird zuerst ein zufälliger Float-Wert zwischen 0 und 1 gleichverteilt bestimmt. Besitzt die Spanne der Intensitätswerte für diese Region die maximale Größe von $|\{0, 1, 2\}| = 3$, dann kann die Region entweder die Intensität 0 oder die Intensität 2 annehmen. Da ein vom Nutzer höher gewählter Gewichtungsparameter intuitiv auch höhere Durchschnittstemperaturen bzw. einen höheren Durchschnittsniederschlag bewirken soll, muss an dieser Stelle der Zufallswert mit dem Gegenwert $1-weightFactor$ des Gewichtungsparameters verglichen werden. Damit diese Mechanik funktioniert, müssen auch die Gewichtungsparameter *Durchschnittstemperatur* und *Durchschnittsniederschlag* stets im Intervall zwischen 0 und 1 gewählt werden, was mit einer Parameterbeschränkung im Unity-Inspektor sichergestellt werden kann. Fällt der Zufallswert niedrig aus, dann wird in der Regel,

```
1  float dieValue = Random.Range(0f, 1f);
2  if((highestIntensityID - lowestIntensityID) == 3) {
3      if(dieValue < 1-weightFactor) {
4          intensityID = 0;
5      } else {
6          intensityID = 2;
7      }
8  } else {
9      if(lowestIntensityID == 0) {
10         if(dieValue < 1-weightFactor)
11             intensityID = 0;
12         else
13             intensityID = 1;
14     } else {
15         if(dieValue < 1-weightFactor)
16             intensityID = 1;
17         else
18             intensityID = 2;
19     }
20 }
```

Abbildung 16:
Gewichtete Zufallsbestimmung von Intensitätswerten

abhängig vom Gewichtungsparameter, auf diese Weise der niedrigere Intensitätswert gewählt. Umgekehrt, mit einer höheren Gewichtung, fallen auch die Intensitätswerte in der Regel höher aus. Der gleiche Prozess wird in den Anwendungsfällen durchlaufen, in denen die Intensitätsspanne lediglich von 0 bis 1 oder von 1 bis 2 reicht, wofür natürlich die potenziellen Ergebniswerte entsprechend angepasst werden (Code 16, Zeile 11, 13, 16 und 18).

5.1.4 Berechnung von Höhen- und Gewässereinfluss. Der Einfluss von Höhenlage und Gewässern auf die Intensitätswerte der Temperatur- und Niederschlagskarte erfolgt direkt nach der Berechnung der Intensitäts-Grundwerte. In Code-Abschnitt 17 ist zu erkennen, wie der Grenzwert in Weltkoordinaten ermittelt wird, der bestimmt, ab welcher Höhenlage sämtliche Intensitätswerte der Temperaturkarte um eine Stufe gesenkt werden. Dafür wird der benutzerdefinierte relative Höhenparameter herangezogen, der festlegt, wie viel Prozent der gesamten Höhenspanne des Terrains zum obersten abgekühlten Bereich gehört. Dieser Parameter bewegt sich zwischen den Werten 0 und 1. Die Parameter *maxHeight* und *minHeight* hingegen stellen die Höhenobergrenze und Höhenuntergrenze des Terrains in Weltkoordinaten dar. Im Anschluss wird für jeden Knoten im Mesh überprüft, ob dieser höher gelegen ist als der berechnete Grenzwert. Für jeden auf diese Weise gefundenen Knoten wird der zugehörige Temperatur-Intensitätswert um eine Stufe oder auf minimal 0 gesenkt.

Die Berechnung des Gewässereinflusses startet auf ähnliche Weise wie die Berechnung des Höheneinflusses. Zuerst wird mithilfe des Parameters für die Bestimmung der relativen Wasserhöhe der Wasserstand im Terrain in Weltkoordinaten (siehe Code-Abschnitt 18 Zeile 1) berechnet. Danach wird für alle Knoten des Meshs überprüft, ob diese unterhalb des Wasserstands liegen. Ist dies der Fall, dann wird in einer Karte (Bool-Array) zu den Koordinaten des aktuellen Knotens vermerkt, dass dieser ein Gewässerknoten ist. Der dritte Schritt in der Prozesskette sieht vor, den benutzerdefinierten *Feuchtigkeitsausbreitung*-Parameter auf die ermittelten Gewässerknoten anzuwenden. Der Parameter gibt als Knotenanzahl an, wie weit der Feuchtigkeitseinfluss eines Gewässers auf die umliegenden Mesh-Knoten in alle Richtungen übergreifen soll. Zu diesem Zweck wird die binäre Dilatationsoperation aus der Bildverarbeitung so oft auf

```
float coldHeightsLimit =
    maxHeight - ((maxHeight - minHeight)
    * realitveColdHeightsLimit);

for(int meshY = 0; meshY < meshSize.y; meshY++) {
    for(int meshX = 0; meshX < meshSize.x; meshX++) {
        int meshIndex = meshX + meshY * meshSize.x;
        if(meshX < textureSize.x
        && meshY < textureSize.y
        && terrainVertices[meshIndex].y > coldHeightsLimit) {
            int textureX = meshIndex % meshSize.x;
            int textureY =
                Mathf.FloorToInt(meshIndex / meshSize.x);
            int textureIndex =
                textureX + textureY * textureSize.x;
            intensityArray[textureIndex] =
                Mathf.Max(
                    intensityArray[textureIndex] - 1,
                    0
                );
            }
    }
}
```

Abbildung 17:
Funktion zur Berechnung des Temperatureinflusses großer Höhenlagen

die Boolean-Gewässerknotenkarte angewendet, wie der *Feuchtigkeitsausbreitung*-Parameter vorgibt. Abschließend wird bei jedem Knoten, der als Gewässerknoten vermerkt ist, der Feuchtigkeits-Intensitätswert um eine Stufe auf maximal 2 angehoben.

5.1.5 Zusammenführung zur Biom-Karte. Jeder Mesh Knoten soll in den generierten Biom-Karten genau einem Biom angehören. Dieses Biom wird über das Tupel von Temperatur- und Feuchtigkeitsintensitätswert des Knotens bestimmt. Diese Tupel zusammen mit den Knotenkoordinaten können in weiterführenden Anwendungen dafür genutzt werden, zu bestimmen, wie die Flora und Fauna in konkreten Regionen des Terrains aussehen soll. Im Rahmen dieses Projektes werden aus den Intensitätskarten lediglich Texturen generiert, die dem Nutzer einen Überblick über die Biomverteilung geben sollen und dazu auf das Terrain projiziert werden. Zu diesem Zweck werden sowohl jedem Intensitätswert der Temperaturkarte als auch jedem Intensitätswert der Feuchtigkeitskarte unterscheidbare Farben zugeordnet. Jede generierte Textur besitzt eine Größe von $X \times Y$ Pixeln, wobei X die Länge und Y die Breite des Terrain-Mesh angegeben in Knoten sind. Jedes Pixel der Textur wird mit der Farbe eingefärbt, die der Temperatur- bzw. Feuchtigkeitsintensitätswert des Mesh-Knotens, mit den gleichen Koordinaten wie die vom Pixel, vorgibt. Sind auf diese Weise die Temperatur- und die Feuchtigkeitstextur erstellt worden, werden die beiden Texturen zu einer gemeinsamen Biom-Textur vereint. Dafür wird für jede Pixelkoordinate zwischen den Farben des zugehörigen Temperatur-Pixels und des zugehörigen Feuchtigkeit-Pixels linear interpoliert und die Ergebnisfarbe in den entsprechenden Pixeln der Biom-Karte übernommen.

5.2 Funktionen zur Nachbearbeitung

5.2.1 Eliminierung kleiner Regionen. Die erste vom Benutzer ausführbare Programmroutine zur Nachbearbeitung dient der Eliminierung von Regionen, deren Größe unter einem vom Nutzer festgelegten Schwellwert liegt. Hierzu muss zuerst mittels

```
float waterLevel =
    minHeight + (maxHeight - minHeight)
    * relativeWaterHeight;

bool[] isWaterNode = new bool[terrainVertices.Length];
for(int meshY = 0; meshY < meshSize.y; meshY++) {
    for(int meshX = 0; meshX < meshSize.x; meshX++) {
        int meshIndex = meshX + meshY * meshSize.x;
        if(meshX < textureSize.x
        && meshY < textureSize.y
        && terrainVertices[meshIndex].y < waterLevel) {
            isWaterNode[meshIndex] = true;
        } else {
            isWaterNode[meshIndex] = false;
        }
    }
}

for(int i = 0; i < humiditySpread; i++) {
    isWaterNode = dilateWaterNodes(isWaterNode);
}

for(int meshY = 0; meshY < meshSize.y; meshY++) {
    for(int meshX = 0; meshX < meshSize.x; meshX++) {
        int meshIndex = meshX + meshY * meshSize.x;
        if(isWaterNode[meshIndex]) {
            int textureX = meshIndex % meshSize.x;
            int textureY =
                Mathf.FloorToInt(meshIndex / meshSize.x);
            int textureIndex =
                textureX + textureY * textureSize.x;
            intensityArray[textureIndex] =
                Mathf.Min(
                    intensityArray[textureIndex] + 1,
                    2
                );
        }
    }
}
```

Abbildung 18:
Funktion zur Berechnung des Feuchtigkeitseinflusses von Gewässern

Breitensuche die Größe von sämtlichen Regionen in Knoten ermittelt werden (siehe Code-Abschnitt 19). Es wird über jedes Pixel der Biom-Karte iteriert (Code-Zeilen 6 und 7) und für jedes auf diese Weise betrachtete Pixel eine iterative Breitensuche nach Pixeln in der direkten Vierernachbarschaft ausgeführt, die die gleiche Farbe wie das aktuell betrachtete Pixel besitzen und in noch keiner anderen Breitensuche betrachtet wurden (Zeilen 19 bis 44). Sämtliche auf diese Weise gefundenen Pixel werden als zur Region zugehörig vermerkt und zu der Liste von Pixeln hinzugefügt, deren Nachbarn in den zukünftigen Durchläufen auf gleiche Weise betrachtet werden müssen (Zeilen 29, 33, 37 und 41). Fertig betrachtete Pixel werden aus der Liste entfernt (Zeile 18). Sobald die *pixelsToCheck*-Liste leer ist, endet die Breitensuche für das aktuelle Pixel (Zeile 15).

Der Schwellenwert, der festlegt, ab welcher Größe eine Region nach der Ausführung der Programmroutine erhalten bleibt, wird mithilfe eines Parameters bestimmt, der durch den Nutzer im Intervall zwischen 0 und 1 festgelegt werden kann. Dieser Parameter gibt an, wie viel Prozent der Gesamtgröße der Biom-Karte eine Region besitzen muss, um nicht eliminiert zu werden. Er wird mit der Gesamtpixelanzahl der Biom-Karte multipliziert, wodurch ein Größen-Grenzwert für Regionen als Pixelanzahl

```
Dictionary<int, int> regionSizes =
    new Dictionary<int, int>();
    //(RegionID, RegionSize)

int regionCount = 0;
for(int x = 0; x < textureSize.x; x++) {
    for(int y = 0; y < textureSize.y; y++) {
        Vector2Int startingPixel = new Vector2Int(x,y);
        if(!pixelRegions.ContainsKey(startingPixel)) {
            regionSizes.Add(regionCount, 0);
            List<Vector2Int> pixelsToCheck =
                new List<Vector2Int>();
            pixelsToCheck.Add(startingPixel);

            while(pixelsToCheck.Count > 0) {
                Vector2Int currentPixel =
                    pixelsToCheck[0];
                pixelsToCheck.RemoveAt(0);
                if(isPixelInTexture(currentPixel)
                && isPixelColorEqual(
                    startingPixel, currentPixel
                )
                && !pixelRegions.ContainsKey(
                    currentPixel)
                ) {
                    pixelRegions.Add(
                        currentPixel, regionCount);
                    regionSizes[regionCount]++;
                    pixelsToCheck.Add(new Vector2Int(
                        currentPixel.x+1,
                        currentPixel.y)
                    );
                    pixelsToCheck.Add(new Vector2Int(
                        currentPixel.x-1,
                        currentPixel.y)
                    );
                    pixelsToCheck.Add(new Vector2Int(
                        currentPixel.x,
                        currentPixel.y+1)
                    );
                    pixelsToCheck.Add(new Vector2Int(
                        currentPixel.x,
                        currentPixel.y-1)
                    );
                }
            }
            regionCount++;
        }
    }
}
```

Abbildung 19:
Iterative Breitensuche zur Ermittlung der Regionsgrößen

entsteht. Abschließend werden alle Pixel in den Regionen umgefärbt, die kleiner als der berechnete Grenzwert sind. Mindestens bleiben hierbei jedoch die beiden größten Regionen bestehen. Die neue Farbe eines Pixels, der umgefärbt werden soll, wird durch die am nächsten liegende Region bestimmt, die nicht eliminiert wird. Dafür werden die Pixel um das umzufärbende Pixel kreisförmig nach außen überprüft.

5.2.2 Opening der Biom-Karte. Die Programmroutine zum Glätten der Biomgrenzen und zum Entfernen kleinerer Biom-Fragmente führt eine morphologische Opening-Operation (siehe Abschnitt 3.3) für jede Farbe aus, die in der Biom-Karte existiert. Da Opening-Operationen nur auf Binärbildern ausgeführt werden können, wird zu jedem Zeitpunkt für die jeweilige Opening-Farbe der Wert 1 und für alle anderen Farben der Wert 0 angenommen.

```
for(int y = 0; y < textureSize.y; y++) {
    for(int x = 0; x < textureSize.x; x++) {
        if(texture.GetPixel(x,y) == erosionColor) {
            if(x-1 >= 0
            && texture.GetPixel(x-1,y) != erosionColor)
            {
                resultTexture.SetPixel(
                    x,y,texture.GetPixel(x-1,y)
                );
            } else if(x+1 < textureSize.x
            && texture.GetPixel(x+1,y) != erosionColor)
            {
                resultTexture.SetPixel(
                    x,y,texture.GetPixel(x+1,y)
                );
            } else if(y-1 >= 0
            && texture.GetPixel(x,y-1) != erosionColor)
            {
                resultTexture.SetPixel(
                    x,y,texture.GetPixel(x,y-1)
                );
            } else if(y+1 < textureSize.y
            && texture.GetPixel(x,y+1) != erosionColor)
            {
                resultTexture.SetPixel(
                    x,y,texture.GetPixel(x,y+1)
                );
            }
        }
    }
}
```

Abbildung 20:
Abgewandelte Erosionsoperation für nicht-binäre Texturen

Abbildung 20 zeigt den Code für die Erosionsoperation, die beim Opening zuerst durchgeführt werden muss. Dabei wird die Vierernachbarschaft jedes Pixels in der Biom-Karte, der die aktuelle Opening-Farbe besitzt, überprüft. Besitzt einer der Nachbarpixel eine andere Farbe, dann wird des aktuell betrachtete Pixel in dieser anderen Farbe eingefärbt.

Abbildung 21 zeigt den Code, der auf die Erosion folgenden Dilatationsoperation. Diese überprüft für jedes Pixel, der nicht die aktuelle Opening-Farbe besitzt, ob ein anderes Pixel in seiner direkten Vierernachbarschaft die Opening-Farbe besitzt. Ist dies der Fall, dann wird die Farbe des aktuell betrachtete Pixels in die Opening-Farbe abgeändert.

5.2.3 Zeichenwerkzeug für die Biom-Karte. Die Grundlage für das Zeichen-Tool, das dem Nutzer zur Verfügung gestellt wird, bildet ein Skript aus dem Unity Asset Store namens Free Draw [15]. Das Skript bietet grundlegende Zeichenfunktionen für Sprites und wurde auf das Konzept dieses Projekts angepasst. In der aktuellen Version werden in jedem Durchlauf der Update-Funktion die Unity-Weltkoordinaten des Mauszeigers in Pixelkoordinaten innerhalb der aktuell angezeigten Textur umgerechnet. Damit kann das Pixel bestimmt werden, das sich zu diesem Zeitpunkt unterhalb des Mauszeigers befindet. Innerhalb jedes Durchlaufs der Update-Funktion werden zusätzlich Betätigungen der linken Maustaste, der rechten Maustaste und des Mausrads verfolgt. Eine Betätigung der rechten Maustaste speichert die Farbe des Pixels, der sich unter dem Mauszeiger befindet, als aktuelle Zeichenfarbe ab. Das Betätigen oder Halten der linken Maustaste hingegen ändert die Farbe aller Pixel auf die aktuelle Zeichenfarbe, die sich unterhalb der Pinselfläche befinden. Die Pinselfläche ist eine kreisförmige Fläche, die stets auf das Pixel unterhalb des

```
for(int y = 0; y < textureSize.y; y++) {
    for(int x = 0; x < textureSize.x; x++) {
        if(texture.GetPixel(x,y) != dilatationColor) {
            if(x-1 > 0
            && texture.GetPixel(x-1,y)
            == dilatationColor) {
                resultTexture.SetPixel(
                    x,y,dilatationColor
                );
            } else if(x+1 < textureSize.x
            && texture.GetPixel(x+1,y)
            == dilatationColor) {
                resultTexture.SetPixel(
                    x,y,dilatationColor
                );
            } else if(y-1 > 0
            && texture.GetPixel(x,y-1)
            == dilatationColor) {
                resultTexture.SetPixel(
                    x,y,dilatationColor
                );
            } else if(y+1 < textureSize.y
            && texture.GetPixel(x,y+1)
            == dilatationColor) {
                resultTexture.SetPixel(
                    x,y,dilatationColor
                );
            }
        }
    }
}
```

Abbildung 21:
Abgewandelte Dilatationsoperation für nicht-binäre Texturen

Mauszeigers zentriert ist. Die Größe der Pinselfläche kann mithilfe des Mausrades beliebig vergrößert oder verkleinert werden.

6 TECHNISCHE EVALUATION

Dieser Abschnitt beschäftigt sich mit der Auswertung der Software, die im Rahmen dieser Arbeit entstanden ist. Insbesondere werden dabei die in Abschnitt 1.2 verfassten Ziele berücksichtigt sowie die Gebrauchstauglichkeit der Anwendung zur Generierung von Spielwelten überprüft.

6.1 Überprüfung des Realismus

Im Folgenden gilt es anhand einiger Kriterien festzustellen, wie wirklichkeitsnah die von der Anwendung generierten Biom-Karten sind.

6.1.1 Kriterium: Biom-Bestimmung. Die Biom-Bestimmung erfolgt auf Basis des Klassifikationsschemas nach Whittaker. Als ausschlaggebende Faktoren werden hierfür sowohl die Temperatur als auch die Feuchtigkeit der jeweiligen Region herangezogen. Zu diesem Zweck generiert die Anwendung eine Temperaturkarte (Abbildung 25) und eine Feuchtigkeitskarte (Abbildung 26), die im Anschluss zu einer Biom-Karte (Abbildung 27) vereint werden. Sowohl in der Temperaturkarte als auch der Feuchtigkeitskarte gilt, dass hellere Farben einen niedrigeren Intensitätswert einer Region anzeigen. Die vielen Flecken, die über die Karten verstreut sind, ergeben sich aus dem Einfluss lokaler Höhenlagen und Gewässer.

Hierbei erreichen die Karten der Anwendung einen höheren Grad an Realismus als einige vergleichbare Karten aus unterschiedlichen Spielen. In der Karte aus dem Spiel *ARK: Survival Evolved* (Abbildung 12) wird beispielsweise der Einfluss von Feuchtigkeit teilweise missachtet, wodurch eine Wüste im Süden existiert, die von Gewässern eingeschlossen und durchzogen ist. Zusätzlich grenzt diese Wüste sowohl im Westen an tropischen Regenwald als auch im Osten an Sumpfland an. Das ist eine Konstellation, die in der realen Welt so nicht vorgefunden werden kann.

Whittakers Klassifikationsschema vernachlässigt jedoch einige untergeordnete natürliche Faktoren, darunter die Bodenbeschaffenheit und die Sonneneinstrahlung. Aus diesem Grund können die von der Anwendung generierten Karten nicht die gleiche Komplexität wie reale Ökosysteme erreichen.

6.1.2 Kriterium: Biom-Verteilung. Die Anwendung vernachlässigt Breitengrade bei der Temperaturverteilung, wodurch der schichtartige Biom-Aufbau der Erde von den Polen zum Äquator in den generierten Karten vollständig fehlt. Diese Vereinfachung dient einer variableren Nutzbarkeit der Anwendung, wie in Abschnitt 4.2 genauer erläutert wird. Dennoch müssen hierdurch Einbußen im Kriterium *Realismus* hingenommen werden.

Unrealistische Nachbar-Biome, wie in der Beispielkarte von *ARK: Survival Evolved*, werden für einen höheren Grad an Realismus bei der Generierung vermieden. Der Algorithmus stellt dabei sicher, dass sich Nachbar-Biome in keinem Fall um mehr als einen Intensitätswert bezüglich der Temperatur oder des Niederschlags unterscheiden. Durch dieses Verfahren kann beispielsweise verhindert werden, dass ein feuchter tropischer Regenwald an eine trockene subtropische Wüste angrenzt.

Zuletzt muss zu diesem Kriterium angemerkt werden, dass durch die Parameter *Durchschnittstemperatur* und *Durchschnittsniederschlag* die Verantwortung für eine realistische Verteilung der Biome zu einem gewissen Teil dem Nutzer überlassen wird. Dieser kann mithilfe der Parameter eine für seine Spielwelt passende Verteilung von Temperatur und Niederschlag wählen. Die Auswirkungen des *Durchschnittsniederschlag*-Parameters in verschiedenen Ausprägungen auf die Generierung von Feuchtigkeitskarten wird in den Abbildungen 22, 23 und 24 aufgezeigt. Da die beiden Parameter auf gleiche Weise implementiert wurden, gelten die gezeigten Veränderungen analog auch für Temperaturkarten mit dem *Durchschnittstemperatur*-Parameter.

6.1.3 Kriterium: Biom-Größe. In Abschnitt 4.3.3 wird beschrieben, dass die Größen der Biome, die durch die Anwendung generiert werden, indirekt durch den *Minima-Suchradius* Parameter bestimmt werden. Dadurch liegt auch bei diesem Kriterium die Verantwortung für realistische Biom-Größen zum größten Teil beim Nutzer. Dieser muss die Größe des Bereichs, in dem nach lokalen Minima zur Verwendung als Centroiden gesucht wird, für seine Zwecke realistisch und passend auswählen. Dadurch können Biom-Karten für unterschiedlichste Anforderungen generiert werden. Ein Beispiel für eine Biom-Karte, die unter Verwendung eines *Minima-Suchradius* von 70 generiert wurde, ist in Abbildung 32 zu sehen. Im Vergleich mit der Abbildung 27, die mit einem *Minima-Suchradius* von 24 generiert wurde, fallen die Größenunterschiede der Biome deutlich auf.

6.1.4 Kriterium: Biom-Übergänge. Die Anwendung generiert Karten mit steilen Biom-Übergängen aufgrund der Verwendung des Konzepts der diskreten Biome (vgl. 4.1.3). Durch diesen Ansatz befinden sich die generierten Karten bezüglich der Biom-Übergänge abseits der Realität. Wichtig ist hierbei jedoch zu erkennen, dass dieses Kriterium weniger ausschlaggebend im Kontext digitaler Spiele ist. Abhängig vom Game Design können Karten mit diskreten Biomen genauso wie Karten mit stetigen Biomen durchaus sinnvoll in Spielen verwendet. Beispiele hierfür

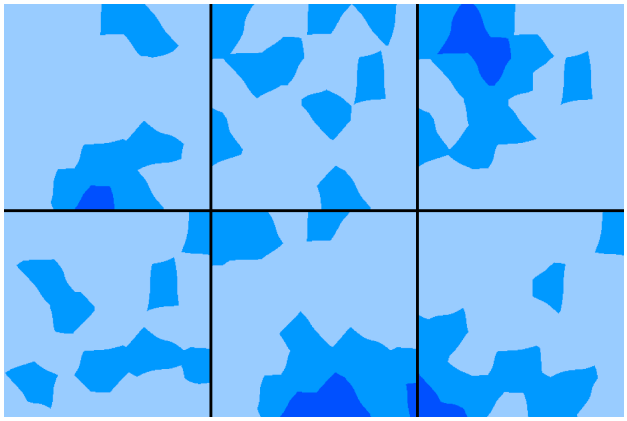

Abbildung 22: Vergleich mehrerer Feuchtigkeitskarten
Durchschnittsniederschlag-Parameter: 0.2
Quelle: Eigene Darstellung

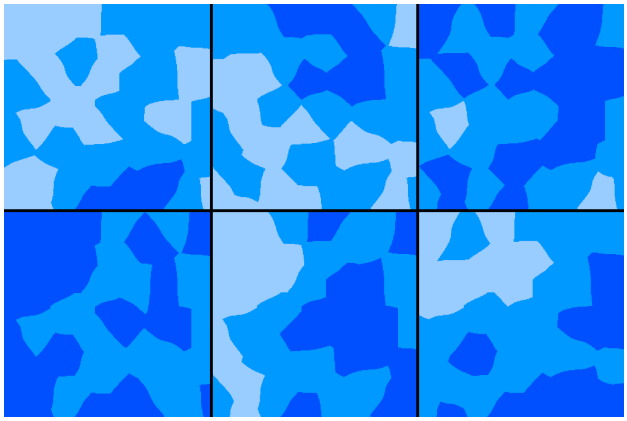

Abbildung 23: Vergleich mehrerer Feuchtigkeitskarten
Durchschnittsniederschlag-Parameter: 0.5
Quelle: Eigene Darstellung

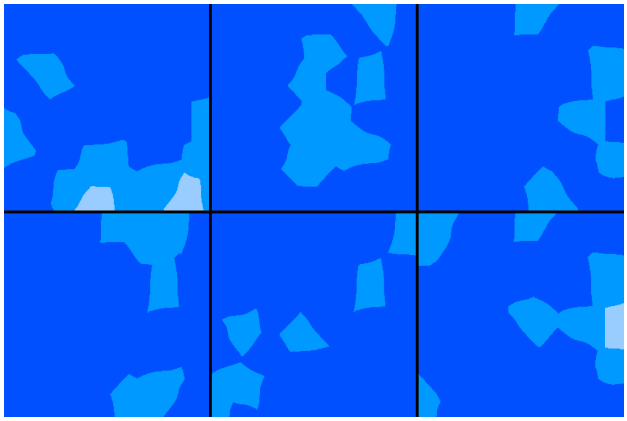

Abbildung 24: Vergleich mehrerer Feuchtigkeitskarten
Durchschnittsniederschlag-Parameter: 0.8
Quelle: Eigene Darstellung

sind die Spiele *RimWorld* und *ARK: Survival Evolved* (Abbildung 10 und 12), bei deren Umsetzung Weltkarten mit diskreten Biomen von den Entwicklern vorgezogen wurden.

6.2 Überprüfung der Gebrauchstauglichkeit

Nachfolgend wird die Tauglichkeit der Anwendung für den normalen Gebrauch untersucht. In diesem Zusammenhang werden insbesondere Laufzeittests einzelner Funktionen ausgewertet sowie die Nutzerfreundlichkeit betrachtet.

6.2.1 Laufzeiten. Im Folgenden wird die Laufzeitskalierung des Programms mit den einzelnen benutzerdefinierten Parametern ausgewertet. Bei der Durchführung der Laufzeittests wurden alle Parameter einzeln getestet. Alle anderen Parameter, abgesehen von dem jeweils getesteten, wurden zu jeder Zeit auf folgende Standardwerte festgelegt:

- Terraingröße: 100×200
- Minima-Suchradius: 24
- Durchschnittstemperatur: 0.5
- Durchschnittsniederschlag: 0.5
- Minkowski-P-Wert: 3
- Temperaturabfallhöhe: 0.7
- Feuchtigkeitsausbreitung: 5
- Eliminierungs-Größenfaktor: 0.005

Die Parameter *Durchschnittstemperatur*, *Durchschnittsniederschlag*, *Minkowski-P-Wert*, *Temperaturabfallhöhe* und *Feuchtigkeitsausbreitung* haben bei den Tests auch bei extremen Werten keine Schwankungen in der Laufzeit aufgezeigt. In jedem Fall liegt bei beliebiger Einstellung dieser Parameter und der standardmäßigen Einstellung aller anderen Parameter die Gesamtlaufzeit für die Generierung einer Biom-Karte bei etwa 220 Millisekunden. Die Berechnung eines einzelnen Voronoi-Diagramms nimmt dabei etwa 80 Millisekunden der Gesamtlaufzeit in Anspruch. Eine Ausnahme in diesem Muster entsteht, wenn der *Minkowski-P-Wert* auf 1 (Manhatten-Distanz) festgelegt wird. In diesem Fall verringert sich die Berechnungszeit eines Voronoi-Diagramms auf etwa 35 Millisekunden und die Gesamtlaufzeit auf ungefähr 130 Millisekunden.

Besondere Beachtung sollte der Wahl der übrigen drei Parameter zufallen, um ausufernde Laufzeiten zu vermeiden. Die Terraingröße gibt die Anzahl der Knoten vor, über denen die Biom-Karte berechnet wird. Dementsprechend sind mit steigender Terraingröße vor allem bei den Voronoi-Diagrammberechnungen auch höhere Laufzeiten vorzufinden. Die Berechnungszeit eines Voronoi-Diagramms kann hierbei von 5 Millisekunden für kleines Terrain mit 2500 Knoten bis hin zu 560 Millisekunden für ein größeres Terrain mit 90000 Knoten variieren. Da die Laufzeit quadratisch mit der Anzahl der Knoten skaliert, sind Terraingrößen oberhalb der 100000 Knoten nicht mit einer flüssigen Verwendung der Anwendung vereinbar.

Beim Einstellen des *Minima-Suchradius*-Parameters sollte beachtet werden, dass bei kleinen Werten im Bereich von 0 bis 10 sämtliche lokalen Minima im Terrain in weniger als 10 Millisekunden gefunden werden. Da durch einen kleinen *Minima-Suchradius* jedoch tendenziell die Anzahl der Centroiden für die Voronoi-Diagrammberechnung stark ansteigt, kann der gesamte Algorithmus trotzdem bis zu 800 Millisekunden in Anspruch nehmen. Wählt man wiederum besonders große Werte für den *Minima-Suchradius*, dann reduziert sich die Berechnungszeit der Voronoi-Diagramme drastisch, da nur einige wenige Centroiden berücksichtigt werden müssen. Die Zeit für das Finden der lokalen Minima innerhalb des Terrains steigt in diesem Fall jedoch, weil wesentlich mehr Nachbarn für jedes Pixel betrachtet werden müssen. Wählt man beispielsweise 160 als Parameterwert, dann erreicht man ebenfalls etwa 800 Millisekunden Gesamtlaufzeit.

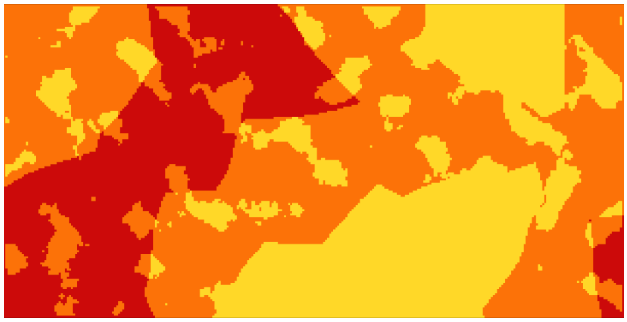

Abbildung 25: Standardmäßige Temperaturkarte
Quelle: Eigene Darstellung

Abbildung 26: Standardmäßige Feuchtigkeitskarte
Quelle: Eigene Darstellung

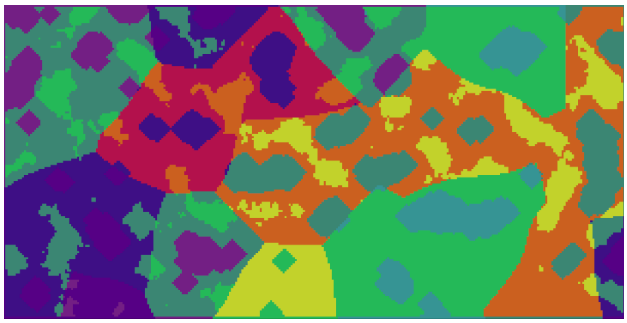

Abbildung 27: Biom-Karte, kombiniert aus 25 und 26
Quelle: Eigene Darstellung

Abbildung 28: Nachbearbeitungsroutinen auf 27 ausgeführt
Quelle: Eigene Darstellung

Abbildung 29: Oranges Biom manuell in 28 eingezeichnet
Quelle: Eigene Darstellung

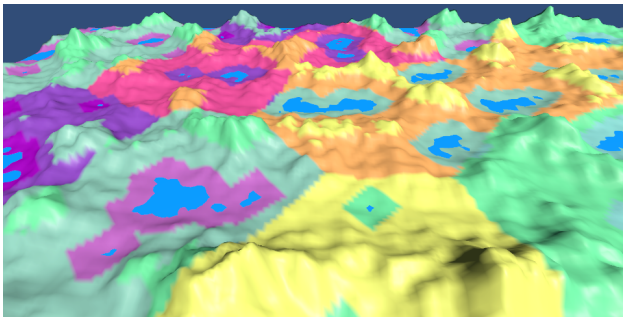

Abbildung 30: Terraindarstellung der Biom-Karte 27
Quelle: Eigene Darstellung

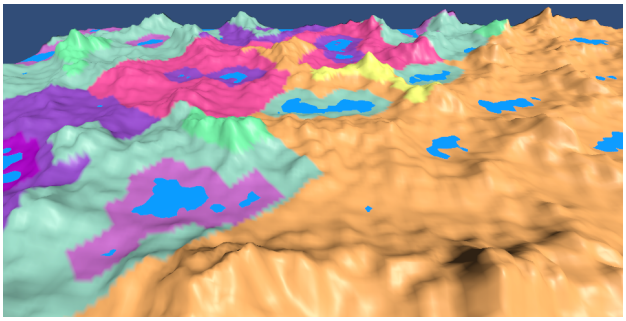

Abbildung 31: Terraindarstellung der Biom-Karte 29
Quelle: Eigene Darstellung

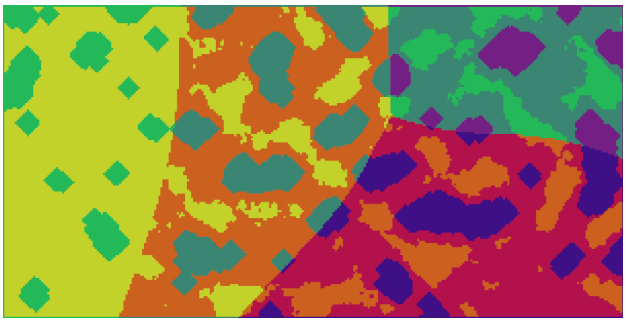

Abbildung 32: Neue Karte mit *Minima-Suchradius* von 70
Quelle: Eigene Darstellung

Für die geringsten Laufzeiten muss hier ein Mittelwert gewählt werden, der sowohl die Minima-Berechnung als auch die Voronoi-Diagrammberechnung minimiert. In den Tests lag dieser Wert bei 35, wodurch eine Gesamtlaufzeit von 182 Millisekunden erzielt wurde.

Zuletzt wird die Wahl des *Eliminierungs-Größenfaktors* betrachtet, um zu kleine Biome in einer zumutbaren Laufzeit von der Karte entfernen zu können. Dieser wird zwischen 0 und 1 festgelegt. Unter Verwendung der Standardeinstellungen bewegen sich die möglichen Laufzeiten für die Durchführung der Programmroutine zwischen 62 Millisekunden für den Parameterwert 0 und 974 Millisekunden für den Parameterwert 1. Für alle Werte bis 0.5 wird dabei eine Laufzeit von 100 Millisekunden nicht überschritten. Da der Parameter in einen Prozentsatz der Gesamtfläche des Terrains umgerechnet wird, skaliert der Einfluss des Parameters auf die Laufzeit auch mit der Terraingröße. Wird die Terraingröße beispielsweise von den standardmäßigen 20000 Knoten auf 90000 Knoten angehoben, dann nimmt die Programmroutine schon bei relativ geringen Werten wie 0.02 einige Sekunden in Anspruch. Steigert man den Parameter auf 0.05, dann muss man bereits mit bis zu 5 Minuten Laufzeit rechnen. Die Laufzeitskalierung des Parameters ist also, vor allem bei größerem Terrain, eher schlecht zu bewerten. Wichtig ist hierbei jedoch anzumerken, dass der Standardwert für den Parameter von 0.005 eine geringe Laufzeit und damit eine flüssige Bedienbarkeit der Anwendung für die gängigen Terraingrößen garantiert und für den normalen Gebrauch nicht abgeändert werden muss.

6.2.2 Nutzerfreundlichkeit. Im Rahmen dieses Projektes wurde lediglich die Funktionalität des Biom-Generators ohne eine gesonderte Benutzeroberfläche erstellt. Stattdessen müssen die einzelnen Parameter in der aktuellen Version im Unity-Inspektor eingestellt werden. Aus diesem Grund sind für die reibungslose Bedienung der Anwendung Kenntnisse mit der Unity Game Engine sowie eine Einweisung in die Projekthierarchie notwendig, was sich negativ auf die Zugänglichkeit der Anwendung auswirkt. Außerdem können die schlechten Laufzeitskalierungen einiger Parameter ohne Warnung zu unerwartet langen Programmaussetzern führen. Diese Aussetzer könnte der Nutzer als störend wahrnehmen. Im Weiteren ermöglicht die farbliche Biom-Darstellung der aktuellen Programmversion zwar die Unterscheidung der unterschiedlichen Biome auf der Karte, es ist jedoch nicht möglich einzelne Biome zu bestimmen. An dieser Stelle fehlt eine Art Legende, die jeder Biom-Farbe entweder eine Biom-Bezeichnung zuordnet, oder zumindest das identifizierende Tupel aus Temperatur- und Feuchtigkeitswert für den Nutzer anzeigt. Um dabei auch noch für Barrierefreiheit zu sorgen, wäre es vorteilhaft, wenn der Nutzer die Biom-Farben selbstständig anpassen könnte. Zusätzlich fehlt dem Nutzer bei der Verwendung des Zeichenwerkzeugs Feedback dazu, welche Pixel genau durch einen Klick abgeändert werden. Hier wäre eine Vorschau der zu zeichnenden Fläche durch temporäres Einfärben der Pixel innerhalb der Pinselfläche um den Mauszeiger sinnvoll.

Andererseits ist die Anwendung jedoch auch sehr universell anwendbar, was sich positiv auf die Nutzerfreundlichkeit auswirkt. Die vielen benutzerdefinierten Parameter ermöglichen die Anpassung des Generators auf unterschiedlichste Bedürfnisse und Game Designs. Unter anderem kann der Nutzer die Biom-Größe, die Biom-Verteilung, die Krümmung der Biom-Grenzen sowie den Einfluss von Höhe und Gewässern auf die Temperatur und den Niederschlag einstellen. Zusätzlich sind die Terrainausmaße durch die Missachtung von Breitengrad-Temperaturen nicht an die Einhaltung einer minimalen oder maximalen Weltengröße geknüpft und können stattdessen beliebig gewählt werden. Außerdem existieren innerhalb der Anwendung sowohl 2D als auch 3D Kartenansichten, die es dem Nutzer ermöglichen, seine Ergebnisse besser zu betrachten und im Folgenden weiter zu bearbeiten.

7 ZUSAMMENFASSUNG UND AUSBLICK

Diese Arbeit befasst sich mit unterschiedlichen Ansätzen zu Umsetzung eines Biom-Generators für bestehendes Terrain zur Verwendung in digitalen Spielen. Es wird auf die zugrundeliegenden theoretischen Grundlagen eingegangen und eines von vielen möglichen Konzepten für einen solchen Generator erstellt. Dabei mussten jedoch auch einige Qualitätseinbußen hingenommen werden, um die in Abschnitt 1.2 gesetzten Ziele erreichen zu können. Im weiteren Verlauf der Arbeit wurde ein Prototyp implementiert und dokumentiert, der das Konzept umsetzt.

Im Verlauf der Arbeit sind einige Erweiterungsmöglichkeiten für das Projekt aufgekommen, die im Folgenden besprochen werden. Zum einen wäre es denkbar, dass das Biom-Modell noch näher an die Bedingungen der realen Welt angepasst wird. Dafür könnte man zusätzliche Faktoren bei der Biom-Bestimmung simulieren, die im Klassifikationsschema von Whittaker vernachlässigt werden. Hierzu gehören beispielsweise der Einfluss der Bodenbedingungen sowie der Sonneneinstrahlung auf die Biome. Außerdem wird in der aktuellen Version der Einfluss des Windes vollständig ignoriert. Mithilfe eines Systems zur Simulation von Hoch- und Tiefdruckgebieten könnten Windstärken und Windrichtungen bestimmt werden, die sich wiederum auf den Niederschlag auswirken würden. In diesem Zusammenhang wäre die Berechnung von Regenschatten an hohen Gebirgen im Terrain denkbar.

Des Weiteren existieren verschiedene Probleme auf Seiten der Nutzerfreundlichkeit der Anwendung, die in zukünftigen Arbeiten adressiert werden könnten. In Abschnitt 6.2.2 wird näher auf die Problematik eingegangen. Der wichtigste Punkt ist in diesem Zusammenhang die Umsetzung einer intuitiven Benutzeroberfläche, um die Einarbeitungszeit zu minimieren und den Nutzer bei der Auswahl geeigneter Parameter zur Biom-Generierung zu unterstützen.

Abschließend muss natürlich auch der Ausbau der generierten Spielwelten mit unterschiedlichsten Assets als Entwicklungsmöglichkeit aufgeführt werden. Dazu zählen neben Vegetation auch Tiere, Siedlungen und NPCs. Da die Auswahl an Gestaltungsmöglichkeiten in diesem Punkt jedoch beinahe unbegrenzt ist, muss einer Erweiterung der Anwendung in diesem Bereich ein passendes Game Design, ein Spiel-Genre oder zumindest ein Thema zugrunde liegen, um auf ein konkretes Ziel hinarbeiten zu können.

LITERATUR

[1] Mayara Albuquerque. 2020. 13 Tips For Writing a Good Text Adventure Game. (2020). https://www.davidepesce.com/2020/02/26/13-tips-for-writing-a-good-text-adventure-game/ (besucht am 18.01.2023).

[2] ark.fandom.com. 2022. Crystal Isles. (2022). https://ark.fandom.com/wiki/Crystal_Isles (besucht am 18.01.2023).

[3] Kurt E. Brassel and Douglas Reif. 1979. A Procedure to Generate Thiessen Polygons. *Geographical Analysis* 11, 3 (1979), 289–303. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1538-4632.1979.tb00695.x

[4] Timo Conradi, Jasper A. Slingsby, Guy F. Midgley, Henning Nottebrock, Andreas H. Schweiger, and Steven I. Higgins. 2020. An operational definition of the biome for global change research. *New Phytologist* 227, 5 (2020), 1294–1306. https://nph.onlinelibrary.wiley.com/doi/abs/10.1111/nph.16580

[5] Datenbanken-verstehen.de. 2023. Graphalgorithmen. (2023). https://www.datenbanken-verstehen.de/lexikon/graphalgorithmen/ (besucht am

16.01.2023).
[6] Marcus Gagelmann. 2022. Entwicklung eines prozeduralen Landschaftsgenerators. (2022).
[7] geeksforgeeks.org. 2022. Flood fill Algorithm – how to implement fill() in paint? (2022). https://www.geeksforgeeks.org/flood-fill-algorithm-implement-fill-paint/ (besucht am 16.01.2023).
[8] Randy Kalista. 2020. Mount Blade II: Bannerlord Preview – A Banner Year. (2020). https://www.gamingnexus.com/Article/6098/Mount--Blade-II-Bannerlord-Preview-e28093-A-Banner-Year/ (besucht am 18.01.2023).
[9] George Law. 2013. Quantitative comparison of flood fill and modified flood fill algorithms. *International Journal of Computer Theory and Engineering* 5, 3 (2013), 503–508.
[10] U. Mortens. 2023. Multidimensionale Skalierung. (2023). http://www.uwe-mortensen.de/MDS.pdf (besucht am 16.01.2023).
[11] Ladislav Mucina. 2019. Biome: evolution of a crucial ecological and biogeographical concept. *New Phytologist* 222, 1 (2019), 97–114. https://nph.onlinelibrary.wiley.com/doi/abs/10.1111/nph.15609
[12] A.M Raid, Wael Khedr, Mohamed El-dosuky, and Mona Aoud. 2014. Image Restoration Based on Morphological Operations. *International Journal of Computer Science, Engineering and Information Technology* 4 (07 2014), 9–21. https://doi.org/10.5121/ijcseit.2014.4302
[13] C. A. Rishikeshan. 2018. Researchgate. (2018). https://www.researchgate.net/figure/Illustration-of-an-a-Opening-and-b-Closing-operation-Where-f-is-the-input-image-B_fig3_327498264 (besucht am 16.01.2023).
[14] Steam. 2016. Quick teaser: The Planet. (2016). https://store.steampowered.com/news/app/294100/view/2909844527767358331 (besucht am 18.01.2023).
[15] TeslaKim. 2019. Free Draw. (2019). https://assetstore.unity.com/packages/tools/painting/free-draw-simple-drawing-on-sprites-2d-textures-113131 (besucht am 13.02.2023).
[16] Robert Harding Whittaker et al. 1970. Communities and ecosystems. *Communities and ecosystems.* (1970).
[17] Wikipedia. 2022. Biome. (2022). https://en.wikipedia.org/wiki/Biome (besucht am 12.01.2023).
[18] Wikipedia. 2022. Voronoi diagram. (2022). https://en.wikipedia.org/wiki/Voronoi_diagram#Informatics (besucht am 12.01.2023).