

Bachelor-Thesis

Rendern von Voxeln mit mehreren Materialien

zur Erlangung des akademischen Grades

Bachelor of Science der Informatik

vorgelegt von:

Matthias Mettenleiter

6. März 2018

1. Gutachter: Prof. Dr. Daniel Scherzer
2. Gutachter: Prof. Dr. rer. nat. Sebastian Mauser

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben. Die Arbeit wurde bisher, in gleicher oder ähnlicher Form, keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Unterschrift

Ort, Datum

Zusammenfassung

Diese Bachelorarbeit befasst sich mit Methoden zur Darstellung von Voxeln, mit dem Ansatz mehrere Materialien innerhalb eines Voxels zu speichern. Hierfür wurden mehrere Methoden entwickelt, die an bekannte Techniken zum Rendern von Voxeln mit einem Material angelehnt sind. Es wurde eine praktische Anwendung umgesetzt deren Ergebnisse in einem Vergleich verwendet werden.

Ein Teil der entwickelten Methoden ist nur von den Materialien innerhalb des darzustellenden Voxels abhängig. Dadurch ist das Gesamtbild, das durch mehrere Voxel entsteht nicht immer homogen.

Der andere Teil der Methoden befasst sich mit Abhängigkeit der Darstellung von der Verteilung der Materialien in benachbarten Voxeln. Dabei wird der Renderprozess aufwendiger, während ein homogeneres Gesamtbild entsteht.

Abstract

This bachelor thesis concerns itself with methods for rendering voxels with the approach of having more than one material defined within each voxel. In order to achieve this, several methods were developed using known techniques for rendering voxels with one material as a starting point. The results of an application which was created for the thesis were used to compare these methods.

A part of the developed methods is only dependent on the materials inside the voxel which is being rendered. As a result, the whole picture created by multiple voxels does not always look homogenous.

The other methods describe the visualization of voxels in relation to their surroundings. By using the relations between voxels, the rendering process becomes more complicated, but the overall result looks more homogenous.

Danksagung

Ich möchte mich hiermit bei allen bedanken, die mir bei der Bachelorarbeit geholfen haben. Dabei möchte ich mich besonders bei Georg Guetschow, Anja Bußjäger und Cornelia Mettenleiter bedanken, die die Arbeit Korrektur gelesen haben und mich motiviert haben. Auch möchte ich mich bei meinem Betreuer Daniel Scherzer bedanken, der mich mit Literaturhinweisen unterstützt hat und mich bei Problemen in der praktischen Umsetzung auf codetechnische Alternativen hingewiesen hat.

Inhaltsverzeichnis

1 Einleitung	6
1.1 Motivation	6
1.2 Fragestellung	6
1.3 Zielsetzung	6
1.4 Voraussetzung	7
1.5 Erklärung des Begriffs Voxel	7
2 Stand der Technik	8
2.1 Einsatzbeispiele von Voxel	8
2.1.1 Medizinischer Bereich	8
2.1.2 Spielebereich	9
2.2 Technologien zum Rendern	12
2.2.1 Rendern als Meshes	12
2.2.2 Rendern via Raytracing	16
2.3 Chunks als Optimierungsmethode	17
3 Methode	18
3.1 Konzepte zur Grundstruktur	18
3.1.1 Aufbau eines Voxels	18
3.1.2 Verbindung der Voxel zum Rendern	19
3.2 Verwendete Struktur in praktischer Umsetzung	20
3.3 Ansätze zum Rendern	21
3.3.1 Eine Textur pro Voxel	21
3.3.2 Eine Textur pro Richtung	24
3.3.3 Materialien entlang eines Vektors	27
3.3.4 Abhängigkeiten zwischen Materialien	33
4 Ergebnisse	34
4.1 Vergleich	34
4.1.1 Kompatibilität mit üblichen Techniken	34
4.1.2 Verhalten bei Darstellung einer Landschaft	35
4.2 Fazit	42
5 Ausblick	43
Abbildungsverzeichnis	44
Literatur	45

1 Einleitung

1.1 Motivation

Eine Vielzahl von Anwendungen arbeitet inzwischen mit Voxeln. Die meisten davon verwenden Voxel um Landschaften darzustellen. Dabei wird jedem Voxel ein Material zugeteilt, damit die Landschaft nicht nur von der Form, sondern auch von der Textur her realistisch gestaltet wird. Dabei werden häufig Materialtypen erzeugt, die eigentlich Mischungen aus Materialien sind, wie sie in unserer Welt natürlich vorkommen. Beispielsweise wird oft ein Voxel als Material Gras definiert, wird dann aber als Gras mit Erde dargestellt.

1.2 Fragestellung

Diese Thesis befasst sich mit dem Ansatz, mehrere Materialien pro Voxel zu definieren. Dadurch soll die Möglichkeit für Materialverteilungen innerhalb und zwischen den Voxeln verbessert werden. Dies ermöglicht es, verschiedene Materialgemische zu beschreiben. Dabei ist die Frage, wie diese Gemische dann dargestellt werden. So sind in einer natürlichen Landschaft zwar auch Gemische vorhanden, jedoch ist bei Betrachtung der Landschaft auch immer eine gewisse Homogenität gegeben. Wird beispielsweise eine Wiese betrachtet so ist es möglich, dass diese einen gewissen Anteil an Steinen und anderen Materialien wie Holz enthält, jedoch ist erstmal nur das Gras zu sehen.

1.3 Zielsetzung

Das Ziel dieser Arbeit ist es, Methoden zu vergleichen, die es ermöglichen Voxel darzustellen, für die jeweils, nicht wie üblich ein Material, sondern eine Gruppe von Materialien definiert ist. Dies ermöglicht das Anzeigen von Materialverteilungen zwischen und innerhalb der Voxel.

Im Zuge dieser Arbeit wird beschrieben, wie Voxel aufgebaut werden, um mehrere Materialien zu enthalten. Weiterhin wird darauf eingegangen welche Möglichkeiten bestehen Verbindungen zwischen Voxeln herzustellen, um die Verteilung der Materialien innerhalb mehrerer Voxel zu betrachten.

Es werden Methoden zur Darstellung der Materialien gezeigt, die an bewährte Technologien, zum Rendern von Voxeln mit nur einem Material, angelehnt sind. Auch werden Ansätze beschrieben, die spezifisch auf die Verteilung der Materialien innerhalb der Voxel und ihrer Nachbarn eingehen.

Der Vergleich wird mithilfe der Anwendung der Methoden in einer praktischen Arbeit durchgeführt.[10]

1.4 Voraussetzung

Diese Thesis bespricht Methoden im Bereich der Computergrafik und ist an eine Leserschaft gerichtet, die ein grundlegendes Verständnis in diesem Bereich besitzt. Es wird von einer Grundkenntnis über Begriffe, wie beispielsweise UV-Mapping, Vertices und Meshes ausgegangen. Auch werden Kenntnisse im Bereich der Vektorrechnung vorausgesetzt.

Genauere Erklärungen dieser Grundlagen sind in den Quellen [11] und [13] zu finden.

1.5 Erklärung des Begriffs Voxel

Der Begriff Voxel steht für Volumetric Pixel oder Volume Element. Ein Voxel ist das 3D-Pendant zu einem Pixel. Während ein Pixel, das von Picture Element hergeleitet wird, eine Möglichkeit darstellt einen 2D-Raum in einheitliche Zellen zu unterteilen, macht ein Voxel das gleiche für einen 3D-Raum. Dabei teilt ein Pixel den 2D-Raum meist in Quadrate und ein Voxel den 3D-Raum meist in Würfel ein. Eine Eigenschaft, die sich die Beiden teilen ist, dass sie ihre Positionskoordinaten nicht explizit speichern. Stattdessen wird die Position eines Voxels über die relative Position zu andern Voxeln und dem Koordinatenursprung bestimmt.[5]

Voxel werden auch häufig im Zusammenhang mit Point-Clouds genannt. Hierbei ist jedoch der Unterschied, dass in einer Point-Cloud jeder Punkt seine Koordinaten speichert. Auf das Rendern von Point-Clouds wird im weiteren nicht eingegangen, da der Schwerpunkt dieser Arbeit bei Methoden für Voxel liegt.

2 Stand der Technik

2.1 Einsatzbeispiele von Voxel

Im Folgenden werden Beispiele für die Einsatzgebiete von Voxeln gezeigt. Dabei wird zuerst auf den medizinischen Bereich eingegangen, woraufhin gezeigt wird, wo und wie Voxel in digitalen Spielen verwendet werden.

2.1.1 Medizinischer Bereich

Voxel haben ihren Ursprung in der Medizin. Ursprünglich wurden sie verwendet, um das Ergebnis einer Computertomographie(CT) darzustellen. Diese wurde 1973 von Godfrey Hounsfield vorgestellt.[5] Diese Technologie wird heute noch verwendet. Dabei wird der abzubildende Teil des Körpers schichtweise über Röntgen eingescannt. Oft liegen diese Schichten weiter auseinander, als die Voxel im darzustellenden Ergebnis groß sind, weshalb dann zwischen den Schichten interpoliert wird, um eine volle volumetrische Darstellung zu erhalten.[13] Hierfür wird inzwischen auch häufig das in Abschnitt 2.2 beschriebene Marching Cubes-Verfahren verwendet, um die Ergebnisse glatter darzustellen und näher an die Originalform heranzukommen, wie es in Abbildung 1 zu sehen ist.[14]

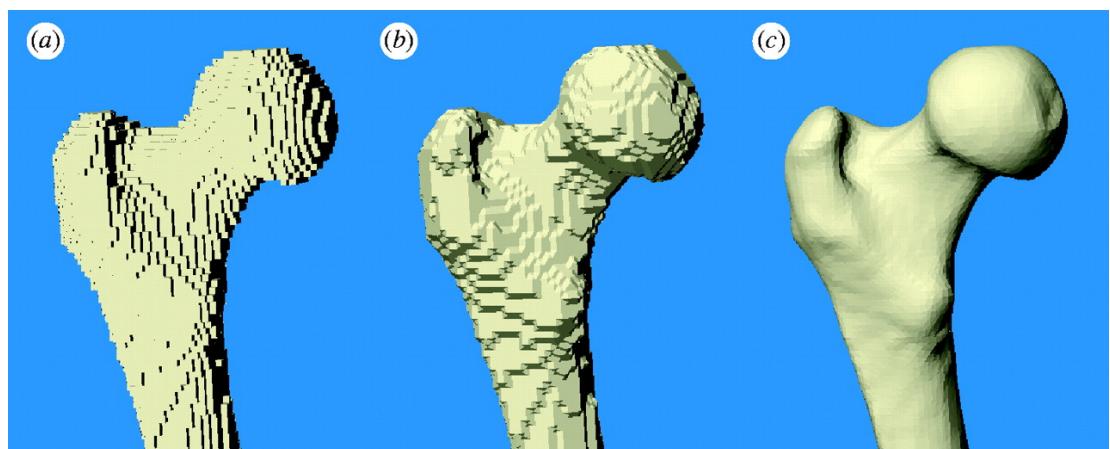


Abbildung 1: Rekonstruktion des CT Scans eines Knochens: (a) Voxel mesh, (b) Mesh mit Marching Cubes Algorythmus (ohne Interpolation), (c) Mesh mit Marching Cubes Algorythmus (mit Interpolation).

2.1.2 Spielebereich

Im Spielebereich kamen Voxel zum ersten mal 1992 mit *Comanche: Maximum Overkill* zum Einsatz. Es handelt sich dabei um einen Flugsimulator, in dem Voxel zur Darstellung von Terrain verwendet werden, wie es in Abbildung 2 zu sehen ist. Verglichen mit den zu dieser Zeit gängigen Vektor-Grafiken bietet die Voxel-Darstellung eine weit realistischere und detailliertere Darstellung der Landschaft.[5] Im Folgenden werden Beispiele aus anderen Genres genannt.



Abbildung 2: Screenshot aus Comanche: Maximum Overkill.

2008 erschien als Beispiel für einen Shooter das Spiel *Voxelstein 3D* als spiritueller Nachfolger des Spiels *Wolfenstein 3D* aus dem Jahr 1992. *Voxelstein 3D* ist komplett aus Voxeln aufgebaut, die, wie in Abbildung 3 dargestellt ist, auch teilweise zerstört werden können.



Abbildung 3: Screenshot aus Voxelstein 3D

Auch wenn vorher bereits andere Spiele mit dieser Struktur bekannt waren, ist das 2009 erschienene Spiel *Minecraft* aktuell das meist bekannte voxelbasierte Sandbox-Spiel, bei der die gesamte Welt aus Voxeln aufgebaut ist, wie in Abbildungen 4 und 5 gezeigt wird. Alle Voxel können abgebaut werden und es können neue Voxel platziert werden. So hat das Spiel inzwischen zwar einen Survival-Modus, in dem der Spieler überleben und sich Werkzeuge, Waffen und Nahrungsmittel herstellen muss, ist aber vom Kern her ein Spiel, bei dem der Spieler, ähnlich wie mit einer unbegrenzten Anzahl Lego-Steinen, kreativ alles bauen kann.



Abbildung 4: Minecraft im Adventure Modus



Abbildung 5: Frei gebautes Gebäude in Minecraft

2.2 Technologien zum Rendern

Die zwei meist verwendeten Technologien zum Rendern von Voxeln sind die Umwandlung von Voxeln in Meshes und das Raytracing von Voxeln. Auf diese Beiden wird im Folgenden näher eingegangen.

2.2.1 Rendern als Meshes

Voxel können auf verschiedene Arten in Meshes umgewandelt werden, um sie darzustellen. Die Techniken, die dazu am häufigsten verwendet und im folgenden beschrieben werden, sind das Verwenden von fixen Meshes pro Voxel, Marching Cubes und Blobby Objects.

Als fixer Mesh

Voxel werden als ein definierter Mesh dargestellt. Dieser kann beliebig aussehen. Meist wird jedoch ein Würfel verwendet. So werden beispielsweise die meisten Voxel in dem in Abschnitt 2.1.2 beschriebenen Spiel *Minecraft* als Würfel dargestellt. Jedoch gibt es auch Voxel wie zum Beispiel den Amboss, Fackeln und weitere Gegenstände, wie sie auf Abbildung 6 zu sehen sind, die andere Meshes verwenden. Im Normalfall werden hierbei die Materialien dadurch dargestellt, dass, wie beim in Abschnitt 2.1.2 gezeigten Spiel *Voxelstein 3D*, eine Farbe pro Voxel dargestellt wird oder wie in *Minecraft* Texturen auf die Seiten der Voxel projiziert werden. Dabei werden teilweise, wie auch in Abbildung 6 beim Amboss, verschiedene Texturen pro Seite verwendet.



Abbildung 6: Verschiedene Meshes in Minecraft

Marching Cubes

Der Marching Cubes Algorithmus wurde 1987 von Lorensen und Cline vorgestellt. Hierbei werden immer acht Voxel zugleich betrachtet, um zwischen diesen eine Schnittfläche zu zeichnen, die die gefüllten Voxel von den nicht gefüllten trennt. Es gibt so 256 Möglichkeiten diese Voxel zu unterteilen. Diese Anzahl kann aufgrund der Symmetrie auf die in Abbildung 7 gezeigten 15 Möglichkeiten reduziert werden.[13]

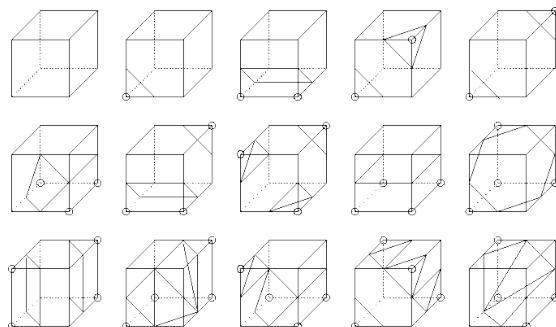


Abbildung 7: Möglichkeiten für Marching Cubes

Für gewöhnlich liegen die Eckpunkte der Polygone hierbei immer auf der Mitte der Strecke zwischen zwei Voxeln. Zur Darstellung von gescannten Objekten werden diese jedoch oft auf dieser Strecke verschoben, um näher an die Proportionen des originalen Objektes heranzukommen. Wie in Abbildungen 8 und 9 gezeigt, werden die Vertices hierbei auf den Schnittpunkt des Vektors zwischen den Voxeln mit der eingescannten Grenze des Objektes geschoben.[4]

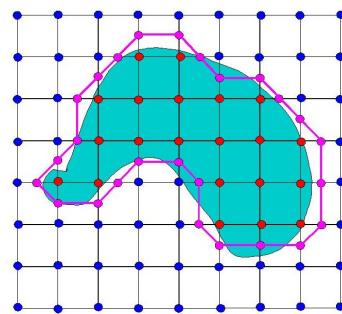


Abbildung 8: Verschiebung bei Marching Cubes vorher

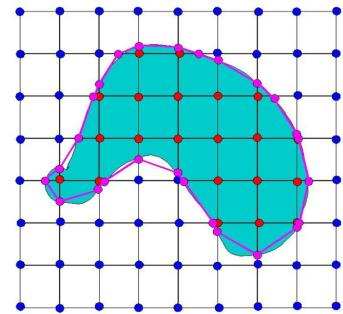


Abbildung 9: Verschiebung bei Marching Cubes nachher

Zum Darstellen von Materialien werden hier die selben Techniken verwendet, wie bei der Darstellung von fixen Meshes. Dadurch, dass die Polygone immer zwischen 8 Voxeln generiert werden, kann auch gut zwischen den Materialien auf der Polygonfläche interpoliert werden, wie es auf Abbildung 10 gezeigt wird.[7]



Abbildung 10: Marching Cubes mit interpolierten Texturen

Dazu werden wie in Abbildung 11 jedem Vertex eines Polygons noch drei UV-Positionen für die drei Materialien und ein Vektor zur Gewichtung der Materialien übergeben. Anhand des Gewichtungsvektors werden die Werte der drei Texturen dann gemischt.[7]

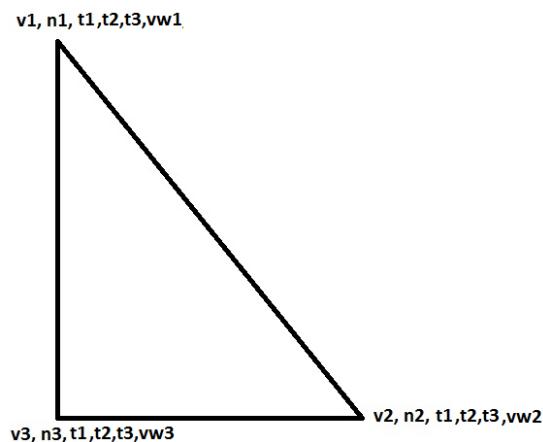


Abbildung 11: Polygon mit zusätzlichen Werten zur Interpolation: (v^*) Vertex Position, (n^*) Normale, (t_1, t_2 und t_3) Materialien der drei Richtungen, (vw^*) Gewicht der Materialien am Vertex

Blobby Objects

Die grundsätzliche Idee hinter Blobby Objects, die auch als Metaballs bekannt sind, ist es, um Punkte Kugeln zu zeichnen, die sich gegenseitig anziehen und miteinander verschmelzen können, wie es auf Abbildung 12 gezeigt ist. Dazu werden unterschiedliche Formeln definiert, die zum Einen den Abstand des Materials um den Punkt und zum Anderen den Einflussbereich des Punktes auf das Material benachbarter Punkte berechnen.[6]

Blobby Objects können nicht nur für Voxel verwendet werden, sondern werden auch verwendet um eine Point-Cloud zu rendern.

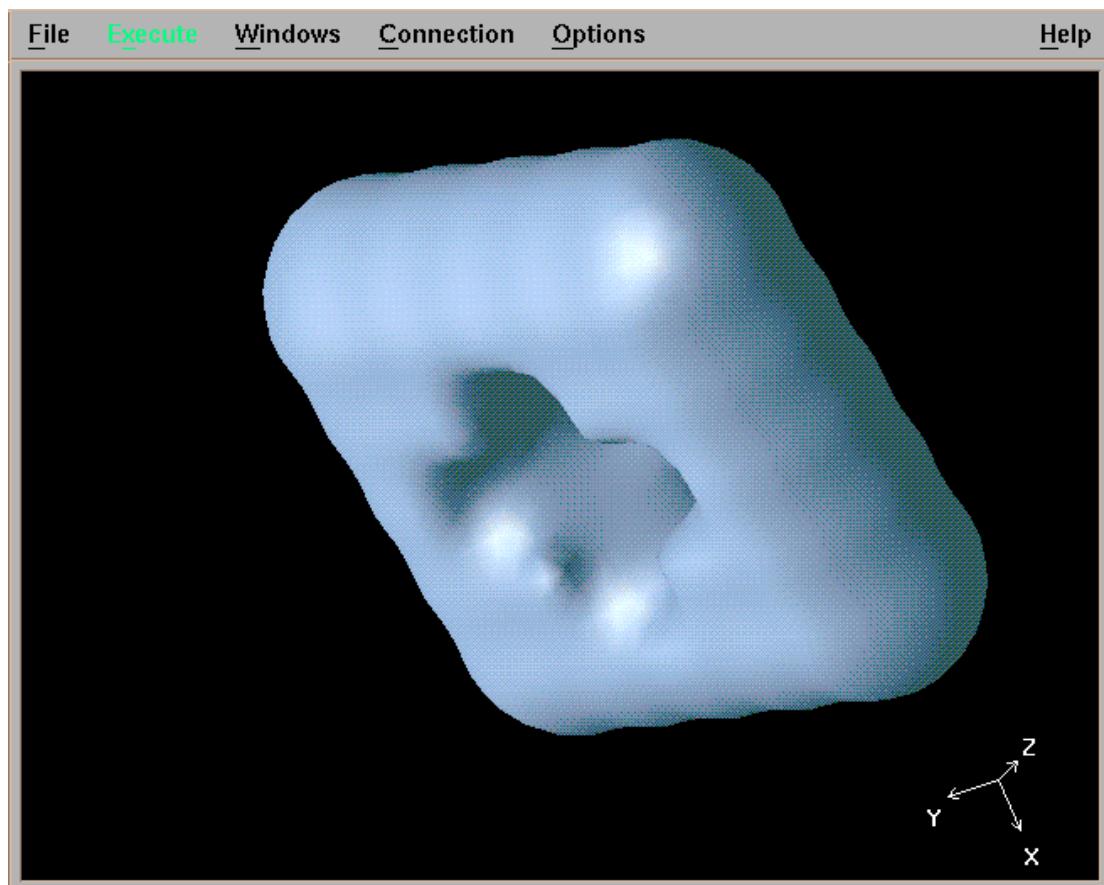


Abbildung 12: Blobby Objects

2.2.2 Rendern via Raytracing

Beim Raytracing werden für jeden zu rendernden Pixel ein oder mehrere Strahlen (sogenannte Rays) in die Welt geschickt und geben dann zurück, mit was sie kollidiert sind, wie es in Abbildung 13 gezeigt wird. Diese Informationen werden dann verwendet, um zu bestimmen, wie dieser Pixel gesetzt werden soll. Dazu werden dann die Position der Kolission des Rays mit dem Voxel, die Normale des Voxels an dieser Stelle und die Positionen und Farben von Lichtquellen zur Berechnung der darzustellenden Farbe an diesem Pixel verwendet.[11]

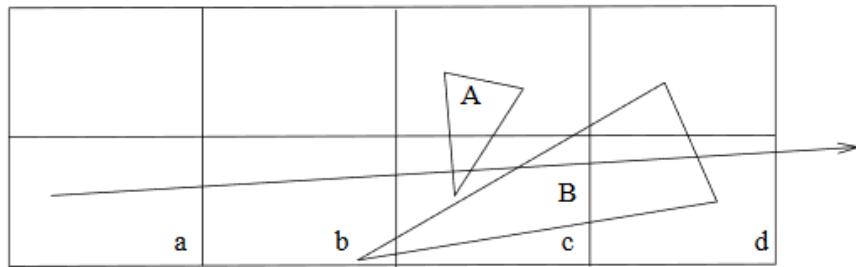


Abbildung 13: Darstellung zu Raytracing in 2D

Dadurch, dass hier keine Polygone definiert werden, auf die eine zweidimensionale Textur abgebildet werden kann, wird hier eine 3D-Textur zur Texturierung eines Objektes verwendet, wie es auf Abbildung 14 der Fall ist.[8]



Abbildung 14: Texturierter Voxel-Hase

2.3 Chunks als Optimierungsmethode

Ein Chunk ist eine Ansammlung von Voxeln, die sowohl die Performance des Programmes als auch die Erweiterbarkeit und Veränderbarkeit der Welt optimiert. Würden die Voxel alle direkt in der Welt sein, so müsste zum Rendern für jedes Voxel ein Zeichenaufruf an die Grafikkarte gesendet werden. Selbst wenn die Voxel instanziert dargestellt werden, um die Anzahl der Zeichenaufrufe zu verringern, müssten die Instanzen bei jeder Änderung oder Erweiterung der Welt neu erzeugt werden, nachdem berechnet wird, welche Voxel sichtbar und welche verdeckt sind. Diese Prüfung würde jedes Mal längere Zeit in Anspruch nehmen. Werden stattdessen die Voxel in einzelne Chunks unterteilt, so können die Voxel der Chunks, in denen keine Änderungen vorkommen, jedes Frame neu gezeichnet werden, ohne erneut berechnet werden zu müssen.[2] Folgende Möglichkeiten sind dabei vorhanden, die Voxel innerhalb der Chunks darzustellen:

Vorgehen mit Instances

Die Voxel werden pro Chunk als Instanzen des selben Meshes gerendert. Durch das instanzierte Aufrufen ist nur ein Zeichenaufruf an die Grafikkarte für alle Instanzen zusammen notwendig.[12] Der Vorteil ist hierbei, dass im Shader noch Berechnungen pro Instanz durchgeführt werden können.

Vorgehen mit Mesh per Chunk

Hierbei wird ein Mesh für den gesamten Chunk angelegt, der aus den sichtbaren Faces aller Voxel besteht. Dieser wird erzeugt, sobald der Chunk an den Renderer übergeben wird und wird dort solange wiederverwendet, bis dieser Chunk verändert wird. Hierbei werden die UVs auf dem Mesh bei Erstellung des Meshes so festgelegt, dass die entsprechenden Materialien der Voxel an diesen Stellen dargestellt werden.[1]

3 Methode

In diesem Kapitel werden Konzepte zur Grundstruktur erläutert, die die Darstellung mehrerer Materialien pro Voxel ermöglicht. Es wird darauf eingegangen, welche Konzepte bei der praktischen Arbeit Verwendung finden und weshalb diese gewählt werden. Dazu werden dann mehrere Ansätze zur Umsetzung des Renderns auf Basis dieser Struktur beschrieben.

3.1 Konzepte zur Grundstruktur

Zur Darstellung mehrerer Materialien pro Voxel wurden verschiedene Ansätze ausgearbeitet, die aufzeigen, wie die verschiedenen Materialien innerhalb eines Voxels angelegt werden und wie diese Daten an den Renderer übergeben werden.

3.1.1 Aufbau eines Voxels

Ein Voxel wird, wie in den in Abschnitt 2.2.1 beschriebenen Ansätzen, mit nur einem Material pro Voxel so angelegt, dass seine Position anhand seiner Indizes innerhalb eines dreidimensionalen Arrays festgelegt wird. Innerhalb eines Voxels werden somit nur noch die Materialinformationen gespeichert. Die Materialien werden hierbei in Form von IDs als Integer in einer Liste gespeichert. Folgende Ansätze werden betrachtet um die Mengen der Materialien abzubilden:

Liste von Materialien mit je einem Mengenwert als Float

Die Menge der Materialien wird hier jeweils als Wert zwischen 0 und 1 dargestellt. 1 entspricht hier einem vollen Voxel und die Gesamtmenge der Materialien pro Voxel darf diesen Wert nicht überschreiten. Durch das Verwenden von Float-Werten für die Menge ist eine große Anzahl verschiedener Materialien pro Voxel möglich. Durch Ungenauigkeiten bei Float-Berechnungen ist jedoch immer eine gewisse Fehleranfälligkeit vorhanden.

Liste von Materialien mit je einem Mengenwert als Integer

Die Menge eines Materials wird auf einen Wert zwischen 0 und einem definierten Maximalwert festgelegt. Dieser Maximalwert legt auch die maximale Gesamtmenge an Materialien in einem Voxel fest. Die Mengenangabe kann sowohl abhängig als auch unabhängig von der Verteilung der Materialien innerhalb des Voxels erfolgen. Hierbei ist es möglich festzulegen, ob pro Material nur ein oder auch mehrere Einträge in der Liste erfolgen dürfen.

Liste mit fixer Länge, in welcher pro Stelle ein Material eingetragen wird

Hierbei ist die Menge der Materialien auf die Länge der Liste begrenzt. Diese Methode ist immer verbunden mit der Verteilung der Materialien innerhalb des Voxels. Jedoch ist hier die Anzahl der Einträge konstant hoch, auch bei niedriger Anzahl verschiedener Materialien. Im Fall eines Voxels, das mit nur einem Material komplett gefüllt ist, müsste bei einer Liste mit fixer Länge n die ID dieses Materials n -Mal eingetragen werden.

3.1.2 Verbindung der Voxel zum Rendern

Da beim Rendern mit mehreren Materialien pro Voxel, die Verteilung der Materialien teilweise nicht innerhalb des Voxels selbst festgelegt werden, sondern auch Informationen der benachbarten Voxel, zur Bestimmung der Verteilung dienen sollen, wird eine Verbindung zwischen den Voxeln angelegt. Dazu werden folgende Möglichkeiten betrachtet:

Information über Nachbarn in Voxel speichern

Wenn Referenzen zu den Nachbarn direkt in jedem Voxel anlegt werden, ist ein schneller Zugriff auf diese Nachbarn möglich. Hierbei erhöht sich jedoch die Größe eines Voxels um zusätzliche Informationen, die auch aus der umgebenden Struktur bezogen werden können.

Voxel in Welt speichern und Nachbarn durch Indizes beziehen

Durch die Indizes innerhalb des Arrays kann nicht nur die Position bestimmt, sondern auch die Nachbarn gefunden werden. Jedoch lässt sich, wie in Abschnitt 2.3 beschrieben, eine auf nur einem Array basierte Welt schwer dynamisch erweitern.

Voxel in Chunks speichern und Nachbarn durch Indizes beziehen

Wie auch in einer Welt werden die Nachbarn eines Voxels in einem Chunk anhand der Indizes im Array bestimmt. Dadurch bleiben die in Abschnitt 2.3 genannten Vorteile erhalten und die Welt ist dynamisch erweiterbar. Der Zugriff auf Nachbarn wird allerdings an den Grenzen der Chunks problematisch. Folgende Ansätze sind zur Behebung dieser Problematik verfügbar:

Finden des Nachbar-Voxels über Nachbar-Chunk innerhalb der Welt

Hierbei enthält jeder Chunk eine Referenz auf die Welt, durch welche er auf den Nachbar-Chunk zugreift. Hierzu müssen die Nachbar-Chunks des zu rendernden Chunks bereits geladen sein. Es ist auch möglich die Chunks aus der Welt zu entfernen, sobald alle Nachbarn und der Chunk selbst im Renderer instanziert wurden, um den verbrauchten Arbeitsspeicher zu minimieren.

Erweitern jedes Chunks um einen Rand

Hier werden die Nachbarn der Grenz-Voxel zusätzlich im Chunk gespeichert. Im Falle einer Welt, die vom Spieler verändert und gespeichert werden kann, müssen hier bei der Erstellung eines Chunks seine Nachbar-Chunks immer geladen werden.

3.2 Verwendete Struktur in praktischer Umsetzung

In der praktischen Anwendung sollte die Verteilung der Materialien innerhalb des Voxels nicht immer durch den Aufbau des Voxels bestimmt werden, sondern ist, wie auch in Abschnitt 3.1.2 beschrieben, mit den Informationen der Nachbarn erreichbar. Durch dieses Kriterium wird, von den in Abschnitt 3.1.1 beschriebenen Möglichkeiten, der Ansatz mit einer Liste fixer Länge ausgeschlossen. Da es einfacher ist mit Integer-Werten zu rechnen und diese dabei weniger fehleranfällig sind als Floats, wird eine Liste verwendet, in die zu jedem Material eine Menge zwischen 0 und einem Maximalwert als Integer abgelegt wird. Der Maximalwert ist hierbei auf 64 festgelegt, da diese Menge zur Veranschaulichung ausreicht. Auch ließe sich ein Würfel problemlos in 64 gleiche Würfel unterteilen, was bei einer in Abschnitt 2.2 beschriebenen kubischen Darstellung vorteilhaft ist.

Um die dynamische Erweiterbarkeit der Welt zu ermöglichen, wurde der in Abschnitt 3.1.2 beschriebene Ansatz mit der Verwendung von Chunks weiter verfolgt. Dabei wurde die in Abschnitt 2.3 beschriebene Vorgehensweise mit Instanced Rendern der Voxel gewählt, da manche der in Abschnitt 3.3 beschriebenen Methoden nicht sinnvoll in einer Struktur mit Mesh per Chunk Rendern, wie es ebenfalls in Abschnitt 2.3 beschrieben ist, umsetzbar sind. Hierbei wurde das Problem mit den Nachbarn an der Chunk-Grenze durch die in Abschnitt 3.1.2 beschriebene Lösung des Speicherns dieser Nachbar-Voxel gelöst. Dafür wurden die Voxel an der Grenze eines Chunks so modifiziert, dass diese auf ihre Nachbarn, welche außerhalb des Chunks liegen, eine Referenz enthalten. Später wurde das Programm so umgebaut, dass der Zugriff auf die Voxel der Nachbarchunks, wie ebenfalls in Abschnitt 3.1.2 beschrieben, mit einer übergeordneten Welt umgesetzt wird. Dabei wird darauf geachtet, dass ein Chunk erst an den Renderer übergeben wird, wenn alle zum Rendern benötigten Nachbarchunks in der Welt bereits geladen sind. Wenn ein Chunk und all seine Nachbarn bereits im Renderer instanziert wurden, wird dieser Chunk wieder aus dem Speicher der Welt entfernt, da er dann nicht mehr zum Rendern benötigt wird.

3.3 Ansätze zum Rendern

Es gibt mehrere Möglichkeiten wie beim Rendern mit den Materialinformationen innerhalb eines Voxels umgegangen werden kann, um diese dann entsprechend darzustellen. Von den in Abschnitt 2.2 beschriebenen Technologien liegt der Schwerpunkt dieser Arbeit beim Rendern als Meshes. Da die Darstellung von Materialien in Voxeln mit einem Raytracer hauptsächlich über eine 3D-Textur umgesetzt wird, wäre die Problematik hier die Speicheroptimierung bei dynamischer Änderung einer 3D-Textur oder einer Anzahl von 3D-Texturen. Dies wird hier nicht weiter bearbeitet.

3.3.1 Eine Textur pro Voxel

Ein Ansatz ist, von den Materialien innerhalb des Voxels eines zu wählen und dieses auf den gesamten Mesh zu projizieren. Dieser Ansatz ist für die verschiedenen Arten der Meshes verwendbar und kann sowohl für einen fixen Mesh, wie einen Würfel, als auch für Marching Cubes oder Blobby Objects verwendet werden, solange klar definiert ist, welches Polygon welchem Voxel zugeordnet ist. Dabei ist es sowohl möglich, vor der Übergabe zum Shader die UVs auf dem Mesh anzupassen, als auch allgemeine UVs zwischen 0 und 1 zu definieren, die dann innerhalb des Shaders anhand einer übergebenen MaterialID auf die Textur angepasst werden. Die folgenden Kriterien sind hier zur Auswahl des zu rendernden Materials verwendbar:

Abhängigkeit von der Menge der Materialien im Voxel

Das darzustellende Material wird anhand seiner Menge innerhalb des Voxels ausgewählt. Hierbei stehen zwei Ansätze zur Auswahl:

Häufigstes Material wird dargestellt

Hierdurch wird beispielsweise eine Landschaft homogen dargestellt und es ist nicht direkt erkennbar, welche Materialien sich noch in der Umgebung befinden, bevor nicht Materialien entfernt werden.

Seltenstes Material wird dargestellt

Wenn das Material mit der geringsten Menge angezeigt wird, ist in einer Landschaft direkt erkennbar, welche Materialien sich in der Umgebung befinden.

Abhängigkeit von den Materialien in den umgebenden Voxeln

Das Verhältnis der Materialien der Voxel, die das zu rendernde Voxel umgeben, zu den Materialien des zu rendernden Voxels wird betrachtet, um zu bestimmen, welches Material dargestellt werden soll. Dabei sollen die Materialien des zu rendernden Voxels angezeigt werden, die sich von den Materialien der Nachbarvoxel, die bei Betrachtung des zu rendernden Voxels nicht sichtbar sind, unterscheiden.

Dazu werden zunächst die Mengen der Materialien innerhalb der Nachbarvoxel und des zu rendernden Voxels in Verhältnisse umgerechnet. Zwischen den Nachbarvoxeln, welche neben einer sichtbaren Seite des zu rendernden Voxels liegen und dem zu rendernden Voxel selbst, wird nun der Durchschnitt dieser Verhältnisse gebildet. Auch wird der Durchschnitt zwischen den Verhältnissen der restlichen Nachbarvoxel gebildet. Die Werte, des ersten so erhaltenen Durchschnittsverhältnisses werden mit den Werten des zweiten Durchschnittsverhältnisses dividiert. Sollte hierbei durch 0 dividiert werden, wird stattdessen durch einen sehr kleinen Wert wie 0.00001 dividiert. Das Material, mit dem höchsten Wert des so erhaltenen Verhältnisses, wird nun als darzustellendes Material bestimmt.

Beispiel: Im folgenden zweidimensionalen Beispiel ist die Maximalanzahl der Materialien eines Voxels auf 8 gesetzt. Unter Anderem ist das zu rendernde Voxel (grau umrandet) nicht vollständig gefüllt. Bei der Umrechnung in Verhältnisse wird der nicht gefüllte Anteil nicht beachtet.

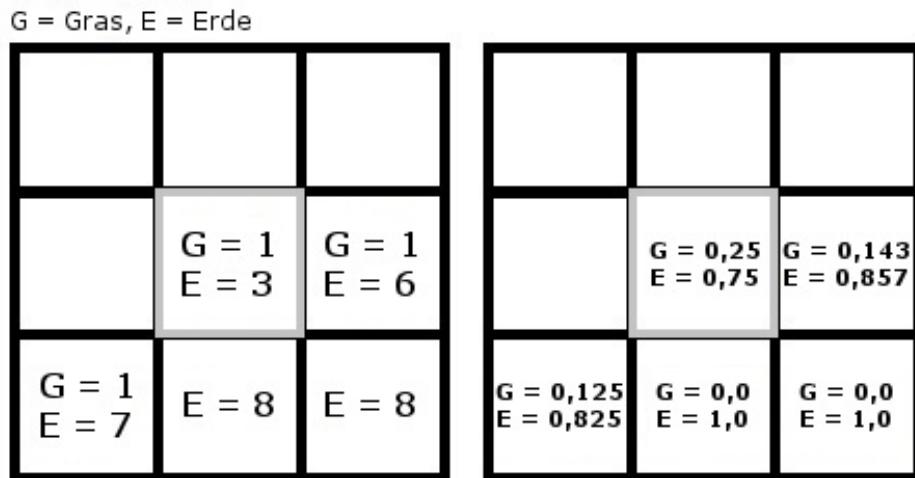


Abbildung 15: Umrechnung zur Materialbestimmung pro Voxel anhand der Umgebungsvoxel

Die sichtbaren Flächen sind hier oben und links. Die Voxel neben den sichtbaren Flächen sind hier demnach das Voxel links unten und das Voxel rechts. Die Berechnung des Durchschnitts dieser drei Voxel ergibt folgendes:

$$\frac{\begin{pmatrix} 0.125 \\ 0.875 \end{pmatrix} + \begin{pmatrix} 0.25 \\ 0.75 \end{pmatrix} + \begin{pmatrix} 0.143 \\ 0.857 \end{pmatrix}}{3} = \begin{pmatrix} 0.173 \\ 0.827 \end{pmatrix} \quad (1)$$

Die nicht sichtbaren Nachbarvoxel im Beispiel sind die beiden übrigen Voxel, die mit Erde gefüllt sind. Der Durchschnitt hierbei bleibt 1 für Erde und 0 für Gras:

$$\frac{\begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix} + \begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix}}{2} = \begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix} \quad (2)$$

Wird nun der Durchschnitt der sichtbaren Voxel durch den Durchschnitt der restlichen Voxel dividiert, so ergibt sich folgendes. Dabei wird 0 durch 0.0001 ersetzt:

$$\begin{pmatrix} 0.173 \\ 0.827 \end{pmatrix} / \begin{pmatrix} 0.00001 \\ 1.0 \end{pmatrix} = \begin{pmatrix} 17300 \\ 0,827 \end{pmatrix} \quad (3)$$

Da 17300 als Wert von Gras im Ergebnis größer ist als der Wert für Erde von 0,827 wird in diesem Beispiel der zu rendernde Voxel als Gras gerendert.

Hierdurch wird in einer Landschaft die Oberfläche homogen gerendert, wobei nicht direkt die Menge des Materials innerhalb des zu rendernden Voxels, sondern die Verteilung der Materialien innerhalb des Voxels und seiner Nachbarn bestimmen, wie das Voxel gerendert wird. So ist es denkbar, dass das Voxel im Beispiel auch noch eine Einheit eines dritten Materials zusätzlich besitzt. Dennoch würde weiterhin das Gras dargestellt werden, welches auch in seinen sichtbaren Nachbarn vorkommt.

Als weitere Abwandlung steht die im Abschnitt 3.3.2 beschriebene Bestimmungsvariante zur Festlegung der Materialien pro Seite zur Verfügung, um den Durchschnitt zwischen den Ergebnissen der einzelnen Seiten zur Bestimmung des Materials des Voxels zu verwenden. Hierbei wird jedoch der Rechenaufwand pro Voxel erhöht.

Vorgabe durch Reihenfolge der Materialien im Voxel

Wie in Abschnitt 3.1.1 beschrieben ist es möglich die Materialien in einer beliebigen Reihenfolge innerhalb des Voxels zu speichern. Diese Reihenfolge wird genutzt, um das darzustellende Material zu bestimmen. So wird immer das Material dargestellt, welches oben in der Liste ist.

Dadurch ist weder eine Homogenität der Oberfläche einer Landschaft, noch die direkte Sichtbarkeit von seltenen Materialien gegeben. Die Engine, die im Hintergrund läuft und die Landschaft zur Verfügung stellt, hat hier viel mehr Einfluss auf die Darstellung der Umgebung, wobei hier beispielsweise auch bessere Algorithmen bei einer generischen Erstellung der Landschaft von Nöten wären, um einem Spieler diese glaubhaft darzustellen.

Das Entfernen von Material ist mit allen Kriterien so umsetzbar, dass immer das sichtbare Material entfernt wird, als wären sie in Schichten angeordnet. Eine Alternative ist es immer Material per Zufallsprinzip zu entfernen, um die Materialien wie ein Gemisch abzubauen.

3.3.2 Eine Textur pro Richtung

Dieses Vorgehen ist angelehnt an das in Abschnitt 2.2.1 beschriebene Standardverfahren bei einer kubischen Voxeldarstellung mit verschiedenen Texturen auf unterschiedlichen Seiten eines Würfels. Das Voxel wird in sechs Richtungen, die den sechs Seiten eines Würfels entsprechen, aufgeteilt. Nun wird pro Richtung ein Material des Voxels bestimmt, das in dieser Richtung dargestellt wird. Diese Methode ist für Marching Cubes oder Blobby Objects in soweit verwendbar, dass die Normalen verwendet werden, um die Richtung zu bestimmen, in die eine Fläche am ehesten zeigt. Dabei ist es hier auch möglich die Bestimmung der Textur anhand einer Übergabe von sechs MaterialIDs im Shader umzusetzen, jedoch bleibt die Übergabe von sechs mal der selben ID, bei einem Voxel mit nur einem Material erspart, wenn die UVs außerhalb des Shaders angepasst wird.

Die Bestimmung der Materialien für jede Seite erfolgt anhand der Verhältnisse der Materialien im Voxel, zu den Verhältnissen der Materialien in den Nachbarvoxeln. Es wird dazu eine abgewandelte Form des in Abschnitt 3.3.1 beschriebenen Bestimmungsalgorithmus mit Abhängigkeit von den Nachbarvoxeln verwendet. Für jede Seite wird separat der Durchschnitt zwischen der Materialverteilung direkt benachbarter Voxeln und dem zu rendernden Voxel bestimmt, um dann die Division mit der Materialverteilung der restlichen Nachbarvoxel durchzuführen. Dann wird von diesem Ergebnis das Material mit dem höchsten Wert für diese Seite bestimmt.

Beispiel: Die Voxel in folgendem Beispiel besitzen eine maximale Anzahl von 8 Materialien. Der nicht gefüllte Anteil wird bei der Umrechnung in Verhältnisse nicht beachtet.

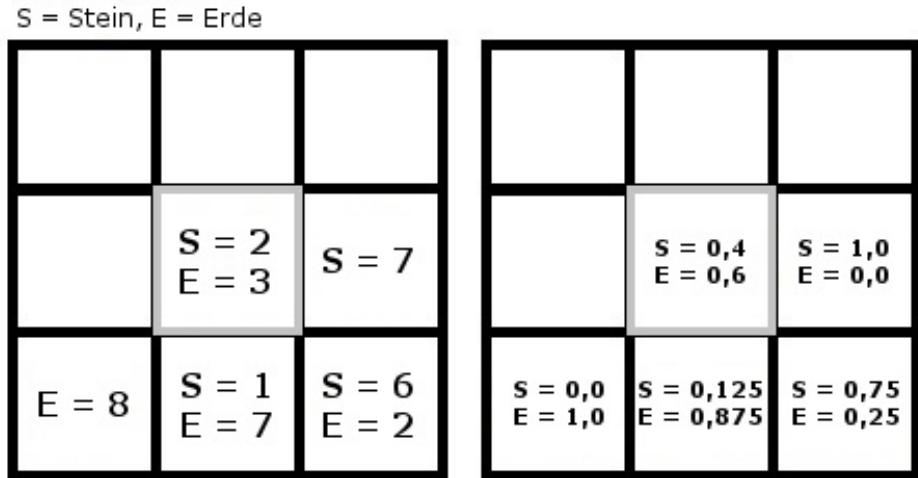


Abbildung 16: Umrechnung zur Materialbestimmung pro Richtung anhand der Umgebungsvoxel

Zuerst wird das Material für die linke sichtbare Fläche bestimmt. Dazu wird der Durchschnitt der Materialien des zu rendernden Voxels und den direkten Nachbarvoxeln der sichtbaren Fläche (hier links unten) berechnet:

$$\frac{\begin{pmatrix} 0.4 \\ 0.6 \end{pmatrix} + \begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix}}{2} = \begin{pmatrix} 0.2 \\ 0.8 \end{pmatrix} \quad (4)$$

Daraufhin wird der Durchschnitt der Materialien der restlichen Nachbarvoxel berechnet:

$$\frac{\begin{pmatrix} 1.0 \\ 0.0 \end{pmatrix} + \begin{pmatrix} 0.125 \\ 0.875 \end{pmatrix} + \begin{pmatrix} 0.75 \\ 0.25 \end{pmatrix}}{3} = \begin{pmatrix} 0.625 \\ 0.375 \end{pmatrix} \quad (5)$$

Vom ersten Durchschnitt wird durch den zweiten dividiert:

$$\begin{pmatrix} 0.2 \\ 0.8 \end{pmatrix} / \begin{pmatrix} 0.625 \\ 0.375 \end{pmatrix} = \begin{pmatrix} 0.32 \\ 2.13 \end{pmatrix} \quad (6)$$

Da im Ergebnis Erde mit 2.13 einen höheren Wert hat als Stein mit 0.32, wird für die linke Richtung Erde dargestellt.

Für die zweite sichtbare Seite oben werden nun die selben Rechenschritte durchgeführt. Dabei ist rechts der einzige direkte Nachbarvoxel der oberen Seite. Es werden als nächstes die Durchschnitte berechnen und diese dann voneinander abgezogen:

$$\frac{\begin{pmatrix} 0.4 \\ 0.6 \end{pmatrix} + \begin{pmatrix} 1.0 \\ 0.0 \end{pmatrix}}{2} = \begin{pmatrix} 0.7 \\ 0.3 \end{pmatrix} \quad (7)$$

$$\frac{\begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix} + \begin{pmatrix} 0.125 \\ 0.875 \end{pmatrix} + \begin{pmatrix} 0.75 \\ 0.25 \end{pmatrix}}{3} = \begin{pmatrix} 0.292 \\ 0.708 \end{pmatrix} \quad (8)$$

$$\begin{pmatrix} 0.7 \\ 0.3 \end{pmatrix} / \begin{pmatrix} 0.292 \\ 0.708 \end{pmatrix} = \begin{pmatrix} 2.397 \\ 0.424 \end{pmatrix} \quad (9)$$

Da hier der Wert von Stein mit 2.397 größer ist als der Wert von Erde mit 0.424 wird an der oberen Seite Stein dargestellt.

So werden in diesem Beispiel für den Voxel an der linken und der oberen Seite zwei verschiedene Materialien gerendert.

3.3.3 Materialien entlang eines Vektors

Innerhalb des Voxels wird ein Vektor bestimmt, entlang dessen alle Materialien des Voxels gerendert werden. Dieser Vektor wird im Folgenden als Materialvektor bezeichnet. Anhand des Vektors wird der Koordinatenbereich um den Voxel auf einen Bereich zwischen 0 und 1 projiziert. Die Materialien werden anhand ihrer Menge innerhalb des Voxels in der bestimmten Reihenfolge ebenfalls auf einen Bereich zwischen 0 und 1 abgebildet. So wird bestimmt, in welchen Bereichen des Voxels welche Texturen gerendert werden.

Die Abbildung des Koordinatenbereiches eines Voxels auf einen Bereich zwischen 0 und 1 wird über die Skalarprodukte zwischen dem normierten Vektor zur Materialverteilung und der aktuell zu rendernden Position innerhalb des Voxels bestimmt. Dabei wird vorher der Raum des Voxels auf einen Würfel abgebildet und der maximale Wert bestimmt, der bei diesem Skalarprodukt entstehen kann. Die Division des Skalarprodukts durch den errechneten Maximalwert ergibt dann einen Wert zwischen 0 und 1. Da die Ecken des Raumes, in dem das Voxel dargestellt wird, vom Wert her nicht direkt festlegbar sind, muss der Ursprung des Koordinatensystems ins Zentrum des Voxels verlagert werden. Der Voxelraum wird daher so dargestellt, dass er auf jeder Koordinatenachse einen Bereich zwischen -0.5 und 0.5 hat. Wird in diesem Zustand die Abbildung der aktuell zu rendernden Positionen nach zuvor beschriebenem Verfahren durchgeführt, so sind die Ergebnisse Werte zwischen -1 und 1. Dieser Bereich wird auf den gewünschten Bereich zwischen 0 und 1 abgebildet.

Veranschaulichung der Umwandlung: Die Berechnung des Maximalwertes \max bei Verwendung des normierten Materialvektors \vec{mat} wird durch das Skalarprodukt mit den Koordinaten der Ecke des Voxelraumes, die in Richtung des Vektors \vec{mat} liegt, berechnet. Dies bedeutet, dass bei einem rein positiven Vektor \vec{mat} auch die Ecke mit der rein positiven Koordinate \vec{edge} verwendet wird.

$$\max = \vec{mat} \circ \vec{edge} \quad (10)$$

Mit diesem Maximalwert wird über folgende Formel für jede Position \vec{pos} an der Oberfläche des Meshes ein Wert y zwischen 0 und 1 bestimmt. Dabei wird am Ende, zur Umwandlung des Bereiches von -1 bis 1 auf 0 bis 1, die Addition mit 1 und die Division durch 2 durchgeführt:

$$y = \left(\frac{\vec{mat} \circ \vec{pos}}{\max} + 1 \right) / 2 \quad (11)$$

In Abb. 17 ist dargestellt, wie Materialvektoren im Zentrum eines Voxels angesetzt werden. Dazu ist die Abbildung von Voxeln auf einen Bereich zwischen 0 und 1 dargestellt.

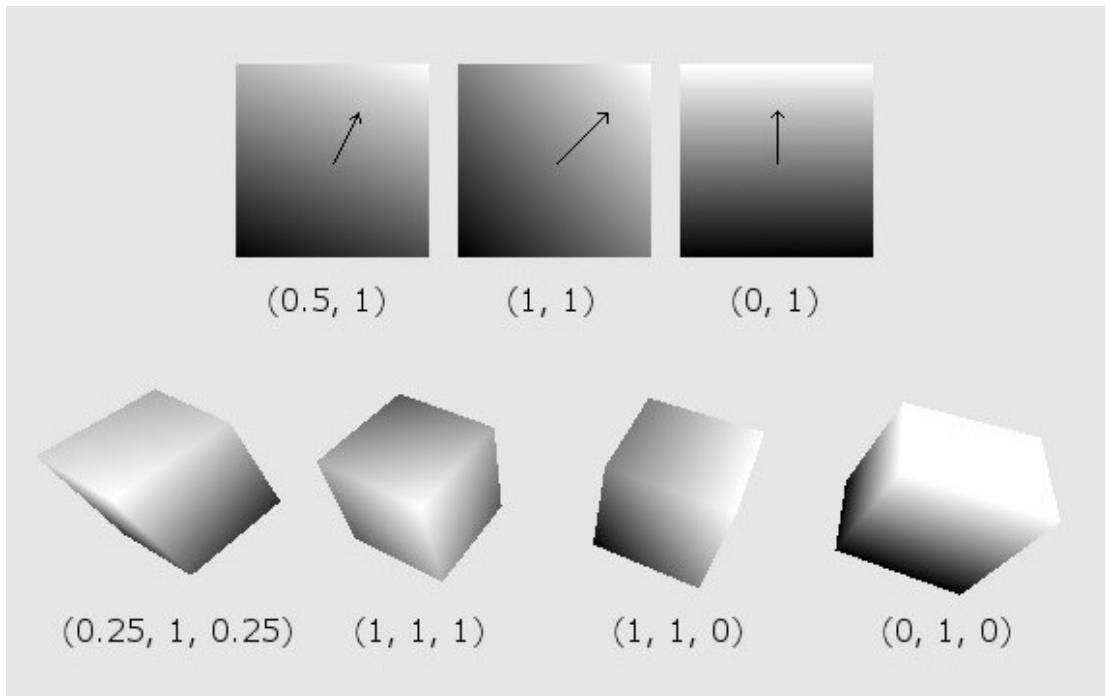


Abbildung 17: Projektion: Vektor in Voxel auf den Bereich 0 (schwarz) bis 1 (weiß)

Für die Bestimmung der Reihenfolge der Materialien und des Materialvektors werden folgende Ansätze betrachtet:

Reihenfolge der Materialien wie im Voxel gespeichert

Die in Abschnitt 3.1.1 beschriebene Möglichkeit, die Materialien in beliebiger Reihenfolge innerhalb des Voxels zu speichern, wird genutzt, um diese Reihenfolge direkt als Reihenfolge der darzustellenden Materialien zu verwenden. Der Materialvektor wird über eine der folgenden Methoden ermittelt:

Festgelegte Richtung

Es wird festgelegt, dass der Vektor innerhalb jedes Voxels in die selbe Richtung zeigt. So wird beispielsweise bestimmt, dass der Vektor immer von unten nach oben zeigt. Dadurch ist es möglich, das oberste Material in der Liste des Vektors auch immer als oberstes Material bei der Darstellung innerhalb der Welt darzustellen. Durch dieses Vorgehen wird, wie bereits im in Abschnitt 3.3.1 beschriebenen Ansatz, mit Verwendung der Reihenfolge der Materialien im Voxel, die Arbeit eine homogene Oberfläche zu erhalten auf die Instanz verlagert, die die Landschaft erzeugt.

Zu den nicht verdeckten Seiten

Hierzu werden die Nachbarn des zu rendernden Voxels betrachtet, um zu bestimmen, welche Seiten auf den Koordinatenachsen durch einen Voxel verdeckt sind. Die Vektoren der so verdeckten Richtungen werden addiert. Das Ergebnis wird mit -1 multipliziert um den Materialvektor zu erhalten.

Beispiel: Wird beispielsweise ein Voxel an der Position (0, 0, 0) betrachtet und dieses Voxel ist durch die Voxel an den Positionen (-1, 0, 0), (1, 0, 0), (0, -1, 0) und (0, 0, -1) durch andere Voxel verdeckt, so ist der Materialvektor:

$$\left(\begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \right) * -1 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \quad (12)$$

Abhängig von Materialverteilung in der Umgebung

Wie in der vorherigen Methode werden die Nachbarn betrachtet, um den Materialvektor zu bestimmen. Dabei werden hier die Füllmengen der umgebenden Voxel mit deren Richtungsvektor multipliziert, bevor diese dann aufaddiert werden. Der Materialvektor entsteht wieder durch negieren dieses Ergebnisses.

Beispiel: Wird beispielsweise ein Voxel an der Position (0, 0, 0) betrachtet und dieses Voxel ist durch die Voxel an den Positionen (-1, 0, 0), (1, 0, 0), (0, -1, 0) und (0, 0, -1) durch andere Voxel verdeckt, während das Voxel an der Position (-1, 0, 0) nur zur Hälfte und die restlichen Voxel voll gefüllt sind, so ist der Materialvektor:

$$(0.5 * \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} + 1 * \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + 1 * \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} + 1 * \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}) * -1 = \begin{pmatrix} -0.5 \\ 1 \\ 1 \end{pmatrix} \quad (13)$$

Bestimmung anhand der Materialverteilung in den Umgebungsvoxeln

Die Reihenfolge der Materialien und der Materialvektor werden beide in Abhängigkeit der Verteilung der Materialien in der Umgebung bestimmt. Dabei gibt es die folgenden Möglichkeiten:

Bestimmung durch Unterteilung in Achsenrichtungen

Hierzu wird der in Abschnitt 3.3.2 verwendete Algorithmus zur Bestimmung der Materialwerte pro Richtung auf den Koordinatenachsen angewandt. Aus dem Durchschnitt dieser Werte wird die Reihenfolge der Materialien anhand ihrer jeweiligen Wertung bestimmt.

Die Wertungen pro Richtung des in der Reihenfolge als Erstes bestimmten Materials werden mit den Vektoren der jeweiligen Richtung multipliziert. Der Materialvektor wird durch die Addition der sich daraus ergebenden Vektoren berechnet.

Beispiel: In diesem Beispiel werden die berechneten Werte von dem Beispiel auf Seite 25 weiter verwendet. Der Durchschitt dieser Werte entspricht:

$$\frac{\begin{pmatrix} 0.32 \\ 2.13 \end{pmatrix} + \begin{pmatrix} 2.397 \\ 0.424 \end{pmatrix}}{2} = \begin{pmatrix} 1.313 \\ 1.277 \end{pmatrix} \quad (14)$$

In diesem Fall ist die Reihenfolge der Materialien erst Stein mit 1.3135 und dann Erde mit 1.277.

Da das erste Material Stein ist, wird der Materialvektor nun über die Werte für Stein und die Vektoren der jeweiligen Richtungen folgendermaßen bestimmt:

$$0.32 * \begin{pmatrix} -1 \\ 0 \end{pmatrix} + 2.397 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -0.32 \\ 2.397 \end{pmatrix} \quad (15)$$

In diesem Beispiel wird somit Stein links oben und Erde rechts unten dargestellt. Das Verhältnis dabei entspricht dem Verhältnis der Mengen der Materialien im Voxel. In diesem Beispiel ist das Verhältnis von Stein zu Erde 2 zu 3.

Definition von Vektoren anhand der Materialverteilung

Für jedes Material innerhalb des zu rendernden Voxels wird ein Vektor bestimmt. Dieser Vektor ist der Durchschnitt der Vektoren, die entstehen, wenn für jeden Nachbar ein Vektor erstellt wird, indem der Vektor der Richtung des Nachbarn mit dem Verhältnis des jeweiligen Materials innerhalb dieses Nachbarn gewichtet wird. Zwischen den Vektoren für die verschiedenen Materialien wird nun jeweils der Winkel

bestimmt. Die zwei Materialien, die am weitesten auseinander liegen werden dadurch bestimmt. Von diesen beiden Materialien wird das mit der kleineren Menge im zu rendernden Voxel das Erste und das Andere wird das letzte Material in der Reihenfolge. Die restlichen Materialien werden dann entsprechend der Größe des Winkels mit dem Vektor des als Erstes bestimmten Materials in die Reihenfolge eingesortiert.

Der Materialvektor wird bestimmt, indem vom Vektor des ersten Materials der Vektor des letzten Materials abgezogen wird.

Beispiel: Die Maximalanzahl der Materialien eines Voxels wird in diesem Beispiel auf 8 gesetzt. Bei der Umrechnung in Verhältnisse wird auch in diesem Beispiel der nicht gefüllte Anteil nicht beachtet.

$S = \text{Stein}$, $E = \text{Erde}$, $G = \text{Gras}$

		$S = 3$			$S = 1,0$ $E = 0,0$ $G = 0,0$
	$S = 2$ $E = 2$ $G = 1$	$S = 8$	$S = 0,4$ $E = 0,4$ $G = 0,2$	$S = 1,0$ $E = 0,0$ $G = 0,0$	
$S = 2$ $E = 5$ $G = 1$	$S = 6$ $E = 2$	$S = 8$	$S = 0,25$ $E = 0,625$ $G = 0,125$	$S = 0,75$ $E = 0,25$ $G = 0,0$	$S = 1,0$ $E = 0,0$ $G = 0,0$

Abbildung 18: Umrechnung zur Materialbestimmung entlang eines Vektors

Zuerst erfolgt die Berechnung der Vektoren für die verschiedenen Materialien innerhalb des zu rendernden Voxels. Es ergeben sich folgende Vektoren für Gras,

$$0.125 * \begin{pmatrix} -1 \\ -1 \end{pmatrix} = \begin{pmatrix} -0.125 \\ -0.125 \end{pmatrix} \quad (16)$$

für Erde,

$$0.625 * \begin{pmatrix} -1 \\ -1 \end{pmatrix} + 0.25 * \begin{pmatrix} 0 \\ -1 \end{pmatrix} = \begin{pmatrix} -0.625 \\ -0.875 \end{pmatrix} \quad (17)$$

und für Stein:

$$0.25 * \begin{pmatrix} -1 \\ -1 \end{pmatrix} + 0.75 * \begin{pmatrix} 0 \\ -1 \end{pmatrix} + 1 * \begin{pmatrix} 1 \\ -1 \end{pmatrix} + 1 * \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 1 * \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2, 75 \\ -1 \end{pmatrix} \quad (18)$$

Zwischen diesen drei Vektoren werden nun die Winkel anhand folgender Formel bestimmt bestimmt:

$$\varphi = \cos^{-1}\left(\frac{\vec{u} \circ \vec{v}}{|\vec{u}| * |\vec{v}|}\right) \quad (19)$$

Dabei beträgt der Winkel zwischen Gras und Erde $9,46^\circ$, zwischen Gras und Stein $115,02^\circ$ und zwischen Erde und Stein $105,56^\circ$. Da der Winkel zwischen Gras und Stein der größte ist und Gras davon den geringsten Anteil innerhalb des zu rendernden Voxels hat, wird Gras das erste und Stein das letzte Material in der Reihenfolge. Dazwischen wird dann die Erde eingefügt.

Nachdem die Reihenfolge festgelegt wurde, wird der Vektor für Stein, da dies das letzte Material in der Reihenfolge ist, von dem Vektor für Gras, als erstes Material in der Reihenfolge, subtrahiert um den Materialvektor zu erhalten:

$$\begin{pmatrix} -0.125 \\ -0.125 \end{pmatrix} - \begin{pmatrix} 2, 75 \\ -1 \end{pmatrix} = \begin{pmatrix} -2, 875 \\ -1.125 \end{pmatrix} \quad (20)$$

Diese Ergebnisse zeigen, dass in diesem Beispiel die Materialien so angeordnet werden, dass Gras links unten und Stein rechts oben gerendert wird. Ob dazwischen noch die Erde gerendert wird ist optional. Diese kann auch weggelassen und nur das Gras und der Stein in den entsprechenden Verhältnissen gerendert werden. Alternativ dazu ist es möglich, wie im Beispiel in Abschnitt 3.3.4 beschrieben, zu bestimmen, dass die Erde nach der Berechnung als Gras dargestellt wird.

Wenn mehr als 2 Materialien in einem Voxel enthalten sind, wird entschieden, ob alle Materialien im gegebenen Verhältnis angezeigt werden, oder ob nur ein paar der Materialien angezeigt werden sollen. Dazu werden entweder Schwellen bestimmt, ab wann ein Material zusätzlich gerendert werden soll, oder Abhängigkeiten zwischen den Materialien verwendet, um ein homogeneres Ergebnis zu erhalten.

3.3.4 Abhängigkeiten zwischen Materialien

Für sämtliche zuvor beschriebenen Ansätze ist es möglich Abhängigkeiten zwischen den Materialien anzulegen, so dass, wenn zwei bestimmte Materialien zusammen vorkommen, nur eines davon oder sogar ein ganz anderes Material dargestellt wird. Dieses Prinzip ist invers zu der in Abschnitt 2.2.1 genannten Methode mehrere Materialien zu rendern, wo nur eines im Voxel gegeben ist. Dadurch bleibt der Vorteil vorhanden, zusätzlich zu den mit Abhängigkeiten versehenen Materialien, weitere Materialien innerhalb des Voxels zu haben, die auch beim Rendern beachtet werden.

Beispiel 1: Beispielsweise wird definiert, dass wenn sich Erde in einem Voxel befindet, in dem sich zusätzlich mindestens eine Einheit Gras befindet, für die entsprechende Menge Erde auch Gras gerendert wird, da Gras bekanntermaßen auf Erde wächst. Dabei ist es möglich zu berücksichtigen, welche Seiten des Voxels gerendert werden, so dass beispielsweise Gras nicht an der Unterseite eines Voxels dargestellt wird.

Beispiel 2: Ein anderes Beispiel für die Möglichkeit ein völlig anderes Material zu rendern wäre zu bestimmen, dass wenn Eisen und Stein sich zusammen in einem Voxel befinden stattdessen Eisenerz für die entsprechende Menge von beiden Materialien gerendert wird.

4 Ergebnisse

In diesem Abschnitt wird ein Vergleich der verschiedenen in Abschnitt 3.3 beschriebenen Methoden durchgeführt und zur Zusammenfassung am Ende ein Fazit abgegeben.

4.1 Vergleich

Bei diesem Vergleich wird spezifisch darauf eingegangen, wie die Methoden aus Abschnitt 3.3 mit denen aus den in Abschnitt 2.2.1 aufgezeigten Techniken verwendet werden können und wie gut diese mit Chunks, wie sie in Abschnitt 2.3 beschrieben sind, kompatibel sind. Des Weiteren wird anhand von im Rahmen der Projektarbeit entstandenen Bildern betrachtet, wie die Darstellung einer Landschaft mit den Methoden funktioniert.

4.1.1 Kompatibilität mit üblichen Techniken

Es ist zwar möglich, sämtliche Techniken als Basis für jede der Methoden zu verwenden, jedoch entstehen dabei verschiedene Problematiken, auf Basis derer gesagt werden kann, welche Technik für welche Methode besser und welche weniger gut geeignet ist.

Eine Textur pro Voxel (Abschnitt 3.3.1)

Diese Methode projiziert die Welt, die mehrere Materialien pro Voxel enthält, auf eine Welt, in der pro Voxel nur ein Material dargestellt wird. Dadurch ist sie sehr nah an den in Abschnitt 2.1 beschriebenen Methoden. Der Ansatz ein Material pro Voxel darzustellen ist, sowohl mit der Technik einen fixen Mesh darzustellen, als auch mit Marching Cubes und Blobby Objects gut umzusetzen. Diese Methode kann sowohl mit Instanced Rendern der Voxel, als auch mit einem Mesh pro Chunk verwendet werden.

Eine Textur pro Richtung (Abschnitt 3.3.2)

Wenn die Methode, pro Achsenrichtung ein Material des Voxels zu bestimmen, verwendet wird, dann ist dies am besten mit fixen Meshes in Form eines Würfels umzusetzen, da dieser lediglich 6 Seiten in die 6 Achsenrichtungen besitzt. Diese Methode kann auch für Marching Cubes und für Blobby Objects verwendet werden. Jedoch können Faces von deren Meshes in sämtliche Richtungen zeigen. Um diese Methode trotzdem darauf anzuwenden, wird die Achsenrichtung bestimmt, zwischen der der Winkel zur Normalen des Faces am kleinsten ist. Für die Anwendung dieser Methode empfiehlt es sich mit einem Mesh pro Chunk zu arbeiten, da die Faces pro Voxel hier sowieso

separat definiert werden und dadurch auch direkt die Faces wie in Abschnitt 2.3 zu einem Gesamtmesh für den Chunk hinzugefügt werden können. Werden die Voxel als einzelne Instanzen gerendert, müssen für jede Instanz die UVs separat definiert werden.

Materialien entlang eines Vektors (Abschnitt 3.3.3)

Die Methode, die Materialien entlang eines Vektors innerhalb des Voxels anzugeben, ist mit fixen Meshes, Marching Cubes und Blobby Objects gut umzusetzen. Dazu müssen die den Voxeln zugeordneten Vertices auf einen Bereich zwischen 0.5 und -0.5 auf den Achsen skalierbar sein. Da die Berechnungen für diese Methode teilweise pro Instanz im Shader durchgeführt werden, kann diese Methode nicht als Mesh pro Chunk sondern lediglich via Mesh per Instanz umgesetzt werden.

4.1.2 Verhalten bei Darstellung einer Landschaft

Für den Vergleich werden die verschiedenen Methoden zum Rendern der selben Landschaft innerhalb des praktischen Teils der Arbeit angewandt. Es wird verglichen, wie homogen und natürlich die Landschaft dargestellt wird. Hierbei werden auch die verschiedenen Algorithmen innerhalb der Methoden betrachtet, da diese eine starke Auswirkung auf die Darstellung der Landschaft haben.

Eine Textur pro Voxel (Abschnitt 3.3.1)

Wird pro Voxel lediglich ein Material gerendert, ist auf den ersten Blick kein Unterschied zu einer Voxeldarstellung mit nur einem Material pro Voxel erkennbar. Die Eigenschaft von mehreren Materialien pro Voxel ist hier erst erkennbar, wenn die Verhältnisse der Materialien innerhalb des Voxels so verändert werden, dass das jeweilige Auswahlkriterium ein anderes Material zur Darstellung präferiert. Dies passiert beispielsweise in einer Umgebung wie einem Spiel, wenn Material entfernt oder hinzugefügt wird.

Wird dabei immer das Material angezeigt, das am häufigsten innerhalb des Voxels vorkommt, so müsste bei der Verteilung der Materialien innerhalb der Welt dieses Kriterium beachtet werden, um eine natürliche Oberfläche zu erhalten. Dabei würden die Materialien innerhalb der Voxel jedoch in Mengen vorkommen, wie sie in natürlichen Verhältnissen nicht auftreten. Zum Beispiel befindet sich auf einer Menge Erde häufig nur eine dünne Schicht Gras. Um mit diesem Algorithmus Gras gerendert zu bekommen, muss allerdings mehr Gras als Erde in einem Block vorhanden sein. In Abbildung 19 ist zu sehen, wie mit diesem Algorithmus eine Welt mit natürlicheren

Mengenverhältnissen dargestellt wird. Hier wird Erde gerendert, auch wenn die Voxel Einheiten von Gras enthalten. Die Übergänge zwischen den Materialien werden hier korrekt dargestellt.

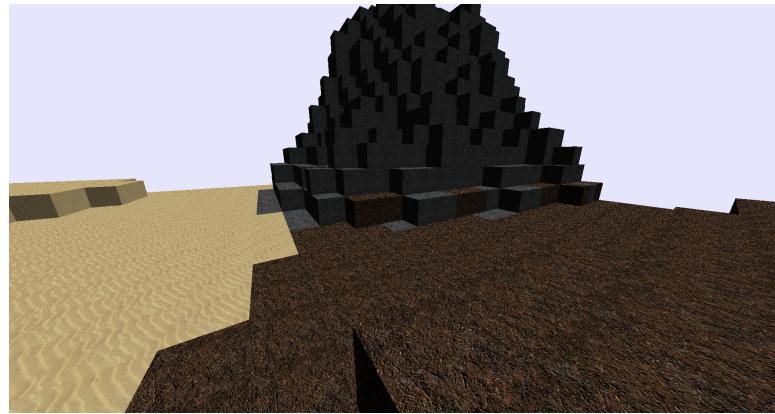


Abbildung 19: Ein Material pro Voxel: Häufigstes Material

Bei Darstellung des seltensten Materials des Voxels, muss dies ebenfalls bei der Verteilung der Materialien innerhalb der Welt beachtet werden. Es wird zwar Gras korrekt über der Erde gerendert, wenn mehr Erde im selben Voxel ist, jedoch werden Übergänge zwischen den Materialien falsch dargestellt. Wenn die Häufigkeit eines Materials abnimmt, während ein anderes Material hinzukommt, so wird das hinzukommende Material gerendert. Dadurch entsteht beim Übergang ein ungewollter mehrfacher Wechsel zwischen den Materialien. Wie in Abbildung 20 zu sehen ist, werden beispielsweise bei den Übergängen von Stein zu Sand, von Stein zu Erde und von Erde zu Sand diese Materialien an der Stelle des Übergangs abgewechselt.



Abbildung 20: Ein Material pro Voxel: Seltenstes Material

Die Betrachtung der Nachbarn, um anhand der Verhältnisse der Materialien innerhalb dieser festzulegen, welches Material dargestellt werden soll, sorgt dafür, dass die Übergänge korrekt dargestellt werden. Auch werden bei realistischen Materialverteilungen, wie beispielsweise, dass Gras an der Oberfläche vorhanden ist und die darunter liegenden Schichten kein Gras enthalten, solche seltener Materialien dann korrekt dargestellt. Die Abbildungen 21 und 22 zeigen die Ergebnisse des in Abschnitt 3.3.1 gezeigten Verfahrens und der genannten Abwandlung anhand der Berechnung pro Seite aus Abschnitt 3.3.2. Dabei sind nur kleine Unterschiede erkennbar. Es ist hierbei nicht zu sagen, welches der beiden Verfahren das bessere Ergebnis liefert, da dies eine Frage des Geschmacks ist.



Abbildung 21: Ein Material pro Voxel: Betrachtung der Nachbarn



Abbildung 22: Ein Material pro Voxel: Durchschnitt aus Betrachtung der Nachbarn pro sichtbare Seite

Bei der Festlegung des zu rendernden Materials anhand der Reihenfolge in der die Materialien im Voxel gespeichert sind, sind die Natürlichkeit und die Homogenität der Darstellung komplett davon abhängig, wie die Welt erstellt wurde. Jedes der in Abbildung 19 bis Abbildung 22 gezeigten Ergebnisse könnte dadurch mit den gleichen Mengen an Materialien in den Voxeln entstehen. Da in der praktischen Umsetzung die Voxel so implementiert sind, dass die Materialien ohne spezifische Reihenfolge innerhalb der Voxel gespeichert werden, werden hierfür keine Bilder gezeigt.

Eine Textur pro Richtung (Abschnitt 3.3.2)

Der Unterschied zwischen dieser Methode und der Methode lediglich ein Material pro Voxel zu rendern liegt darin, dass hier direkt erkennbar ist, dass ein Voxel mehr als ein Material enthält, wenn in verschiedenen Richtungen des Voxels verschiedene Materialien dargestellt werden. Dieses Verfahren ist von den in Abschnitt 2.2.1 beschriebenen Techniken abgeleitet, bei denen Voxel mit nur einem Material auf verschiedenen Seiten verschiedene Texturen erhalten, um diese besser in die Umgebung einzubringen. So verwendet beispielsweise das in Abschnitt 2.1.2 aufgeführte Spiel Minecraft, wie in Abbildung 23 zu sehen, diese Methode, um einen Gras-Block so zu rendern, als würde er von oben nach unten einen Übergang zu Erde durchführen. Da der Algorithmus der selbe ist, sehen die Ergebnisse ähnlich aus, wie das in Abbildung 22 mit einem Material pro Voxel gezeigte. Aufgrund der Notwendigkeit von größeren Umstellungen innerhalb des Projektcodes und der starken Ähnlichkeit mit dem vorherigen Ergebnis durch den selben Algorithmus ist diese Methode in dem praktischen Teil nicht implementiert.



Abbildung 23: Minecraft Gras-Block

Materialien entlang eines Vektors (Abschnitt 3.3.3)

Bei dieser Methode ist der Vorteil, dass alle Materialien innerhalb eines Blocks dargestellt werden. Die Materialien werden hierbei in Schichten angeordnet, was dafür sorgt, dass die Oberflächen bei korrekter Ausrichtung der Schichten für den Betrachter sehr homogen wirkt. Dabei werden Materialien, die in sehr kleiner Menge vorkommen und dadurch eine sehr dünne Schicht ergeben, in so kleinen Mengen dargestellt, wenn der Materialvektor diagonal liegt, dass sie kaum noch erkennbar sind.

Bei der Bestimmung der Materialreihenfolge anhand der Reihenfolge im Voxel ist das Ergebnis wieder sehr abhängig von der Erstellung der Welt und es können auch Ergebnisse, wie bei den folgenden gezeigten Methoden herauskommen. Da in der praktischen Umsetzung die Voxel so implementiert sind, dass die Materialien ohne spezifische Reihenfolge innerhalb der Voxel gespeichert werden, werden hierfür keine Bilder gezeigt.

Die Methode, die die Berechnungen aus Abschnitt 3.3.2 verwendet, um die Reihenfolge und die Richtung festzulegen, ergibt bei realistischen Mengenverhältnissen eine relativ homogene Darstellung. Dabei hat der Algorithmus wie beschrieben Probleme mit der Darstellung von kleinen Mengen bei diagonalem Materialvektor, wodurch an den Stufen in einer Grasfläche hauptsächlich Erde dargestellt wird und nur noch entlang der Kante ein kleiner Streifen Gras vorhanden ist, wie es in Abbildung 24 zu sehen ist. Die Übergänge zwischen Materialien werden gut dargestellt, da noch Abstufungen innerhalb der Voxel zu sehen sind, wie es in Abbildung 25 am Berg gezeigt wird.



Abbildung 24: Material entlang eines Vektors: Berechnung über sichtbare Seiten



Abbildung 25: Material entlang eines Vektors: Berechnung über sichtbare Seiten aus näherer Ansicht

Werden zuerst Vektoren für die Materialien bestimmt, um diese dann zum Rendern zu verwenden, so ist das Ergebnis ähnlich wie bei der Berechnung über die Seiten. Es ergeben sich jedoch andere Materialvektoren, wodurch bei flachen Materialübergängen teilweise kleine Ecken des anderen Materials in einer ansonsten homogenen Oberfläche auftauchen, wie es in Abbildung 26 zu sehen ist. Andererseits werden Übergänge bei Stufen, wie in Abbildung 27, sehr gut dargestellt, da hier noch Diagonalen entstehen, die die Übergänge fließender erscheinen lassen.



Abbildung 26: Material entlang eines Vektors: Berechnung über Vektoren

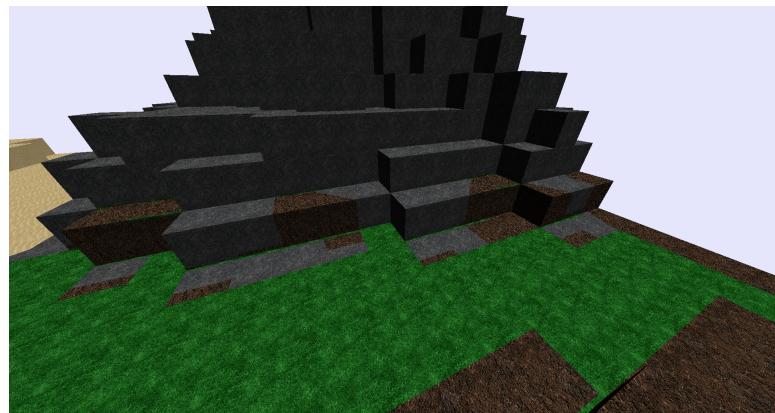


Abbildung 27: Material entlang eines Vektors: Berechnung über Vektoren aus näherer Ansicht

Abhängigkeiten zwischen Materialtypen (Abschnitt 3.3.4)

Das Hinzufügen von Materialabhängigkeiten verbessert die Homogenität der gerenderten Landschaft. So ist, wie in Abbildungen 28 und 29 gezeigt, die Verteilung des Grases weit besser als in den ursprünglichen Ergebnissen. Hierbei werden die beiden unterschiedlichen Algorithmen zur Darstellung der Materialien entlang eines Vektors genommen und definiert, dass Erde immer als Gras gerendert werden soll, wenn der Vektor sowohl Erde als auch Gras enthält.



Abbildung 28: Material entlang eines Vektors: Berechnung über sichtbare Seiten mit Abhängigkeit bei Gras



Abbildung 29: Material entlang eines Vektors: Berechnung über Vektoren mit Abhängigkeit bei Gras

4.2 Fazit

Es gibt keine Methode, die klar als beste genannt werden kann, da jede der Methoden Vor- und Nachteile mit sich bringt. Es ist letztendlich vom Projekt und dem gewünschten Effekt abhängig, welche Methode bevorzugt wird. So ist ein Ergebnis, wie in Abbildung 20, durch die Fehler bei den Übergängen im gezeigten Beispiel nicht optimal, jedoch kann auch ein Szenario entworfen werden, in dem beispielsweise die Welt überwiegend aus dem selben Material besteht und die seltenen Materialien immer hervorgehoben werden sollen. Auch ist eine Methode, die pro Voxel nur ein Material darstellt in einer veränderbaren Umgebung wie einem Spiel verwendbar, wenn ein Voxel mehrere Materialien enthalten kann. In einer Anwendung, die jedoch statische Bilder erzeugt, würde es keinen Sinn machen solch eine Methode zu verwenden, wenn die Voxel mit mehreren Materialien gefüllt werden, da dann kein Unterschied zu einer Welt mit einem Material pro Voxel sichtbar ist. Auch können sämtliche gezeigte Methoden noch weiter verbessert oder verändert werden, wodurch wieder komplett andere Darstellungen entstehen können. So ist, wie gezeigt schon durch so etwas wie die Definition von Abhängigkeiten zwischen Materialien eine starke Veränderung des Ergebnisses vorhanden.

5 Ausblick

In diesem Abschnitt werden weitere Punkte genannt, die auf Basis dieser Arbeit bearbeitet werden können.

Verwendung der Methoden mit anderen Rendertechniken

Im Verlauf der praktischen Umsetzung dieser Arbeit wurden die Methoden immer auf fixe Meshes in Form von Würfeln angewandt. Im weiteren wäre ein Schritt diese Methoden auch mit anderen der in Abschnitt 2.2 beschriebenen Technologien umzusetzen.

Darstellung der Menge des Materials innerhalb der Voxel

Es kann zusätzlich zur Bestimmung der Materialeigenschaften auch die Menge der Materialien zur Darstellung betrachtet werden. Dabei können die Voxel verformt werden, wie es bereits in Abschnitt 2.2.1 für medizinische Anwendungen bei Marching Cubes beschrieben ist. Dabei existiert bei fiktiven Landschaften kein eingescanntes Objekt, auf das die Form angepasst wird, sondern es müssten Regeln definiert werden, wie diese Verformung anhand der Materialmenge umgesetzt werden soll.

Interpolation der Texturen

Es können die Farben oder Texturen interpoliert werden um fließendere Übergänge zu erhalten. Hierzu können Verfahren ähnlich des in Abschnitt 2.2.1 gezeigten Verfahrens zur Interpolation von Texturen bei Marching Cubes verwendet werden.

Interpolation der Texturen

Neben der Betrachtung von Vorgehen zur Darstellung von Voxeln mit mehreren Materialien können auch noch Möglichkeiten zur Anwendung dieser Voxel gestaltet werden. So wäre es beispielsweise denkbar ein Spiel umzusetzen, bei dem die Materialien aus diesen Voxeln entnommen, verarbeitet und wieder platziert werden können.

Abbildungsverzeichnis

1	Rekonstruktion des CT Scans eines Knochens[14]	8
2	Screenshot aus Comance: Maximum Overkill (Volles Video: https://www.youtube.com/watch?v=51E_G7NCXVM)	9
3	Screenshot aus Voxelstein 3D (Volles Video: https://www.youtube.com/watch?v=6ZUdsRdok04)	10
4	Screenshot aus Minecraft	11
5	Screenshot aus Minecraft (Volles Video: https://www.youtube.com/watch?v=QmBt0MrLn3U)	11
6	Verschiedene Meshes in Minecraft	12
7	Möglichkeiten für MarchingCubes[9]	13
8	Marching Cubes Verschiebung vorher[4]	13
9	Marching Cubes Verschiebung nacher[4]	13
10	Interpolierte Marching Cubes[7]	14
11	Polygon mit zusätzlichen Werten zur Interpolation[7]	14
12	Blobby Objects[6]	15
13	Raytracing[3]	16
14	Texturierter Voxel-Hase[8]	16
15	Beispiel Farbe pro Voxel	22
16	Beispiel Farbe pro Richtung	25
17	Beispiel Vektorprojektion in Voxeln	28
18	Beispiel Farbe entlang Vektor	31
19	Projektscreenshot Farbe pro Voxel 1	36
20	Projektscreenshot Farbe pro Voxel 2	36
21	Projektscreenshot Farbe pro Voxel 3	37
22	Projektscreenshot Farbe pro Voxel4	37
23	Minecraft Gras-Block	38
24	Material entlang eines Vektors: Berechnung über sichtbare Seiten	39
25	Material entlang eines Vektors: Berechnung über sichtbare Seiten aus näherer Ansicht	40
26	Material entlang eines Vektors: Berechnung über Vektoren	40
27	Material entlang eines Vektors: Berechnung über Vektoren aus näherer Ansicht	41
28	Material entlang eines Vektors: Berechnung über Berechnung über sichtbare Seiten mit Abhängigkeit bei Gras	41
29	Material entlang eines Vektors: Berechnung über Vektoren mit Abhängigkeit bei Gras	42

Literatur

- [1] ALEXSTV: *Unity voxel block tutorial pt. 1.* <http://alexstv.com/index.php/posts/unity-voxel-block-tutorial>.
- [2] ALWAYSGEEKY: *Let's Make a Voxel Engine.* <https://sites.google.com/site/letsmakeavoxelengine/home/chunks>.
- [3] AMANATIDES, JOHN und ANDREW WOO: *A Fast Voxel Traversal Algorithm for Ray Tracing*, 2011. <http://www.cse.chalmers.se/edu/year/2011/course/TDA361/grid.pdf>.
- [4] ANDERSON, BEN: *An Implementation of the Marching Cubes.* http://www.cs.carleton.edu/cs_comps/0405/shape/marching_cubes.html.
- [5] GEBHARDT, SCOTT, ELIEZER PAYZER, LEO SALEMANN, ALAN FETTINGER, EDUARD ROTENBERG und CHRISTOPHER SEHER: *Polygons, Point-Clouds, and Voxels, a Comparison of High-Fidelity Terrain Representations.* Paper on Simulation Interoperability Workshops, Herbst 2009. https://www.sisostds.org/DesktopModules/Bring2mind/DMX/Download.aspx?Command=Core_Download&EntryId=28745&PortalId=0&TabId=105.
- [6] MATTHEW WARD, WPI CS DEPARTMENT: *An Overview of Metaballs/Blobby Objects.* <https://web.cs.wpi.edu/~matt/courses/cs563/talks/metaballs.html>.
- [7] MECHAM, JOHN: *Introduction to Multimaterial Terrain*, Mai 2014. <http://voxiworld.blogspot.de/>.
- [8] NOOBODY: *GPU voxel raytracing.* <http://www.blitzbasic.com/Community/posts.php?topic=89584>.
- [9] POLLEFEYS, MARC: *Marching Cubes*, 2002. <http://www.cs.unc.edu/~marc/tutorial/node130.html>.
- [10] PROJEKT: *Cubic-engine.* <https://github.com/MMDaemon/Cubic-engine>.
- [11] SHIRLEY, PETER and STEVE MARSCHNER (editors): *Fundamentals of Computer Graphics*. A K Peters, Ltd, 3rd edition, 2009. ISBN:978-1-56881-469-8.
- [12] SHREINER, DAVE, GRAHAM SELLERS, JOHN KESSENICH, and BILL LICEA-KANE (editors): *OpenGL Programming Guide*. Addison-Wesley, 8th edition, 2013. ISBN-10:0-321-77303-9, ISBN-13: 978-0-321-77303-6.
- [13] WATT, ALAN (editor): *3d Computer Graphics*. Addison-Wesley, 3rd edition, 2000. ISBN:0-201-39855-9.

- [14] YOUNG, P.G, T.B.H BERESFORD-WEST, S.R.L COWARD, B NOTARBER-ARDINO, B WALKER, and A ABDUL-AZIZ: *An efficient approach to converting three-dimensional image data into highly accurate computational models.* Physiological transactions of a royal society A, 366, September 2008. <http://rsta.royalsocietypublishing.org/content/366/1878/3155>.