Institut für Parallele und Verteilte Systeme
Abteilung Simulation großer Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 332

Visualisierung von oktalbaumbasierten kartesischen Gittern

Felix Kohlgrüber

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Miriam Mehl
Betreuer/in:	Dipl.-Inf. Michael Lahnert
Beginn am:	16. Mai 2016
Beendet am:	30. September 2016
CR-Nummer:	G.1.8, I.3.8

Kurzfassung

Forests of Octrees sind eine Generalisierung von oktalbaumbasierten numerischen Gittern und können verwendet werden, um Adaptive Mesh Refinement (AMR) für numerische Anwendungen umzusetzen. Eine Umsetzung von parallelen AMR-Algorithmen, die Forests of Octrees verwenden, ist in der p4est Softwarebibliothek implementiert.

In dieser Bachelorarbeit werden verschiedene Möglichkeiten untersucht, einen Forest of Octrees sowie einige seiner Eigenschaften zu visualisieren. Im Zuge dieser Betrachtungen wurde ein Visualisierungsprogramm erstellt, das die p4est-Bibliothek einbindet und die Visualisierung sowie die interaktive Manipulation eines Forest of Octrees ermöglicht.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1 Einleitung

Ein Grundproblem numerischer Simulation ist die Wahl eines passenden Gitters, auf dem die numerischen Berechnungen ausgeführt werden sollen. Die Wahl hängt dabei stark von der Art des numerischen Problems ab und beeinflusst sowohl die Effizienz als auch Qualität des Ergebnisses der Berechnung. Je feiner die Auflösung das gewählten Gitters ist, desto geringer ist der Fehler durch die Diskretisierung. Andererseits erhöht sich mit der Auflösung des Gitters ebenfalls der Rechenaufwand zur Lösung des numerischen Problems. Die Wahl der Auflösung ist dementsprechend immer eine Abwägung zwischen dem Berechnungsaufwand und der Qualität des Ergebnisses.

Während sich einige numerische Probleme gut mit Gittern gleichmäßiger Auflösung beschreiben lassen, existieren ebenfalls Probleme, die innerhalb eines großen Definitionsbereichs stark lokale Phänomene enthalten (z.B. [HGS+03]). Diese Art von Problemen benötigt in bestimmten Bereichen eine deutlich feinere Auflösung als in anderen Bereichen. Wird ein solches Problem mit einem Gitter gleichmäßiger Auflösung simuliert, muss zwischen Qualität und Effizienz entschieden werden. Einerseits können lokale Phänomene nicht mehr genau simuliert werden, wenn ein grob aufgelöstes Gitter verwendet wird, andererseits verursacht ein fein aufgelöstes Gitter einen deutlichen Mehraufwand in der Berechnung, der in großen Teilen des Definitionsbereichs keinen signifikanten Mehrwert bringt. Zudem können lokale Phänomene auch erst im zeitlichen Verlauf einer Simulation auftreten.

Adaptive Mesh Refinement and Coarsening (AMR) [Ber82] ist eine Methode, die Gitter mit lokal unterschiedlichen Auflösungen ermöglicht, die sich zudem dynamisch im zeitlichen Verlauf der Simulation verändern können. Somit können lokale Phänomene fein aufgelöst werden, ohne die Auflösung des gesamten Gitters erhöhen zu müssen. Durch diese Eigenschaft können AMR-Methoden lokale Phänomene mit hoher Genauigkeit abbilden und dabei im Vergleich zu gleichmäßig aufgelösten Gittern ein Vielfaches an Gitterpunkten einsparen, was den Berechnungsaufwand deutlich reduziert.

Eine Methode, Gitter mit lokal unterschiedlichen Auflösungen umzusetzen, sind baumbasierte Gitter [FLS+97; GZ99; Pop03]. Diese Gitter können durch die rekursive Verfeinerung, die durch einen Quaternär- oder Oktalbaum vorgegeben wird, lokal unterschiedlich aufgelöst sein. Baumbasierte Gitter können allerdings nur würfelförmige Geometrien darstellen, was für einige Anwendungen nicht ausreicht.

Forests of Octrees [BWG11] verbinden mehrere baumbasierte Gitter miteinander und sind so in der Lage, eine größere Anzahl an Geometrien abzudecken. Die einzelnen Bäume des Forest of Octrees können dabei individuell verfeinert werden.

Die Bibliothek p4est [BWG11; BWI14] implementiert Algorithmen für paralleles AMR auf Forest of Octrees. P4est stellt dabei die Gitterstrukturen zur Verfügung, sodass die Bibliothek mit verschiedenen numerischen Lösungsverfahren verwendet werden kann. ALPS [BBG+09] und deal.II [BHH+16; BHK07] sind Beispiele für numerische Anwendungen, die p4est verwenden können. Eine wichtige Eigenschaft dieser Bibliothek ist ihre Skalierbarkeit auf stark parallelisierten Rechnersystemen und wurde bereits auf über 220.000 CPUs skaliert [BWG11].

Da Forests of Octrees komplexe Geometrien und Verfeinerungen darstellen können, ist eine Visualisierung der Gitterstruktur ein nützliches Hilfsmittel, um die Funktionsweise des Gitters nachzuvollziehen. P4est bietet die Möglichkeit, Forests of Octrees als vtk-Dateien zu exportieren. Diese Dateien können anschließend von einem externen Programm gelesen und visualisiert werden. Neben der Struktur des Forest of Octrees können ebenfalls Daten aus der numerischen Anwendung in die zu exportierende Datei geschrieben werden. So ist es möglich, die Werte eines Oktanten zu einem bestimmten Zeitpunkt in der Simulation zu visualisieren. Durch periodisches Erstellen von vtk-Dateien während der Simulation kann zusätzlich im Nachhinein der Verlauf der Simulation nachvollzogen werden.

Da die Visualisierung nur indirekt über einen Dateiexport möglich ist, können lediglich die im Dateiexport enthaltenen Daten visualisiert werden. Zusätzlich ist es nicht möglich, direkt aus der Visualisierung Änderungen an dem visualisierten Forest of Octrees vorzunehmen. Um einen Forest of Octrees zu visualisieren, muss zunächst ein Programm geschrieben werden, das einen Forest of Octrees erstellt und exportiert. Anschließend kann der Forest of Octrees dann wie bereits genannt visualisiert werden.

Das Ziel dieser Arbeit ist es, Möglichkeiten der direkten Visualisierung eines Forest of Octrees zu untersuchen. Hierfür wird ein Visualisierungsprogramm entwickelt, das die Bibliothek p4est einbindet und so einen direkten Zugriff auf die Datenstrukturen erhält, die einen Forest of Octrees definieren. Durch die direkte Einbindung von p4est können sämtliche Eigenschaften des Forests of Octrees dargestellt werden und zudem interaktiv Manipulationen an bestimmten Eigenschaften des Forests of Octrees vorgenommen werden.

Diese Arbeit ist folgendermaßen aufgebaut: Kapitel 2 enthält theoretische Betrachtungen zu Quad- bzw. Octrees und Forests of Octrees, die die Grundlagen der im Folgenden beschriebenen Sachverhalte darstellen. Auf dieses Kapitel folgt in Kapitel 3 eine detaillierte Beschreibung des Visualisierungsprogramms, die auf Konzepte, Funktionen und Eigenschaften der Implementierung eingeht. Kapitel 4 fasst die Ergebnisse der Arbeit zusammen und gibt einen Ausblick auf mögliche zukünftige Weiterentwicklungen.

2 Theoretische Betrachtungen

2.1 Quad- / Octrees

Quaternärbäume bzw. Oktalbäume sind hierarchische Datenstrukturen, die eine Spezialisierung von Bäumen darstellen und aufgrund ihrer Eigenschaften als Basis für rekursiv verfeinerte mehrdimensionale Gitterstrukturen verwendet werden können.

Bäume sind weit verbreitete Datenstrukturen, in denen eine Menge von Elementen, die als Knoten bezeichnet werden, hierarchisch angeordnet werden. Jeder Baum besitzt dabei genau einen Ursprungsknoten, der als Wurzel bezeichnet wird und die höchste Ebene der Hierarchie darstellt. Ein Knoten kann mit beliebig vielen untergeordneten Knoten verbunden sein, die als Nachfolger oder Kinder des Knotens bezeichnet werden. Analog dazu wird der Knoten der nächsthöheren Hierarchieebene als Vorgänger oder Elternknoten bezeichnet. Mit Ausnahme der Wurzel hat jeder Knoten genau einen Elternknoten, sodass eine eindeutige Hierarchie entsteht, die keine Zyklen enthält. Ein Knoten, der keine Kinder hat, wird als Blatt bezeichnet. Jedem Knoten wird ein Level zugeordnet, das der Distanz zwischen dem Knoten und der Wurzel entspricht. Somit hat die Wurzel Level 0, ihre Nachfolger Level 1, deren Nachfolger Level 2, usw.

Quaternär- bzw. Oktalbäume haben zusätzlich die Eigenschaft, dass jeder Knoten entweder ein Blatt ist oder exakt vier bzw. acht Kindknoten besitzt.

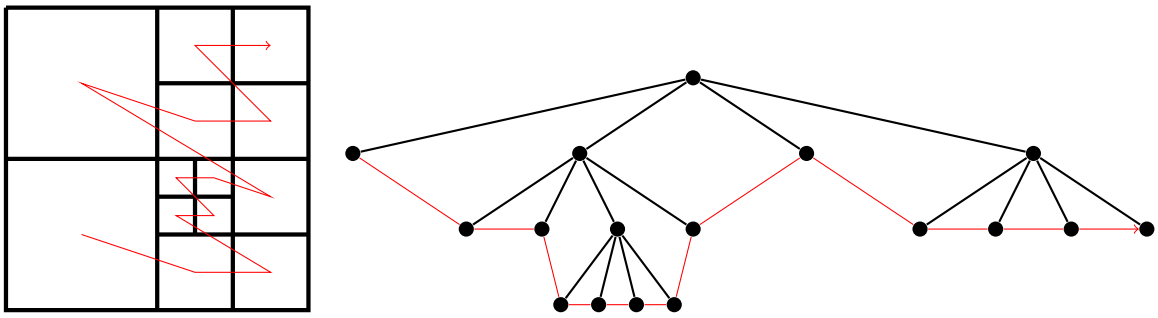


Abbildung 2.1: Quadtree-basiertes Gitter (a) und zugehöriger Quadtree (b)

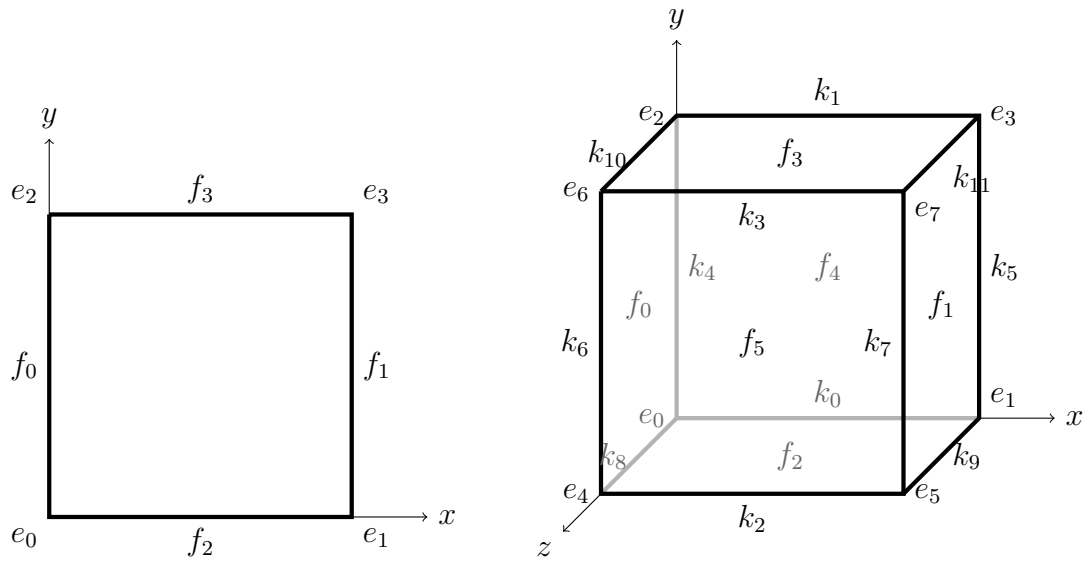


Abbildung 2.2: Benennung der Flächen f_i, Kanten k_i und Ecken e_i eines Quadtrees (links) und eines Octrees (rechts).

Durch diese Eigenschaft können sie verwendet werden, um einen zwei- bzw. dreidimensionalen Raum rekursiv in kleinere Bereiche zu unterteilen [FB74; Mea82]. Abbildung 2.1 zeigt ein rekursiv verfeinertes Gitter und den dazugehörigen Quaternärbaum.

Die einzelnen Zellen des Gitters werden als Quadranten bzw. Oktanten bezeichnet. Jeder Quadrant bzw. Oktant entspricht einem Blattknoten des Quad- bzw. Octrees. Innere Knoten des Baums stehen für Quadranten bzw. Oktanten, die verfeinert und somit durch vier bzw. acht Nachfolger ersetzt wurden.

Im folgenden Text werden zur Vereinfachung Gitter, die auf Quaternär- oder Oktalbäumen basieren, als Octree und Quadranten sowie Oktanten als Oktanten bezeichnet. An Stellen, an denen die Dimensionalität relevant ist, wird dies kenntlich gemacht.

2.1.1 Benennung

Ein Octree der Dimension $d \in \{2, 3\}$ hat $2 * d$ Flächen f_i, 2^d Ecken e_i sowie 12 Kanten k_i (Kanten existieren nur für $d = 3$). Wir übernehmen das Benennungsschema aus [BWG11], das in Abbildung 2.2 dargestellt ist. Die Indices der Flächen, Kanten und Ecken sind 0-basiert und benutzen die sogenannte z-order, was bedeutet, dass Elemente zunächst in x, dann in y, und dann in z-Richtung nummeriert werden.

Die Benennung der Flächen, Ecken und Kanten eines Oktanten innerhalb eines Octrees ist identisch zu der oben genannten Benennung für einen Octree.

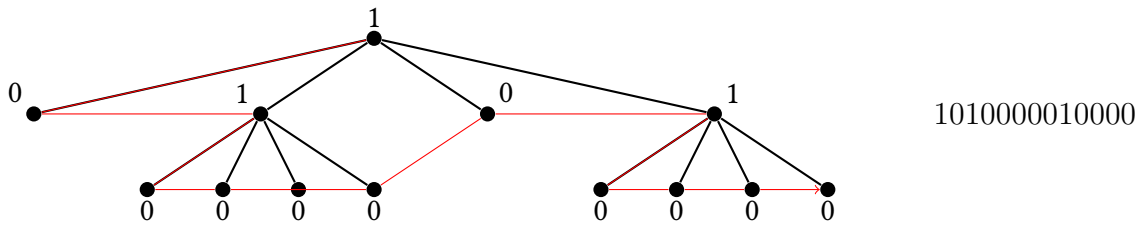


Abbildung 2.3: Quadtree (links) und Binärrepräsentation (rechts)

2.1.2 Binärrepräsentation

Ein Oktalbaum kann als Binärstring codiert bzw. dargestellt werden. Diese Repräsentation bietet sich an, um einen Oktalbaum zu speichern oder textbasiert darzustellen. Die Binärrepräsentation eines Quaternär- bzw. Oktalbaums kann erstellt werden, indem beginnend an der Wurzel des Baums eine Tiefensuche gestartet und für jeden besuchten Knoten ein Bit gespeichert wird. Für Blätter wird dabei das Bit 0 gespeichert, für innere Knoten das Bit 1. Abbildung 2.3 zeigt einen Octree sowie dessen Binärrepräsentation.

2.1.3 Space-Filling Curve

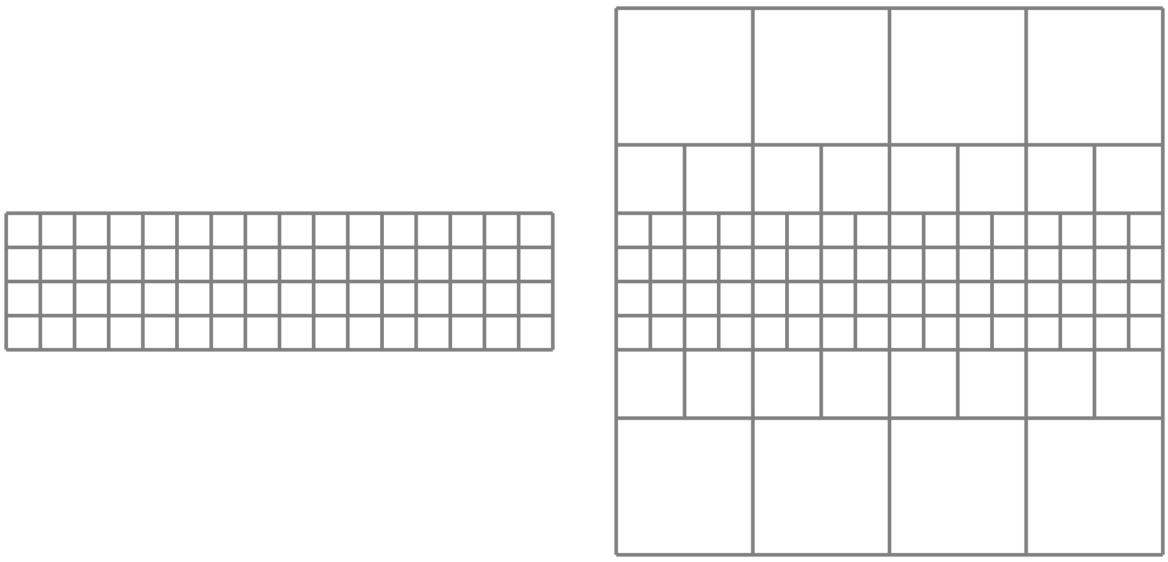
Obwohl die hierarchische Anordnung der Oktanten eines Octrees in vielen Bereichen Vorteile bringt, benötigen andere Features eine “flache“ Ordnung der Oktanten.

Eine solche Ordnung der Oktanten kann hergestellt werden, indem die Blattknoten des Octrees mittels Tiefensuche traversiert werden. Die dadurch entstehende Reihenfolge entspricht in der geometrischen Darstellung einer Space-Filling Z-Shaped curve. Abbildung 2.1 zeigt die Baumstruktur sowie die geometrische Darstellung der Space-Filling Curve eines Octrees. Im Folgenden wird diese Kurve der Einfachkeit halber als Z-Kurve bezeichnet.

Eine Anwendung der Z-Kurve ist die Partitionierung und Verteilung der Oktanten auf eine Menge von Prozessoren. Dabei wird die Z-Kurve in gleichmäßige Teilstücke aufgeteilt, die dann jeweils einem Prozess zugeordnet werden.

2.1.4 Darstellbare Geometrien

Ein Octree deckt einen quadrat- bzw. würfelförmigen Raum ab. Obwohl dies für bestimmte Anwendungen passend ist, sind für viele Problemstellungen allgemeinere Geometrien erforderlich. Bereits einfache Gitter, die in verschiedenen Dimensionen unterschiedliche Größen haben, sind mit einem Octree nicht mehr gut darstellbar. Abbildung 2.4 zeigt links ein Gitter, das in Dimension x verglichen mit Dimension y eine viermal so hohe Auflösung besitzt. Die rechte Grafik zeigt einen Octree, der dieses Gitter enthält. Zusätzlich zu den 64 Oktanten



(a) **(b)**

Abbildung 2.4: Gitter mit vierfacher Auflösung in Dimension x (links) und Octree, der dieses Gitter enthält (rechts)

des Gitters enthält der Octree weitere 16 Oktanten, die aufgrund der Geometrie des Octrees nötig sind, aber keinen Mehrwert für die numerische Anwendung bringen. Die Anzahl dieser Oktanten steigt exponentiell mit dem Größenunterschied der Dimensionen und erschwert die Berechnung dieser Art von Gittern erheblich. Dieser Overhead kann durch eine flexiblere Geometrie vermieden werden. Im Folgenden werden Forests of Octrees vorgestellt, die auf Octrees aufbauen und allgemeinere Geometrien darstellen können.

2.2 Forest of Octrees

Forests of Octrees [BWG11] erweitern das Konzept von Octrees, indem mehrere Bäume über gemeinsame Kanten oder Flächen miteinander verbunden werden (Kanten existieren nur im Fall eines dreidimensionalen Octrees). Sie können als zweistufige Unterteilung des geometrischen Raumes gesehen werden, wobei auf der Makroebene verschiedene Octrees miteinander verbunden werden und auf der Mikroebene die Verfeinerung jedes Octrees in eine Menge von Oktanten stattfindet.

Die Makroebene teilt den Raum $\Omega_d \subset \mathbb{R}^3$ in K Unterräume, von denen jeder mittels einer glatten Funktion ϕ_k von einem Referenzwürfel in den dreidimensionalen Raum abgebildet wird.

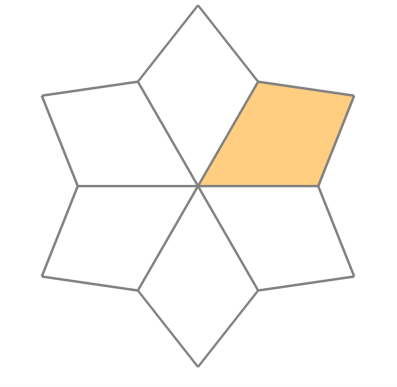


Abbildung 2.5: Forest of Octrees bestehend aus sechs Octrees, die sternförmig angeordnet sind und über den Mittelpunkt miteinander verbunden sind

$$\bar{\Omega}_d = \bigcup_k \phi_k([0,1]^d)\,, \quad \phi_k : [0,1]^d \to \mathbb{R}^3\,, \quad 0 \leq k < K\,, \quad d \in \{2,3\} \tag{2.1}$$

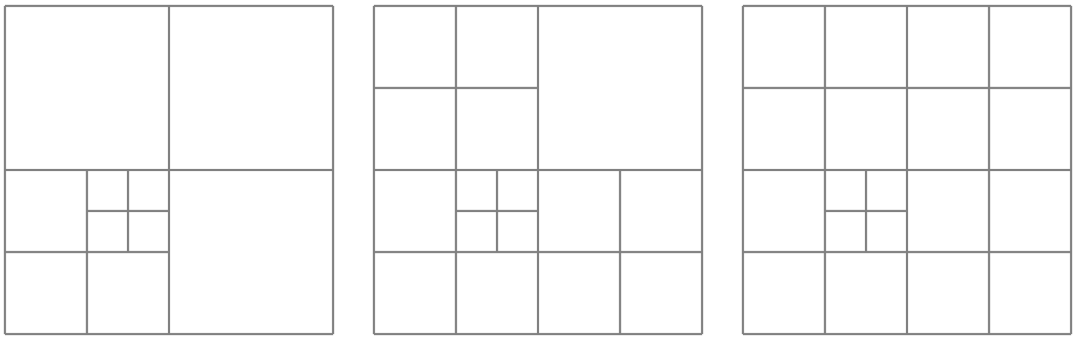
Genauer sind beliebige Mannigfaltigkeiten erlaubt, die von Abbildungen einer endlichen Menge von Octrees abgedeckt werden können, wobei zusätzlich die Einschränkung gilt, dass gemeinsame Makroflächen oder Makrokanten (nur für $d = 3$) zwischen Octrees komplett geteilt sein müssen. Diese Einschränkung verbietet sogenannte hanging faces bzw. edges, also Makroflächen oder Makrokanten, die nur teilweise von mehreren Octrees geteilt werden.

Mit diesem Ansatz erlaubt verglichen mit den Möglichkeiten eines einzelnen Octrees die Darstellung einer größeren Menge an Geometrien. Mit einem Forest of Octrees, der aus vier Octrees besteht, die in x-Richtung nebeneinander angeordnet sind und sich benachbarte Makroflächen teilen, kann beispielsweise das Gitter aus Abbildung 2.4a ohne zusätzliche Oktanten dargestellt werden. Andere Beipiele für Geometrien, die mit Forests of Octrees dargestellt werden können, sind das Möbiusband sowie der Torus.

Die Mikroebene beschreibt die rekursive Aufteilung eines Octrees in eine Menge von Oktanten. Im Gegensatz zur Makroebene sind hier allerdings auch hanging faces bzw. edges erlaubt, wodurch benachbarte Oktanten verschiedener Größen möglich werden.

2.2.1 Nachbarschaft

Betrachtet man Nachbarschaften bezüglich der Makroebene, können Octrees entweder über gemeinsame Flächen, Kanten oder Ecken verbunden sein. Da hanging faces auf der Makroebene nicht erlaubt sind, kann jede Makrofläche maximal eine benachbarte Fläche haben. Makrokanten und Makroecken können beliebig viele Nachbarn haben, da sie im Gegensatz zu Oktanten innerhalb eines Octrees nicht auf eine würfelförmige Geometrie festgelegt sind.



(a) Nicht balanciert **(b)** Flächen balanciert **(c)** Ecken balanciert

Abbildung 2.6: 2:1 Balance eines Quadtrees

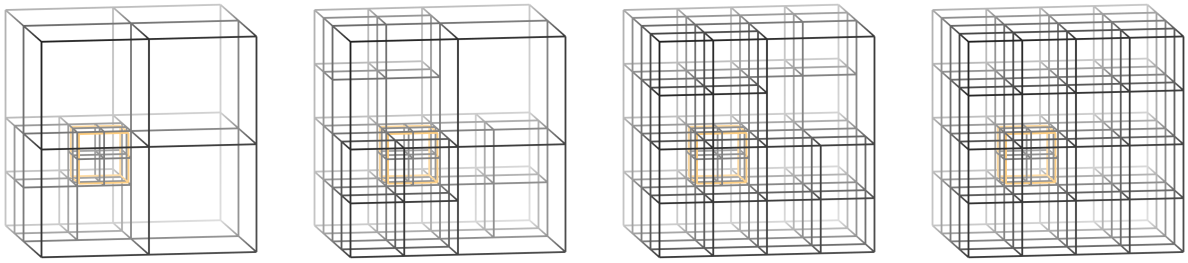
Abbildung 2.5 zeigt einen Forest of Octrees, dessen sechs Octrees eine gemeinsame Ecke im Mittelpunkt besitzen.

Auf der Mikroebene betrachten wir Nachbarschaften zwischen Oktanten, sowohl innerhalb desselben Octrees als auch zwischen Oktanten, die über eine Makrofläche, -kante oder -ecke verbunden sind. Da die Mikroebene hanging nodes erlaubt, können benachbarte Oktanten unterschiedliche Größen haben, woraus folgt, dass ein Oktant beliebig viele Flächen- und Kantennachbarn haben kann. Innerhalb eines Octrees ist die Anzahl der Eckennachbarn über eine Ecke des Oktanten exakt $2^d - 1$, an einer Makroecke können Ecken von beliebig vielen Octrees zusammentreffen.

Benachbarte Flächen und Kanten können unterschiedlich orientiert sein. Während bei Flächennachbarn 2^{d-1} verschiedene Orientierungen möglich sind, können benachbarte Kanten auf exakt zwei verschiedene Arten miteinander verbunden sein. Diese Orientierung bestimmt zusammen mit der Auswahl der Flächen bzw. Kanten der benachbarten Octrees die Ausrichtung ihrer lokalen Koordinatensysteme. Da alle Oktanten eines Octrees dasselbe Koordinatensystem verwenden, ist ihre Ausrichtung identisch. Oktanten verschiedener Octrees, die über Makroflächen bzw. kanten miteinander verbunden sind, können unterschiedliche Orientierungen haben.

2.2.2 2:1 Balance

Forests of Octrees erlauben benachbarte Oktanten beliebiger Größe. Für viele numerische Anwendungen werden allerdings Gitter benötigt, die den Größenunterschied zwischen benachbarten Oktanten limitieren. 2:1 Balance ist eine Eigenschaft einer Menge von Oktanten, die aussagt, dass alle benachbarte Oktanten entweder die gleiche Größe haben (1:1) oder dass sich die Größe der Oktanten um Faktor 2 unterscheidet (2:1). Existiert ein Paar von benachbarten



(a) Nicht balanciert **(b)** Flächen balanciert **(c)** Kanten balanciert **(d)** Ecken balanciert

Abbildung 2.7: 2:1 Balance eines Octrees

Oktanten, deren Größe sich um einen Faktor größer als 2 unterscheidet, ist die Eigenschaft verletzt.

Ist die Eigenschaft erfüllt, schränkt dies die möglichen Nachbarschaften eines Oktanten stark ein. Während ein Oktant generell beliebig viele Nachbarn pro Fläche bzw. Kante haben kann, können die Nachbarschaften eines 2:1 balancierten Forest of Octrees in drei Fälle eingeteilt werden:

- Ein Nachbar gleicher Größe
- Ein Nachbar doppelter Größe
- Zwei bzw. vier Nachbarn halber Größe

Zusätzlich kann 2:1 Balance auf verschiedene Arten von Nachbarschaften beschränkt werden. 2:1 Balance kann für Oktanten mit gemeinsamen Flächen, Kanten oder Ecken gefordert werden. Abbildung 2.6 und 2.7 zeigen Octrees, die auf unterschiedliche Arten balanciert wurden. Für die verschiedenen Arten von 2:1 Balance gelten folgende Implikationen:

- 2:1 Balance benachbarter Ecken $\Longrightarrow$ 2:1 Balance benachbarter Kanten
- 2:1 Balance benachbarter Kanten $\Longrightarrow$ 2:1 Balance benachbarter Flächen

2.2.3 Space-Filling Curve

Die Space-Filling Curve eines Forest of Octrees wird erstellt, indem die Z-Kurven aller Octrees des Forests aneinandergefügt werden. Die Reihenfolge der Octrees ist hierbei die Reihenfolge, in der diese definiert wurden. In der Theorie ist es möglich, die Reihenfolge der Octrees so zu wählen, dass aufeinander folgende Octrees geometrisch nicht zusammenhängen, praktisch sollte allerdings (falls möglich) eine Ordnung der Bäume in z-Reihenfolge gewählt werden.

3 Visualisierungsprogramm

3.1 Überblick

Das Visualisierungsprogramm besitzt eine grafische Benutzeroberfläche, über die die Anzeige und Manipulation eines Forest of Octrees möglich ist. Das Hauptfenster ist dabei in vier Bereiche eingeteilt, deren Funktionen im Folgenden kurz erläutert werden: Das zentrale Element ist die zweidimensionale Abbildung eines Forest of Octrees, die sich in der Mitte des Hauptfensters befindet. Rechts befindet sich eine Seitenleiste, in der die Makrostruktur des Forest of Octrees ausgewählt, Darstellungsoptionen angepasst und Informationen über den Forest of Octrees angezeigt werden können. Zusätzlich wird die Verfeinerung des ausgewählten Octrees in den Seitenleisten links und unten angezeigt und kann dort ebenfalls angepasst werden. In der

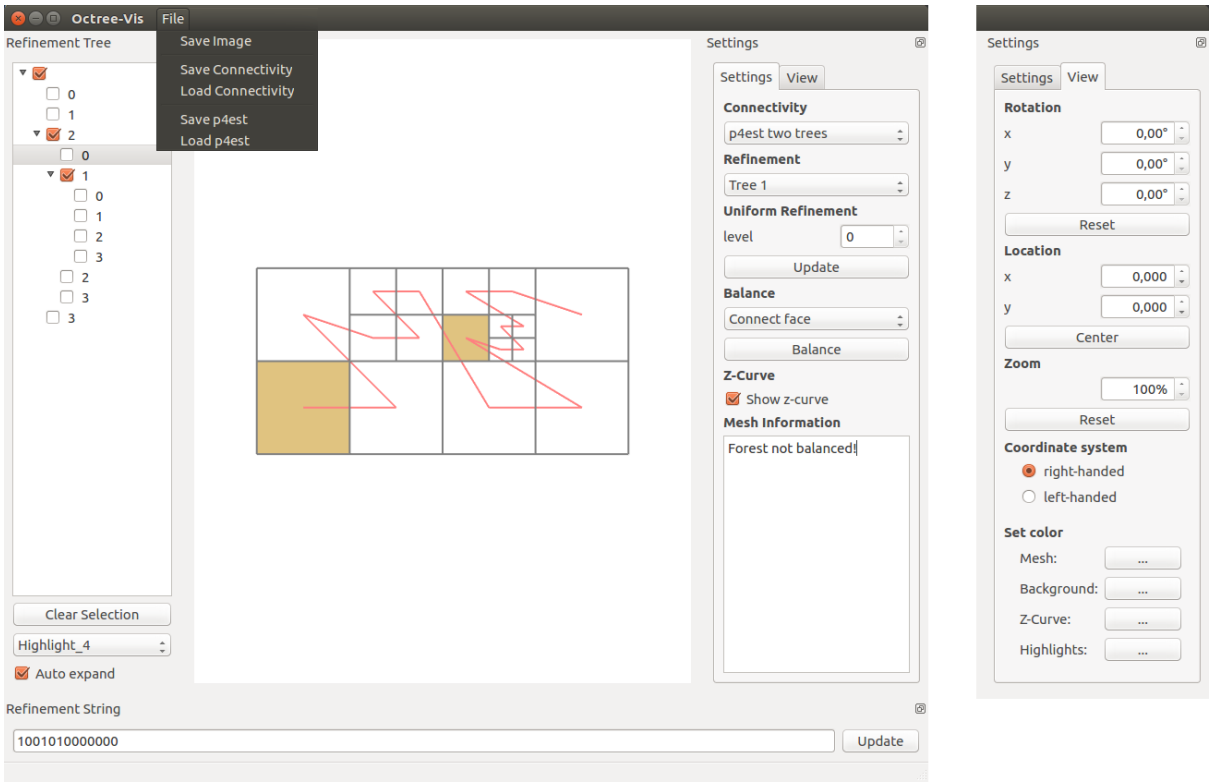


Abbildung 3.1: Hauptfenster des Visualisierungsprogramms

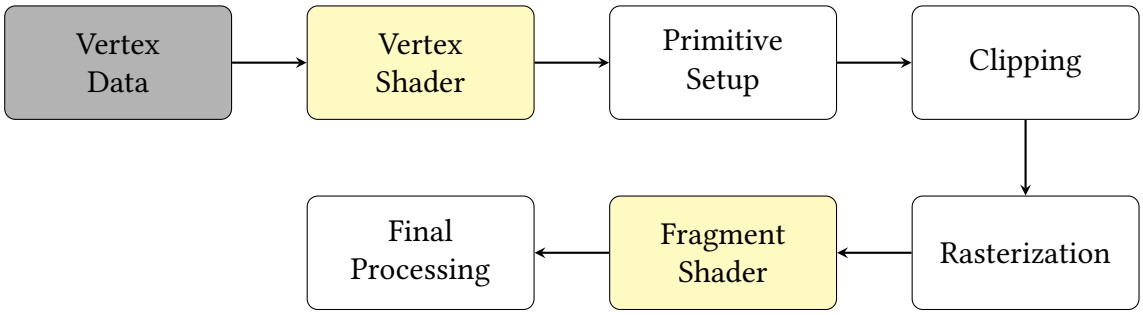


Abbildung 3.2: Verwendete OpenGL-Pipeline

unteren Seitenleiste wird die Verfeinerung dabei als Binärstring angezeigt, während sie in der linken Seitenleiste als Baumstruktur dargestellt wird. Über die linke Seitenleiste können außerdem einzelne Zellen des ausgewählten Octrees ausgewählt und hervorgehoben werden. Weitere Funktionen können über das Menü am oberen Rand des Hauptfensters erreicht werden.

Die Größe des Hauptfensters und der Seitenleisten ist anpassbar. Außerdem ist es möglich, die Seitenleisten aus dem Hauptfenster herauszulösen und als eigene Fenster zu verwenden, wodurch die Größe der einzelnen Elemente flexibler angepasst werden kann. Diese Funktion ist besonders hilfreich, wenn beispielsweise eine größere Anzeige des Forest of Octrees benötigt wird oder die Elemente des Hauptfensters auf mehrere Bildschirme verteilt werden sollen.

3.2 Verwendete Sprachen und Bibliotheken

Das Visualisierungsprogramm ist in C++ geschrieben und verwendet die Bibliotheken OpenGL, Qt und p4est. Die Programmiersprache C++ wurde ausgewählt, da es einerseits durch ihre große Verbreitung eine Vielzahl an kompatiblen Bibliotheken für grafische Anwendungen gibt und andererseits die Einbindung der in C implementierten Bibliothek p4est einfach möglich ist.

3.2.1 OpenGL

Für die Darstellung des Forests of Octrees wird die Bibliothek OpenGL [SSKL13] verwendet, die das performante Zeichnen von geometrischen Primitiven (Linien und Flächen) im dreidimensionalen Raum sowie die Abbildung der Szene auf eine zweidimensionale Zeichenfläche ermöglicht. Die hohe Performance der Bibliothek wird dabei durch die Verwendung der Grafikhardware (GPU) erreicht und ist Lösungen, die ausschließlich die CPU verwenden, in Geschwindigkeit und Effizienz deutlich überlegen.

Listing 3.1 Shaderprogramm des Vertex Shaders des Visualisierungsprogramms

```
attribute highp vec3 position;
attribute lowp vec3 color;
uniform mat4 transformation;
uniform vec3 bg_color;
smooth out vec3 color;
void main() {
  gl_Position = transformation * position;
  float alpha = (-gl_Position.z + 1) / 2;
  color = color * alpha + bg_color * (1 - alpha);
}
```

OpenGL führt eine Reihe von Arbeitsschritten aus, um ein Bild zu erzeugen. Diese Folge von Arbeitsschritten wird als Rendering-Pipeline bezeichnet. Abbildung 3.2 zeigt die Pipeline, die für das Visualisierungsprogramm verwendet wird. Die einzelnen Schritte der Pipeline werden im Folgenden vorgestellt:

Vertex Data

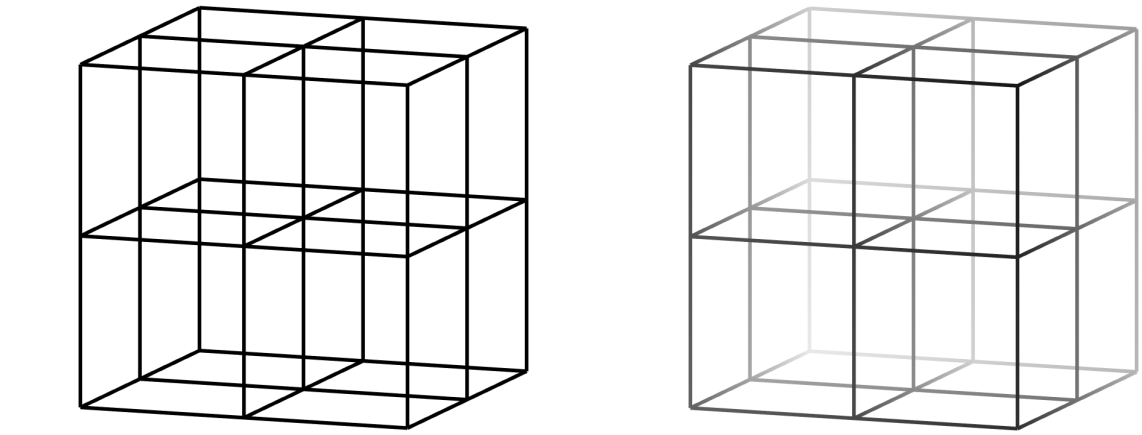
Die Eingabe der Pipeline besteht aus den geometrischen Daten, die gezeichnet werden sollen. Diese müssen im Speicher der Grafikkarte abgelegt werden, bevor sie gezeichnet werden können.

Die Szene, die das Visualisierungsprogramm zeichnet, besteht aus den Linien des Gitters, der Z-Kurve und den Linien bzw. Flächen der Hervorhebung. Diese Daten werden als Array von Fließkommazahlen gespeichert. Jeweils 6 Fließkommazahlen repräsentieren einen farbigen Punkt, wobei die ersten drei Komponenten die Position im dreidimensionalen Raum und die restlichen drei Komponenten die Farbe des Punktes im RGB-Format angeben. Zusätzlich zu diesem Array sind außerdem die Indices der ersten Punkte jedes Teils der Darstellung bekannt. Mithilfe dieser Indices können einzelne Teile der Szene unabhängig voneinander gezeichnet werden.

Vertex Shader

Shader sind spezielle Recheneinheiten der Grafikkarte, die programmiert werden können und so performant Daten einer 3D-Szene verarbeiten können. Das Visualisierungsprogramm verwendet die Vertex und Fragment Shader, die in jeder OpenGL Pipeline vorhanden sein müssen. OpenGL unterstützt weitere Shadertypen, die weitere 3D-Effekte ermöglichen, diese werden allerdings nicht für die Darstellung des Visualisierungsprogrammes benötigt.

Der Vertex Shader ist der erste Verarbeitungsschritt der Pipeline und verarbeitet die Eingabedaten. Abhängig von der Anwendung kann dieser Shader vom Weiterreichen der eingegebenen Daten bis zu komplexen Berechnungen viele verschiedene Aufgaben erfüllen. Listing 3.1 zeigt

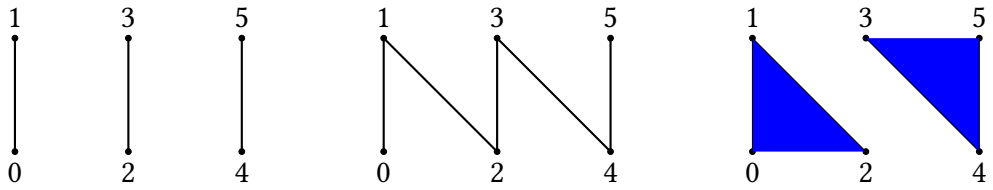


(a) Ohne Farbanpassung **(b)** Mit Farbanpassung

Abbildung 3.3: Tiefenabhängige Farbanpassung

das Shaderprogramm, das für den Vertex Shader des Visualisierungsprogramms verwendet wird. Dieses Shaderprogramm wird für jeden Punkt aufgerufen und erhält die Position und Farbe des Punktes aus den Eingabedaten. Zusätzlich erhält das Shaderprogramm die Farbe des Hintergrundes sowie eine Transformationsmatrix als Attribute. Diese beiden Attribute werden gesetzt, bevor die Szene gezeichnet wird, und haben unabhängig von dem Punkt, für den das Shaderprogramm aufgerufen wird, denselben Wert. Sie eignen sich damit für Daten, die für alle zu zeichnenden Punkte identisch sind.

Das Shaderprogramm des Vertex Shaders bestimmt die Position und Farbe des Punktes, für den es aufgerufen wird. Die Position wird berechnet, indem die Transformationsmatrix auf die Position des Punktes angewendet wird. Diese Transformationsmatrix bildet dabei Punkte vom globalen Koordinatensystem der Szene auf die Bildschirmkoordinaten ab. Die Farbe des Punktes wird abhängig von der Entfernung des Punktes zur Kamera angepasst, um einen Tiefeneindruck in der zweidimensionalen Abbildung zu erstellen. Je weiter ein Punkt von der Kamera entfernt ist, desto weniger hebt er sich vom Hintergrund ab. Abbildung 3.3 zeigt diese Farbanpassung. Die Entfernung eines Punktes zur Kamera (gl_position.z) liegt nach der Transformation in Bildschirmkoordinaten zwischen -1 (nah) und 1 (fern), sodass ein Punkt mit minimaler Distanz zur Kamera seine Farbe behält und einem Punkt mit maximaler Distanz zur Kamera die Hintergrundfarbe zugewiesen wird.



(a) GL_LINES **(b)** GL_LINE_STRIP **(c)** GL_TRIANGLES

Abbildung 3.4: Vergleich verschiedener OpenGL Darstellungsmethoden

Primitive Setup

Die Aufgabe dieses Verarbeitungsschrittes ist es, eine Menge von Punkten nach der Verarbeitung im Vertex Shader zu geometrischen Primitiven zusammenzusetzen. OpenGL bietet verschiedene Methoden, dies zu tun. Im Folgenden werden die drei Methoden vorgestellt, die von dem Visualisierungsprogramm verwendet werden. Zusätzlich zeigt Abbildung 3.4 die entstehenden geometrischen Primitive am Beispiel einer Menge von 6 Punkten.

- GL_LINES

  Die Methode GL_LINES setzt jeweils zwei Punkte zu einer Linie zusammen, die diese beiden Punkte verbindet. Ein Punkt wird dabei jeweils nur für eine Linie verwendet, sodass zum Zeichnen von n Linien $2n$ Punkte benötigt werden.

- GL_LINE_STRIP

  Die Methode GL_LINE_STRIP verbindet jeden Punkt der Eingabe mit seinem Nachfolger. So entsteht ein durchgängiger Pfad, der alle Punkte verbindet. Eine Eingabe von m Punkten wird zu $m - 1$ Linien zusammengesetzt.

- GL_TRIANGLES

  Die Methode GL_TRIANGLES setzt jeweils drei Punkte zu einem Dreieck zusammen. Ein Punkt wird dabei jeweils nur für ein Dreieck verwendet, sodass zum Zeichnen von n Dreiecken $3n$ Punkte benötigt werden.

Das Visualisierungsprogramm verwendet die Methode GL_LINES, um die Gitterlinien des Forest of Octrees sowie die linienbasierten Hervorhebungen zu zeichnen. Die Methode GL_LINE_STRIP wird verwendet, um die Z-Kurve zu zeichnen und GL_TRIANGLES findet bei der Zeichnung der flächenbasierten Hervorhebung Verwendung.

Clipping

Geometrische Primitive können außerhalb des sichtbaren Bereichs der Darstellung liegen. Clipping ist ein Verarbeitungsschritt, in dem diese so modifiziert werden, dass sie komplett im sichtbaren Bereich liegen. Dieser Schritt wird automatisch von OpenGL ausgeführt.

Rasterization

Der nächste Verarbeitungsschritt ist die Rasterisierung, in dem die geometrischen Primitive in eine Menge von Fragmenten umgewandelt werden. Fragmente können als Kandidaten für Bildpunkte im finalen Bild angesehen werden. Ob ein Fragment im entgültigen Bild sichbar ist, hängt allerdings von den weiteren Verarbeitungsschritten ab.

Fragment Shader

Der Fragment Shader ist der letzte Shader der Pipeline und bestimmt die Farbe eines Fragmentes. Ein Fragment Shader kann beispielsweise verwendet werden, um eine Textur auf einem Fragment anzuzeigen oder Fragmente anhand bestimmter Bedingungen zu verwerfen. Das Visualisierungsprogramm verwendet einen sogenannten pass-through Shader, der die Farbe des Fragments, für das er aufgerufen wurde, ohne Modifikationen an den nächsten Verarbeitungsschritt weitergibt.

Final Processing

In diesem letzten Schritt der Pipeline werden die Fragmente auf ihre Sichtbarkeit geprüft und gegebenenfalls in den Framebuffer geschrieben, der die Farbwerte für jeden Bildpunkt der Abbildung enthält.

3.2.2 Qt

Als Framework für die grafische Benutzeroberfläche wird die Bibliothek Qt verwendet, die neben einer großen Auswahl an Anzeige- und Kontrollwidgets auch Widgets enthält, die die einfache Einbindung von OpenGL Zeichenflächen ermöglichen. Widgets sind vorgefertigte Komponenten, die bestimmte Funktionen enthalten und unkompliziert in eine grafische Benutzeroberfläche integriert werden können. Die umfangreiche Auswahl an Tools und die verwendeten Konzepte der Bibliothek vereinfachen zudem die Entwicklung komplexer grafischer Benutzeroberflächen erheblich.

3.2.3 P4est

Das Visualisierungsprogramm verwendet die Bibliothek p4est und nutzt deren Datenstrukturen als Quelle der Daten, die visualisiert werden. Im Folgenden werden die Funktionen der p4est Bibliothek beschrieben, die für das Visualisierungsprogramm verwendet wurden:

Verwendete Funktionalität

Methoden und Module, die mit p4est beginnen, beziehen sich auf Forests of Quadtrees. Analog dazu enthalten Methoden und Module, die das Prefix p8est besitzen, Funktionalität für Forests of Octrees. Zur Vereinfachung werden im Folgenden jeweils nur die Funktionen p4est_* genannt, analog zu diesen existieren allerdings auch Implementierungen mit dem Prefix p8est.

Die Makrostruktur des Forest of Octrees wird in p4est als connectivity bezeichnet und kann über die Methode p4est_connectivity_new erstellt werden. Neben der Möglichkeit, eine eigene Makrostruktur zu erstellen, können auch spezielle Methoden p4est_connectivity_new_* verwendet werden, um bestimmte vordefinierte Makrostrukturen zu erstellen.

Die Methode p4est_new kann verwendet werden, um eine p4est Datenstruktur zu erstellen. Diese Datenstruktur enthält sämtliche Informationen über den Forest of Octrees. Bei der Erstellung eines Forest of Octrees wird die Makrostruktur als Parameter übergeben und kann anschließend nicht mehr modifiziert werden.

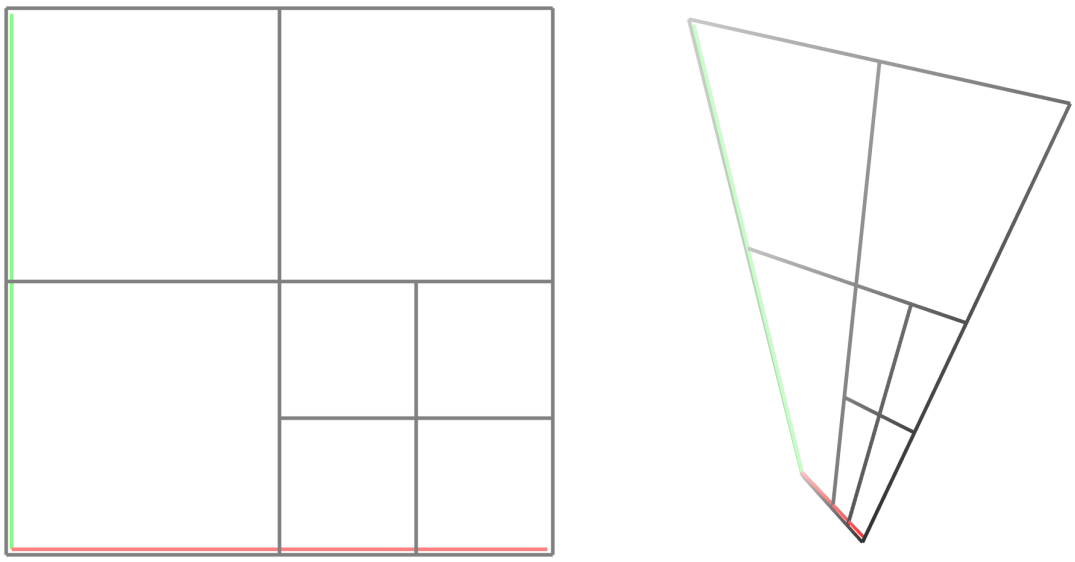
Die Mikrostruktur der Octrees des Forest of Octrees kann mit den Methoden p4est_refine und p4est_coarsen dynamisch angepasst werden.

Die Methode p4est_balance kann verwendet werden, um die Mikrostruktur eines Forest of Octrees so anzupassen, dass der Forest of Octrees anschließend 2:1 balanciert ist. Es ist ebenfalls möglich, die Art der Nachbarschaften zu wählen, für die diese Eigenschaft gelten soll. Die Methode p4est_is_balanced kann verwendet werden, um abzufragen, ob ein Forest of Octrees 2:1 balanciert ist.

Ist der Forest of Octrees 2:1 balanciert, kann die Methode p4est_mesh_new verwendet werden, um eine Datenstruktur zu erstellen, die die Nachbarschaftsinformationen aller Oktanten des Forest of Octrees enthält.

3.3 Koordinatensysteme

Das Visualisierungsprogramm verwendet verschiedene Koordinatensysteme, um einen Forest of Octrees darzustellen. Die verwendeten Koordinatensysteme und die Transformationen zwischen ihnen werden im Folgenden dargestellt.



(a) (b)

Abbildung 3.5: Octree im lokalen (links) und globalen (rechts) Koordinatensystem.

3.3.1 Lokale Koordinatensysteme

Jeder Octree besitzt ein lokales Koordinatensystem, bezüglich dessen die Positionen seiner Oktanten angegeben werden. Alle Oktanten eines Octrees liegen im Bereich $[0, 1]^d$, wobei d für die Dimensionalität des Octrees steht.

3.3.2 Weltkoordinaten

Weltkoordinaten werden verwendet, um die komplette zu zeichnende Szene in einem globalen Koordinatensystem darzustellen. Die Makrostruktur des Forest of Octrees ist bereits in Weltkoordinaten angegeben, die Oktanten eines Octrees müssen von dem lokalen Koordinatensystem des Octrees in das globale Koordinatensystem umgerechnet werden. Diese Umrechnung entspricht der Abbildungsfunktion $\phi_k : [0, 1]^d \to \mathbb{R}^3$ und wird im Folgenden beschrieben:

Da die Abbildungsfunktion die lokalen Koordinaten eines Octrees auf ein beliebiges Volumen im globalen Koordinatensystem abbilden kann, kann die Abbildung nicht mittels linearer Transformation der Koordinaten umgesetzt werden. Zudem ist die Funktion ϕ_k nicht vollständig definiert, da lediglich die Positionen der Eckpunkte der Octrees bekannt sind. Abbildung 3.5 zeigt einen Octree in lokalen und globalen Koordinaten.

Das Visualisierungsprogramm verwendet deshalb baryzentrische Koordinaten, um Punkte von dem lokalen Koordinatensystem eines Octrees in Weltkoordinaten umzurechnen. Ist ein Punkt Q als gewichtete Summe $Q = \lambda_1 P_1 + \lambda_2 P_2 + \cdots + \lambda_k P_k$ mit $\lambda_1 + \lambda_2 + \cdots + \lambda_k = 1$ darstellbar, bezeichnet man die Gewichte $(\lambda_1, \lambda_2, \ldots, \lambda_k)$ als die baryzentrischen Koordinaten des Punktes Q bezüglich der Basispunkte $P_1, \ldots, P_k \in \mathbb{R}^3$.

Um einen Punkt Q, der bezüglich des lokalen Koordinatensystems des Octrees angegeben ist, in Weltkoordinaten umzurechnen, werden die 2^d Eckpunkte des Octrees sowohl in lokalen Koordinaten P_i als auch in Weltkoordinaten P_i' benötigt. Zunächst werden die baryzentrischen Koordinaten $(\lambda_1, \lambda_2, \ldots, \lambda_k)$ von Q bezüglich der Basispunkte P_i (lokalen Koordinaten) berechnet. Anschließend werden die baryzentrischen Koordinaten λ_i auf die Basispunkte P_i' (globalen Koordinaten) angewendet. Der resultierende Punkt $Q' = \lambda_1 P_1' + \lambda_2 P_2' + \cdots + \lambda_k P_k'$ gibt die Position von Q in Weltkoordinaten an.

3.3.3 Normalisierte Weltkoordinaten

Die Makrostruktur des Forest of Octrees kann beliebige Punkte $P \in \mathbb{R}^3$ enthalten. Damit die Darstellung des Forest of Octrees in der Standardansicht sichtbar ist und einen Großteil der sichtbaren Fläche ausfüllt, wird die komplette Szene durch uniforme Skalierung und Parallelverschiebung auf den Bereich $[-1/\sqrt{3}, 1/\sqrt{3}]^3$ abgebildet. Die Wahl des Bereiches garantiert, dass die komplette Szene bei beliebiger Rotation um den Ursprung im Bereich $[-1, 1]^3$ liegt, was für die folgenden Abbildungen wichtig ist.

Die geometrischen Daten, die im Speicher der Grafikkarte abgelegt werden, befinden sich in normalisierten Weltkoordinaten.

3.3.4 Kamerakoordinaten

Eine OpenGL-Zeichenfläche stellt Szenen, die sich im Bereich $[-1, 1]^3$ befinden dar, indem sie eine orthographische Projektion der 3D-Koordinaten auf den zweidimensionalen Bereich $[-1, 1]^2$ ausführt. Die x- bzw. y-Koordinate bestimmt die horizontale bzw. vertikalen Position eines Punktes auf der Zeichenfläche. Die z-Koordinate bestimmt die Entfernung eines Punktes von der Kamera, wobei die Entfernung zu der Kamera steigt, je kleiner die z-Koordinate des Punktes ist. Abbildung 3.6 zeigt den sichtbaren Raum der OpenGL-Zeichenfläche sowie die Projektion auf den zweidimensionalen Bildbereich. OpenGL verwendet ein rechtshändiges Koordinatensystem, sodass die x-Achse nach rechts, die y-Achse nach oben und die z-Achse in Richtung des Betrachters zeigt. Im Gegensatz dazu würde die z-Achse eines linkshändigen Koordinatensystems in Blickrichtung des Betrachters liegen. In der Darstellung einer Szene kann zwischen den beiden Koordinatensystemen gewechselt werden, indem die z-Dimension invertiert wird.

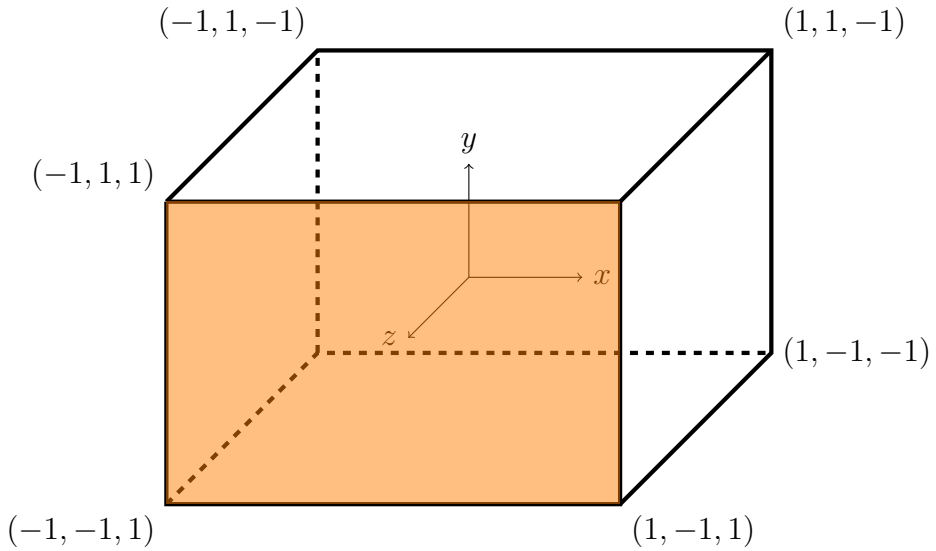


Abbildung 3.6: Darstellung des sichtbaren Bereichs einer OpenGL-Zeichenfläche

Die Transformation von normalisierten Weltkoordinaten in Kamerakoordinaten ist eine affine Abbildung und hängt von der Rotation, Position und Vergrößerung der Ansicht sowie dem Seitenverhältnis der OpenGL-Zeichenfläche und der Auswahl des Koordinatensystems ab. Sie kann als Transformationsmatrix $M = S \cdot T \cdot R$ dargestellt werden, die durch Anwenden der Rotationsmatrix R, der Translationsmatrix T und der Skalierungsmatrix S entsteht.

Die Rotationsmatrix $R = R_z(\gamma)R_y(\beta)R_x(\alpha)$ berechnet sich aus drei einzelnen Rotationen um die Koordinatenachsen, die nacheinander ausgeführt werden.

Die Translationsmatrix verschiebt die Szene in xy-Richtung und kann mittels homogenen Koordinaten folgendermaßen dargestellt werden:

$$T = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}_h \tag{3.1}$$

Die Variable x gibt hierbei den Versatz in horizontaler Richtung an, die Variable y den Versatz in vertikaler Richtung.

Die Skalierungsmatrix S setzt sich aus dem Skalierungsfaktor s, dem Seitenverhältnis $r = b/h$ (b und h sind Breite bzw. Höhe) der Zeichenfläche und der Orientierung des Koordinatensystems $c \in \{-1, 1\}$ folgendermaßen zusammen:

$$S = \begin{pmatrix} s/r & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & c \end{pmatrix} \tag{3.2}$$

Der Skalierungsfaktor wird auf die x- und y-Dimension angewendet, jedoch nicht auf die z-Dimension. Der Grund hierfür ist, dass eine Änderung der Skalierung der z-Dimension keinen Einfluss auf die Position der Elemente der Szene nach der orthographischen Projektion hat. Zudem werden Elemente, deren z-Koordinaten außerhalb des sichtbaren Bereichs $[-1, 1]$ liegen, nicht angezeigt, weshalb ein Skalierungsfaktor in der z-Dimesion zur Folge hätte, dass Elemente, die bezüglich ihrer x- und y-Koordinaten innerhalb der Zeichenfläche lägen, aufgrund ihrer z-Koordinaten nicht angezeigt würden. Die x-Dimension wird zusätzlich durch das Seitenverhältnis der Zeichenfläche geteilt, um eine identische Skalierung der x- und y-Dimensionen sicherzustellen. Der Faktor c kann verwendet werden, um die Darstellung zwischen einem rechts- bzw. linkshängiden Koordinatensystem umzustellen. Für $c = 1$ wird die Szene bezüglich eines rechtshändigen Koordinatensystems angezeigt, für $c = -1$ wird ein linkshändiges Koordinatensystem verwendet.

3.4 Features der Implementierung

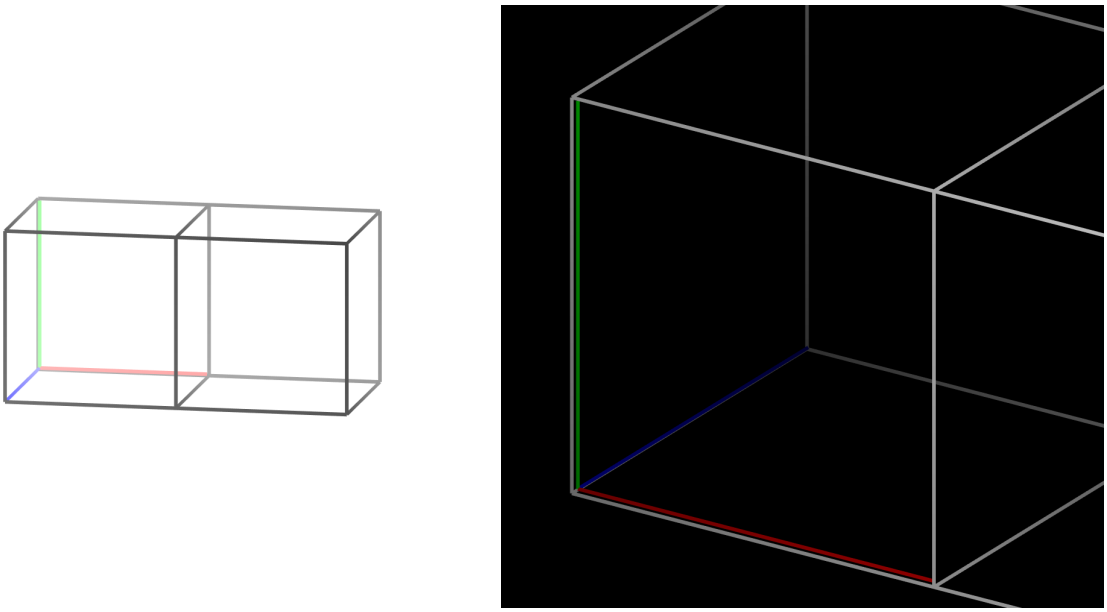
3.4.1 Darstellung eines Forest of Octrees

Das Visualisierungsprogramm enthält in der Mitte eine räumliche Darstellung eines Forest of Octrees. Diese Darstellung kann interaktiv auf verschiedene Weisen angepasst werden, um den Forest of Octrees (oder einen Teil davon) optimal darzustellen. Es ist möglich, die Rotation, Position und Vergrößerung des Forest of Octrees über die entsprechenden Eingabefelder im Tab “View“ oder mit der Maus anzupassen. Zudem ist es möglich, die Farbe des Gitters, des Hintergrunds, der Z-Kurve und des Highlights über entsprechende Schaltflächen in der grafischen Benutzeroberfläche anzupassen. Abbildung 3.7 zeigt zwei Darstellungen eines Octrees, die sich in verschiedenen Darstellungsoptionen unterscheiden.

3.4.2 Verfeinerung eines Octrees

Das Visualisierungsprogramm bietet drei Möglichkeiten, den aktuell ausgewählten Octree zu verfeinern, die im Folgenden vorgestellt werden.

Im Reiter “Settings“ der rechten Seitenleiste kann der aktuelle Octree von Dimension d gleichmäßig bis zu einem bestimmten Level l verfeinert werden, sodass der resultierende Octree $2^{d \cdot l}$ Oktanten gleicher Größe besitzt.



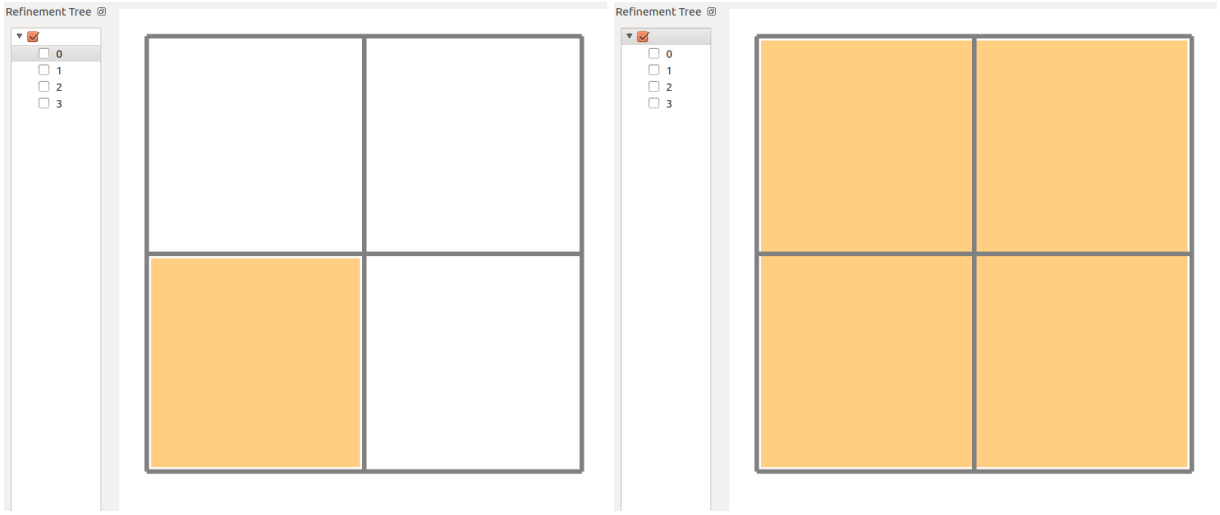
(a) (b)

Abbildung 3.7: Unterschiedliche Darstellungen desselben Forest of Octrees

Die zweite Möglichkeit besteht darin, die Verfeinerung in Form eines Binärstrings vorzugeben, wobei unvollständige Binärstrings implizit mit 0-Bits aufgefüllt werden. Durch diese Annahme ist es beispielsweise möglich, einen Octree von Dimension 3, der gleichmäßig bis Level 1 verfeinert ist, als unvollständigen Binärstring ”1” anzugeben. Der vollständige Binärstring für diese Verfeinerung lautet ”100000000”. Zeichen nach dem letzten relevanten Zeichen des Binärstrings werden ignoriert (”1000000001101” und ”100000000” führen zu derselben Verfeinerung).

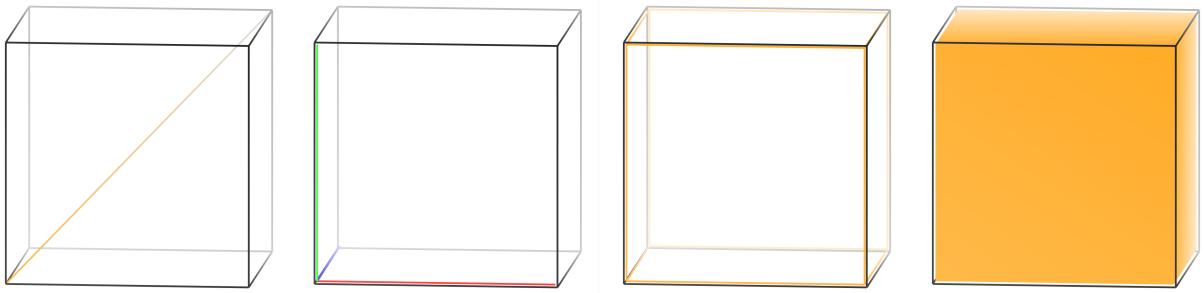
Die letzte Möglichkeit, die Verfeinerung des Octrees zu verändern, ist die Manipulation der hierarchischen Baumstruktur in der linken Seitenleiste. In dieser Darstellung wird für jede Zelle des Octrees ein Kontrollkästchen angezeigt, wobei ein aktiviertes Kästchen für eine verfeinerte Zelle und ein nicht aktiviertes Kästchen für einen Quadranten bzw. ein Blatt des Verfeinerungsbaums steht. Ein Oktant kann verfeinert werden, indem das zugehörige Kontrollkästchen aktiviert wird. Durch das Deaktivieren einer verfeinerten Zelle werden alle Oktanten dieser Zelle zu einem Oktanten zusammengefasst, wodurch diese diese Zelle zu einem Blatt im Verfeinerungsbaum wird.

Wird die Verfeinerung des aktuellen Octrees über eine der drei oben genannten Methoden verändert, werden alle anderen Anzeigen der Verfeinerung automatisch aktualisiert. Die mittige Darstellung des Forest of Octrees, der Verfeinerungsbaum in der linken Seitenleiste und der Binärstring in der unteren Seitenleiste zeigen somit für den ausgewählten Octree stets dieselbe Verfeinerung.



(a) Hervorhebung eines Oktanten **(b)** Hervorhebung einer verfeinerten Zelle

Abbildung 3.8: Hervorhebung einer Zelle



(a) Methode 1 **(b)** Methode 2 **(c)** Methode 3 **(d)** Methode 4

Abbildung 3.9: Vergleich verschiedener Hervorhebungsmethoden

3.4.3 Hervorheben einzelner Quadranten

Das Visualisierungprogram bietet die Möglichkeit, Zellen des angezeigten Forest of Octrees farblich hervorzuheben. Es kann eine Zelle pro Octree ausgewählt werden, wobei die ausgewähle Zelle entweder eine verfeinerte Zelle (innerer Knoten des Verfeinerungsbaums) oder ein Oktant (Blattknoten des Verfeinerungsbaums) sein kann, siehe Abbildung 3.8. Eine Zelle kann hervorgehoben werden, indem das zugehörige Element im Verfeinerungsbaum in der linken Seitenleiste selektiert wird. Durch Drücken der Schaltfläche “Clear Selection“ kann die Hervorhebung aufgehoben werden.

Abbildung 3.9 zeigt die vier verschiedenen Hervorhebungsmethoden, die über das Dropdown-Listenfeld in der linken Seitenleiste ausgewählt werden können. Diese Methoden und deren Eigenschaften werden im Folgenden detailliert vorgestellt:

Hervorhebungsmethode 1

Die erste Hervorhebungsmethode ist zugleich auch die einfachste Hervorhebung, da sie aus lediglich einer Linie besteht (siehe Abbildung 3.9a). Diese Linie ist eine Diagonale, die von den minimalen zu den maximalen lokalen Koordinaten der Zelle verläuft. Ihre Farbe beginnt mit der vom Benutzer festgelegten Hervorhebungsfarbe und endet mit dem Grauwert derselben Farbe. Durch die Position und den Farbverlauf der Diagonale kann die Position des Ursprungs des lokalen Koordinatensystems des Octrees bestimmt werden. Die Position des Ursprungs des lokalen Koordinatensystems reicht allerdings nicht aus, um die Orientierung der Koordinatenachsen eindeutig zu bestimmen. Die zweite Hervorhebungsmethode stellt diese Orientierung eindeutig dar.

Hervorhebungsmethode 2

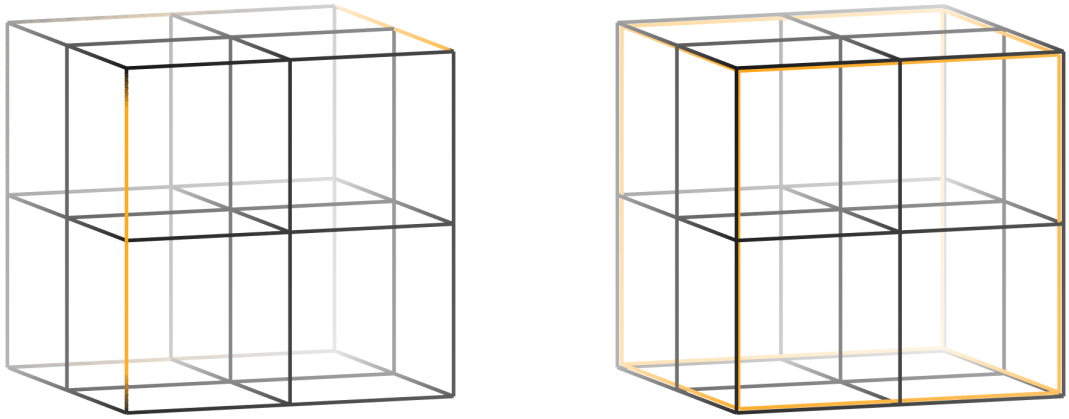
Die zweite Hervorhebungsmethode benutzt zwei bzw. drei farbige Linien, um die Ausrichtung der Koordinatenachsen des lokalen Koordinatensystems darzustellen (siehe Abbildung 3.9b). Die Linien beginnen an den minimalen lokalen Koordinaten der Zelle und verlaufen entlang der Gitterlinien derselben. Die Linien in x-, y- und z-Richtung haben die Farben Rot, Grün und Blau. Diese Hervorhebungsmethode stellt die Orientierung des lokalen Koordinatensystems eines Octrees eindeutig dar und ist besonders hilfreich, um die Orientierung verschiedener Octrees in einem Forest of Octrees zu vergleichen.

Hervorhebungsmethode 3

Die dritte Hervorhebungsmethode benutzt vier bzw. zwölf Linien, die entlang der Gitterlinien der ausgewählten Zelle verlaufen, um die Zelle hervorzuheben (siehe Abbildung 3.9c). Die Farbe der Linien ist die Hervorhebungsfarbe, die vom Nutzer eingestellt wurde. Da diese Methode sämtliche Kanten einer Zelle in einer einheitlichen Farbe hervorhebt, gibt sie einen guten visuellen Eindruck der Größe der Zelle wieder. Zudem bleiben bei Verwendung dieser Hervorhebung Zellen, die sich innerhalb oder hinter der hervorgehobenen Zelle befinden, sichtbar. Im Gegensatz zu den Hervorhebungsmethoden 1 und 2 stellt diese Hervorhebung keine Informationen über die Ausrichtung des lokalen Koordinatensystems des Octrees dar.

Hervorhebungsmethode 4

Die vierte Hervorhebungsmethode nutzt im Gegensatz zu den vorherigen Hervorhebungsmethoden Flächen statt Linien, um die ausgewählte Zelle hervorzuheben (siehe Abbildung 3.9d). Die Hervorhebung besteht aus zwei bzw. zwölf Dreiecken in der Hervorhebungsfarbe, wobei jeweils zwei Dreiecke eine viereckige Fläche darstellen. Durch die Verwendung von Flächen



(a) Ohne Offset **(b)** Mit Offset

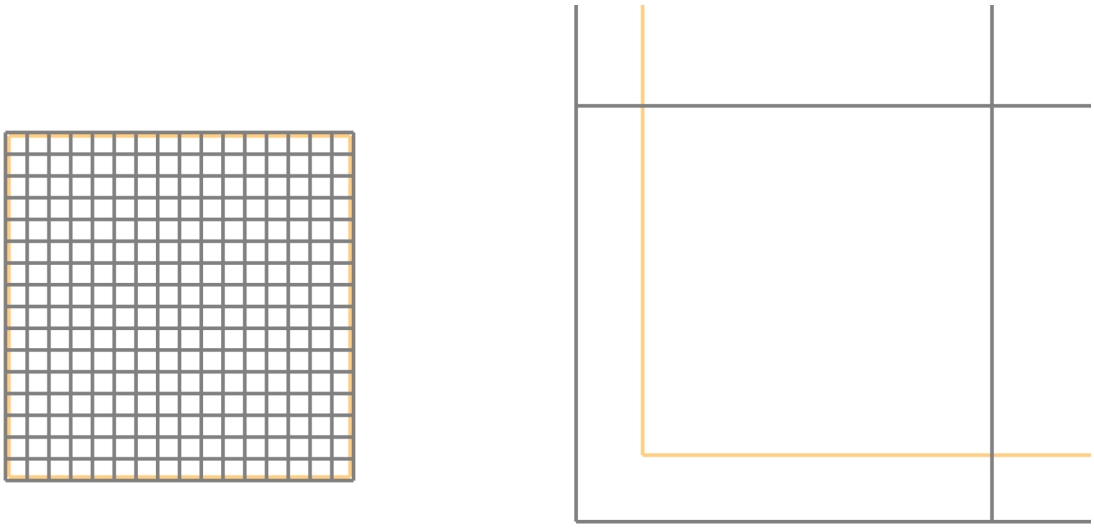
Abbildung 3.10: Z-Fighting zwischen Gitter und Hervorhebung

ist diese Hervorhebung optisch auffälliger als die vorherigen Methoden. Diese Hervorhebungsmethode stellt ebenfalls keine Informationen über die Ausrichtung des lokalen Koordinatensystems des Octrees dar. Durch die Verwendung undurchsichtiger Flächen verdeckt diese Hervorhebung Zellen, die sich innerhalb oder hinter der ausgewählten Zelle befinden.

Z-Fighting

Z-Fighting oder Stitching ist ein Phänomen, das in der Computergrafik auftritt, wenn mehrere Elemente an dieselbe Position gezeichnet werden. Da diese Elemente identische Tiefenwerte haben, kann nicht eindeutig entschieden werden, welches der Elemente sichtbar und welches verdeckt sein soll. Durch Rundungsfehler der Tiefenwerte entstehen visuelle Artefakte, sodass die Elemente jeweils nur teilweise sichtbar sind. Abbildung 3.10a zeigt einen Octree, dessen Außenkanten durch zusätzliche Linien in der Farbe Orange hervorgeboben werden sollen. In der Implementierung des Visualisierungsprogramms entsteht Z-Fighting ausschließlich, wenn die Hervorhebungen 2, 3 oder 4 genutzt werden, da diese Hervorhebungen zusätzliche Linien zeichnen, die entlang bestehender Gitterlinien verlaufen.

Eine Möglichkeit, Z-Fighting bei diesen Hervorhebungen zu verhindern, wäre, dafür zu sorgen, dass nur diejenigen Elemente gezeichnet werden, die auch sichtbar sein sollen. So könnte beispielsweise das Z-Fighting in Abbildung 3.10a verhindert werden, indem alle Linien, die entlang der Außenkanten des Octrees verlaufen, nicht gezeichnet würden. Obwohl dieser Ansatz in der Theorie funktioniert und sogar die Anzahl der zu zeichnenden Linien verringern



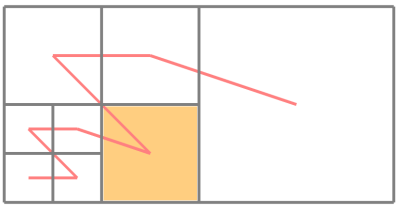
(a) (b)

Abbildung 3.11: Offset der Hervorhebung

könnte, ist die praktische Ermittlung der relevanten Kanten nicht trivial. Sowohl innerhalb eines Octrees als auch zwischen verschiedenen Octrees können sich Linien benachbarter Oktanten verschiedener Größen ganz oder teilweise überdecken, was die korrekte Filterung der Liniensegmente zusätzlich erschwert. Zusätzlich wäre bei diesem Ansatz das Gitter des Forest of Octrees abhängig von der ausgewählen Zelle und Hervorhebungsmethode, sodass bei jeder Änderung der Hervorhebung zusätzlich das komplette Gitter neu berechnet werden müsste.

Eine weitere Möglichkeit, Z-Fighting zu verhindern, ist, sich überdeckende Elemente mithilfe eines Offsets zu verschieben. Durch diese Verschiebung erhalten die Elemente unterschiedliche Positionen, die die eindeutige Berechnung der Sichtbarkeit der Elemente zulassen. Das Visualisierungsprogramm verwendet ein Offset von 0.01 je Dimension bezüglich des lokalen Koordinatensystems des Oktanten, um die Linien der Hervorhebung in Richtung der Mitte der hervorgehobenen Zelle zu verschieben. Der Vorteil dieser Verschiebung besteht darin, dass die Linien der Hervorhebung unabhängig vom Blickwinkel auf die Szene sichtbar sind und die Position der Linien nicht angepasst werden muss, wenn sich der Blickwinkel ändert. Ein Nachteil dieses Offsets ist, dass sich der sichtbare Abstand zwischen Gitter- und Hervorhebungslinie vergrößert, wenn die Szene vergrößert wird. Abbildung 3.11 zeigt den Abstand zwischen Gitter- und Hervorhebungslinien für zwei verschiedene Vergrößerungen desselben Octrees.

Eine weitere Möglichkeit der Implementierung von Offsets zur Verhinderung von Z-Fighting wäre, die Linien der Hervorhebung nach der Transformation in Kamerakoordinaten in Rich-



(a)

Mesh Information

local quadrants: 8
ghost_num_quadrants: 0

Selected quadrant: 4

Face neighbors:
Face 0: half-size face 1 (orientation 0) of quads 1 and 3
Face 1: double-size face 0 (orientation 0, subface 0) of quad 7
Face 2: same-size face 2 (orientation 0) of quad 4
Face 3: same-size face 2 (orientation 0) of quad 6

Corner neighbors:
Corner 0: Quad -3
Corner 1: Quad -3
Corner 2: Quad 5
Corner 3: Quad -1

(b)

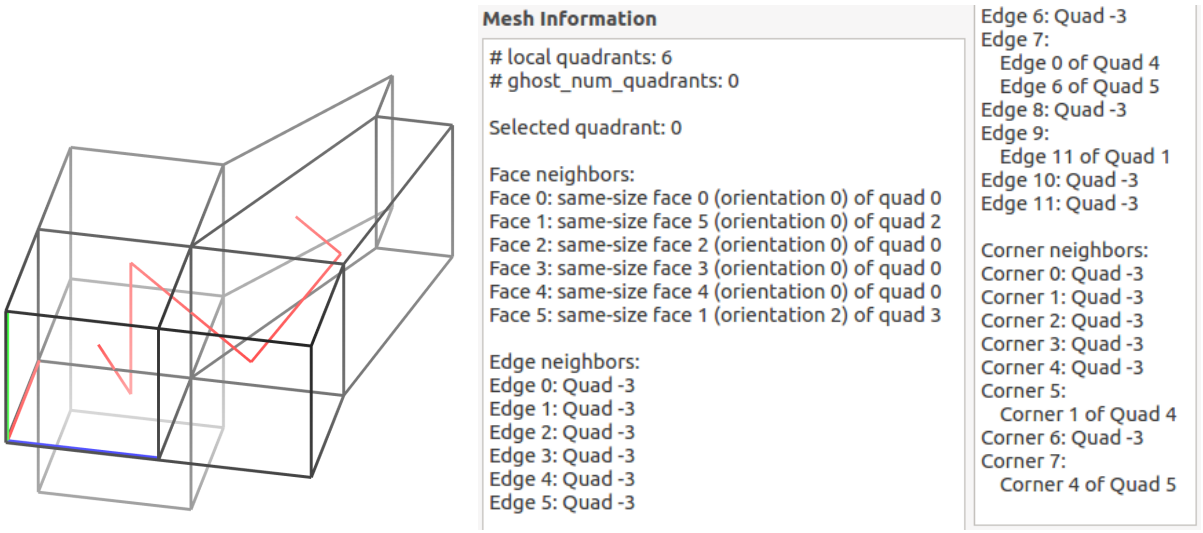
Abbildung 3.12: Nachbarschaftsinformationen eines Forest of Octrees ($d = 2$)

tung der Kamera zu verschieben. Durch diese Art der Verschiebung könnten die Linien der Hervorhebung Gitterlinien überdecken, ohne in der zweidimensionalen Abbildung des Forest of Octrees eine andere Position zu erhalten. Da in diesem Ansatz die Linien der Hervorhebung abhängig vom Blickwinkel auf die Szene wären, müssten sie jedoch bei jeder Änderung des Blickwinkels neu berechnet werden.

3.4.4 Anzeigen von Nachbarschaftsinformationen

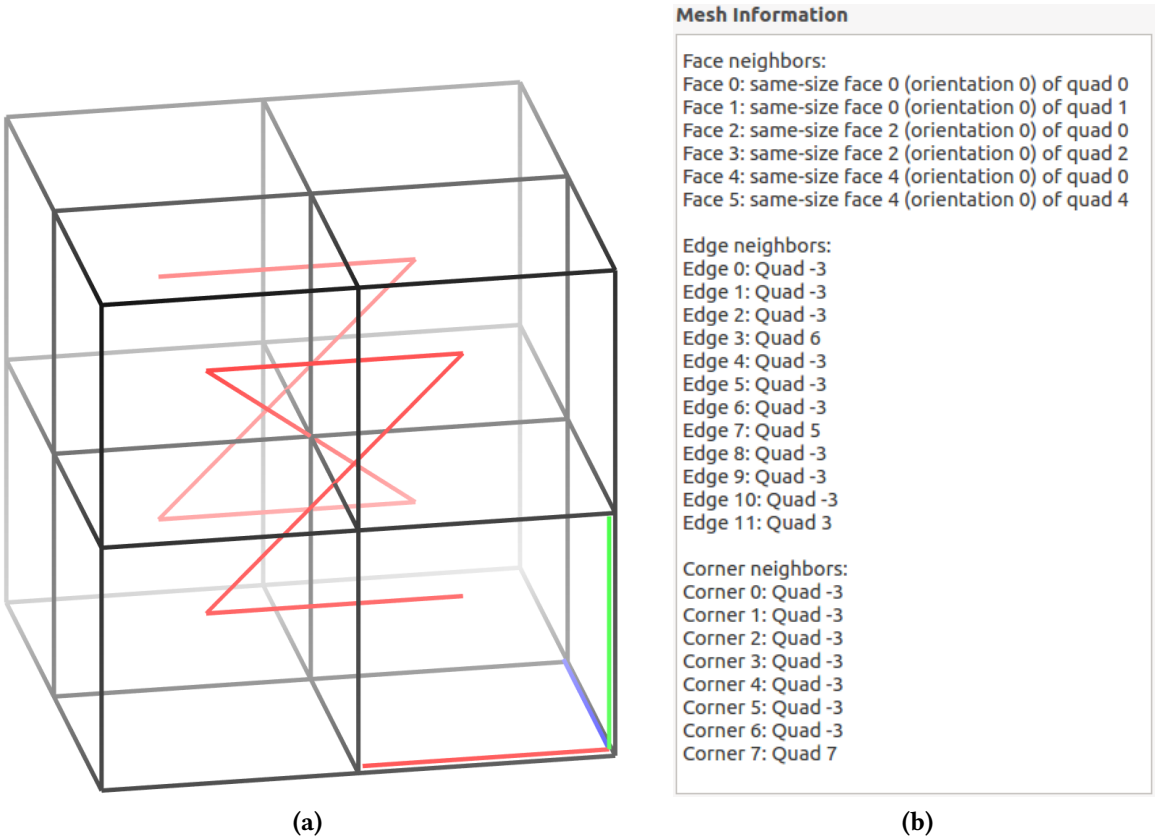
Eine weitere Funktionalität das Visualisierungsprogramms ist, die Nachbarschaftsinformationen des hervorgehobenen Oktanten anzuzeigen. Die Informationen werden aus der Datenstruktur p4est_mesh ausgelesen, die erstellt wird, falls der Forest of Octrees 2:1 balanciert ist. Diese Datenstruktur wird zudem neu berechnet, sobald sich die Mikro- oder Makrostruktur des Forest of Octrees ändert. Ist der Forest of Octrees nicht 2:1 balanciert, können keine Nachbarschaftsinformationen angezeigt werden. Die Nachbarschaftsinformationen, die für einen 2:1 balancierten Forest of Octrees angezeigt werden, sind zudem von den ausgewählten Arten von Nachbarschaften abhängig. Ist ein Forest of Octrees zum Beispiel lediglich bezüglich Flächennachbarn 2:1 balanciert, können auch nur Nachbarschaftsinformationen zu Flächennachbarn angezeigt werden. Sowohl benachbarte Oktanten innerhalb desselben Octrees als auch benachbarte Oktanten verschiedener Octrees werden in der Anzeige der Nachbarschaften berücksichtigt.

Die Nachbarschaftsinformationen werden (falls vorhanden) für jede Fläche, Kante und Ecke des Oktanten textbasiert angezeigt. Die Benennung der Flächen, Kanten und Ecken entspricht dabei



(a) **(b)**

Abbildung 3.13: Nachbarschaftsinformationen eines Forest of Octrees ($d = 3$)



(a) **(b)**

Abbildung 3.14: Nachbarschaftsinformationen innerhalb eines Octrees ($d = 3$)

dem Benennungsschema aus Abbildung 2.2. Außerdem werden die Anzahl der Quadranten sowie der Index des hervorgehobenen Quadranten angezeigt.

Abbildung 3.12 und 3.13 zeigen jeweils einen Forest of Octrees sowie die Nachbarschaftsinformationen des ausgewählten Oktanten des Forest of Octrees.

Folgende Arten von Nachbarschaften können angezeigt werden:

Flächennachbarn

Aufgrund der 2:1 Balance kann eine Fläche eines Oktanten eine Fläche gleicher Größe, eine Fläche doppelter Größe oder zwei bzw. vier Flächen halber Größe als Nachbarn haben. Eine Fläche wird dabei über den Index des Oktanten und den Index der Fläche des Oktanten eindeutig identifiziert. Zusätzlich wird die relative Orientierung der Flächen zueinander sowie der Index der Subfläche (falls benachbarte Fläche doppelte Größe hat) angegeben. Flächen, die keinen Nachbarn haben, verweisen auf sich selbst (siehe Face 2 in Abbildung 3.12).

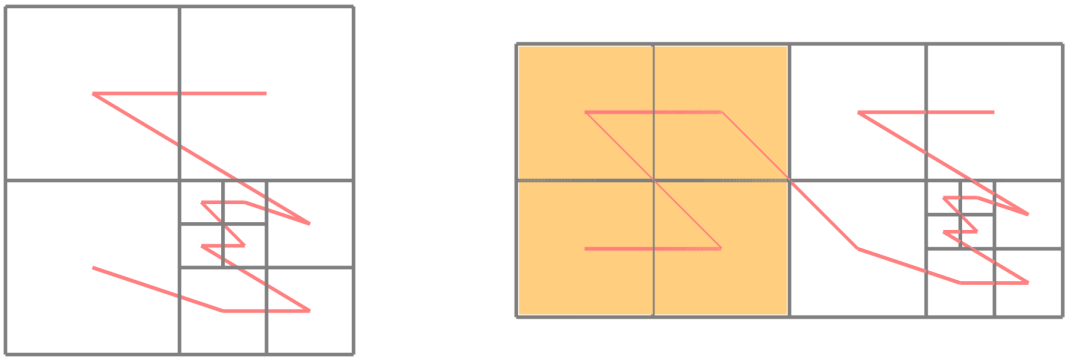
Kantennachbarn

Kantennachbarn existieren nur für Forests of Octrees mit $d = 3$. Kantennachbarn, die ebenfalls Flächennachbarn sind, werden nicht als Kantennachbarn angezeigt.

Innerhalb eines Octrees gibt es für jede Kante eines Oktanten genau einen benachbarten Oktanten, der nicht gleichzeitig auch ein Flächennachbar ist (siehe Edge 3, 7 und 11 in Abbildung 3.14). Am Rand eines Oktanten kann es beliebig viele Kantennachbarn geben. Hat eine Kante keinen Nachbarn, wird der Index -3 angezeigt. Abbildung 3.13 zeigt sowohl Kanten, die mehrere Nachbarn haben (Edge 7), als auch Kanten, die keine Nachbarn haben (Edge 6).

Eckennachbarn

Innerhalb eines Octrees werden Eckennachbarn, die ebenfalls Flächennachbarn oder Kantennachbarn sind, nicht als Eckennachbarn angezeigt. Eckennachbarn verschiedener Bäume, die ebenfalls Flächennachbarn sind, werden ebenfalls nicht angezeigt. Eckennachbarn verschiedener Bäume, die ebenfalls Kantennachbarn sind, können allerdings ebenfalls als Eckennachbarn angezeigt werden (siehe Corner 5 und 7 in Abbildung 3.13).



(a) Z-Kurve eines Octrees **(b)** Z-Kurve eines Forest of Octrees (aus zwei Octrees)

Abbildung 3.15: Z-Kurven eines Octrees und eines Forest of Octrees

3.4.5 Weitere Funktionen

Neben den bereits genannten Features des Visualisierungsprogramms werden im Folgenden weitere Funktionen des Visualisierungsprogrammes vorgestellt:

- Anzeigen der Z-Kurve

  Die Z-Kurve wird als durchgängige Linie, die die Mittelpunkte der Oktanten aller Bäume des Forest of Octrees verbindet, gezeichnet. Abbildung 3.15 zeigt die Z-Kurve eines Octrees sowie eines Forest of Octrees, der aus zwei Octrees besteht.
- Laden einer vorgegebenen Makrostruktur über das Auswahlfeld in der rechten Seitenleiste.
- Speichern bzw. Laden der Connectivity Datenstruktur (Makrostruktur, ohne Verfeinerung der einzelnen Bäume) über das Programmmenü.
- Speichern bzw. Laden der gesamten p4est Datenstruktur (Makrostruktur inklusive Verfeinerung der einzelnen Bäume) über das Programmmenü.
- Speichern der aktuellen Darstellung des Forest of Octrees als Bilddatei (über das Programmmenü).

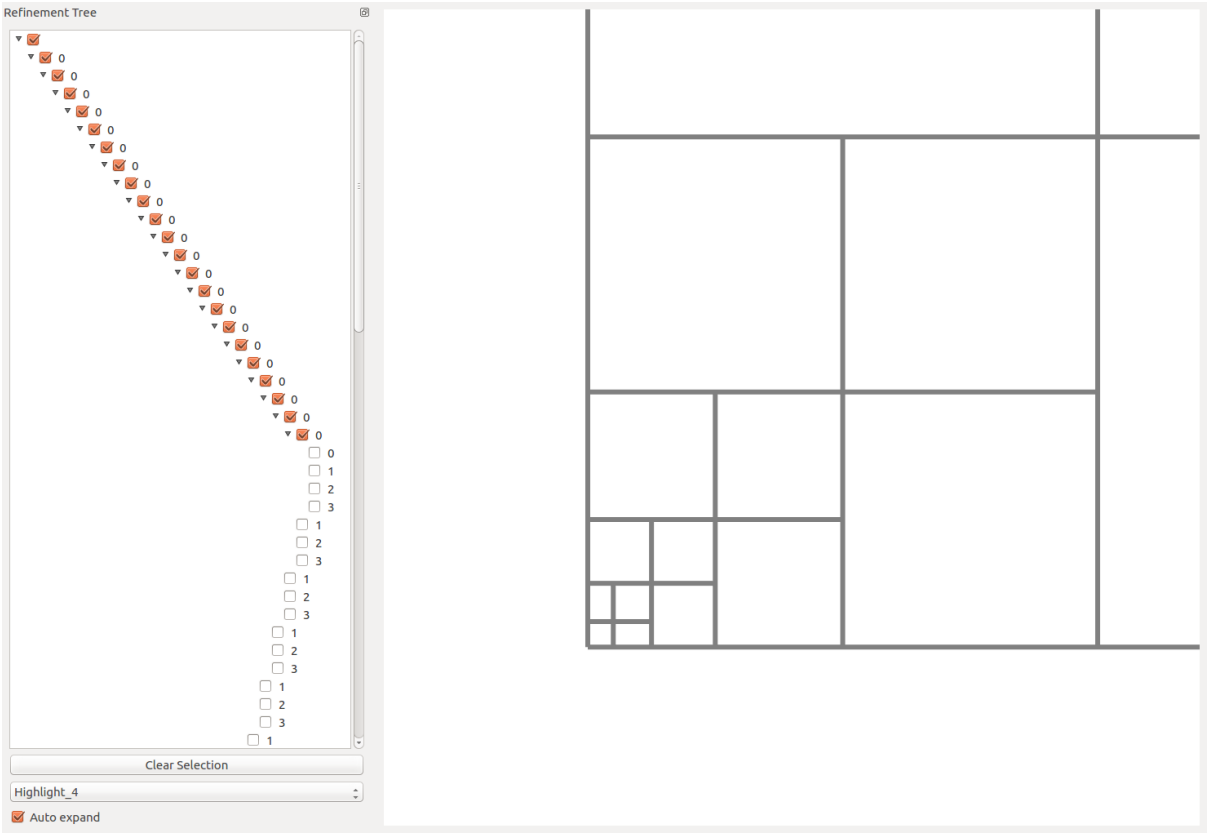


Abbildung 3.16: Genauigkeit bei sehr starker Verfeinerung

3.5 Performance

3.5.1 Genauigkeit

Das Visualisierungsprogramm verwendet Fließkommazahlen des Typs GLFloat, um Punkte im dreidimensionalen Raum anzugeben. Abbildung 3.16 zeigt einen Ausschnitt eines Octrees, der sehr stark verfeinert ist. In der linken unteren Ecke ist erkennbar, dass die vier Oktanten des feinsten Levels unterschiedlich groß dargestellt werden. Die Ursache hierfür ist die begrenzte Genauigkeit der GLFloat-Fließkommazahlen. Sichtbare Darstellungsfehler treten bei einem zentrierten und nicht gedrehten zweidimensionalen Octree ab Level 22 auf. Die Größe eines Oktanten dieses Levels beträgt $2^{-22} \approx 2.38 \cdot 10^{-7}$ je Dimension. Abhängig von der gewählten Geometrie, Verfeinerung und Ansicht der Oktanten kann sich das Level, ab dem Darstellungsfehler auftreten, unterscheiden.

Bei Bedarf könnte die Genauigkeit (sofern von OpenGL unterstützt) durch die Wahl von Fließkommazahlen höherer Genauigkeit verbessert werden, in der praktischen Anwendung des Visualisierungsprogramms ist dieses Problem allerdings kaum relevant, da die visualisierten Forests of Octrees in der Regel bei weitem nicht so stark verfeinert sind.

3.5.2 Speicherbedarf der geometrischen Daten

Das Visualisierungsprogramm berechnet aus den Datenstrukturen, die die p4est-Bibliothek bereitstellt, geometrische Daten, die in den Speicher der Grafikkarte geschrieben und anschließend zur Darstellung des Forest of Octrees, der Hervorhebung sowie der Z-Kurve verwendet werden. Im Folgenden wird der Speicherbedarf dieser Daten untersucht:

Die geometrischen Daten bestehen aus einer Menge von farbigen Punkten, die jeweils in Form von sechs Fließkommazahlen des Typs float gespeichert werden. Die Größe f des Typs float ist nicht fest definiert und kann sich zwischen verschiedenen Architekturen und Compilern unterscheiden. Für die Größe p eines farbigen Punktes gilt $p = 6f$.

Speicherbedarf der Oktanten

Ein Oktant der Dimension $d \in \{2, 3\}$ besteht aus $2^{d-1} \cdot d$ geometrischen Linien, die jeweils aus 2 farbigen Punkten bestehen. Für den Speicherbedarf o eines Oktanten gilt somit $o = 2^d \cdot d \cdot p$. Der Speicherbedarf eines Forest of Octrees, der aus n Oktanten besteht, ist folglich $n \cdot o$.

Speicherbedarf der Z-Kurve

Die Z-Kurve verbindet alle n Oktanten des Forest of Octrees mit einer durchgängigen Linie. Da sie die OpenGL-Zeichenmethode GL_LINE_STRIP verwendet, benötigt sie für jeden Oktanten exakt einen farbigen Punkt. Der Speicherbedarf z der Z-Kurve ist somit $n \cdot p$.

Speicherbedarf der Hervorhebungen

Der Speicherbedarf der Hervorhebung eines Oktanten hängt von der gewählten Hervorhebungsmethode ab. Hervorhebungsmethode 1 besteht aus exakt einer Linie, sodass für den Speicherbedarf $h_1 = 2p$ gilt. Die zweite Hervorhebungsmethode zeichnet d Linien, sodass $h_2 = 2d \cdot p$ gilt. Der Speicherbedarf von Hervorhebungsmethode 3 ist identisch zu dem Speicherbedarf eines Oktanten, sodass $h_3 = o = 2^d \cdot d \cdot p$ gilt. Die vierte Hervorhebungsmethode zeichnet 6^{d-2} Flächen die jeweils aus zwei Dreiecken bzw. sechs farbigen Punkten bestehen. Für den Speicherbedarf gilt dementsprechend $h_4 = 6^{d-1} \cdot p$.

Zusätzlich definieren wir h_{max}, das den maximal möglichen Speicherbedarf einer Hervorhebung in Abhängikkeit der Dimension d des Oktanten angibt:

$$h_{max} = \max\{h_1, h_2, h_3, h_4\} = \begin{cases} h_3 & d = 2 \\ h_4 & d = 3 \end{cases} = \begin{cases} 8p & d = 2 \\ 36p & d = 3 \end{cases}$$

$Dimension$	h_1	h_2	h_3	h_4	h_{max}
2	$12f$	$24f$	$48f$	$36f$	$48f$
3	$12f$	$36f$	$144f$	$216f$	$216f$

Tabelle 3.1: Speicherbedarf der Hervorhebung in Abhängigkeit von Dimension d und Hervorhebungsmethode h_i (mit $i \in \{1, 2, 3, 4, max\}$). f ist die Größe einer Fließkommazahl.

Tabelle 3.1 zeigt den Speicherbedarf einer Hervorhebung für verschiedene Hervorhebungsmethoden und Dimensionen.

Da für jeden Octree eines Forest of Octrees ein Oktant hervorgehoben werden kann, berechnet sich der maximale gesamte Speicherbedarf aller Hervorhebungen h'_i eines Forests of Octrees bestehend aus m Octrees als $h'_i = m \cdot h_i$ mit $i \in \{1, 2, 3, 4, max\}$.

Gesamtspeicherbedarf

Der Gesamtspeicherbedarf g_d berechnet sich folgendermaßen:

$$
\begin{aligned}
g_d &= n \cdot o + n \cdot p + h'_{max} \\
&= n \cdot 2^d \cdot d \cdot p + n \cdot p + m \cdot h_{max} \\
g_2 &= n \cdot 8p + n \cdot p + m \cdot 8p \\
&= n \cdot 9p + m \cdot 8p \\
&= (n \cdot 9 + m \cdot 8) \cdot 6f \\
g_3 &= n \cdot 24p + n \cdot p + m \cdot 36p \\
&= n \cdot 25p + m \cdot 36p \\
&= (n \cdot 25 + m \cdot 36) \cdot 6f
\end{aligned}
$$

Tabelle 3.2 zeigt den Gesamtspeicherbedarf in Abhängigkeit der Dimension d und der Anzahl der Oktanten n. Da die Makrostruktur eines Forest of Octrees in der Regel aus wenigen Octrees besteht, ist der Speicherbedarf für die Hervorhebungen sehr gering und wurde der Einfachkeit halber in Tabelle 3.2 vernachlässigt. Aktuelle Grafikhardware besitzt Speicher im Bereich von einigen Gigabytes, sodass abhängig von der verwendeten Hardware die Darstellung von bis zu $\approx$ 10.000.000 Oktanten möglich sein sollte.

3.5.3 Berechnungsaufwand / Skalierbarkeit

Das Visualisierungsprogramm führt verschiedene Operationen aus, deren Berechnungsaufwand und Speicherbedarf stark von der Komplexität des angezeigten Forest of Octrees abhängt und die im Folgenden genannt sind:

	Dimension	
Anzahl Oktanten	2	3
1	216 Byte	600 Byte
10	2160 Byte	6000 Byte
100	$\approx$ 21 KB	$\approx$ 59 KB
1.000	$\approx$ 211 KB	$\approx$ 586 KB
10.000	$\approx$ 2109 KB	$\approx$ 5859 KB
100.000	$\approx$ 21 MB	$\approx$ 57 MB
1.000.000	$\approx$ 206 MB	$\approx$ 572 MB
10.000.000	$\approx$ 2060 MB	$\approx$ 5722 MB
100.000.000	$\approx$ 20 GB	$\approx$ 56 GB

Tabelle 3.2: Speicherbedarf des Gitters sowie der Z-Kurve in Abhängigkeit der Dimension d und der Anzahl der Oktanten. Die Größe f einer Fließkommazahl wurde mit 4 Byte angenommen.

- Modifizieren der p4est-Datenstruktur
- Berechnung der Nachbarschaftsbeziehungen der Oktanten des Octrees
- Berechnung der geometrischen Daten zur Anzeige des Forest of Octrees

Wird die Laufzeit der oben genannten Funktionen bezüglich verschiedener Forests of Octrees verglichen, zeigt sich, dass die Berechnung der Nachbarschaftsbeziehungen der Oktanten des Octrees die Laufzeit dominiert und bereits für eine Menge von 10^3 Oktanten eine deutliche Verzögerung verursacht. Der Grund hierfür ist, dass die Nachbarschaftsbeziehungen aller Oktanten berechnet und gespeichert werden.

Die Komplexität der anderen beiden Operationen ist wesentlich geringer und ist dementsprechend in der Lage, komplexere Forests of Octrees darzustellen. Die Berechnung der Nachbarschaftsbeziehungen limitiert somit die maximale Komplexität der darstellbaren Forests of Octrees.

4 Zusammenfassung und Ausblick

In dieser Arbeit wurden Möglichkeiten untersucht, wie ein Forest of Octrees sowie seine Eigenschaften visualisiert werden können. Das aus diesen Überlegungen entstandene Visualisierungsprogramm bietet die Möglichkeit, einen Forest of Octrees zu visualisieren und interaktiv zu modifizieren. Zusätzlich ist es möglich, weitere Eigenschaften des Forests of Octrees oder einzelner Octrees bzw. Oktanten anzuzeigen. Die Erstellung und Verwaltung des Forest of Octrees wird dabei von der p4est Softwarebibliothek umgesetzt, auf die das Visualisierungsprogramm aufbaut.

Die Softwarebibliothek p4est bietet durch den Export bestimmer Eigenschaften eines Forest of Octrees eine indirekte Möglichkeit der Visualisierung. Diese Visualisierung erlaubt die Anzeige von numerischen Daten der Anwendung und bietet durch periodisches Exportieren des Forest of Octrees die Möglichkeit, den Verlauf der Simulation nachzuvollziehen. Da das erstellte Visualisierungsprogramm ausschließlich Gitterstrukturen darstellt und keine numerischen Anwendungen auf diesen ausführen kann, bietet es diese Funktionalität nicht. Ein weiterer Unterschied zwischen den beiden Visualisierungsmöglichkeiten ist die Interaktivität. Während im Visualisierungsprogramm die Struktur des Forest of Octrees interaktiv angepasst werden kann, muss in der anderen Visualisierungsmethode zunächst die Struktur im Programmcode angepasst werden, bevor der Forest of Octrees exportiert und anschließend visualisiert werden kann. Ein Vorteil des Visualisierungsprogramms ist zudem, dass es eine grafische Schnittstelle zu der p4est-Bibliothek herstellt, sodass Funktionen selbiger verwendet werden können, ohne Quellcode schreiben zu müssen. Durch die erweiterbare Implementierung des Visualisierungsprogramms ist es zudem unkompliziert möglich, weitere Funktionen zu implementieren.

Ausblick

Die aktuelle Implementierung des Visualisierungsprogrammes enthält Grundfunktionen, die erweitert werden können, um Funktionen zu verbessern oder zusätzliche Funktionen bereitzustellen. Die folgende Auflistung enthält Ideen, die für zukünftige Erweiterungen des Visualisierungsprogramms verwendet werden können:

- Auswahl zwischen Z-Fighting-Strategien

  Um Z-Fighting zu vermeiden, verwendet das Visualisierungsprogramm ein Offset bezüglich der lokalen Koordinaten des hervorgehobenen Oktanten. Diese Vorgehensweise ist

unabhängig von dem Blickwinkel der Darstellung, ist optisch allerdings nicht optimal, da der Versatz, der durch das Offset verursacht wird, sichtbar ist. Die Verwendung anderer Methoden führt zu einer exakteren optischen Darstellung, benötigt allerdings häufigere Neuberechnungen der geometrischen Daten. Eine Option, mit der der Benutzer zwischen verschiedenen Verfahren wählen kann, würde je nach Anwendungsfall eine passend optimierte Darstellung erlauben.

- Erstellung beliebiger Makrostrukturen

  Die grafische Benutzeroberfläche könnte erweitert werden, um die Erstellung von beliebigen Makrostrukturen zu ermöglichen.

- Optionale Berechnung der Nachbarschaftsinformationen

  Die aktuelle Implementierung des Visualisierungsprogramms berechnet automatisch die Nachbarschaftsinformationen des Forest of Octrees. Diese Berechnung hat eine hohe Komplexität und beschränkt die maximale Größe eines darstellbaren Forest of Octrees. Eine Option, diese Berechnung (und damit die Anzeigen der Nachbarschaftsinformationen) zu deaktivieren, würde die Darstellung von komplexeren Octrees erlauben.

- Erweiterung der Hervorhebungsmöglichkeiten

  Die aktuelle Implementierung erlaubt die Hervorhebung eines Oktanten pro Octree mit einer global festgelegten Form bzw. Farbe. Die bestehende Funktionalität könnte erweitert werden, um Oktanten nach bestimmten Kriterien wie deren Level, deren Position innerhalb des Octrees, deren Nachbarschaftsinformationen oder ähnlichem hervorzuheben.

- Verbesserung der dreidimensionalen Anzeige

  Das Visualisierungsprogramm zeigt eine zweidimensionale Projektion des Forest of Octrees. Obwohl durch eine tiefenabhängige Farbanpassung beim Betrachter ein Eindruck der dritten Dimension entsteht, ist es häufig nötig, die Szene zu rotieren, um die dreidimensionale Struktur des Forest of Octrees zu erfassen. Eine tatsächliche dreidimensionale Darstellung des Forest of Octrees würde das Erfassen der dreidimensionalen Struktur deutlich vereinfachen.

  Es gibt verschiedene Möglichkeiten, eine dreidimensionale Darstellung für den Benutzer sichtbar zu machen. Eine Möglichkeit besteht in der Anzeige von Anaglyphenbildern. Werden diese Bilder durch eine Brille mit bestimmten Farbfiltern betrachtet, entsteht ein Eindruck der Tiefenebene der Szene. Der Vorteil dieser Visualisierung besteht in geringen Kosten, da normale Computerbildschirme verwendet werden können und lediglich eine entsprechende Brille angeschafft werden muss. Ein Nachteil dieser Darstellung ist, dass durch die Farbfilter die Qualität der Farbwiedergabe deutlich abnimmt.

  Eine andere Möglichkeit besteht in der Verwendung einer speziellen Brille, die durch eingebaute Bildschirme unterschiedliche Bilder für beide Augen erstellen kann. Die Qualität dieser Darstellung ist der zuletzt genannten Methode deutlich überlegen und

erlaubt zusätzlich durch Tracking der Position bzw. Rotation der Brille eine gewisse Interaktivität.

Literaturverzeichnis

[BBG+09] C. Burstedde, M. Burtscher, O. Ghattas, G. Stadler, T. Tu, L. C. Wilcox. „ALPS: A framework for parallel adaptive PDE solution“. In: *Journal of Physics: Conference Series, Volume 180, Number 1* (2009) (zitiert auf S. 12).

[Ber82] M. J. Berger. „Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations“. Diss. Stanford University, 1982 (zitiert auf S. 11).

[BHH+16] W. Bangerth, T. Heister, L. Heltal, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin. „The deal.II Library, Version 8.3“. In: *Archive of Numerical Software* 4.100 (2016), S. 1–11. DOI: 10.11588/ans.2016.100.23122 (zitiert auf S. 12).

[BHK07] W. Bangerth, R. Hartmann, G. Kanschat. „deal.II – a General Purpose Object Oriented Finite Element Library“. In: *ACM Transactions on Mathematical Software, 33(4):24* (2007) (zitiert auf S. 12).

[BWG11] C. Burstedde, L. C. Wilcox, O. Ghattas. „p4est: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees“. In: *SIAM Journal on Scientific Computing, 33(3), 1103–1133* (2011). DOI: 10.1137/100791634. URL: http://www.p4est.org/ (zitiert auf S. 11, 12, 14, 16).

[BWI14] C. Burstedde, L. C. Wilcox, T. Isaac. „The p4est software for parallel AMR Howto document and step-by-step examples“. In: (2014) (zitiert auf S. 12).

[FB74] R. A. Finkel, J. L. Bentley. „Quad Trees: A Data Structure for Retrieval on Composite Keys“. In: *Acta Informatica 4(1):1-9* (1974). DOI: 10.1007/BF00288933 (zitiert auf S. 14).

[FLS+97] J. E. Flaherty, R. M. Loy, M. S. Shephard, B. K. Szymanski, J. D. Teresco, L. H. Ziantz. „Adaptive Local Refinement with Octree Load-Balancing for the Parallel Solution of Three-Dimensional Conservation Laws“. In: *Journal of Parallel and Distributed Computing, 47(139-152)* (1997) (zitiert auf S. 11).

[GZ99] M. Griebel, G. Zumbusch. „Parallel Multigrid in an Adaptive PDE Solver based on Hashing and Space-Filling Curves“. In: (1999) (zitiert auf S. 11).

[HGS+03] C. E. Hall, M. Gurnis, M. Sdrolias, L. L. Lavier, R. D. Müller. „Catastrophic initiation of subduction following forced convergence across fracture zones“. In: *Earth and Planetary Science Letters 212:15-30* (2003) (zitiert auf S. 11).

[Mea82] D. Meagher. „Geometric modeling using octree encoding“. In: *Computer Graphics and Image Processing, 19(2):129–147* (1982) (zitiert auf S. 14).

[Pop03] S. Popinet. „Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries“. In: *Journal of Computational Physics 190(2):572-600* (2003) (zitiert auf S. 11).

[SSKL13] D. Shreiner, G. Sellers, J. Kessenich, B. Lecea-Kane. *OpenGL Programming Guide, Eighth Edition.* Addison-Wesley, 2013 (zitiert auf S. 22).

Alle URLs wurden zuletzt am 23. 09. 2016 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift