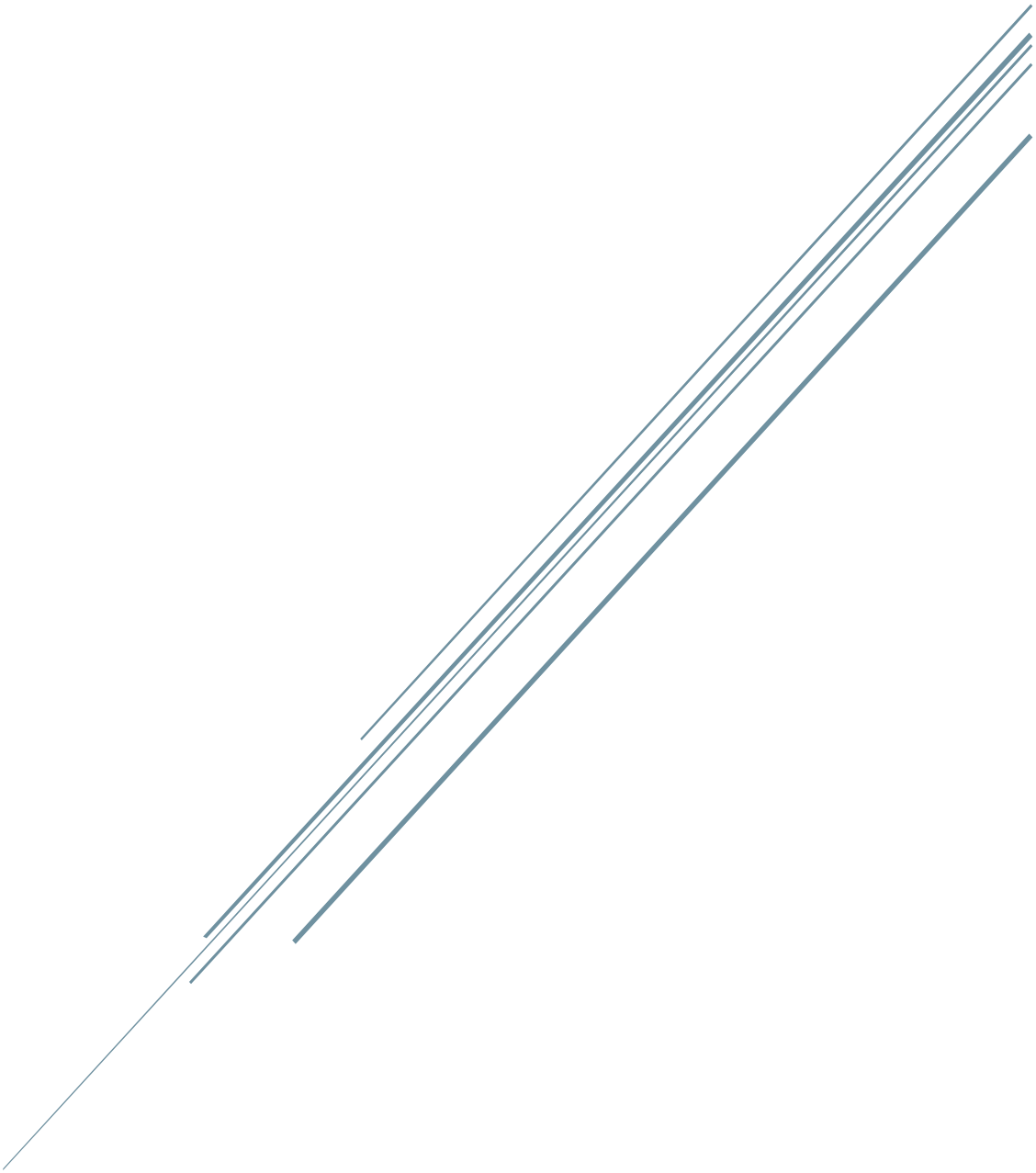
SEMINARVORTRAG

Quadtrees und ihre Anwendungen

Marco Broese, inf104150 SS 20

Ausarbeitung des Seminarvortrages im Rahmen des Seminars zur Zahlentheorie, KI und Eisenbahn von Prof. Dr. Sebastian Iwanowski



FH-Wedel

Seminar zu Zahlentheorie und Eisenbahn

Inhalt

Motivation und Hintergrund zu Quadtree

Die Datenstruktur des Quadtree wurde ursprünglich von Raphael Finkel und Jon Louis Bentley im Jahre 1974 entwickelt. Die Aufgabenstellung, welche zu ihrer Entwicklung führte, war die Suche in 2-dimensionalen Räumen.

Auf Basis dieser Aufgabenstellung wurde die Datenstruktur des Quadtree im Jahre 1983 von Walter Habenicht verallgemeinert. Habenicht entwickelte auf Basis des Quadtree den sogenannten Octree. Diese Datenstruktur ermöglicht eine Verbesserung der Suche in n-dimensionalen Räumen.

Mögliche Anwendungsbeispiele dieser Datenstrukturen sind:

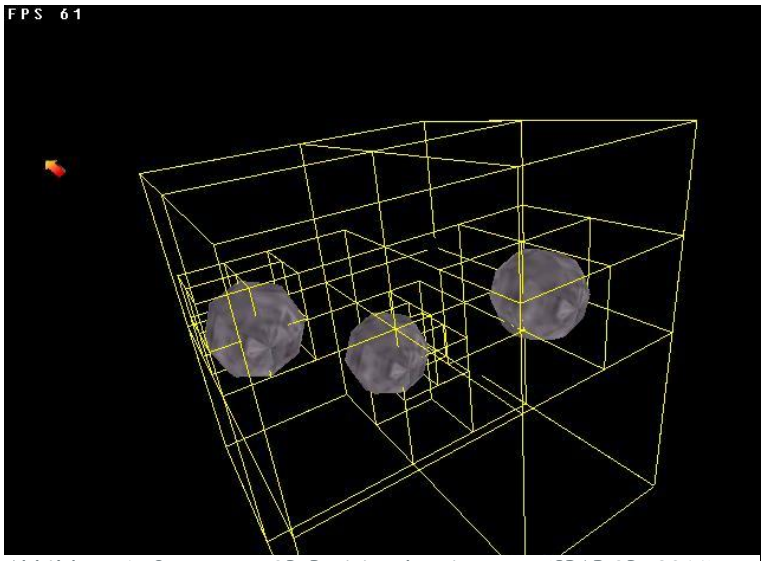


Abbildung 1: Octree zur 3D-Positionsbestimmung (SPAR 3D, 2011)

Abbildung 1:
Zeigt einen Raum mit nur 3 Objekten. Octrees können genutzt werden, um bspw. Raytracing in der Computergrafik effizienter zu implementieren.

Abbildung 2: Karte mit unregelmäßig verteilten Datenpunkten (admin, 2014)

Abbildung 2:
Zeigt eine Karte mit unterschiedlichen Verteilungsdichten von Datenpunkten. Quadtrees können diese unterschiedlichen Verteilungsdichten effizient darstellen und speichern.

Grundlagen

Bäume sind die Grundlage von Quadtrees. Im Folgenden werden die wichtigsten Begriffe erläutert, welche für das Verständnis der restlichen Kapitel benötigt werden.

Baum Begriffe

Ein Baum ist eine Datenstruktur, welche abstrakte Daten in einer hierarchischen Struktur organisiert abbilden kann. (Schramm, 2012)

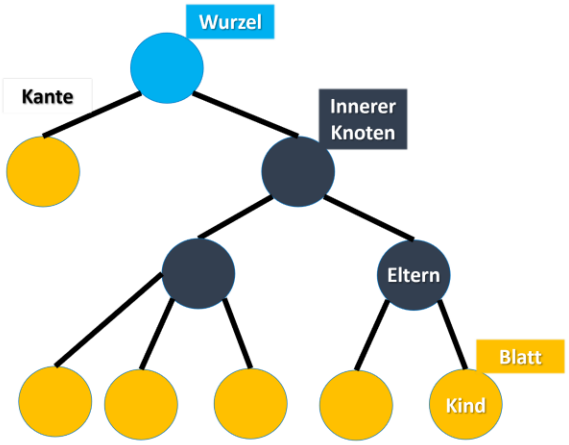


Abbildung 3: Baum Begriffe 1

- Baum: Menge von Knoten und KantenKnoten: ein Objekt im Baum, welches Daten beherbergen kann
- Kante: eine Verbindung zwischen genau zwei Knoten
- Wurzel: der erste Knoten des Baums
- Eltern/Kind: eine Kante verbindet einen Eltern Knoten mit seinem Kind Knoten. Jeder Knoten hat genau einen Eltern Knoten (ausgenommen Wurzel) und beliebig viele Kind Knoten
- Innerer Knoten: ein Knoten, der Kinder hat
- Blatt: ein Knoten ohne Kinder

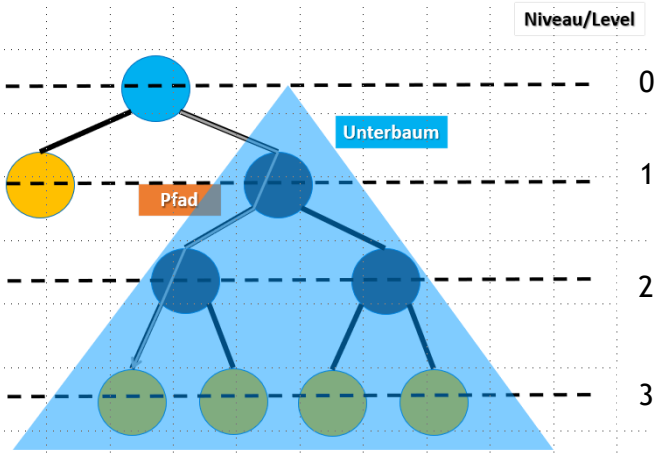


Abbildung 4: Baum Begriffe 2

- Pfad: eine Folge von aufeinander folgenden Knoten und Kanten, welche miteinander verbunden sind

- Niveau/Level: Länge des Pfades von der Wurzel zu diesem Knoten
- Höhe: längster Pfad in einem Baum
- Unterbaum/Teilbaum:
  jeder Knoten kann als Wurzel eines neuen Baums angesehen werden

Anwendungen

Anwendungsgebiete von Baum Strukturen lassen sich im Allgemeinen dort finden, wo hierarchische Strukturen zu modellieren sind.

Anwendungen können sein:

- Familienstammbaum
- Ergebnisse eines Sport-Turniers („KO-System“)
- Gliederung eines Buches
- Gewisse Daten-Strukturen im Rechner
- Arithmetische Ausdrücke
- Spezielle Graphen

Beispiel: Binärbaum

Einer der grundlegendsten Vertreter der Baum Struktur ist der Binärbaum. Anders als bei anderen Baumstrukturen kann ein Knoten in einem Binärbaum höchstens zwei Kinder haben.

Da der Binärbaum einer der grundlegendsten Vertreter der Baumstruktur ist, sind die möglichen Operationen in gleicher oder ähnlicher Form auch bei einem Quadtree zu finden.

Zu den Operationen gehören: (Schramm, 2012)

- insert: einfügen eines neuen Elementes in einen Baum
  insert: Element x Tree → Tree

- remove: entfernen einen Elements aus einem Baum
  remove: Element x Tree → Tree

- empty: erzeugen eines leeren Baumes
  empty: → Tree

- isEmpty: liefert true genau dann, wenn der Baum leer ist
  isEmpty: Tree → boolean

- find: findet ein Element in einem Baum
  find: Element x Tree → Element

Diese Operationen werden in einen späteren Abschnitt von Interesse sein, da verschiedene Arten von Quadtrees bei unterschiedlichen Teilaspekten der Operationen auf Probleme stoßen können.

Quadtree Einführung

Wie schon bei den Binärbäumen erwähnt, haben verschiedene Baum Strukturen verschiedene Charakteristiken.

Bei einem Standard Quadtree wird eine 2-dimensionaler Fläche unterteilt. Diese Fläche wird in vier Abschnitte unterteilt - einmal horizontal und einmal vertikal. Die Unterteilung der Fläche geschieht in ihrer geometrischen Mitte. Die vier daraus entstehenden Flächen werden gemäß den Himmelsrichtungen benannt: NW, NE, SW und SE. Diese so erzeugten Flächen werden auch Zellen genannt.

Jede Zelle eines Quadtree repräsentiert einen Knoten der Baumstruktur, wenn es sich um einen Innerer Knoten handelt, hat dieser exakt vier Kinder.

Hat ein Knoten keine Kinder, kann dieser Daten speichern. Somit sind die Daten in einem Standard Quadtree nur in den Blättern gespeichert. Jedes Blatt hat eine maximale Anzahl an Daten, die in ihm gespeichert werden können. Sobald diese Anzahl überschritten wird, wird das Blatt zu einem Knoten und erhält exakt vier Kinder, in denen die Daten verteilt werde.

Die Daten selbst können von unterschiedlicher Art sein: Punkt, Linie, Fläche, usw.

Durch diese Struktur der Aufteilung repräsentiert der so entstehende Baum die räumliche Zerlegung einer Fläche. (Böhm, WS 2014/15)

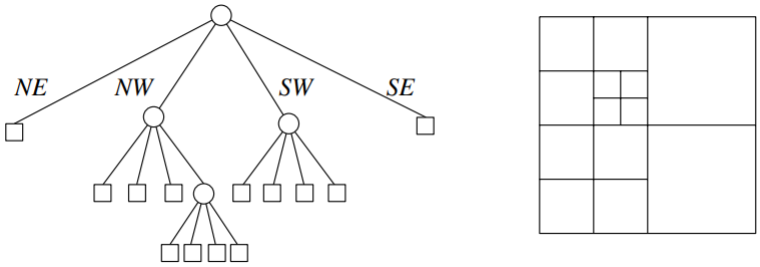


Abbildung 5: Quadtree und zwei seiner geometrischen Repräsentationen (Heide, 2012)

Region Quadtree

Der sogenannte Region Quadtree ist ähnlich zu dem Standard Quadtree. Es gibt zwei entscheidende Unterschiede, welche Region Quadtrees von anderen Quadtrees unterscheiden:

1. ein Blatt kann eine beliebige Anzahl an Daten enthalten (meistens in Form einer Liste)
2. die gesamte Fläche, welche durch eine Zelle definiert wird, wird repräsentiert durch die enthaltenen Daten (bspw. eine Farbe in einem Blatt repräsentiert die Farbe der gesamten Fläche)

Problem

Durch die beliebige Anzahl an Daten, die ein Blatt speichern kann, entsteht das Problem zu entscheiden, ab wann eine Zelle unterteilt werden soll.

Im Allgemeinen gibt es zwei unterschiedliche Lösungsansätze für dieses Problem:

1. Vorgabe einer Maximaltiefe: es wird eine Maximalanzahl an Daten pro Blatt vorgegeben, sobald eine bestimmte Maximaltiefe des Baumes erreicht wurde, werden die Blätter nicht weiter unterteilt und die Daten werden auf der untersten Ebene gespeichert.
2. ein Abbruch-Kriterium auf Basis der gespeicherten Daten, bspw. ein Schwellwert der zwischen den Level nicht unterschritten werden darf

Vor- und Nachteile

Zu den Vorteilen der Region Quadtrees gehören:

- Rasterdaten können gut hinzugefügt werden
- gut zur Darstellung von Karten
- Variable Auflösung / einfaches Zoomen
- Reihenfolge des Einfügens hat keinen Einfluss auf Daten-/Baumstruktur
- einfache Implementierung
- einfaches Löschen von Daten

Zu den Nachteilen der Region Quadtrees gehören:

- abhängig von räumlicher Distanz (z.B. Schriften)
    - Bestimmung von Baumtiefe schwierig
- Bewegung oder Rotation benötigt aufwendige Reorganisation der Baumstruktur
    - Dynamische Bilder schwer darstellbar
- Speicherung von Vektorgrafiken nicht möglich

Beispiele und Anwendungsfälle

Bildkompression

Bei der Bildkompression können die verschiedenen Kompressionsstufen durch einen einzigen Quadtree dargestellt werden. In diesem Verfahren wird die Tiefe des Quadtrees durch zwei Aspekte beeinflusst.
Einerseits durch die Kompressionsstufe, welche die maximale Tiefe darstellt und zum anderen durch einen Schwellwert, welcher den Unterschied der Farben pro Niveau darstellt. Sollte der Wert zwischen zwei Level unter dem Schwellwert liegen, wird diese Zelle nicht weiter unterteilt (Böhm, WS 2014/15).

Abbildung 6: Bildkompression mit Tiefe 2 (Lkjhsdfljsd, 2016)

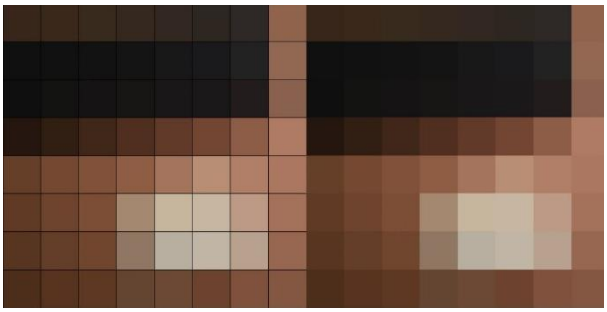


Abbildung 7: Bildkompression mit Tiefe 3 (Lkjhsdfljsd, 2016)

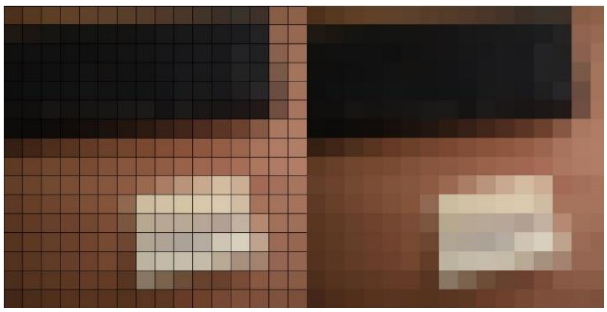


Abbildung 8: Bildkompression mit Tiefe 4 (Lkjhsdfljsd, 2016)

Abbildung 9: Bildkompression mit Tiefe 5 (Lkjhsdfljsd, 2016)

Abbildung 10: Ohne Kompression (Lkjhsdfljsd, 2016)

Punkt Quadtree

Der Punkt Quadtree hat zwei besondere Eigenschaften:

1. die Daten werden in den Knoten und nicht in den Blättern gespeichert
2. die Aufteilung der Zelle erfolgt nicht mehr in vier gleich große Bereiche

Bei einem Punkt Quadtree sind die Daten zumeist 2-dimensionale Punkte, welche gleichzeitig als Schlüssel zur Identifikation dienen. Außerdem sind sie maßgeblich dafür, wie die Struktur des daraus resultierenden Baumes aussieht.

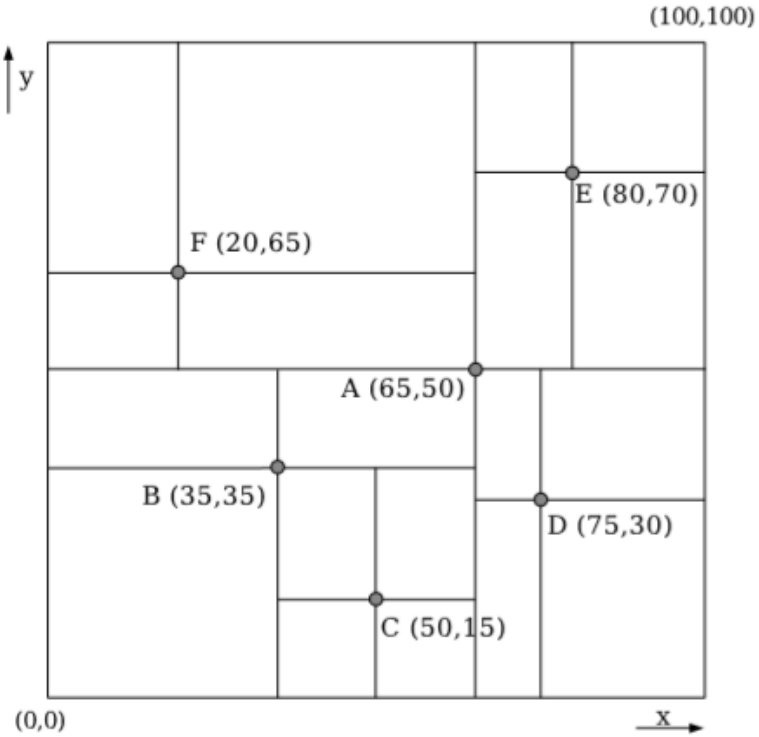


Abbildung 11: Punkt Quadtree Darstellung in der Ebene (Peitsch)

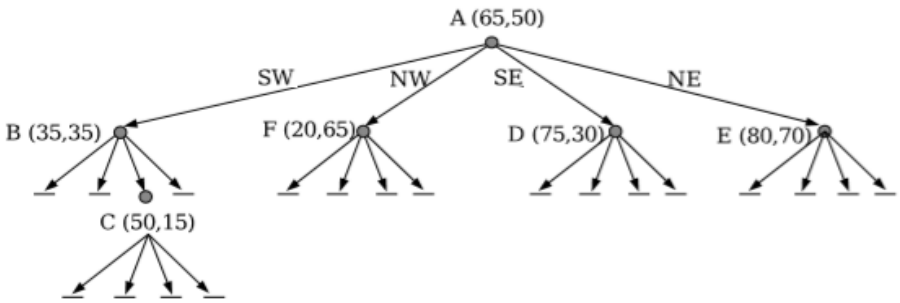


Abbildung 12: Darstellung Punkt Quadtree als Baum (Peitsch)

Operationen

Wie eingangs beschrieben, können unterschiedliche Probleme bei Teilaspekten der Standardoperationen auftreten. Diese Probleme werden im Folgendem verdeutlich und entsprechende Lösungen dargestellt.

Punktsuche

Die Suche in einem Punkt Quadtree ist ähnlich zu der Suche im Binärbaum. Der Startpunkt für eine Suche ist die Wurzel. Um zu entscheiden, welches der vier Kinder entlang gegangen werden muss, sind bei der weiteren Suche jeweils zwei Vergleiche notwendig.
Wie einleitend beschrieben, setzen sich die Schlüssel der Knoten aus ihren beiden Koordinaten zusammen. Somit ist es notwendig die Koordinaten zu vergleichen, um den geeigneten Unterbaum für die weitere Suche zu finden.

Einmal muss anhand der X-Koordinate entschieden werden, ob die nördlichen (NW, NO) oder ob die südlichen Unterbäume (SW, SO) von Interesse sind.
Danach muss anhand der Y-Koordinate entschieden werden, ob die Westlichen oder Östlichen von Interesse sind.

Dieses Prinzip wird solange fortgesetzt, bis der gesuchte Knoten gefunden ist oder ein Blatt erreicht wurde, in diesem Fall ist das gesuchte Element nicht im Baum vorhanden.

Einfügen

Das Einfügen in einen Punkt Quadtree ist vom Prinzip her sehr ähnlich zu der gerade beschriebenen Punktsuche. Im Prinzip wird anhand der X-Koordinate und der Y-Koordinate dasjenige Blatt herausgesucht, welches der Position des neu einzufügenden Knoten am ehesten entspricht. Sobald das Blatt gefunden wurde, wird an dieser Position das Blatt durch einen Knoten mit vier Blättern ersetzt.

Das Einfügen von neuen Knoten in ein Punkt Quadtree hat jedoch das Problem, dass die Einfüge-Reihenfolge von Relevanz ist. Je nachdem in welcher Reihenfolge Knoten zum Baum hinzugefügt werden, entstehen unterschiedliche Baumstrukturen. Ein Problem dieser so erzeugten Baumstrukturen kann sein, dass sie extrem unbalanciert werden, was die Vorteile der Baumstruktur zunichtemachen kann.

Löschen

Wie schon vorher erwähnt, ist die Struktur des Baumes abhängig von der Reihenfolge der eingefügten Knoten. Diese Eigenschaft ist besonders bedeutend beim Löschen von Knoten. Soll ein Knoten gelöscht werden, der mehr als ein Kind hat, welches kein Blatt ist, muss entschieden werden, welches der Kinder ein optimaler Kandidat ist, um den gelöschten Knoten in der Baumstruktur zu ersetzen.

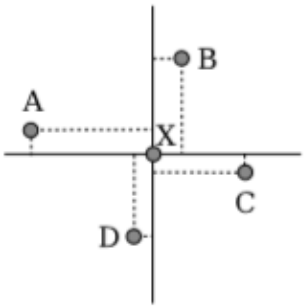


Abbildung 13: Quadtree mit keinem idealen Kandidaten zum Ersetzen von X (Peitsch)

Kriterien zur Bestimmung eines optimalen Kandidaten:

1. es existiert kein Knoten im Fenster zwischen den Kandidaten und dem zu löschenden Knoten
2. Kandidat mit minimalem Abstand

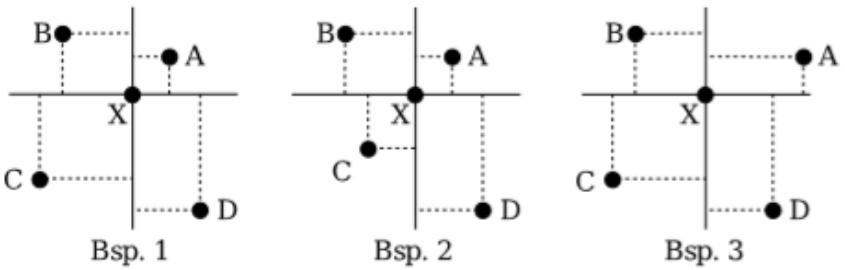


Abbildung 14: Beispiele für Kandidatenwahl (Peitsch)

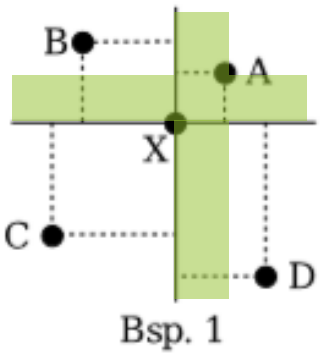


Abbildung 15: Eingezeichnetes Fenster zwischen X und A

Im ersten Beispiel lässt sich der Kandidat A, durch das erste Kriterium als der optimale Kandidat bestimmen. Nur bei Kandidat A ist kein anderer Knoten im eingezeichneten Fenster.

Im zweiten Beispiel erfüllen die Kandidaten C und A das erste Kriterium. Mit Hilfe des zweiten Kriteriums - dem minimalen Abstand, wird der Kandidat A als optimaler Kandidat bestimmt.

Im dritten Beispiel wird das erste Kriterium von keinem Kandidaten erfüllt, somit folgt die Auswahl nur über das zweite Kriterium dem minimalen Abstand. Daraus folgt, dass Kandidat B der optimale Kandidat ist. In diesem Fall muss ein weiterer Unterbaum erzeugt werden (mit D als Kind von B und A als Kind von D).

Bereichssuche

Neben der einfachen Suche erlauben die Punkt Quadtrees die Möglichkeit der Bereichssuche. Dabei werden beispielsweise auf einer Karte der Mittelpunkt eines Kreises und sein Radius angegeben (es sind auch anderen Strukturen denkbar).

Es sollen alle Knoten angegeben werden, die im Inneren des definierten Kreises liegen, dieses stellt eine Standardaufgabenstellung im 2- oder 3-dimensionalen Raum dar.

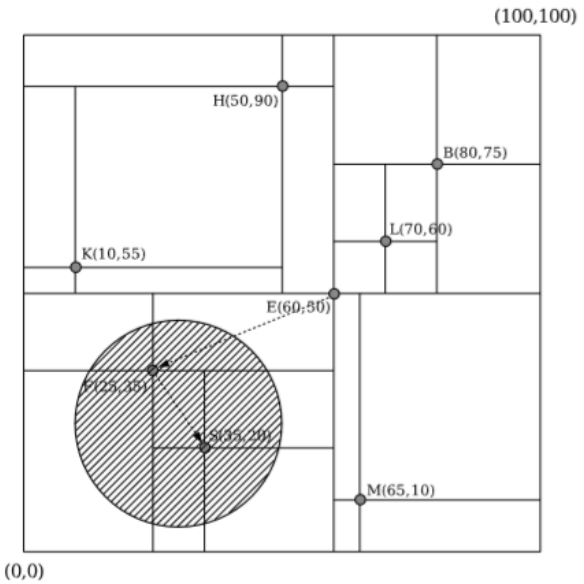


Abbildung 16: Bereichssuchbeispiel (Peitsch)

In diesem Beispiel sollen alle Punkte gefunden werden, die innerhalb des Kreises um den Punkt (25,30) mit dem Radius 20 liegen:

1. Die Suche beginnt bei der Wurzel E (60,50). Da der Kreis komplett in der südwestlichen Zelle liegt, können die anderen 3 Zellen ignoriert werden.
2. Der Knoten F (25,35) ist innerhalb des Radius des Kreises, somit ist F Teil des Ergebnisses. Es müssen alle 4 Kinder von F überprüft werden. F hat nur ein Kind (S 35,30), welches kein Blatt ist. Daher muss nur dieses weiter überprüft werden.
3. Der Knoten S (35,20) ist auch innerhalb des Radius und somit Teil des Ergebnisses. Alle 4 Kinder des Knoten S müssen nun untersucht werden., Da es sich bei den Kindern von S um Blätter handelt, müssen diese nicht weiter geprüft werden.
4. Da kein weiterer Knoten betrachtet werden muss, ist die Suche beendet und als Ergebnis wurden die Knoten F und S innerhalb des gegebenen Kreises gefunden.

Vor- und Nachteile

Zu den Vorteilen der Punkt Quadtrees gehören:

- Rasterdaten können gut hinzugefügt werden
- gut zur Darstellung von Karten
- einfache und schnelle Bereichssuche

Zu den Nachteilen der Punkt Quadtrees gehören:

- Einfüge-Reihenfolge hat Einfluss auf Baumstruktur
- Löschen ist aufwendig
- Gefahr von unbalancierten (entarteten) Bäumen
    - Vorteil der Suche in Bäumen kann dadurch verloren gehen

Speicherung von Geometrie

Problem

Die Speicherung von Geometrien birgt ein entscheidendes Problem. Bei allen vorgestellten Strukturen werden nur einzelne Punkte gespeichert und jeder Punkt hat eine exakte Zugehörigkeit zu einer Zelle, er kann nie in zwei unterschiedlichen Zellen liegen.

Bei Geometrien tritt das Problem auf, dass sie eine Ausdehnung haben und somit über mehrere Zellen verteilt liegen können wie in Abbildung 17 dargestellt. Um dieses Problem zu lösen, werden im Folgenden drei einfache und ein komplexerer Lösungsansatz vorgestellt.

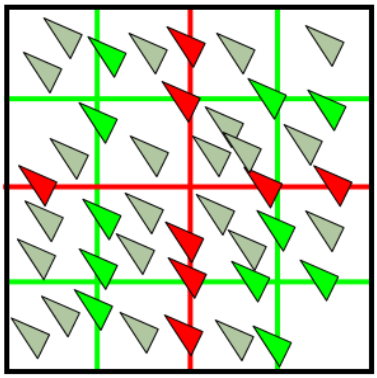


Abbildung 17: Polygone über mehrere Zellen (Heide, 2012)

Lösungsansätze

1. Polygon aufteilen

   Im ersten Ansatz wird ein Polygon entlang der Zellgrenzen aufgeteilt, sodass die einzelnen Teilpolygone nur noch in einer einzigen Zelle liegen.

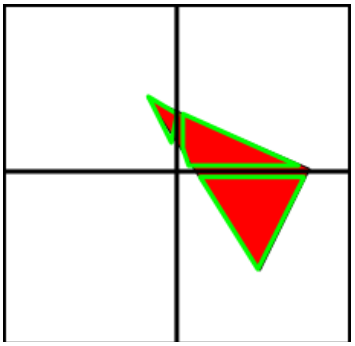


Abbildung 18: Polygon in drei Polygone aufgespaltet (Heide, 2012)

Nachteil: Die Polygon Anzahl erhöht sich und das Verarbeiten des Originalpolygons wird schwieriger, da jetzt mehrere Polygone verarbeitet werden müssen.

Beispiel: Verschiebung eines Polygons:

1. Zusammensetzen des Originalpolygons
2. Verschiebung des Originalpolygons
3. Erneute Aufspaltung in neue Teilpolygone

2. Polygon redundant speichern

Im zweiten Ansatz wird das Polygon redundant abgespeichert, sodass in jeder überdeckten Zelle eine Referenz auf das Polygon abgespeichert wird.

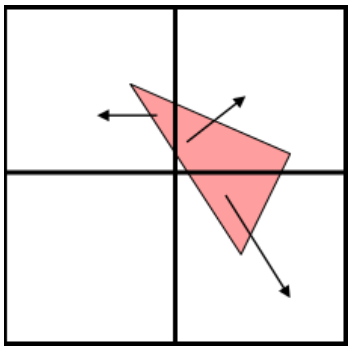


Abbildung 19: Polygon redundant speichern (Heide, 2012)

Nachteil: Erhöhter Verwaltungsaufwand, denn es müssen immer alle Referenzen aktualisiert werden.

3. Polygon in einer nicht unterteilten Zelle speichern

Im dritten Ansatz wird das Polygon in dem Knoten abgespeichert, in dem es komplett enthalten ist, also wo das Polygon durch keine Kante unterteilt wird.

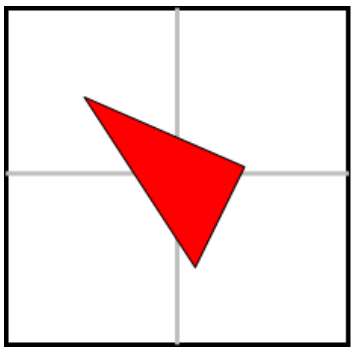


Abbildung 20: Polygon höher im Baum speichern (Heide, 2012)

Nachteil: Viele Polygone liegen höher in der Baumstruktur als es ihrer Größe entspricht.

Loose Quadtree

Neben den bisher beschriebenen simplen Lösungsansätzen gibt es noch weitere, komplexere Lösung. Im Folgendem wird der Ansatz der so genannten Loose Quadtree dargestellt.

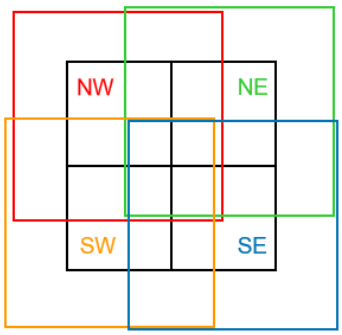


Abbildung 21: Darstellung eines Loose Quadtrees (Heide, 2012)

Die Idee hinter dem Loose Quadtree ist es, Polygone/Objekte so tief wie möglich und so hoch wie nötig in der Baumstruktur zu speichern.

Bei den Loose Quadtrees ist eine Zelle größer als in einem normalen Quadtree, sodass sich die Zellen überschneiden (wie in Abbildung 21 dargestellt).
Die Zellzugehörigkeit eines Objektes wird anhand des Objektmittelpunktes ermittelt. Sollte ein Objekt mit komplettem Umfang innerhalb einer Zelle eines Loose Quadtrees liegen, wird es dieser Zelle zugeordnet. Dieses Verfahren wird soweit fortgesetzt, bis es keinen Loose Quadtrees mehr gibt, der das Objekt komplett beinhaltet (beispielhaft in Abbildung 22 zu sehen).

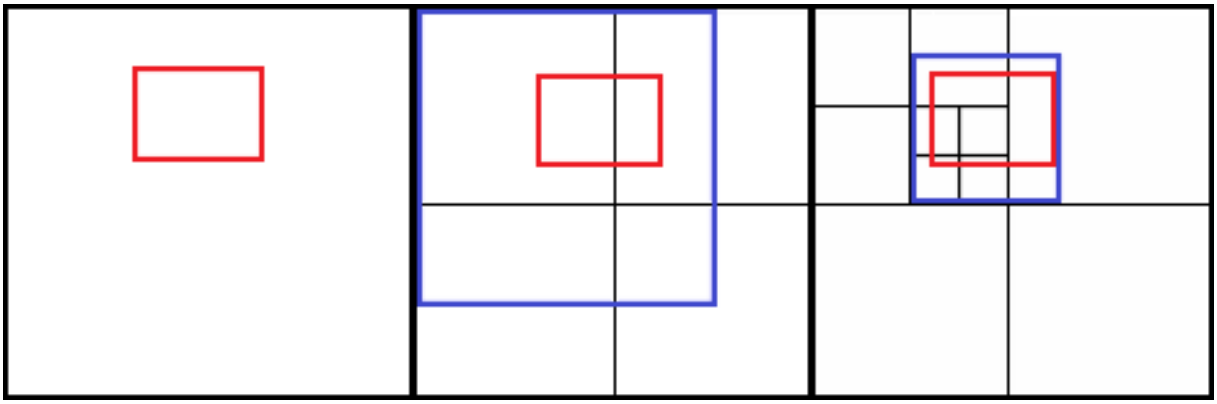


Abbildung 22: Einsortieren in einen Loose Quadtree (Heide, 2012)

Vor- und Nachteile

Zu den Vorteilen der Loose Quadtrees gehören:

- sehr gut geeignet für Objekte von unterschiedlichen Größen
- deutlich weniger Insert/Delete Operationen als bei den einfachen Lösungen
    - bessere Performance

Zu den Nachteilen der Loose Quadtrees gehören:

- schlechtere Performance von Suchanfragen
    - es werden mehr Objekte geprüft

Anwendungsbeispiel: Netzgenerierung mit Quadtrees

Ein Anwendungsgebiet in dem Quadtrees zum Einsatz kommen, ist die Netzgenerierung. Dieses Gebiet ist von äußerstem Interesse, da heutzutage fast alle elektronischen Geräte zur Steuerung und Kontrolle elektrischen Schaltkreis beherbergen.
Diese Schaltkreise werden üblicherweise auf Leiterplatten platziert. Um eine Leiterplatte zu entwerfen, muss entschieden werden, wo die verschiedenen Komponenten platziert und wie sie verbunden werden. Dadurch entstehen einige geometrische Probleme, welche mittels der Netzgenerierung mit Quadtrees gelöst werden können.

Ein Standardproblem beim Design der Leiterplatten ist, dass die elektronischen Komponenten Hitze ausstrahlen. Für eine einwandfreie Funktion muss die Hitze unterhalb eines bestimmten Schwellwerts liegen. Beim Design von Leiterplatinen ist es schwierig, Aussagen darüber zu treffen, ob die Hitze-Verteilung den Anforderungen entspricht. An diesem Punkt setzt die Netzgenerierung mit Quadtrees an.

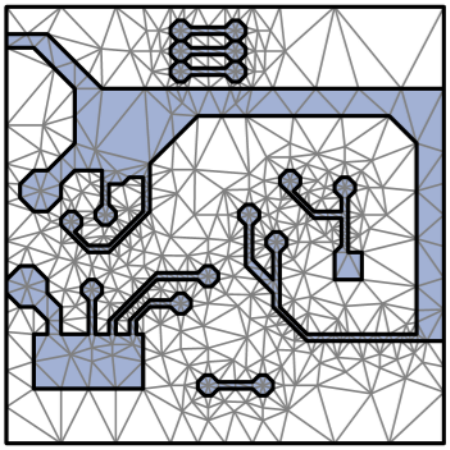


Abbildung 23: Dreiecknetz einer Leiterplatte (Berg, 2008)

Das Ziel ist es, ein Dreiecksnetz der Leiterplatte zu generieren, auf Basis dessen eine Bestimmung der Hitze-Verteilung erfolgen kann.

Das Netz muss bestimmte Eigenschaften aufweisen:

1. Conforming
   - ein Dreieck darf keinen Scheitelpunkt eines weiteren Dreiecks im Inneren seiner Kanten haben
2. Respect the input
   - die Kanten der Komponenten müssen von den Kanten der Dreiecke überdeckt werden
3. Well-shaped
   - die Winkel jedes Dreiecks dürfen nur zwischen 45° - 90° liegen
4. Non-Uniform
   - es muss eine adaptive Netzgröße vorliegen

Begriff: Uniform und Non-Uniform Mesh

In einem Uniform Mesh, wie in Abbildung 24 dargestellt, sind alle Dreiecke von der gleichen Größe, sodass sie ein einheitliches Netz bilden. Infolgedessen hat das Netz sehr viele Dreiecke.

Bei einem Non-Uniform Mesh sind die Dreiecke nicht alle gleich groß. Ein einfaches Ersetzen von vielen kleinen Dreiecken durch ein einziges großes, würde bei der Netzgenerierung jedoch zu einem Verstoß der Conforming Eigenschaft führen. Um ein Non-Uniform Mesh unter den zuvor beschriebenen Eigenschaften zu generieren, wird die Dreiecksgröße schrittweise angehoben.

Wie in Abbildung 24 zu sehen ist, hat ein Uniform Mesh deutlich mehr Dreiecke als ein Non-Uniform Mesh (512 zu 52), sodass durch den Einsatz eines Non-Uniform Meshes ein großer Performance Gewinn erzielt werden kann.

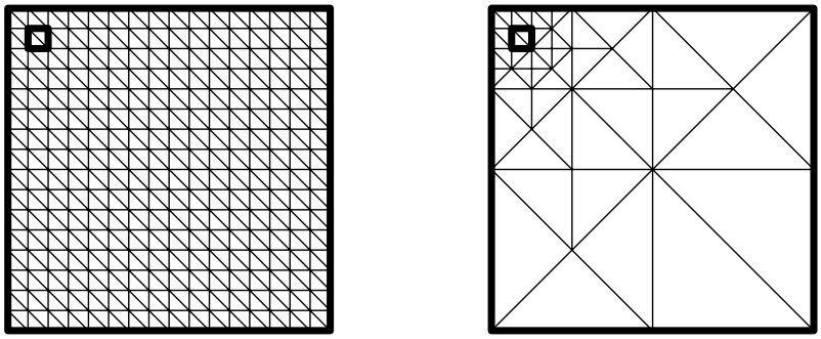


Abbildung 24: Uniform Mesh(links) Non-Uniform Mesh(rechts) (Berg, 2008)

Begriff: Balancierter Quadtree

Um bei der Netzgenerierung die Triangulierung zu vereinfachen, wurde das Konzept des balancierten Quadtree eingeführt.

Die Unterteilung eines Quadtree wird als ausgeglichen (ausgewogen) bezeichnet, wenn zwei benachbarte Quadrate sich in Ihrer Größe maximal um einen Faktor 2 unterscheiden. Wenn der Faktor größer als 2 ist, wird das größere Quadrat weiter unterteilt.

Ein Quadtree wird als balanciert bezeichnet, wenn alle seiner Unterteilungen ausgeglichen sind (siehe Abbildung 25).

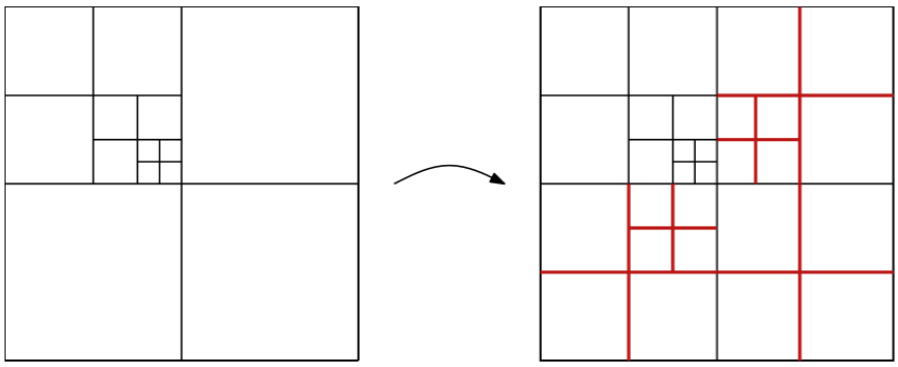


Abbildung 25: Quadtree und seine balancierte Form (Berg, 2008)

Beispiel

Im folgenden Beispiel soll ein Mesh für eine simple Leiterplatte erzeugt werden.

Die Abbildung 26 zeigt die verschiedenen Quadtrees, die im Laufe der Erzeugung des Meshes entstehen. Jede Grafik zeigt eine weitere Verfeinerung des Quadtrees. Die Abbildung 27 zeigt den dazugehörigen balancierten Quadtree.

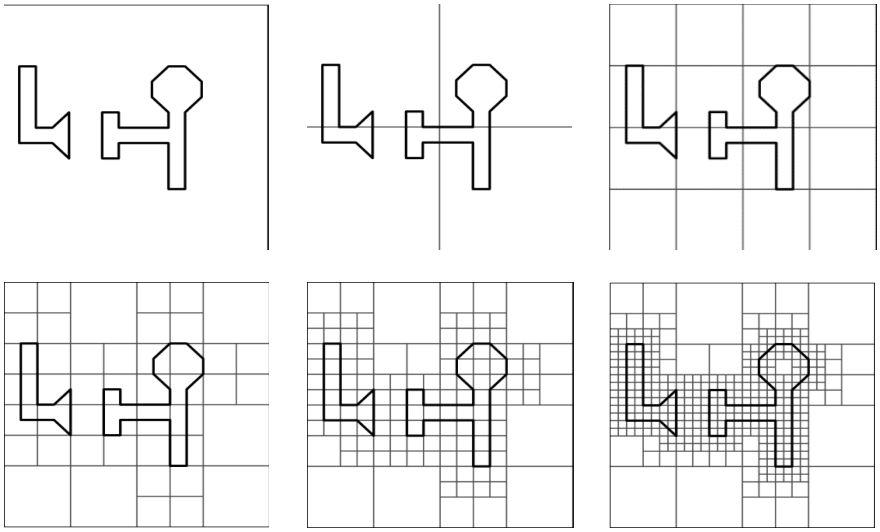


Abbildung 276: Quadtree Verfeinerung (Nöllenburg, 2012)

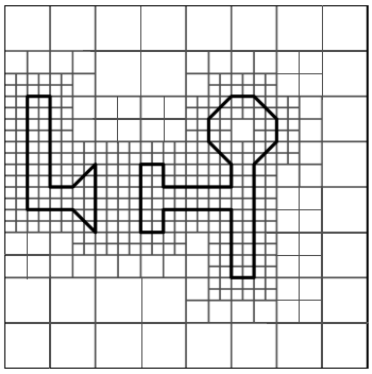


Abbildung 267: Balancierter Quadtree (Nöllenburg, 2012)

Der letzte Schritt zum Erzeugen des Meshes besteht in der Triangulierung der Quadrate.

Dafür gibt es drei verschiedene Ansätze:

1. Diagonale in jedes Viereck einfügen
   - verletzt das conforming Kriterium (siehe Abbildung 28)
2. Nutzung der Unterteilungsknoten für Triangulierung
   - verletzt well-shaped Kriterium (siehe Abbildung 29)
3. Einführung von Steinerknoten
   - erfüllt alle Kriterien (siehe Abbildung 30)

Bei einem Steinerknoten werden alle angrenzenden Knoten von einem Quadrat zu einem neu erzeugten Knoten in der Mitte zusammengeführt.

Auf dem so erzeugtem Mesh, kann die Simulation der Hitze erfolgen und das Design der Leiterplatte überprüft werden (Dreiecks Mesh beispielhaft siehe Abbildung 30).

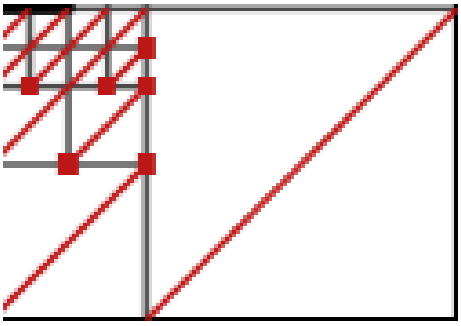


Abbildung 298: Triangulierung durch Diagonalen einfügen (Nöllenburg, 2012)

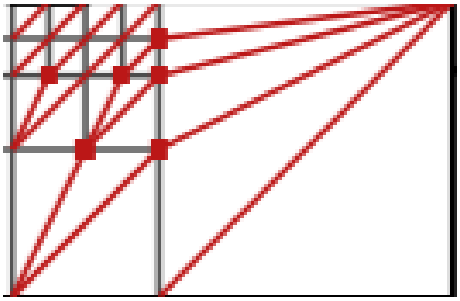


Abbildung 289: Triangulierung durch Nutzung von Unterteilungsknoten (Nöllenburg, 2012)

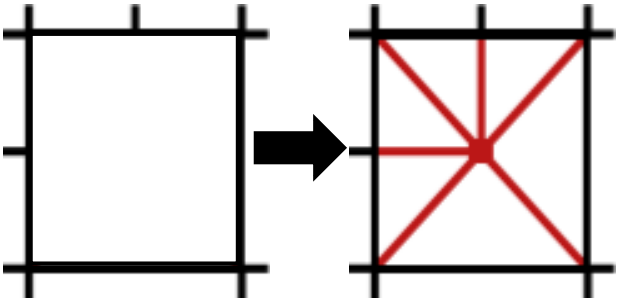


Abbildung 30: Triangulierung einfügen von Steinerknoten (Nöllenburg, 2012)

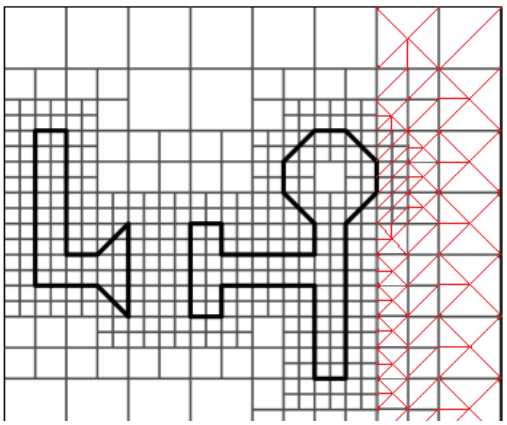


Abbildung 30: Beispielhafte Triangulierung des Meshes

Literaturverzeichnis

admin. (8. Dezember 2014). *Phase2*. Von www.phase2technology.com: https://www.phase2technology.com/blog/using-d3-quadtrees abgerufen

Berg, P. D. (2008). *Computational Geometry: Algorithms and Applications*. Springer-Verlag.

Böhm, P. D. (WS 2014/15). *Ludwig-Maximilians-Universität München*. Von Vorlesung Geo-Informationssysteme: https://www.dbs.ifi.lmu.de/Lehre/GIS/WS1415/Skript/GIS_WS14_04_part3.pdf abgerufen

Heide, P. D. (24. April 2012). *www.hni.uni-paderborn.de*. Von www.hni.uni-paderborn.de: https://www.hni.uni-paderborn.de/fileadmin/_migrated/content_uploads/algocgOctree.pdf abgerufen

Lkjhsdfljsd. (11. März 2016). *Quadtree*. Von wikimedia.org: wikipedia.org/wiki/Quadtree abgerufen

Nöllenburg, M. (19. Juni 2012). *https://i11www.iti.kit.edu/*. Von https://i11www.iti.kit.edu/: https://i11www.iti.kit.edu/_media/teaching/sommer2012/compgeom/algogeom-ss12-vl10.pdf abgerufen

Nunez, S. (13. März 2017). *Map Clustering with QuadTrees using GoogleMaps*. Von AGOSTINI.TECH : https://agostini.tech/2017/03/13/map-clustering-with-quadtrees-using-googlemaps/ abgerufen

Peitsch, T. (kein Datum). *dbs.uni-leipzig*. Von dbs.uni-leipzig: https://dbs.uni-leipzig.de/file/Punkt-Quadtree.pdf abgerufen

Schramm, P. D. (20. Dezember 2012). *Algorithmen & Datenstrukturen*. Von services.informatik.hs-mannheim.de: http://services.informatik.hs-mannheim.de/~schramm/ads/files/Kapitel10_01.pdf abgerufen

SPAR 3D. (20. Dezember 2011). *Octreebig*. Von SPAR3D: https://www.spar3d.com/news/software/urban-robotics-extends-octree-format-to-point-cloud-library-open-source-community/ abgerufen

Abbildungsverzeichnis