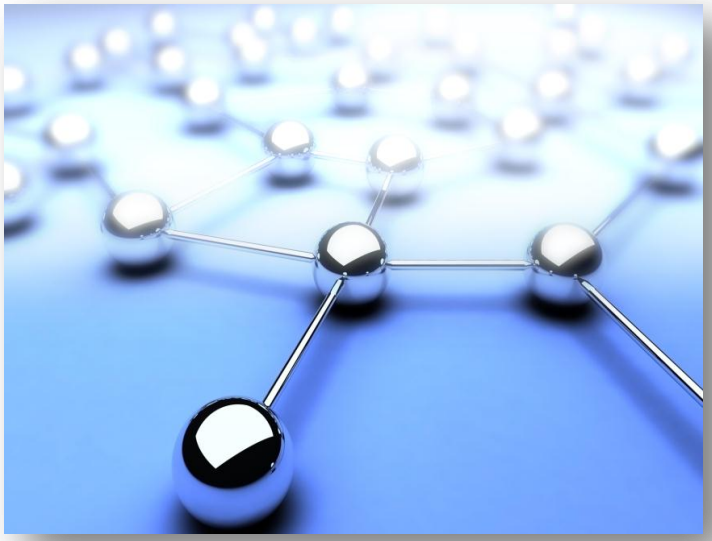
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Vorlesung Algorithmen für hochkomplexe Virtuelle Szenen

Sommersemester 2012



Matthias Fischer
mafi@upb.de

Vorlesung 5
24.4.2012

Quadtrees/Octrees
Übersicht

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Quadtrees / Octrees

- Motivation
- Quadtrees für Punktmengen
- Speicherung von Dreiecken
- Eigenschaften
- Aufbau eines Quadtrees
- Suche im Quadtree
- Balancierung

Loose-Octree

- Einfügen und Partitionieren im Octree
- Einsortierung
- Konstruktion
- Größe der Loose-Octreebox
- Direkte Platzierung
- Frustum Culling

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Quadtrees / Octrees

Computational Geometry - Algorithms and Applications;
Mark de Berg, Marc de Kreveld, Mark Overmars;
Springer Verlag, 2000.

Kapitel: Quadtrees

Loose Octrees

Mark DeLoura
Game Programming Gems
Kap. 4.11 Loose Octrees, von Thatcher Ulrich
Charles River Media, 2000

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Quadtrees/Octrees

Datenstruktur von Finkel und Bentley, 1974

Ziel

- Räumliche Sortierung
- Bereichsanfragen

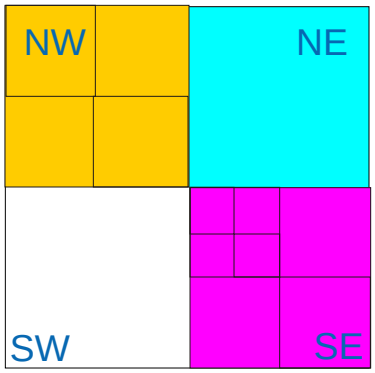
Anwendungen

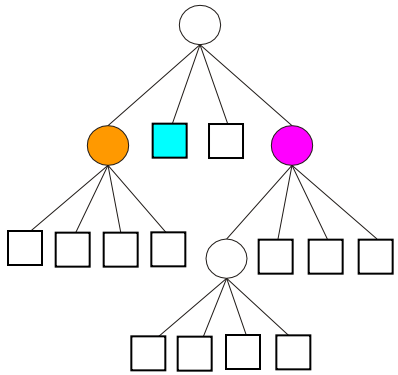
- Finite Elemente Methode (Erzeugung von hochkomplexen Netzen)
- Meshes
- Verdeckungsberechnung

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Quadtree

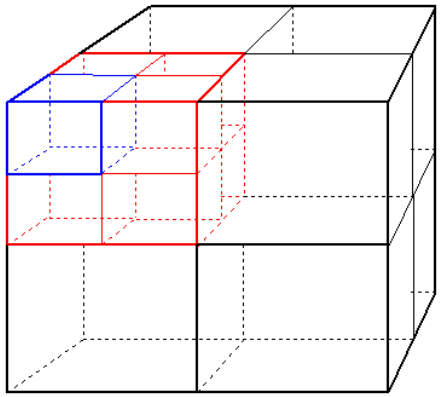
- 2-dimensional
- jeder Knoten korrespondiert mit einem Quadrat (Octreezelle, Quadrant)
- innere Knoten haben vier Kinder
- alle Quadrate in derselben Ebene sind gleich groß

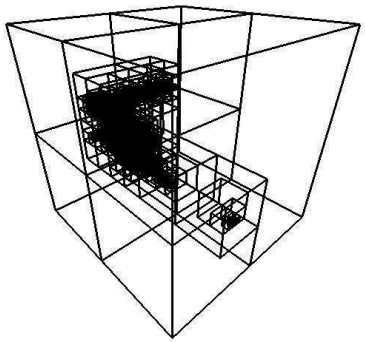




Octree

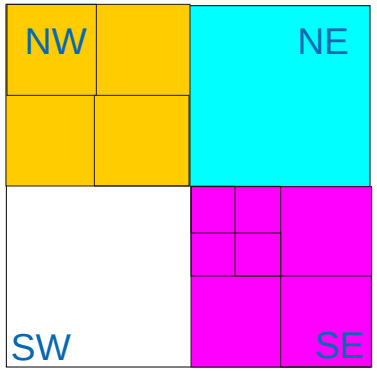
- 3-dimensional
- jeder Knoten korrespondiert mit einem Würfel (Octreezelle, Oktant)
- innere Knoten haben acht Kinder
- alle Würfel in derselben Ebene sind gleich groß





HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Welche Fragen stellen sich?



Wie tief nehmen wir die Unterteilung vor?

- feste maximale Tiefe
- so tief gehen, bis in jeder Zelle < c Elemente liegen

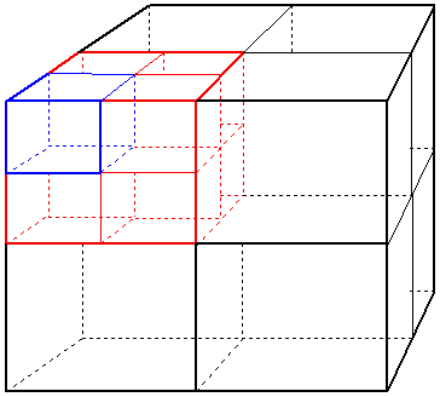
Wo speichern wir die Elemente?

- in den Blättern
- in Blättern und inneren Knoten (bspw. bei Dreiecken)

Wie speichern wir Punkte?

- in den Zellen, in denen sie liegen

Wie speichern wir Dreiecke? .....



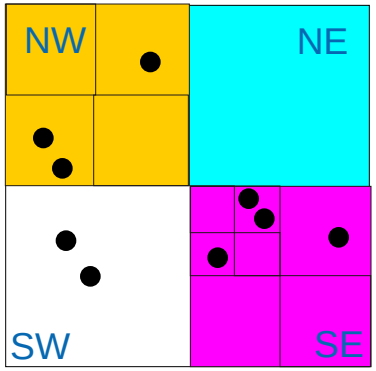
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

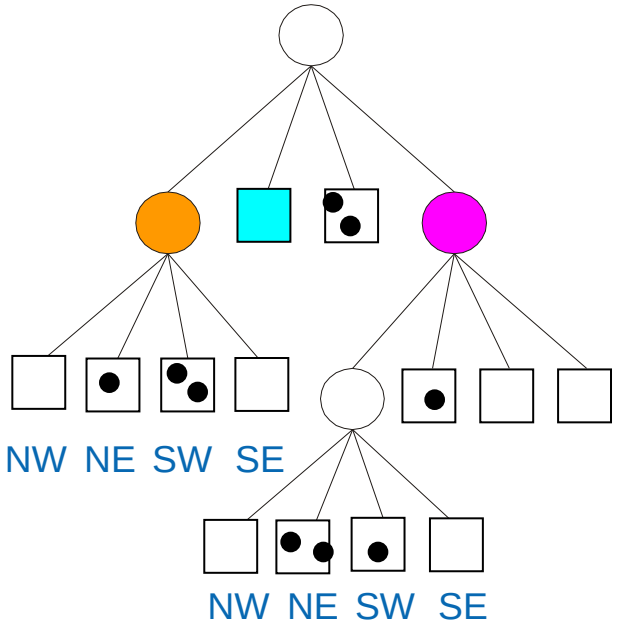
Aufbau eines Quadtrees für Punktmengen

Eingabe: Eine Menge P von n Punkten

Aufbau:

- Jeder Punkt wird in genau einem Quadrat gespeichert,
- das Quadrat ist ein Blatt des Quadtrees,
- die Blätter enthalten maximal k Punkte (im Beispiel rechts k=2),
- innere Blätter speichern nur das Quadrat, aber keine Punkte.






HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Rekursive Definition Quadtree (Octree analog)

Sei $\sigma := [x_\sigma : x'_\sigma] \times [y_\sigma : y'_\sigma]$ das oberste Quadrat für die Punktmenge P

Falls $|P| \leq 1$

- Der Quadtree besteht aus einem Blatt, an dem σ und P gespeichert sind

Falls $|P| \geq 1$

- Seien $\sigma_{NE}, \sigma_{NW}, \sigma_{SW}, \sigma_{SE}$, die vier Quadranten von σ und sei

$$x_{mid} := \frac{(x_\sigma + x'_\sigma)}{2}, y_{mid} := \frac{(y + y'_\sigma)}{2}$$

- Wir definieren:

$$P_{NE} := \{p \in P | p_x > x_{mid} \wedge p_y > y_{mid}\}, \quad P_{NW} := \{p \in P | p_x \leq x_{mid} \wedge p_y > y_{mid}\}$$
$$P_{SW} := \{p \in P | p_x \leq x_{mid} \wedge p_y \leq y_{mid}\}, \quad P_{SE} := \{p \in P | p_x > x_{mid} \wedge p_y \leq y_{mid}\}$$

- Der Quadtree besteht aus einem Knoten v, in dem σ gespeichert ist
- Knoten v hat vier Kinder:
- Das NE-Kind ist die Wurzel eines Quadtree der Punktmenge P_{NE} im Quadrat σ_{NE}

Das NW-Kind	-- “” --	P_{NW} im Quadrat σ_{NW}
Das SW-Kind	-- “” --	P_{SW} im Quadrat σ_{SW}
Das SE-Kind	-- “” --	P_{SE} im Quadrat σ_{SE}

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Algorithmus Quadtree (Octree) Aufbau

1. Beginne mit einer Zelle (Quadrat/Würfel), die alle Punkte enthält, bspw. Boundingbox von allen Punkten.
2. Unterteile das Quadrat (Würfel) in vier (acht) gleich große Quadrate (Oktanten) und verteile alle Punkte in die Quadrate (Oktanten).
3. Unterteile rekursiv die Quadrate (Oktanten) bis maximal k Punkte in einem Quadrat (Würfel) erreicht sind.

   Alternatives Abbruchkriterium ist die Beschränkung der Baumtiefe; hängt von der Anwendung ab.

In praktischen Anwendungen ist der Wert von k entscheidend für die Laufzeit des Algorithmus, der die Datenstruktur anwendet (Renderingalgorithmus) und liegt typischerweise deutlich über 1.

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Wie hängen Tiefe des Quadtrees und die Punktmenge voneinander ab?

Wir können Größe (Anzahl der Knoten) und Tiefe des Quadtrees nicht durch die Anzahl Punkte beschränken.

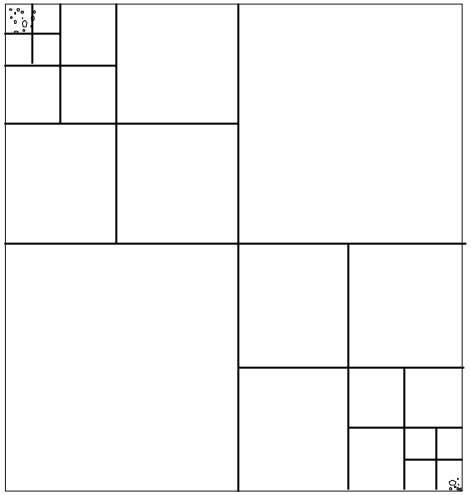
Es gilt die folgende Aussage:

- Die Tiefe eines Quadtrees für eine Punktemenge P in der Ebene ist höchstens $\log\left(\frac{s}{c}\right) + \frac{3}{2}$

Dabei ist

- c die minimale Entfernung zwischen zwei Punkten und
- s die Seitenlänge des äußersten Quadrats.

Warum?



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Weil …

Seitenlänge des Quadrats eines Knoten in Tiefe i ist $\frac{s}{2^i}$

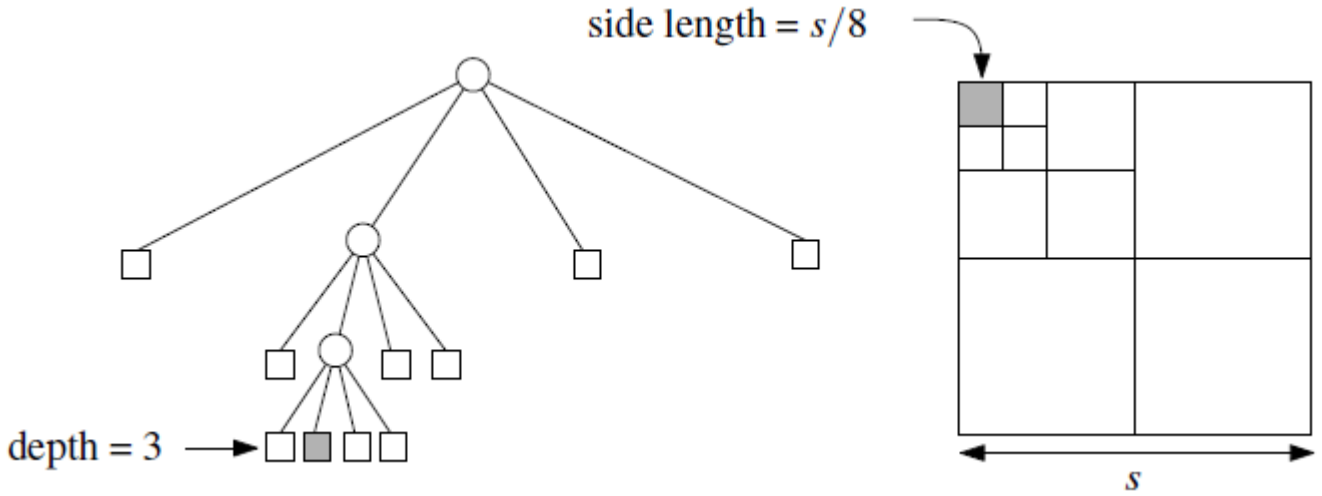
→ Maximale Entfernung zweier Punkte in einem Quadrat ist $\sqrt{2}\frac{s}{2^i}$

Interne Quadrate umfassen mind. 2 Punkte, die mind. c voneinander entfernt sind

→ für jedes interne Quadrat auf Tiefe i gilt: $\sqrt{2}\frac{s}{2^i} \geq c$

→ $\mathrm{i} \leq \log\left(\frac{s}{c}\right) + \frac{1}{2}$

→ die Schranke folgt, da die Tiefe des Quadtrees um eins tiefer ist, als die maximale Tiefe eines internen Knotens



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Was bedeutet die Anhängigkeit von der Tiefe des Baumes für ungünstige Punkteanordnungen?

→ Quadtrees lassen sich nicht so balancieren,
wie man es von anderen Baumstrukturen gewohnt ist

Was kostet der Aufbau eines Quadtrees mit n Punkten?

Ein Quadtree mit n Punkten und Tiefe d hat

- $O((d+1)n)$ Knoten und kann in
- Zeit $O((d+1)n)$ aufgebaut werden

Warum?

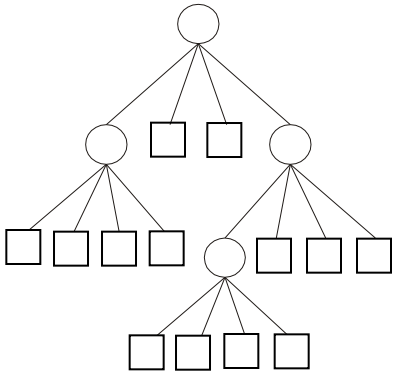
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Weil ….

Zu 1.)

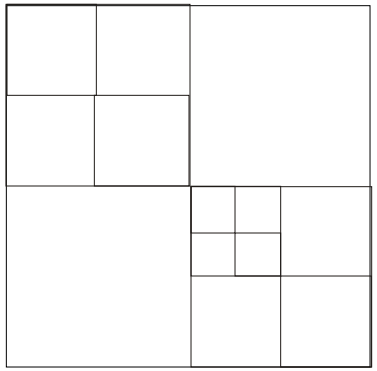
Jeder innere Knoten hat vier Kinder

→ Anzahl Blätter = 1+ 3 x Anzahl innerer Knoten



Wir betrachten nur die inneren Knoten,
über die Anzahl innerer Knoten wissen wir:

- Jeder innere Knoten enthält mindestens einen Punkt,
- Knoten auf gleichem Level sind disjunkt u. überdecken alle Punkte
- → Anzahl innerer Knoten auf jedem Level ist $O(n)$
- → Baum hat $O((d+1)n)$ Knoten



Zu 2.)

Auf jedem Level werden die Punkte eines Quadrats auf die vier Kinder verteilt

→ Zeit hängt linear von der Anzahl Punkte in jedem Quadrat ab

→ Aufbau kostet $O((d+1)n)$ für alle Level

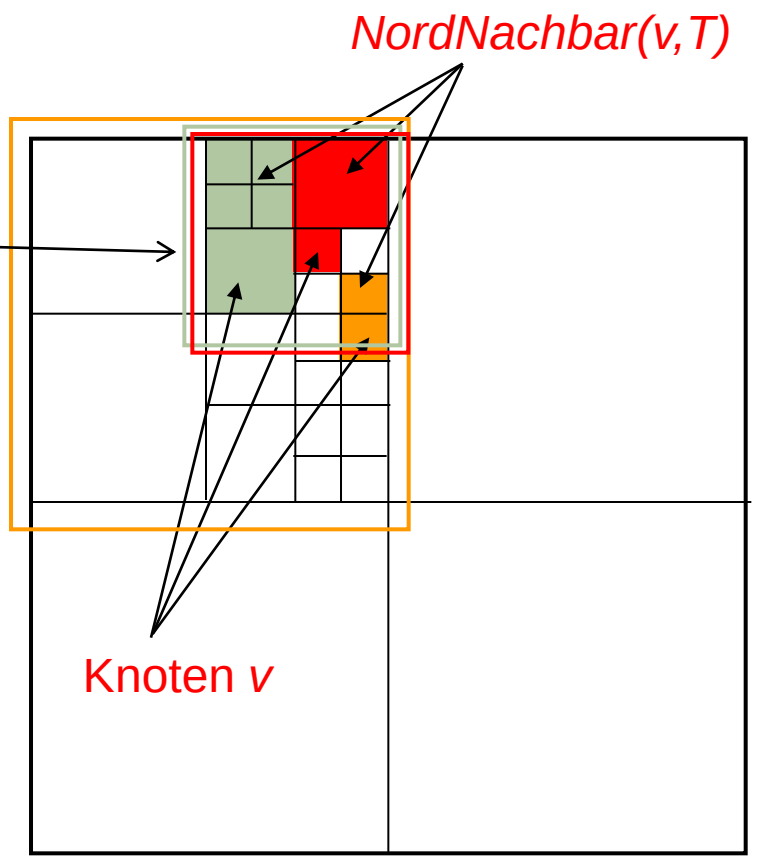
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Suche im Quadtree

Wir haben einen Knoten v im Quadtree und suchen seinen (nördlichen) Nachbarn: $NordNachbar(v,T)$

Idee

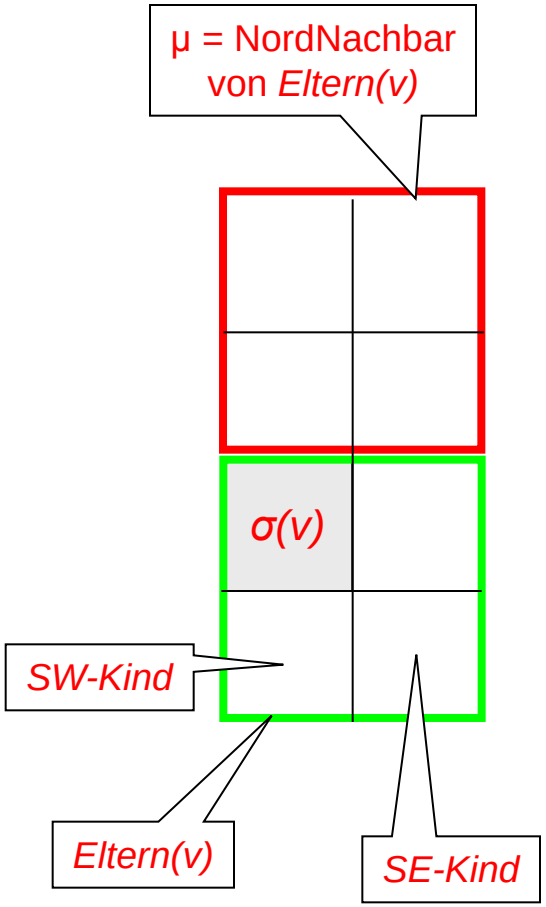
Laufe im Quadtree hoch bis zu dem ersten Knoten, der gemeinsamer Vorfahr von v und $NordNachbar(v,T)$ ist



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Eingabe: Ein Knoten v in einem Quadtree T

Ausgabe: Tiefster Knoten v‘ dessen Tiefe höchstens so tief ist wie die von v, so dass σ(v‘) NordNachbar von σ(v) ist. Falls es keine Knoten gibt, gib nil aus.



Algorithmus NordNachbar(v,T)

1. **If** v=root(T) **then return** nil
2. **If** v=SW-Kind von Eltern(v) **then return** NW-Kind von Eltern(v)
3. **If** v=SE-Kind von Eltern(v) **then return** NE-Kind von Eltern(v)
4. μ := NordNachbar(Eltern(v), T)
5. **If** μ = nil or μ ist ein Blatt
   then return μ
   else if v = NW-Kind von Eltern(v)
   then return SW-Kind von μ
   else return SE-Kind von μ

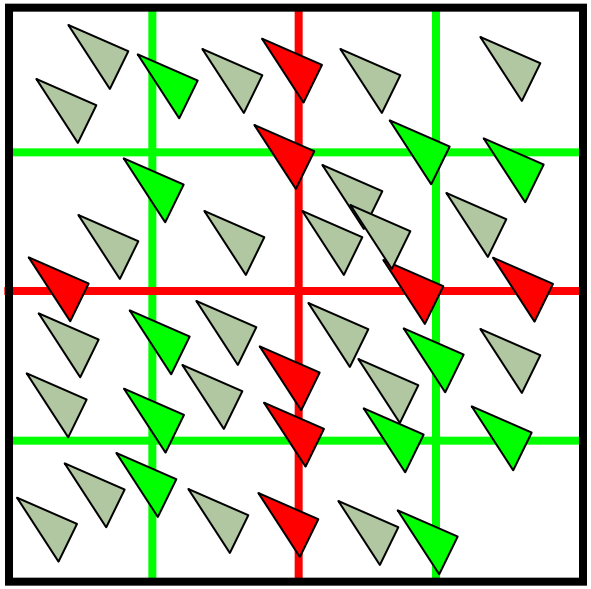
Laufzeit
Nachbar eines Knotens v im Quadtree der Tiefe d wird in Zeit $O(d+1)$ gefunden.

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Was müssen wir beim Abspeichern von Dreiecken beachten?

Problem

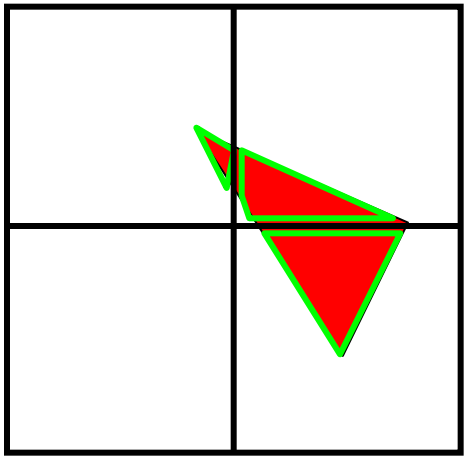
Polygone werden häufig von den Kanten/Begrenzungsflächen der Zellen geschnitten.

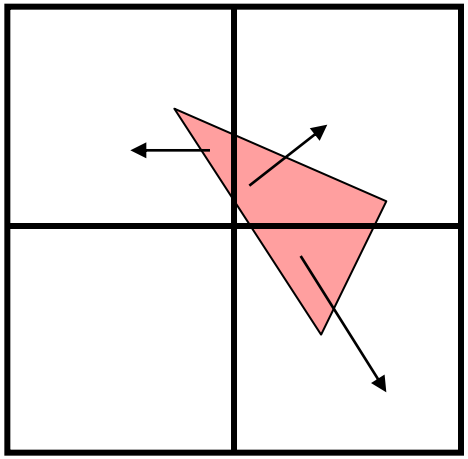


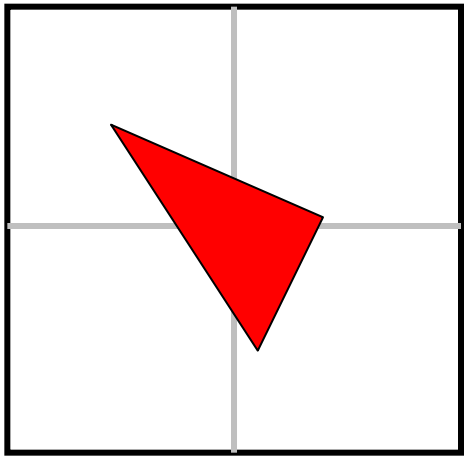
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Praktikable Lösungsansätze

- Polygon aufteilen
  → Polygonanzahl erhöht sich
- Polygon redundant speichern
  → Verwaltungsaufwand erhöht sich, da nur 1x zeichnen sinnvoll
- Polygon in der Zelle speichern, für die es noch nicht unterteilt wird
  → Viele Polygone liegen höher im Baum, als es ihrer Größe entspricht



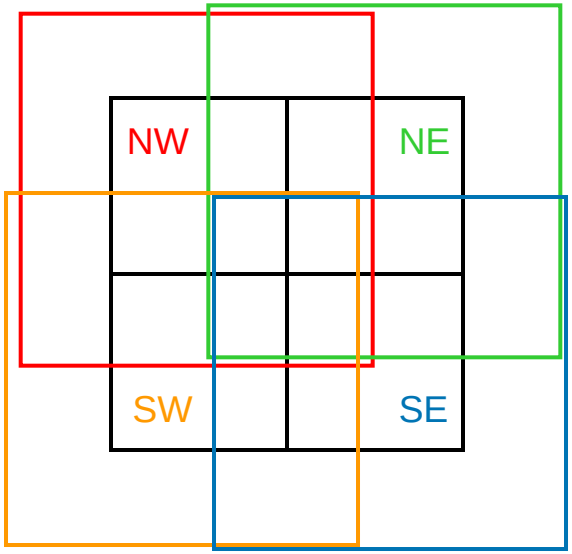


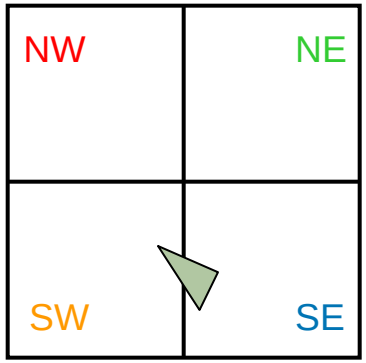


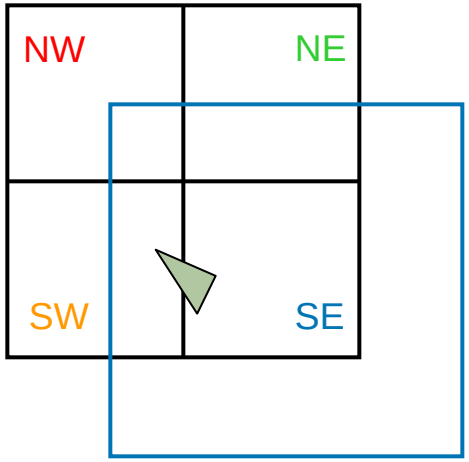
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Loose Octrees (Thatcher Ulrich)

- vergrößere den von den Knoten belegten Raum in allen Dimensionen
- Objekt Mittelpunkt entscheidet, in welchem Knoten ein Objekt liegt
- Ebene hängt nur noch von der Größe des Objekts ab

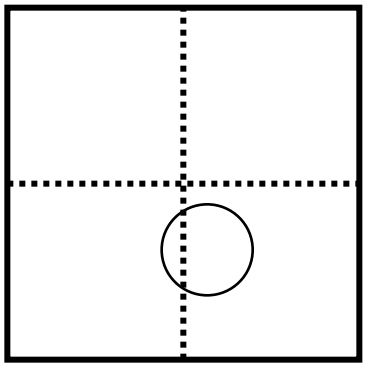






HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Partitionieren von Objekten durch einen Octree



Die Aufteilung des Raumes erfolgt:

- ohne Redundanzen,
- vollständig,
- ohne Berücksichtigung auf die Lage der Objekte.

Wie fügen wir Objekte ein?

Nachteil

- nur Punkte lassen sich einfach und eindeutig einsortieren
- für Objekte mit Ausdehnung sind zusätzliche Lösungen gefragt

Einfachste Lösung

- sortiere Objekt so tief wie möglich ein

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Herkömmliche Traversierung zum Einfügen eines Objektes in einen Octree

Jeder Knoten im Octree enthält:

- Zentrum des Bounding-Box Mittelpunktes
- Referenzen auf die 8 Kinder
- Liste, der im Knoten gespeicherten Objekte (Dreiecke)

```
struct node { Vector3 CubeCenter;
              node* Child[2][2][2];
              ObjectList Objects;
            }
```

Octree

Einfügen und Partitionieren im Octree

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Wir benötigen eine Funktion (**`Classify`**), die klassifiziert, ob ein Objekt (**`v`**) von einer Ebene (**`p`**) durchschnitten wird:

```
int Classify (plane p, volume v)
{
  if ( v ist vollständig hinter p) {
       return 0;
  } else if ( v ist vollständig vor p) {
       return 1;
  } else {
       // v wird von p durchtrennt
       return 2;
}}
```

Octree
Einfügen und Partitionieren im Octree

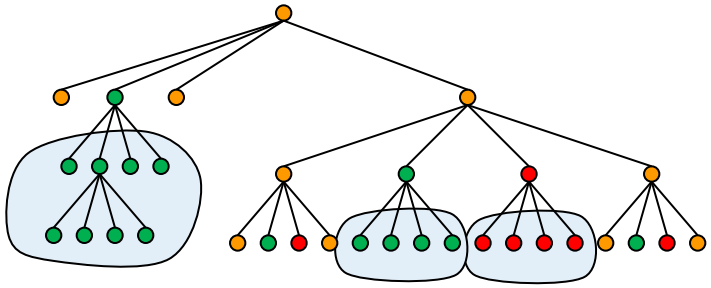
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Die Funktion **`InsertObjectIntoTree`** zum Einfügen eines Objektes **`o`** in einen Knoten **`n`** des Octrees führt schrittweise eine Partitionierung durch:

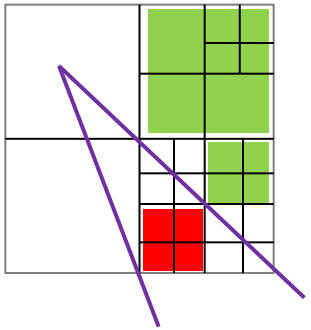
```
void InsertObjectIntoTree( node* n, Object* o)
{

int xc = Classify(plane(1,0,0, CubeCenter.x), o.BoundingVolume);
int yc = Classify(plane(0,1,0, CubeCenter.y), o.BoundingVolume);
int zc = Classify(plane(0,0,1, CubeCenter.z), o.BoundingVolume);

if ( xc == 2 || yc == 2 || zc == 2)
  {     // Objekt wird mind. einer Teilungsebene geschnitten,
        // es passt nicht in einen Kindknoten, speichere es hier
        Objects.Insert(o);
  } else {
        // Objekt passt in einen Kindknoten,
        // Speichere Objekt im Kindknoten
        InsertObjectIntoTree( Child[zc][yc][xc], o);
}}
```

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Wie berechnen wir Frustum Culling mit dem Octree?



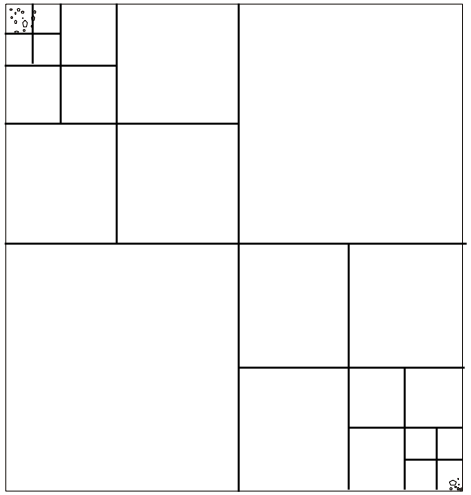
```
enum Visibility {NOT_VISIBLE,
                 PARTLY_VISIBLE,
                 FULLY_VISISBLE};
void Node::Render(Frustum f, Visibility v)
{
  // Falls Elternknoten nicht vollständig sichtbar,
  // Sichtbarkeit prüfen
  if (v != FULLY_VISIBLE) {
       v= ComputeVisibility(this.BoundingBox, f);
       if (v == NOT_VISIBLE) return;
  }
  // Objekte der aktuellen Knoten immer rendern
  this.ObjectList.Render(f,v);
  // Alle Kinder mit PARTLY_VISIBLE und FULLY_VISIBLE
  // traversieren
  for (children) {child.Render(f, v);}
}
```

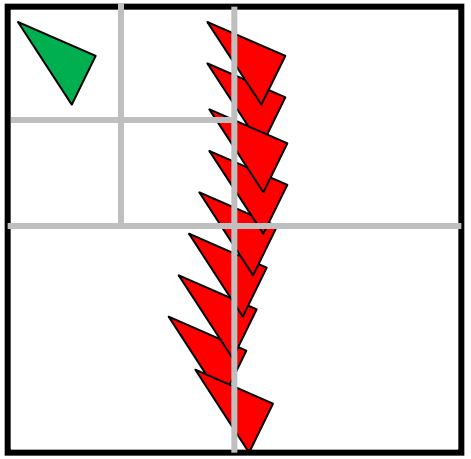


HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Eigenschaften von Ouadtrees/Octrees

- Für dynamische Daten geeignet, einfügen/löschen von Objekten ruft meist nur lokale Änderungen hervor, vgl. kD-Bäume.
- Struktur nur bedingt von der Dichte der Objekte abhängig, s. Anhäufungen von Punkten.
- Polygone können sehr ungünstig positioniert sein, bei „Speicherung in der kleinstmöglichen Zelle“, bleiben sie ggfs. im Wurzelknoten liegen.





Loose-Octree
Konstruktion

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

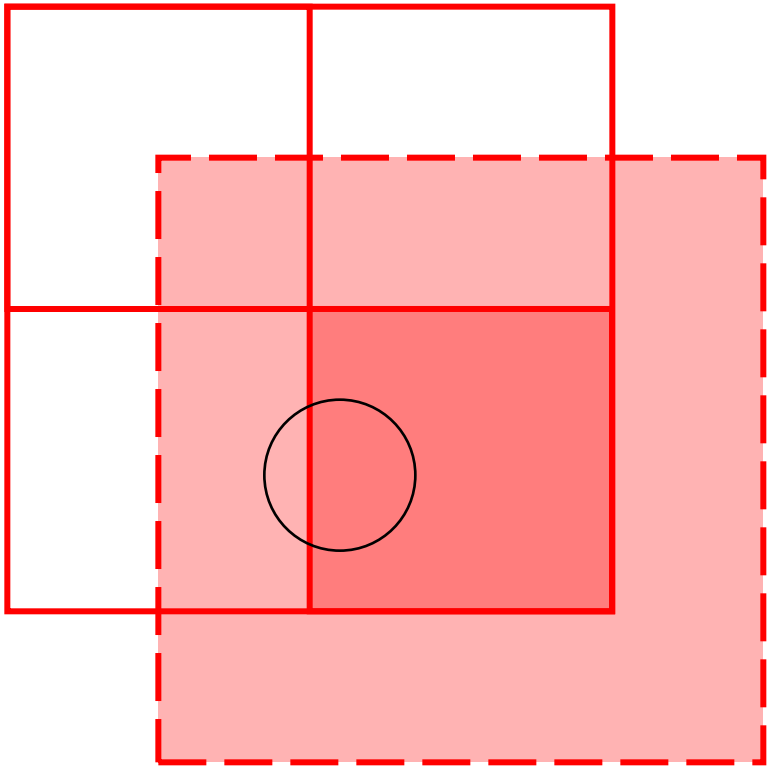
Wie vermeiden wir die Nachteile eines Octrees?

Ziel
Kleine Objekte bleiben nicht mehr auf den Kanten höherer Knoten liegen.

Lösungsansatz
Wir vergrößern die Zellen der Octrees.

Fragen:

- Um wie viel vergrößern wir die Zellen?
- Was ändert sich hierdurch?



Loose-Octree
Konstruktion

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

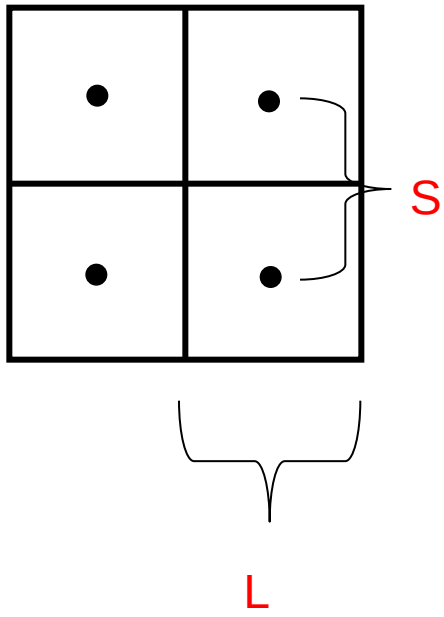
Im herkömmlichen Octree gilt:

Kantenlänge der Octreezelle

$$L(tiefe) = \frac{W}{2^{tiefe}}$$

Abstände der Octreezellen
(in x,y,z-Richtung)

$$S(tiefe) = \frac{W}{2^{tiefe}}$$



$tiefe$:= Tiefe des Octree

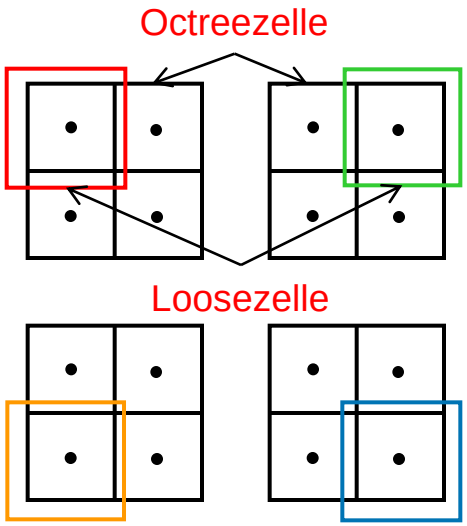
W := Länge der Octreezelle, die die virtuelle Szene umschließt

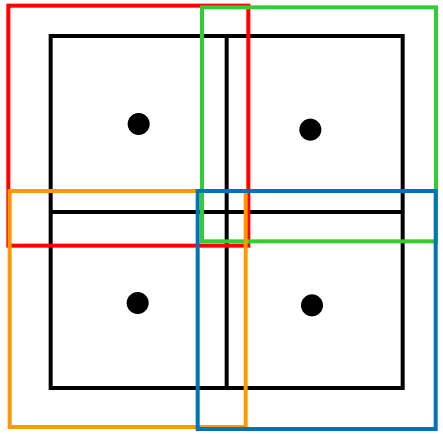
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Loose-Octrees

Vergrößere den Raum einer Octreezelle in allen Dimensionen:

- Wir verändern die Kantenlänge der Octreezellen:
  $L(tiefe) = k\frac{W}{2^{tiefe}}$ $k :=$ ein Faktor > 1
- Wir lassen die Abstände der Zellenmittelpunkte gleich:
  $S(tiefe) = \frac{W}{2^{tiefe}}$
- Wir lassen zu, dass sich die Loosezellen der Kinder überlappen.





Loose-Octree
Konstruktion

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Drei Fragen stellen sich:

1. In welchen Zellen speichern wir die Objekte?
2. Was bewirkt der Faktor k für die Abspeicherung?
3. Wie groß wird k gewählt?

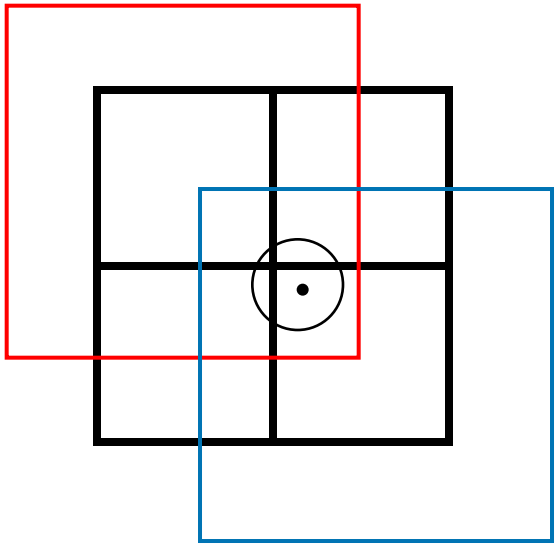
Loose-Octree
Einsortierung

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Frage 1

In welchen Zellen speichern wir die Objekte?

- Objektmittelpunkt entscheidet, in welchem Knoten ein Objekt liegt.
- wähle den Oktanten, der den Objektmittelpunkt enthält.
- aufgrund der Redundanzen sind andere Möglichkeiten denkbar.



Der Loose-Octree hat dann folgende Eigenschaft:

Die Tiefe der Ebene, auf der Objekte gespeichert werden, hängt nur noch von der Objektgröße Objekts ab

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Frage 2

Was bewirkt der Faktor k für die Abspeicherung?

Octree:

„Kleine Objekte“ bleiben bei ungünstiger Position auf höheren Ebenen des Baumes liegen.

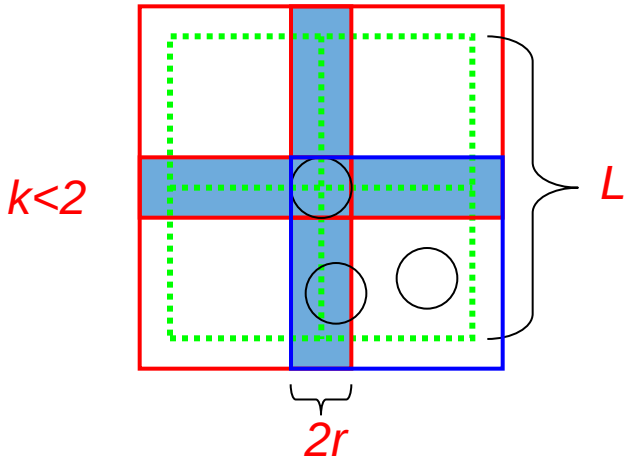
Loose Octree:

„Kleine Objekte“ fallen auf tiefere Ebenen des Baumes.

Loose-Octree
Größe der Loosezelle

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Wie klein darf ein Objekt werden,
um auf einem Knoten mit Tiefe d „liegen zu bleiben“?



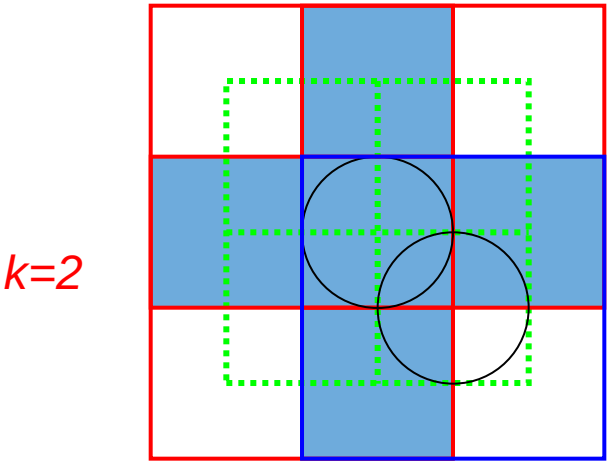
Ein Objekt mit Radius $(k-1) * \frac{L}{4}$ fällt mindestens bis auf Tiefe d, wobei $L = \frac{W}{2^d}$ die Kantenlänge der Octreezelle ist.

Warum?

- Objekte bleiben nur im grau schraffierten Bereich liegen.
- $2r = k * \left(\frac{L}{2}\right) - \left(\frac{L}{2}\right)$

$= (k-1) * \left(\frac{L}{2}\right)$

$r = (k-1) * \frac{L}{4}$



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Frage 3

Wie groß wird k gewählt?

$k < 2$

Nur ganz kleine Objekte „fallen durch“.

$k = 2$

Alle Objekte, deren Durchmesser gleich der Kantenlänge der Octreezelle ist, passen in den entsprechenden Knoten der Loosezelle.

$k > 2$

Überlappungen werden zu groß, zu große Redundanzen entstehen.

Loose-Octree
Größe der Loosezelle

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

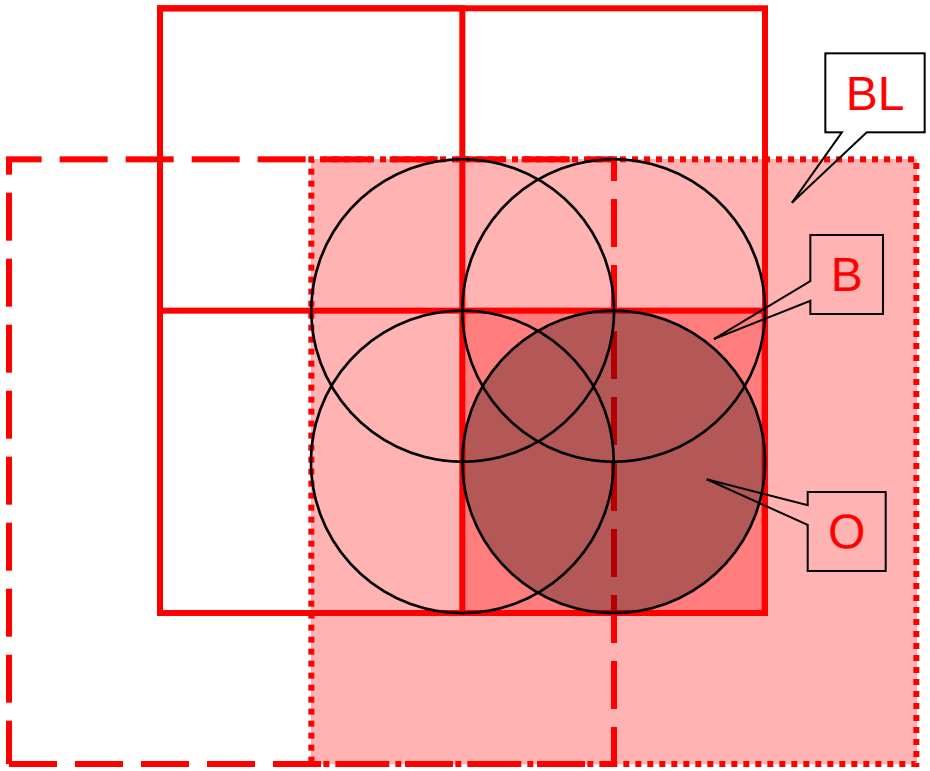
Warum ist für $k = 2$ unser Problem „kleiner ungünstig liegender Objekte“ gelöst ?

Sei

- B, die Octreezelle
- BL, die um $k = 2$ entsprechend vergrößerte Loosezelle

Dann

- passt jedes Objekt O aus dem Elternknoten von BL, das nicht größer als B ist, in ein Kind BL des Elternknotens.



Loose-Octree
Größe der Loosezelle

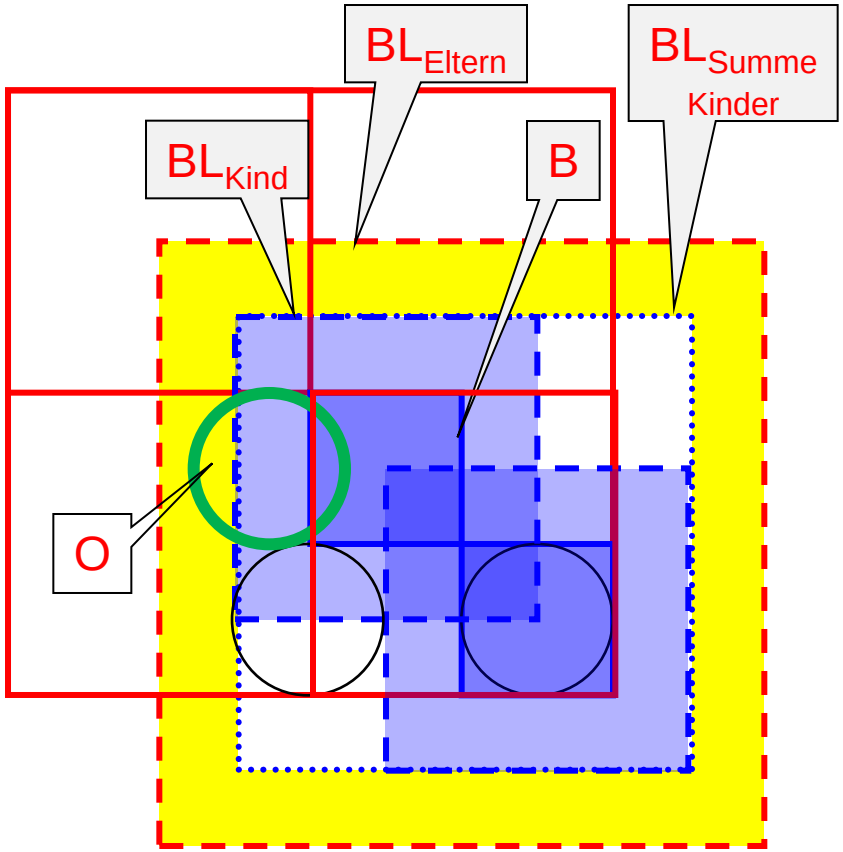
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Verlust der vollständigen Aufteilung

- Der Raum der Loosezelle eines Elternknoten BL_{Eltern} ist größer als die Summe der Loosezellen BL_{Kind} aller Kinder.

Was machen wir mit Objekten O, die so groß sind, wie die Octreezellen B der Kinder, die jedoch außerhalb der Summe aller Loosezellen der Kinder liegen?

- Diese finden in den Kindern der Geschwister der Eltern Platz (Cousin/e).




HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Platzierungsformel

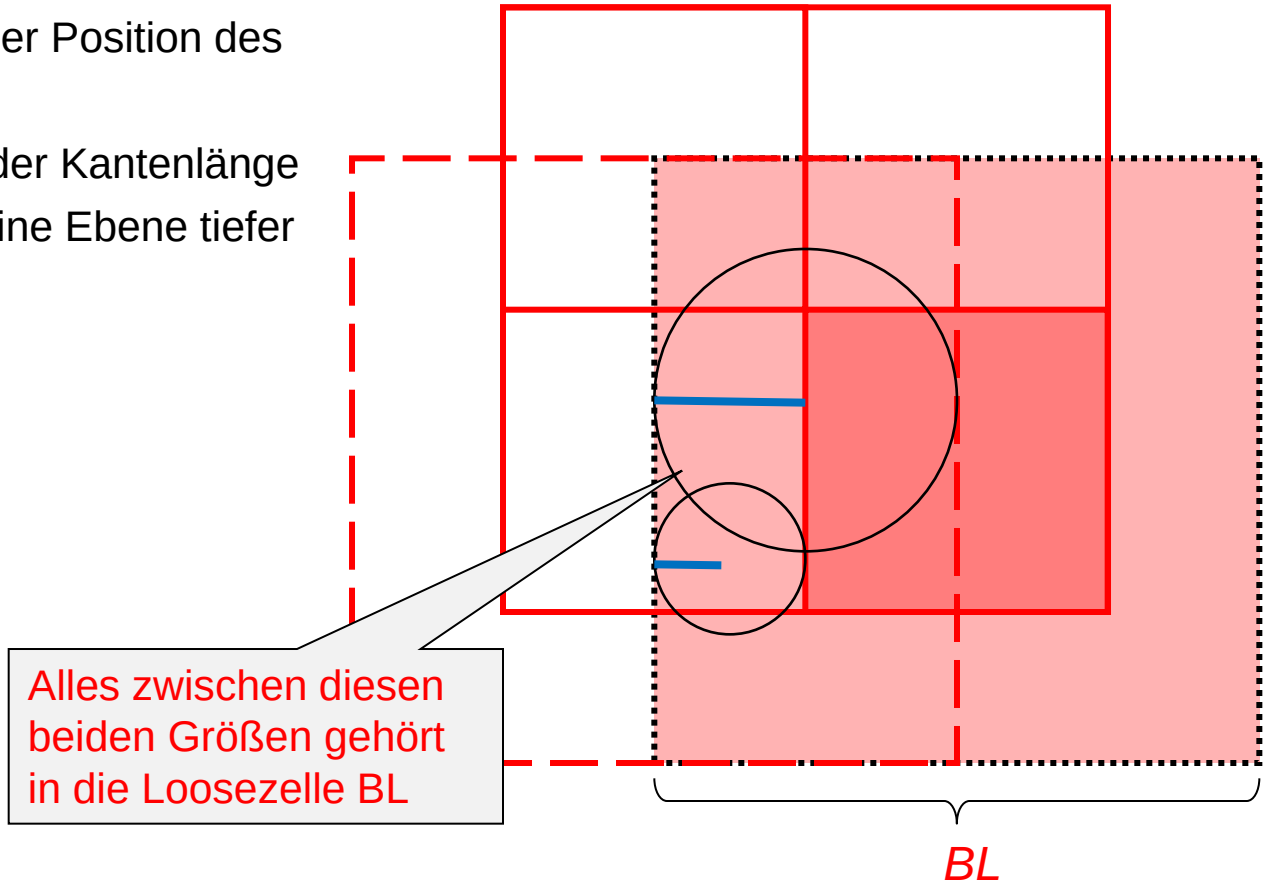
Wir zeigen:

- Für $k = 2$ können alle Objekte direkt im Loose-Octree gespeichert werden (direkte Adressierung) .
- Eine Traversierung und Suche wie beim Octree ist nicht mehr nötig.

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Beobachtung

- Alle Objekte mit $Radius \leq \frac{1}{4}$ der Kantenlänge der Loosezelle BL, passen in die Loosezelle.
- Das ist unabhängig von der Position des Objektes.
- Objekte mit $Radius \leq \frac{1}{8}$ der Kantenlänge der Loosezelle, werden eine Ebene tiefer gespeichert.



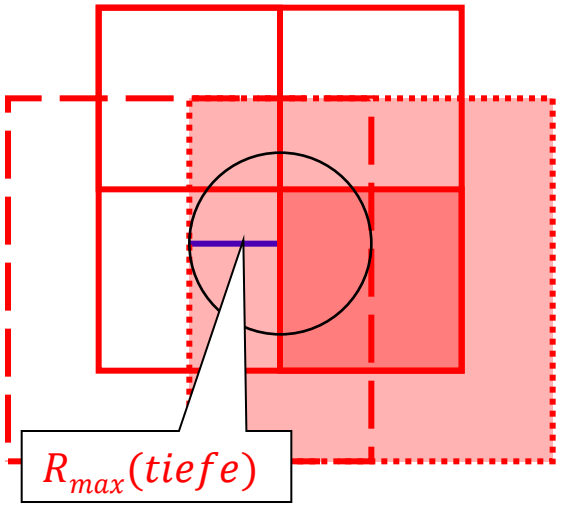
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Berechnung der Tiefe

Idee:

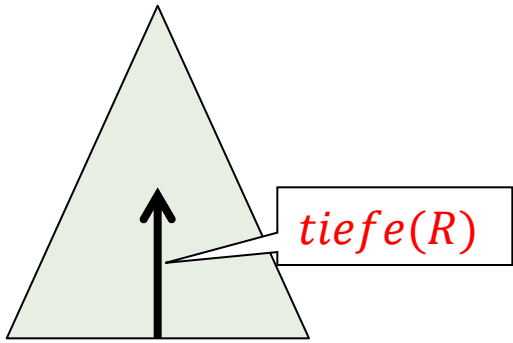
Wir betrachten den maximalen Radius eines Objektes, das in eine Loosezelle passt:

$R_{max}(tiefe)$:= maximaler Objektradius, der in Ebene tiefe gespeichert werden kann



Wir suchen von „unten“ die erste passende Ebene:

$tiefe(R)$:= tiefste Ebene, die ein Objekt mit Radius R aufnehmen kann

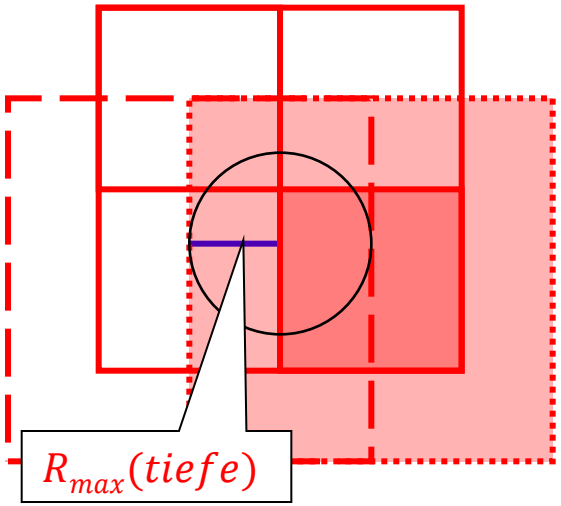


HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Berechnung der Tiefe

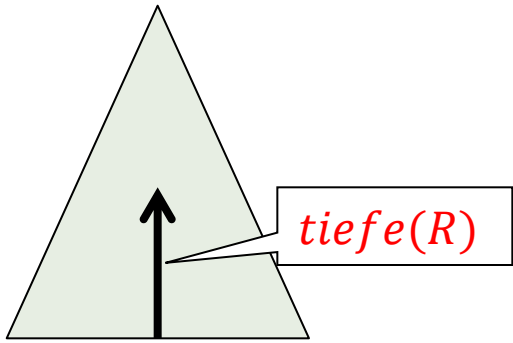
Wir berechnen den maximalen Radius eines Objektes, das in einer Ebene gespeichert werden kann:

- $L(tiefe) = 2 * \frac{W}{2^{tiefe}}$
- $R_{max}(tiefe) = \frac{1}{4} * L(tiefe)$
  $= \frac{1}{2} * \frac{W}{2^{tiefe}}$



Wir suchen von „unten“ nach „oben“ die erste passende Ebene für das Objekt mit Radius R:

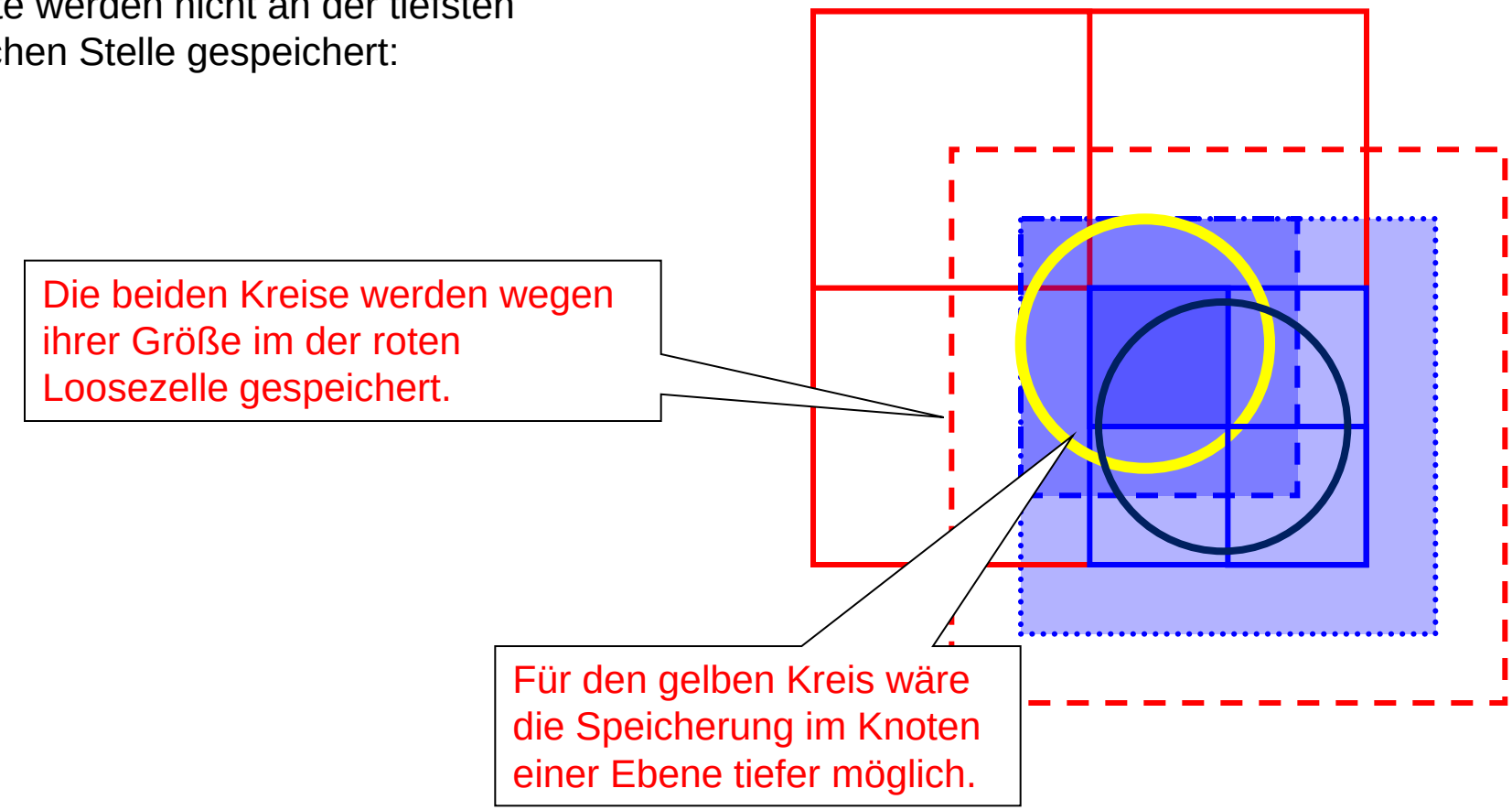
- $R \leq R_{max}(tiefe(R)) \leq \frac{1}{2} * \frac{W}{2^{tiefe(R)}}$
- $tiefe(R) \leq \log(\frac{W}{R}) - 1$



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Nachteil der Platzierungsformel

Objekte werden nicht an der tiefsten möglichen Stelle gespeichert:

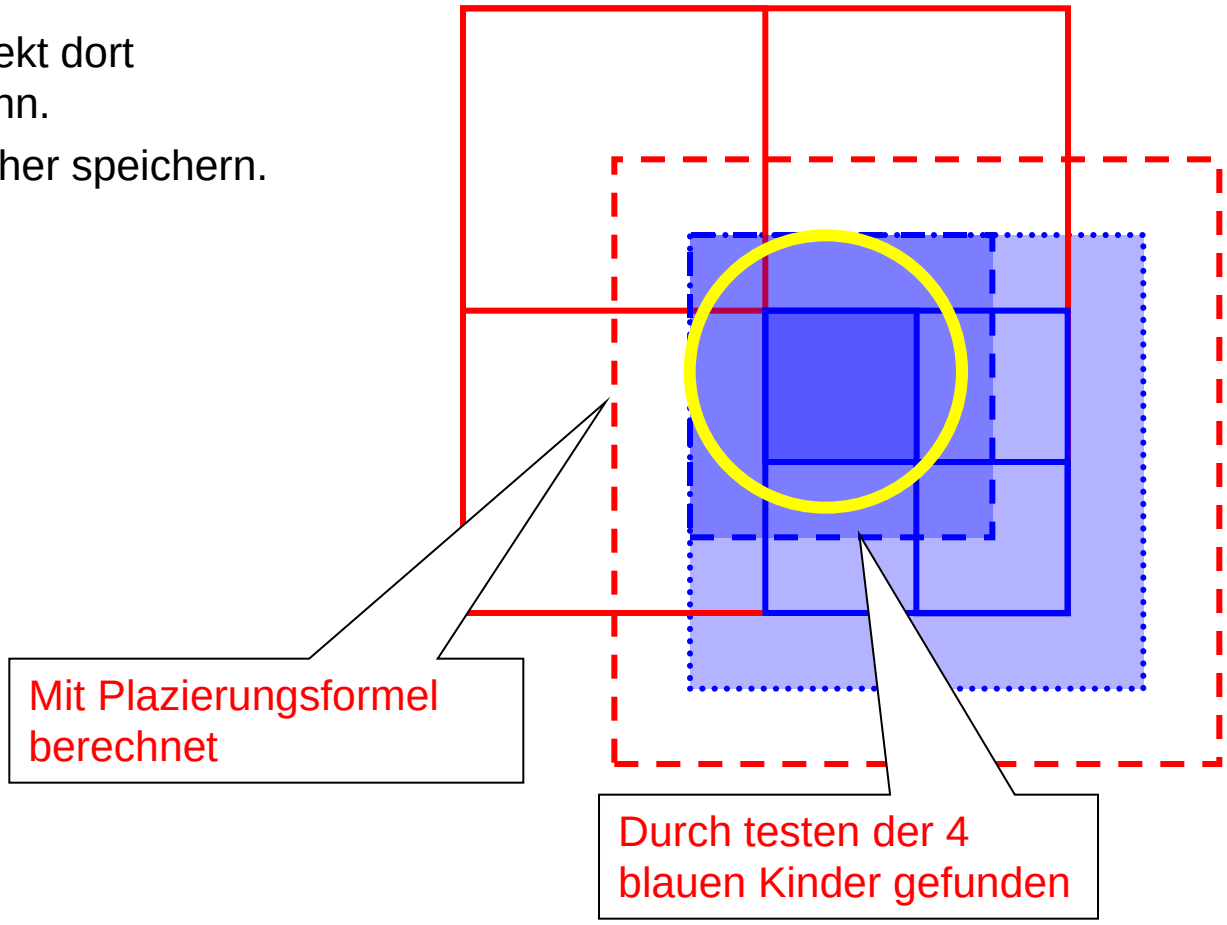


Loose-Octree
Direkte Platzierung

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Wie finden wir die kleinste mögliche Loosezelle ?

- Passende Ebene mit Plazierungsformel berechnen.
- Kinder testen, ob das Objekt dort aufgenommen werden kann.
- Ansonsten eine Ebene höher speichern.



Loose-Octree
Frustum Culling

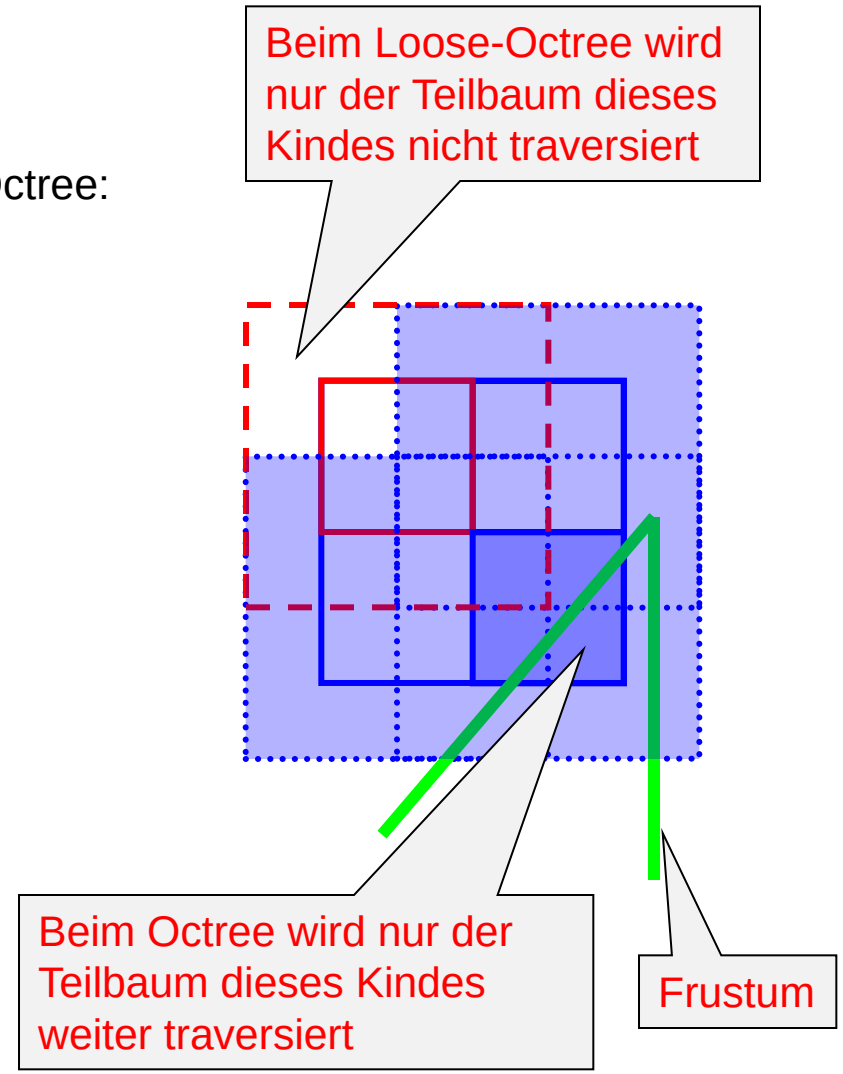
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Welchen Preis bezahlen wir für die gute Platzierung kleiner Objekte?

Empirische Untersuchungen zeigen für den Loose-Octree:

- es werden weniger „möglich sichtbare“ Objekte zurückgeliefert,
- es müssen mehr Knoten getestet werden.

Empirische Daten zum Vergleich
Ouadtree / Loose Quadtree
siehe „Game Programming Gems“ (Kapitel 4.11)

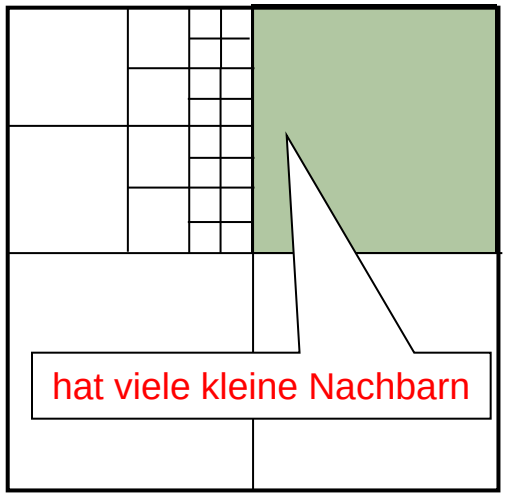


HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Balancierte Quadtrees

Problem

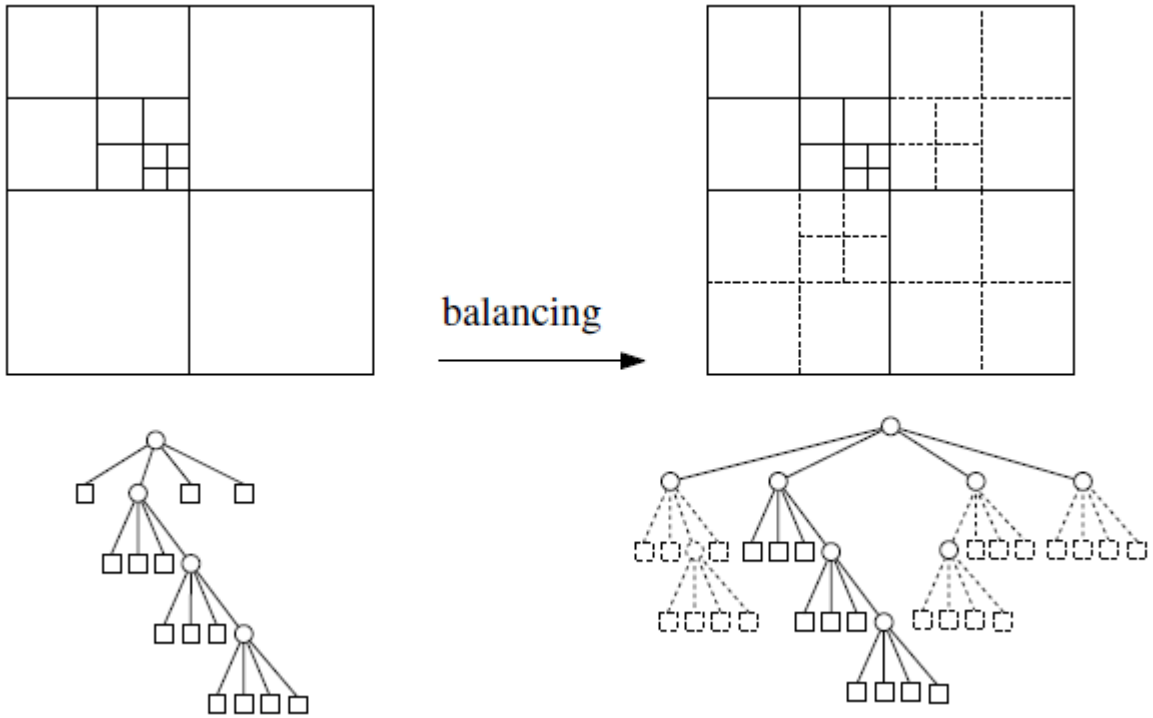
- Große Quadrate können an einer Seite viele kleine Nachbarn haben.
- Was verstehen wir unter einer Balance?
- Wie wird eine Balance konstruiert?



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Definition

Ein Quadtree ist balanciert,
wenn sich zwei benachbarte Quadrate in der Seitenlänge höchstens um einen Faktor zwei unterscheiden.

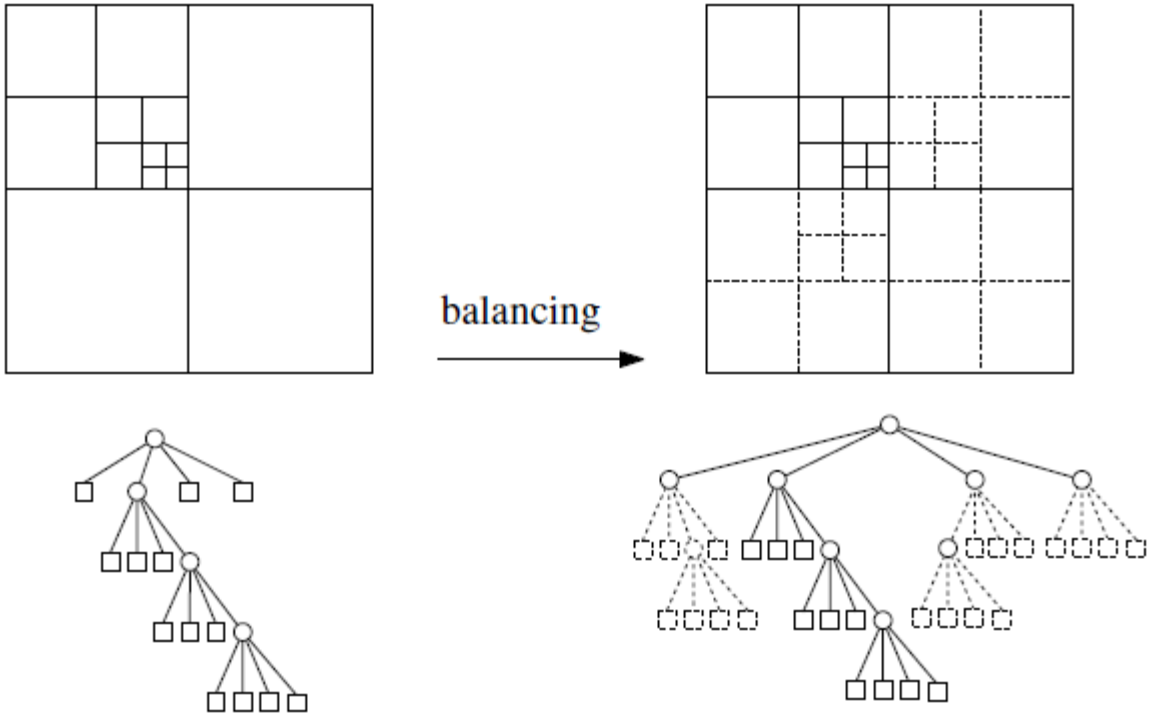


HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Algorithmus zum Aufbau eines balancierten Quadtrees

Idee

1. Baue den Quadtree mit dem mit dem normalen Algorithmus auf.
2. Verfeinere alle unbalancierten Blätter (= weitere Aufteilungen hinzufügen), bis alle Blätter balanciert sind.
3. Teile dazu alle Blätter,
   die zu groß sind, weiter auf.



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Eingabe: umbalancierter Quadtree (aus Punkten)

Ausgabe: balancierte Version des Quadtrees

Algorithmus BalanceQuadtree(T)

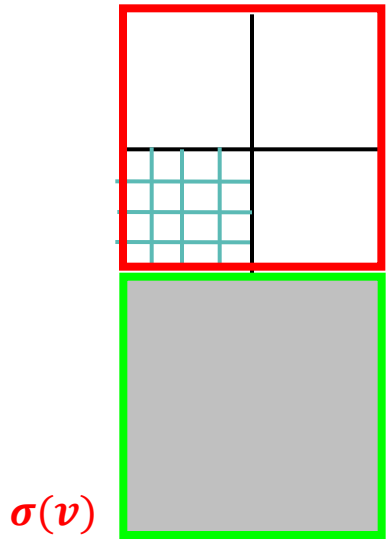
1. Füge alle Blätter in eine lineare Liste L
2. while L ist nicht leer
3. entferne ein Blatt v aus L
4. if $\sigma(v)$ muss aufgeteilt werden
5. then
   - Ersetze Blatt v als internen Knoten mit vier Kindern.
   - Falls v einen Punkt enthält, füge ihn in das entsprechende Blatt (Kind von v).
   - Füge alle vier Kinder von v in L ein.
   - Prüfe, ob $\sigma(v)$ jetzt Nachbarn enthält, die aufzuteilen sind. Füge die ggfs. in L ein.

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Wie wird geprüft, ob ein Blatt aufgeteilt werden muss?

→ verwende den Algorithmus zur Nachbarsuche $NordNachbar(v,T)$

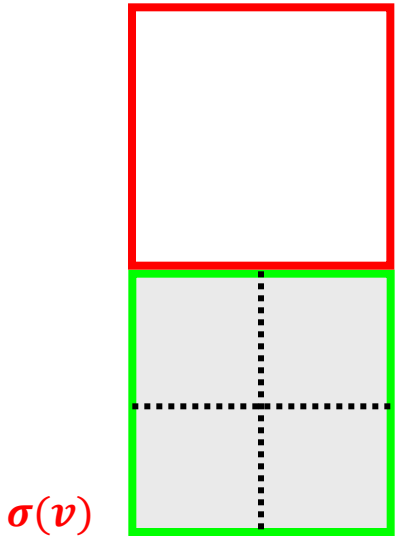
Es gibt unbalancierte Nordnachbarn zu einem Knoten v, wenn der Nordnachbar von v ein SW- oder SE-Kind hat, dass kein Blatt ist.



Wie wird geprüft, ob nach einem Split von v die Nachbarn von $\sigma(v)$ aufgeteilt werden müssen?

→ verwende den Algorithmus zur Nachbarsuche $NordNachbar(v,T)$

Es gibt einen unbalancierten Nordnachbarn zu einem Knoten v, wenn der Nordnachbar größer ist als $\sigma(v)$.

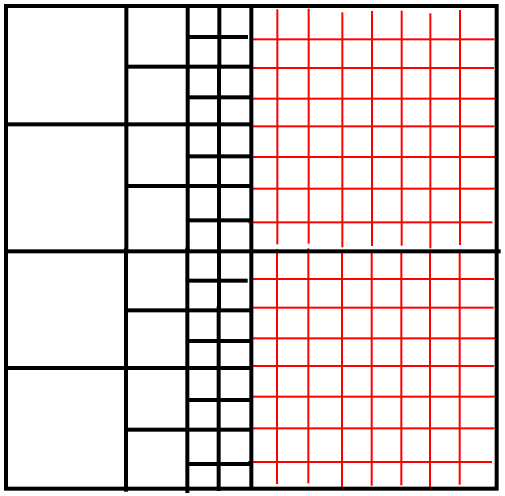


HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Es gilt:

Sei T ein Quadtree mit m Knoten.
Die balancierte Version $T*$ von T hat $O(m)$ Knoten.

Warum bewirkt eine Aufteilung nicht die Erzeugung vieler weiterer Aufteilungen?



Wir zeigen:

Es sind nur $O(m)$ Aufteilungen notwendig.
→ $T*$ hat $O(m)$ Knoten

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

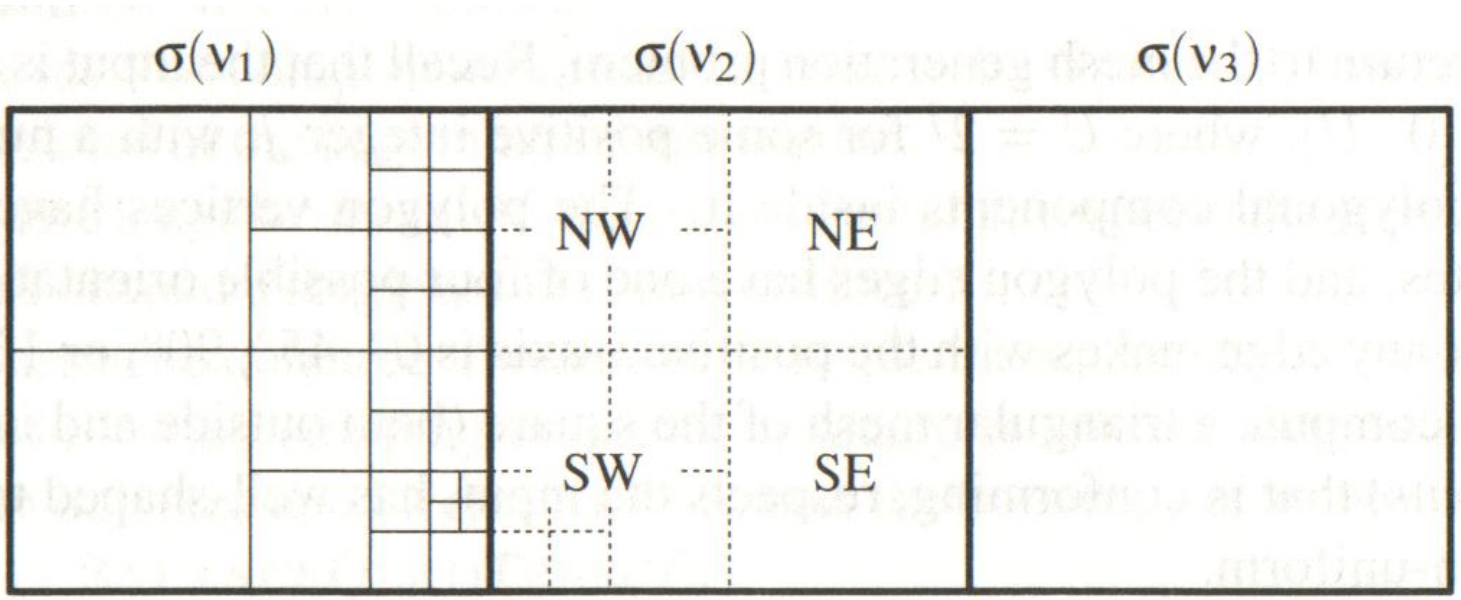
Lemma

Seien v_1, v_2, v_3 drei benachbarte Knoten auf gleicher Tiefe, wobei v_2, v_3 Blätter in T sind. Für jede beliebige Unterteilung von $\sigma(v_1)$ bewirken die balancierenden Aufteilungen durch den Algo. in $\sigma(v_2)$ keine weiteren Aufteilungen in $\sigma(v_3)$.

Beweis: Induktion über die Tiefe d des Teilbaum unter v_1

I.A.

Für $d = 1$ hat $\sigma(v_1)$ eine Unterteilung, $\sigma(v_2), \sigma(v_3)$ werden nicht unterteilt.



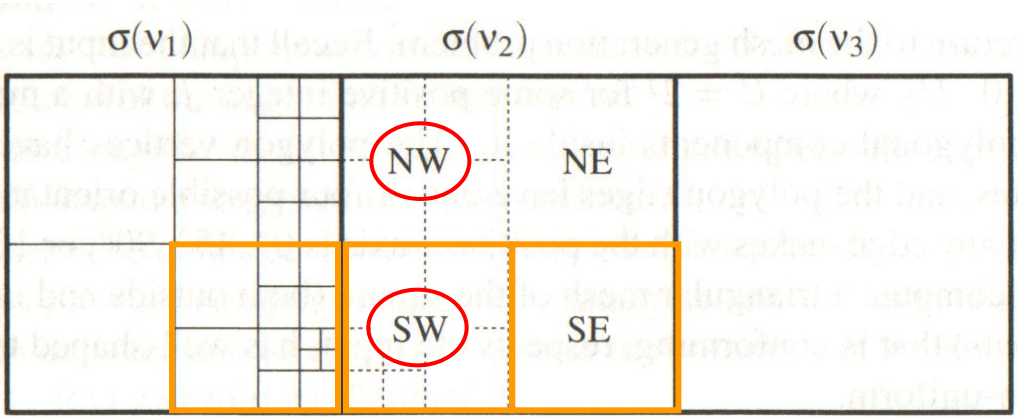
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

I.S.

Habe v_1 jetzt Tiefe d, mit $d > 1$.

- Nach I.V. verursacht die Blancierung im SW-Quadrat von $\sigma(v_2)$ (notwendig wegen kleiner SE-Quadrate in $\sigma(v_1)$ keine Aufteilung im SE-Quadrat von $\sigma(v_2)$.
- Ebenso nach I.V. verursacht die Blancierung im NW-Quadrat von $\sigma(v_2)$ (notwendig wegen kleiner NE-Quadrate in $\sigma(v_1)$) keine Aufteilung im NE-Quadrat von $\sigma(v_2)$.
- → NE und SE in $\sigma(v_2)$ werden nicht aufgeteilt
- → $\sigma(v_3)$ braucht nicht aufgeteilt werden

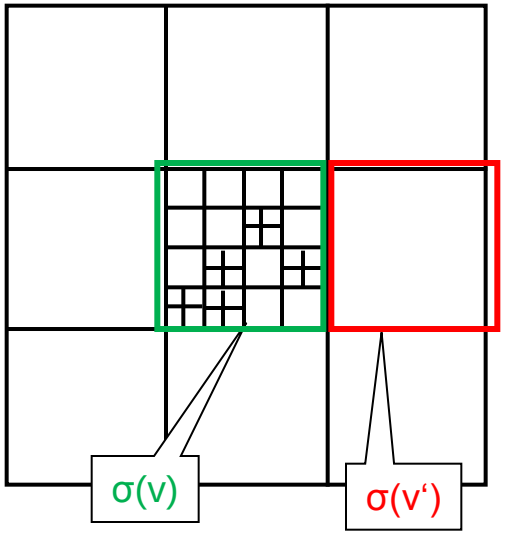
q.e.d. (Lemma)



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Folgerung

Die Unterteilung eines Quadrates $\sigma(v)$ verursacht in anderen Quadraten $\sigma(v')$ gleicher Größe nur dann eine weitere Aufteilung, wenn v' ein Knoten der 8 benachbarten Quadrate von v ist.



Wir müssen jetzt die Quadrate zählen, die aufgeteilt werden:

Dazu unterscheiden wir die Quadrate des ursprünglichen Quadtree und die neu durch die Balancierung hinzugefügten Quadrate.

Alte Quadrate := Quadrate von T
Neue Quadrate := zusätzliche neue Quadrate von $T*$

Quadtrees/Octrees
Balancierung

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

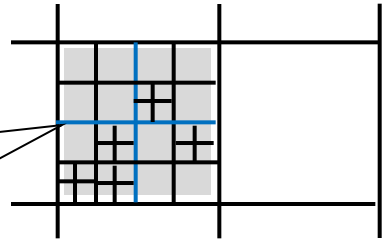
Betrachte den Split eines Quadrates σ:

- Der Split kostet 4 neue Knoten.
- Eines der umgebenden Quadrate muss ein altes Quadrat σ' sein, das den Split verursacht hat.
- Wir weisen die Kosten des Splits (4) dem alten Quadrat σ' zu.

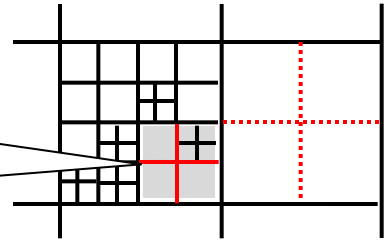
→ Jeder alte Knoten von T bekommt durch die Balancierung so maximal 8-mal Kosten zugewiesen.

q.e.d.

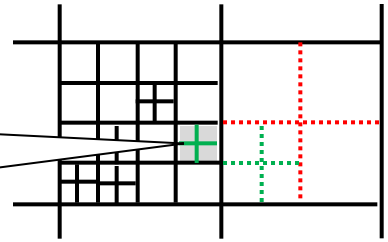
Die blaue Unterteilung des alten Quadrates verursacht keinen Split.



Die rote Unterteilung des alten Quadrates verursacht einen Split.



Die grüne Unterteilung des alten Quadrates verursacht einen Split.



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Laufzeit

Sei T ein Quadtree mit m Knoten.
Die balancierte Version kann in Zeit $O((d+1)m)$ konstruiert werden.

Warum?

- Da wir eine konstante Anzahl von Nachbarschaftssuchen durchführen benötigen wir $O(d+1)$ Zeit, um ein Blatt in der Liste L zu behandeln.
- Da jeder Knoten höchstens einmal behandelt wird und es höchstens $O(m)$ Knoten gibt.
- → Laufzeit ist $O((d+1)m)$