Karlsruhe Institute of Technology

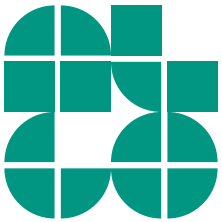
Vorlesung Algorithmische Geometrie

Quadtrees und Meshing

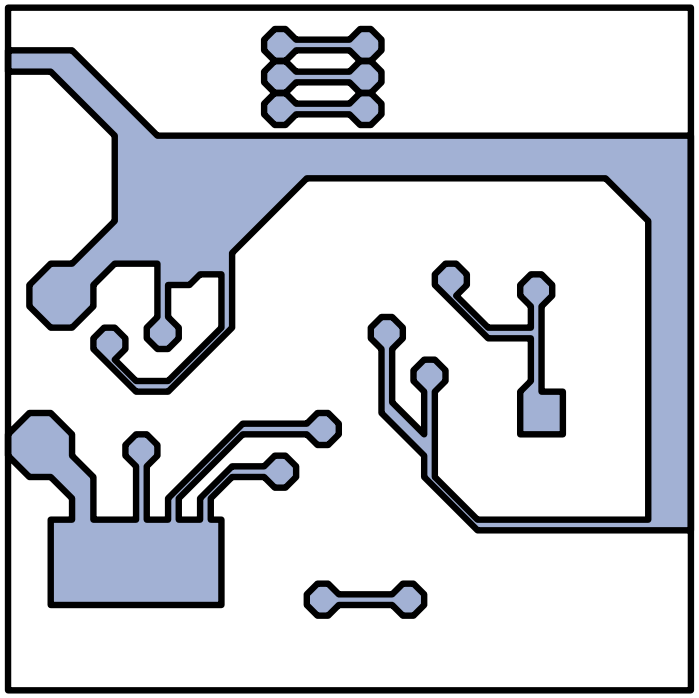
LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Martin Nöllenburg

21.06.2011



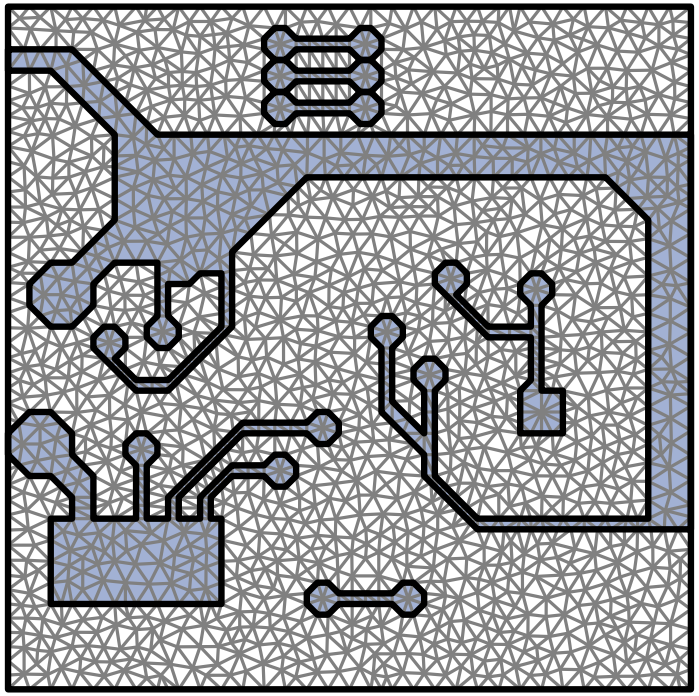
Motivation: Meshing von Platinenlayouts



Zur Simulation der Hitzeentwicklung auf Platinen kann man die *finite Elemente Methode* (FEM) einsetzen:

- zerlege die Platine in kleine homogene Elemente (z.B. Dreiecke) $\rightarrow$ Mesh
- eigene Hitzeentwicklung und Einfluss auf Nachbarn je Element bekannt
- approximiere numerisch die gesamte Hitzeentwicklung der Platine

Motivation: Meshing von Platinenlayouts



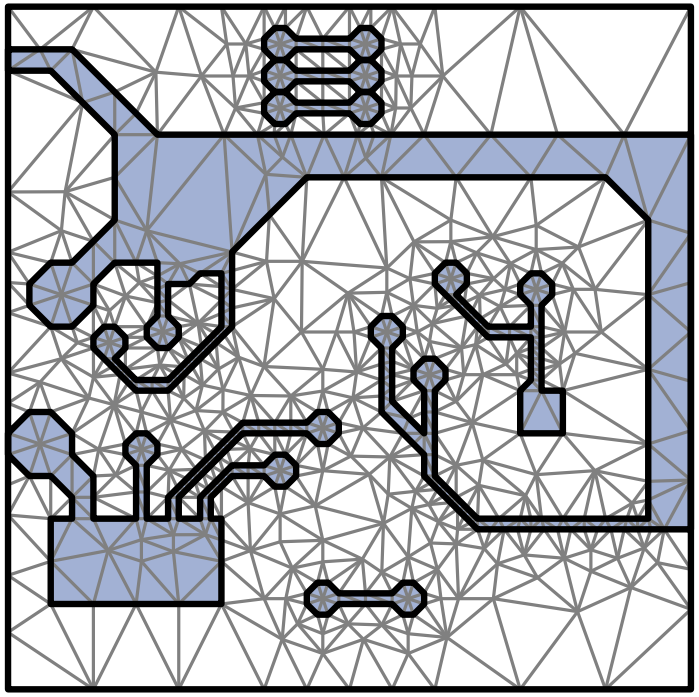
Zur Simulation der Hitzeentwicklung auf Platinen kann man die *finite Elemente Methode* (FEM) einsetzen:

- zerlege die Platine in kleine homogene Elemente (z.B. Dreiecke) $\rightarrow$ Mesh
- eigene Hitzeentwicklung und Einfluss auf Nachbarn je Element bekannt
- approximiere numerisch die gesamte Hitzeentwicklung der Platine

Qualitätseigenschaften von FEM:

- je feiner das Mesh desto besser die Approximation
- je gröber das Mesh desto schneller die Berechnung
- je kompakter die Elemente desto schneller die Konvergenz

Motivation: Meshing von Platinenlayouts



Zur Simulation der Hitzeentwicklung auf Platinen kann man die *finite Elemente Methode* (FEM) einsetzen:

- zerlege die Platine in kleine homogene Elemente (z.B. Dreiecke) $\rightarrow$ Mesh
- eigene Hitzeentwicklung und Einfluss auf Nachbarn je Element bekannt
- approximiere numerisch die gesamte Hitzeentwicklung der Platine

Qualitätseigenschaften von FEM:

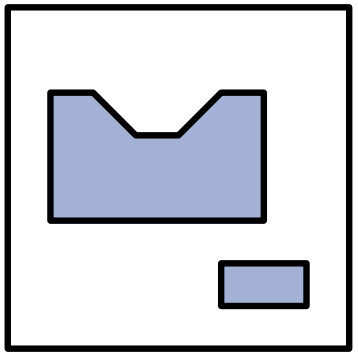
- je feiner das Mesh desto besser die Approximation
- je gröber das Mesh desto schneller die Berechnung
- je kompakter die Elemente desto schneller die Konvergenz

Ziel:

- adaptive Meshgröße (klein an Materialgrenzen, sonst gröber)
- wohlgeformte Dreiecke (nicht zu schmal)

Adaptive Dreiecksnetze

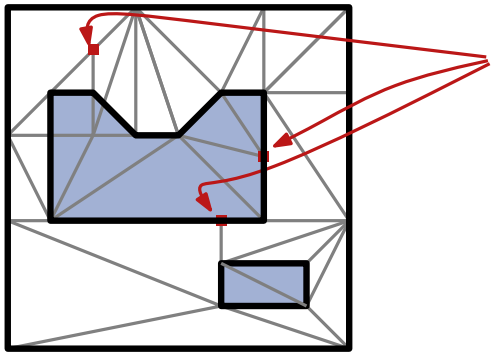
Geg.: Quadrat $Q = [0, U] \times [0, U]$ für Zweierpotenz $U = 2^j$ mit *oktilinearen*, ganzzahligen Polygonen im Inneren.



Ges.: Dreiecksnetz für Q mit folgenden Eigenschaften

Adaptive Dreiecksnetze

Geg.: Quadrat $Q = [0, U] \times [0, U]$ für Zweierpotenz $U = 2^j$ mit *oktilinearen*, ganzzahligen Polygonen im Inneren.



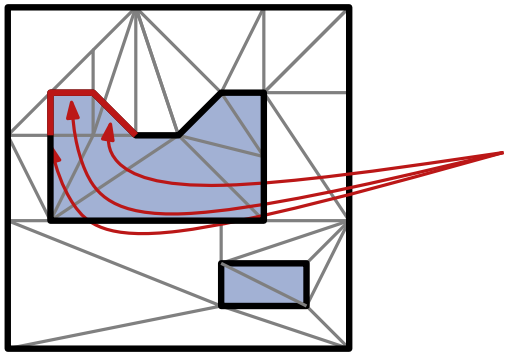
unzulässige Dreiecksknoten

Ges.: Dreiecksnetz für Q mit folgenden Eigenschaften

- keine Dreiecksknoten im Inneren von Kanten

Adaptive Dreiecksnetze

Geg.: Quadrat $Q = [0, U] \times [0, U]$ für Zweierpotenz $U = 2^j$ mit *oktilinearen*, ganzzahligen Polygonen im Inneren.



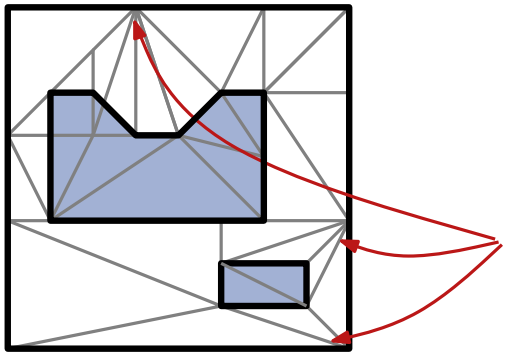
nicht Teil des Dreiecksnetzes

Ges.: Dreiecksnetz für Q mit folgenden Eigenschaften

- keine Dreiecksknoten im Inneren von Kanten
- Eingabekanten müssen Teil des Dreiecksnetzes sein

Adaptive Dreiecksnetze

Geg.: Quadrat $Q = [0, U] \times [0, U]$ für Zweierpotenz $U = 2^j$ mit *oktilinearen*, ganzzahligen Polygonen im Inneren.



unzulässige Winkel

Ges.: Dreiecksnetz für Q mit folgenden Eigenschaften

gültig:

- keine Dreiecksknoten im Inneren von Kanten
- Eingabekanten müssen Teil des Dreiecksnetzes sein
- Dreieckswinkel zwischen 45° und 90°

Adaptive Dreiecksnetze

Geg.: Quadrat $Q = [0, U] \times [0, U]$ für Zweierpotenz $U = 2^j$ mit *oktilinearen*, ganzzahligen Polygonen im Inneren.

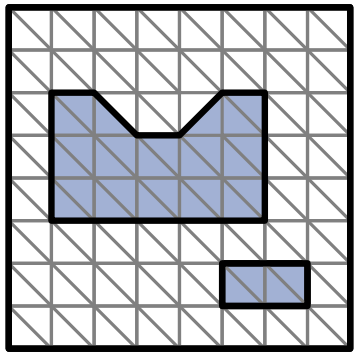
Ges.: Dreiecksnetz für Q mit folgenden Eigenschaften

gültig
- keine Dreiecksknoten im Inneren von Kanten
- Eingabekanten müssen Teil des Dreiecksnetzes sein
- Dreieckswinkel zwischen 45° und 90°

- **adaptiv**, d.h. fein an den Polygonkanten, sonst gröber

Adaptive Dreiecksnetze

Geg.: Quadrat $Q = [0, U] \times [0, U]$ für Zweierpotenz $U = 2^j$ mit *oktilinearen*, ganzzahligen Polygonen im Inneren.



uniformes Dreiecksnetz

Ges.: Dreiecksnetz für Q mit folgenden Eigenschaften

gültig
- keine Dreiecksknoten im Inneren von Kanten
- Eingabekanten müssen Teil des Dreiecksnetzes sein
- Dreieckswinkel zwischen 45° und 90°

- **adaptiv**, d.h. fein an den Polygonkanten, sonst gröber

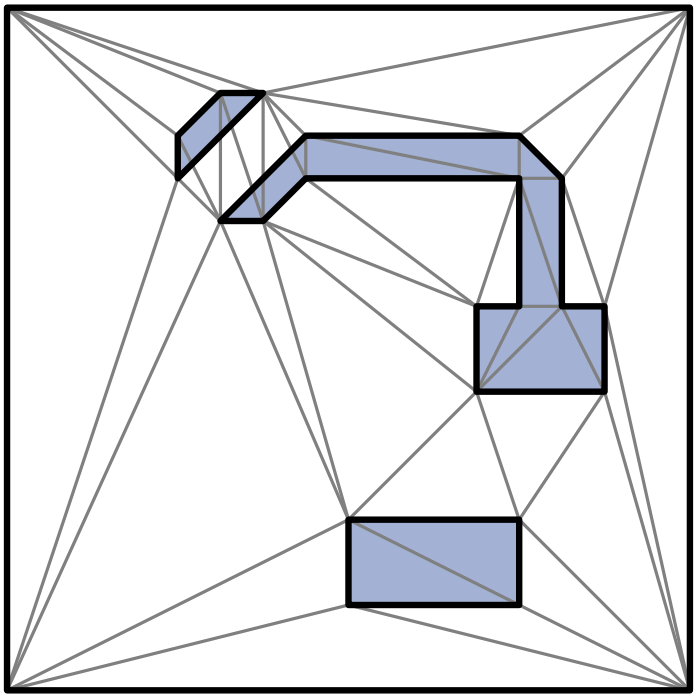
Kennen wir schon sinnvolle Triangulierungen für Q?

Delaunay-Triangulierung?

- maximiert kleinste Winkel

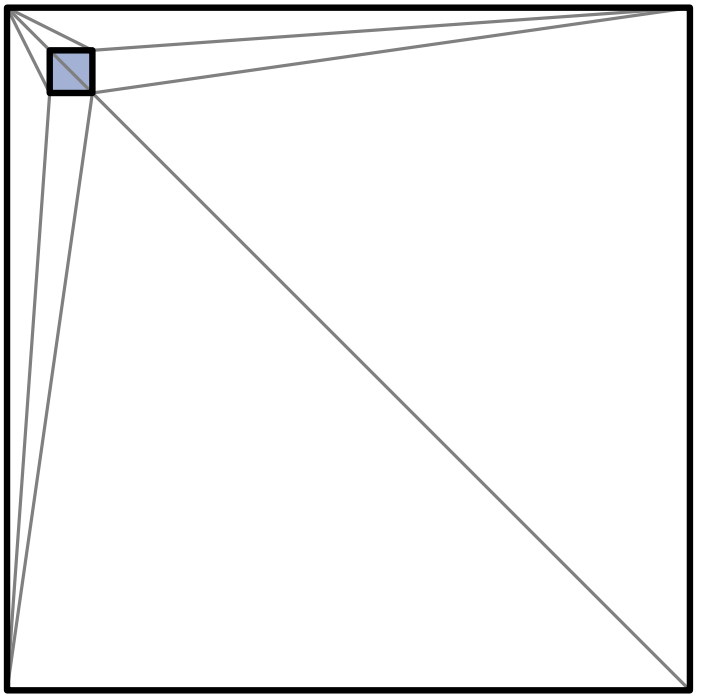
Delaunay-Triangulierung?

- maximiert kleinste Winkel
- ist definiert für Punkte und ignoriert vorhandene Kanten



Delaunay-Triangulierung?

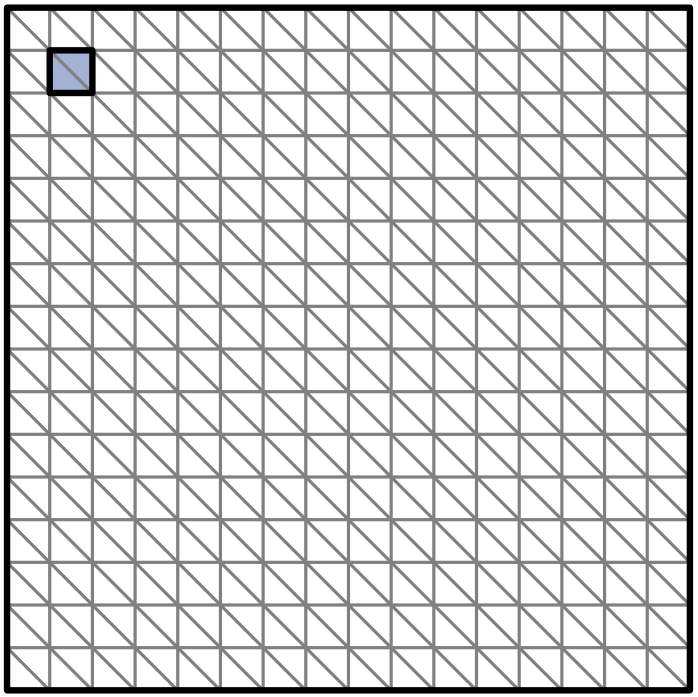
- maximiert kleinste Winkel
- ist definiert für Punkte und ignoriert vorhandene Kanten
- kann trotzdem sehr kleine Winkel produzieren



Delaunay-Triangulierung?

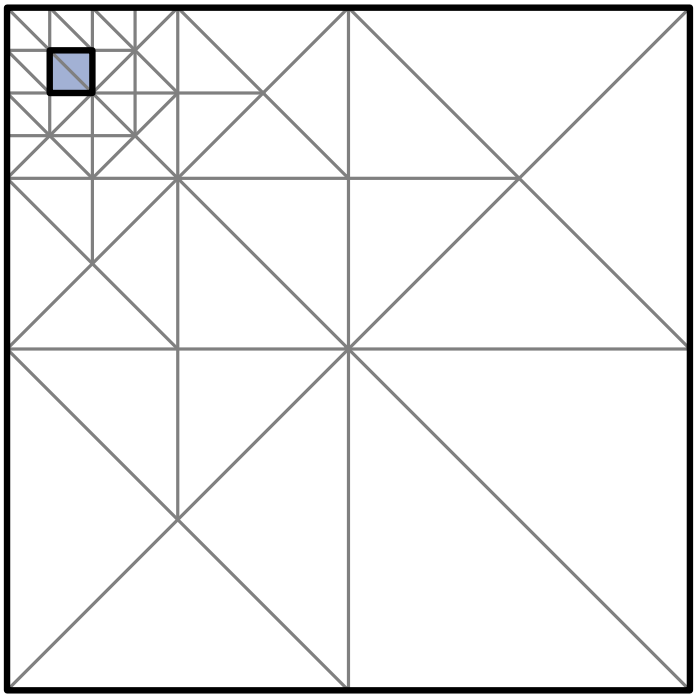
- maximiert kleinste Winkel
- ist definiert für Punkte und ignoriert vorhandene Kanten
- kann trotzdem sehr kleine Winkel produzieren
- verwendet keine zusätzlichen *Steiner*-Punkte

Winkel zulässig, aber uniform



512 Dreiecke

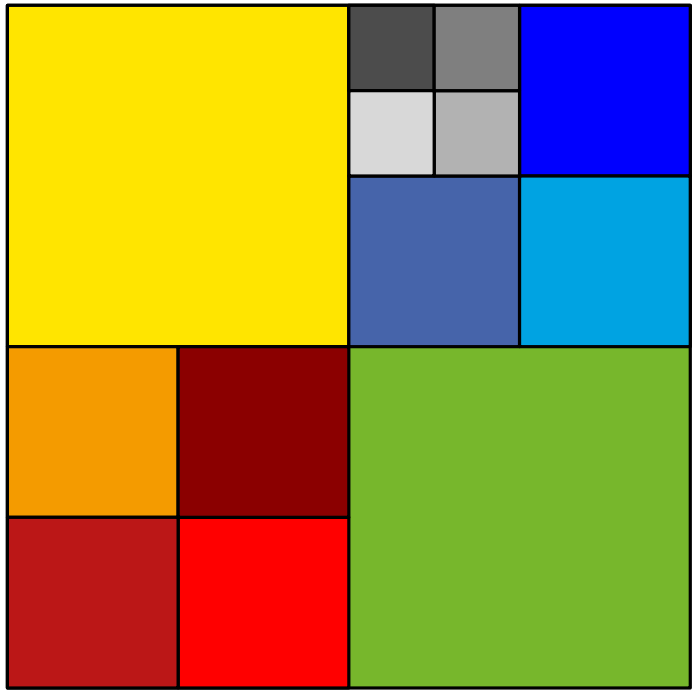
Winkel zulässig und adaptiv

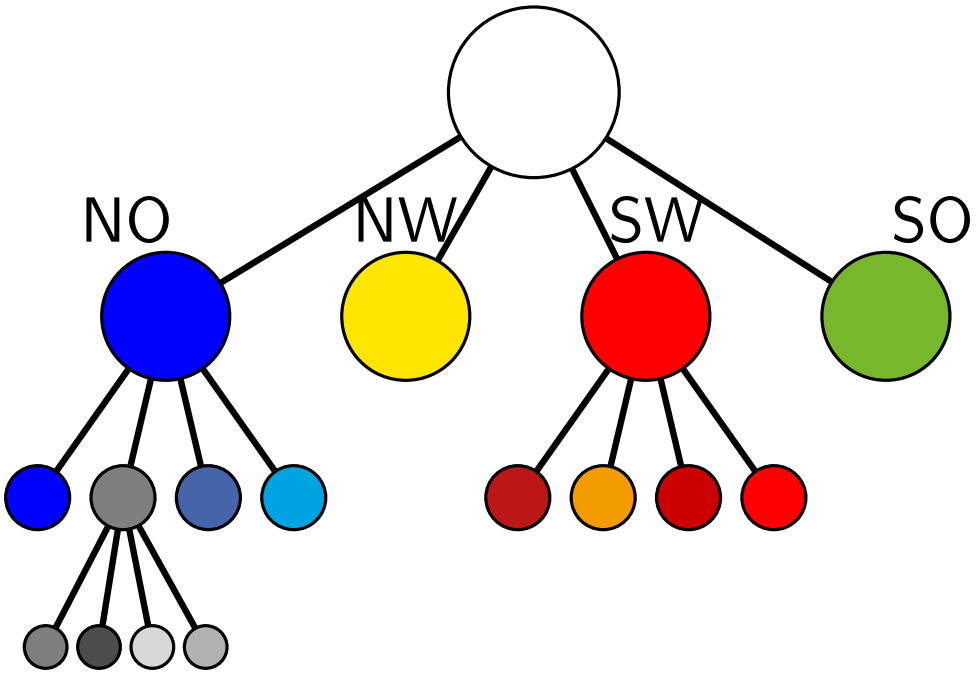


52 Dreiecke

Quadtrees

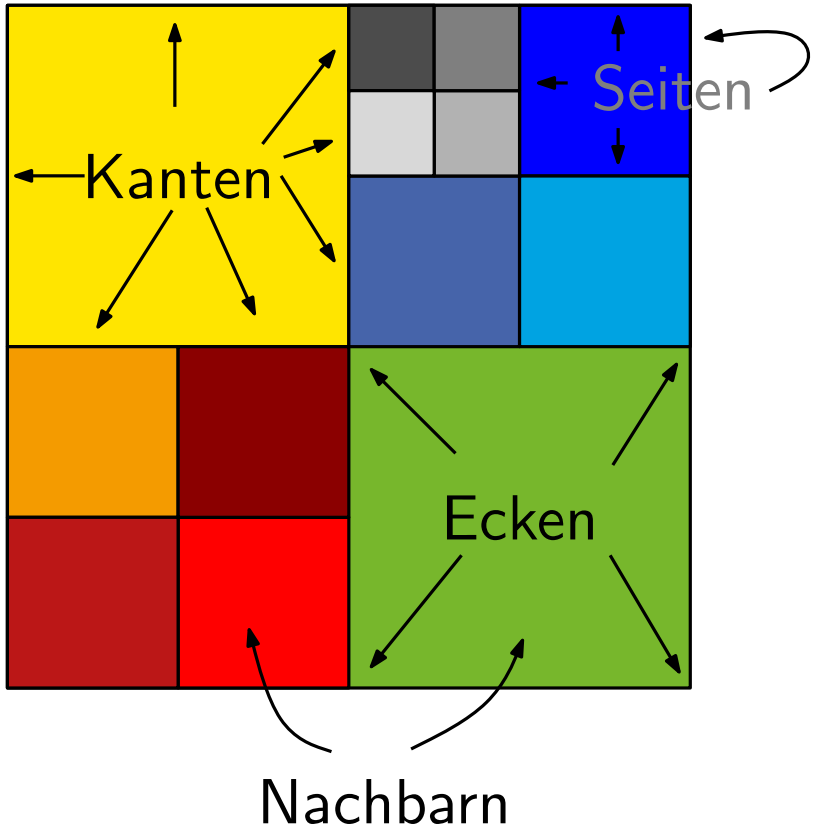
Def.: Ein **Quadtree** ist ein Wurzelbaum, in dem jeder innere Knoten vier Kinder hat. Jeder Knoten entspricht einem Quadrat und die Quadrate der Blätter bilden eine Unterteilung des Wurzelquadrats.

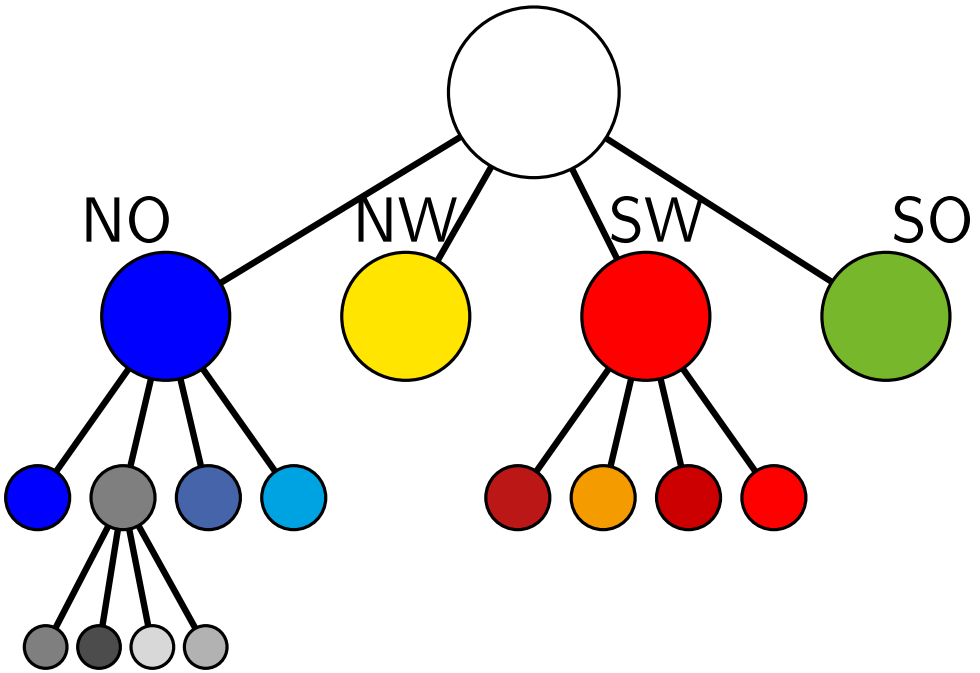




Quadtrees

Def.: Ein **Quadtree** ist ein Wurzelbaum, in dem jeder innere Knoten vier Kinder hat. Jeder Knoten entspricht einem Quadrat und die Quadrate der Blätter bilden eine Unterteilung des Wurzelquadrats.





Quadtrees

Def.: Ein **Quadtree** ist ein Wurzelbaum, in dem jeder innere Knoten vier Kinder hat. Jeder Knoten entspricht einem Quadrat und die Quadrate der Blätter bilden eine Unterteilung des Wurzelquadrats.

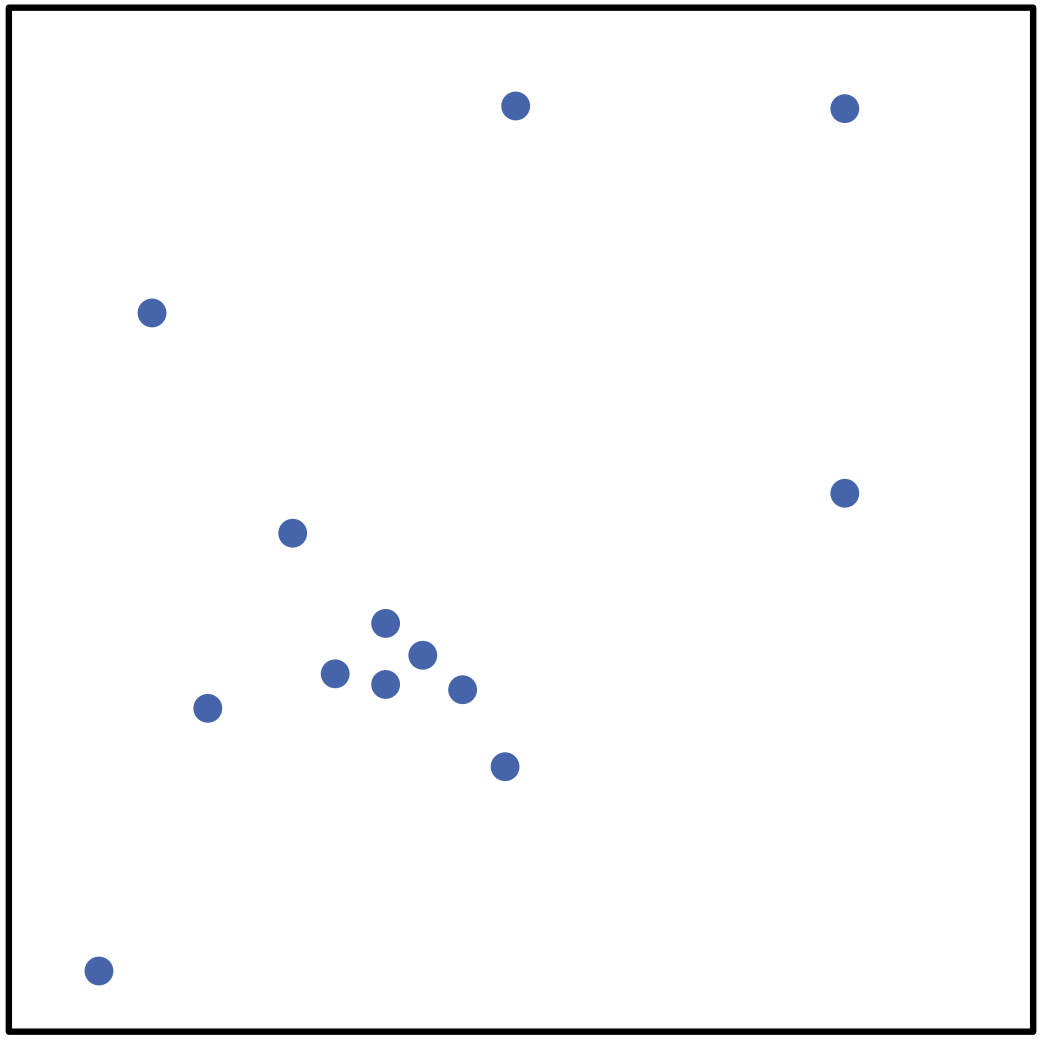
Def.: Für eine Punktmenge P in einem Quadrat $Q = [x_Q, x'_Q] \times [y_Q, y'_Q]$ gilt für den Quadtree $\mathcal{T}(P)$

- ist $|P| \leq 1$, so ist $\mathcal{T}(P)$ ein Blatt, das P und Q speichert
- sonst sei $x_{\text{mid}} = \frac{x_Q + x'_Q}{2}$ und $y_{\text{mid}} = \frac{y_Q + y'_Q}{2}$ und

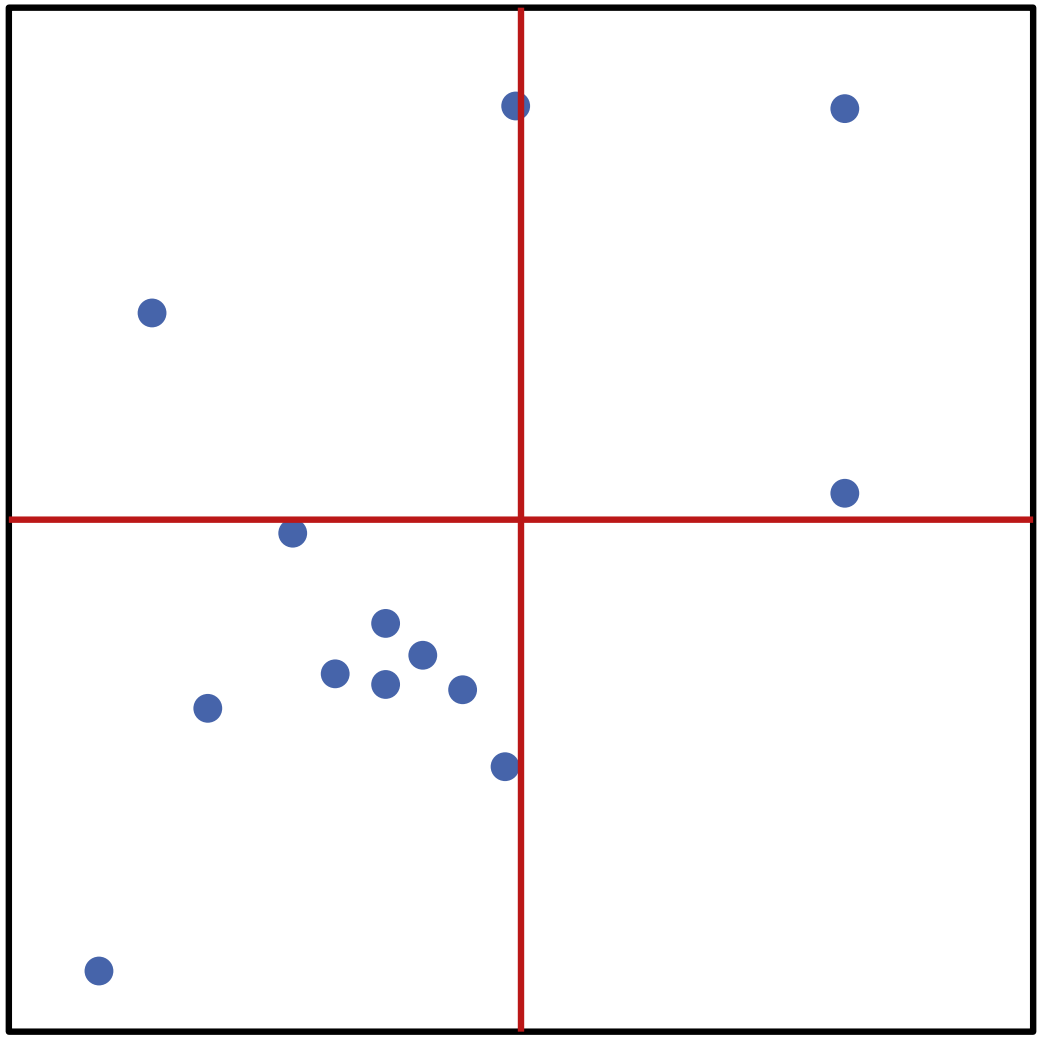
$$
\begin{aligned}
P_{NO} &:= \{p \in P \mid p_x > x_{\text{mid}} \text{ and } p_y > y_{\text{mid}}\} \\
P_{NW} &:= \{p \in P \mid p_x \leq x_{\text{mid}} \text{ and } p_y > y_{\text{mid}}\} \\
P_{SW} &:= \{p \in P \mid p_x \leq x_{\text{mid}} \text{ and } p_y \leq y_{\text{mid}}\} \\
P_{SO} &:= \{p \in P \mid p_x > x_{\text{mid}} \text{ and } p_y \leq y_{\text{mid}}\}
\end{aligned}
$$

$\mathcal{T}(P)$ besteht aus Wurzel v, die Q speichert, mit vier Kindern für P_i und Q_i ($i \in \{NO, NW, SW, SO\}$.

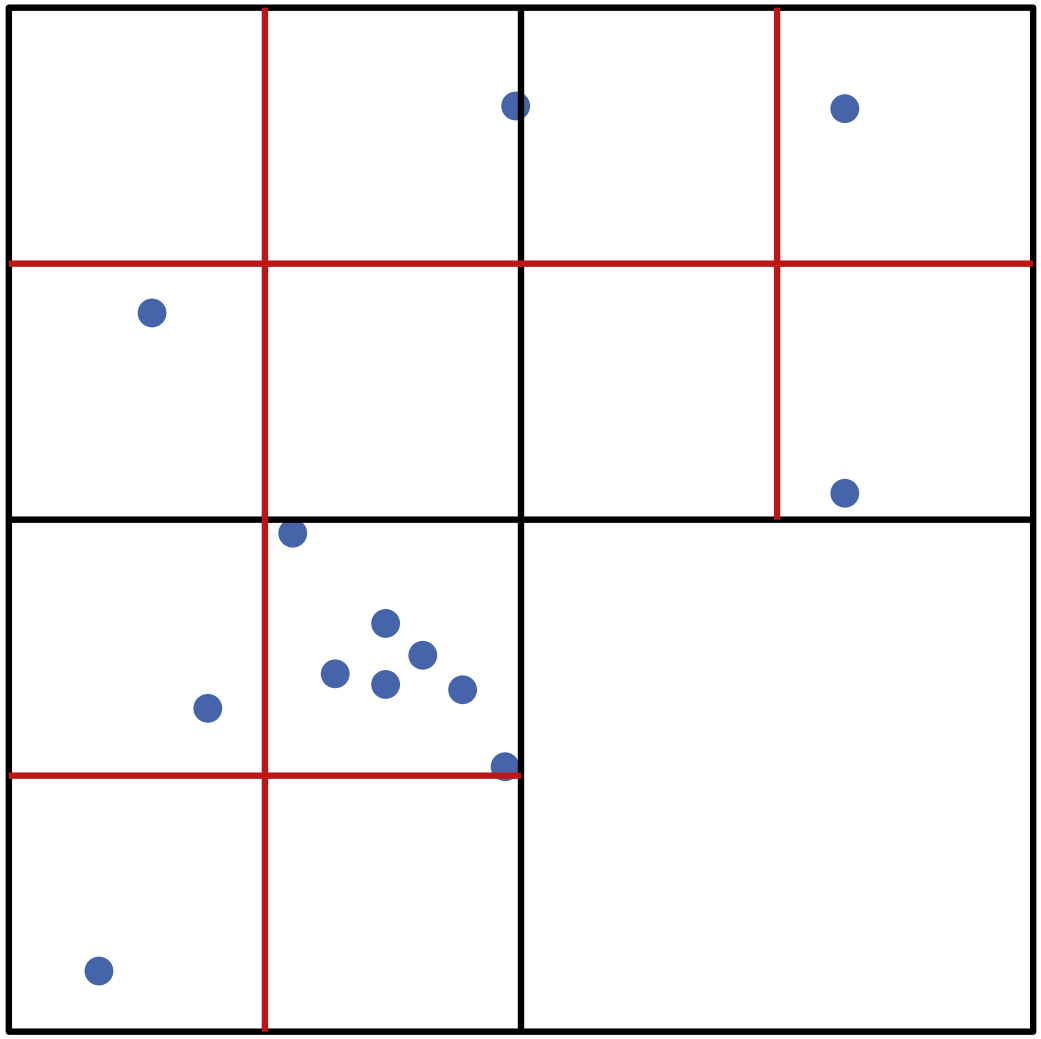
Beispiel



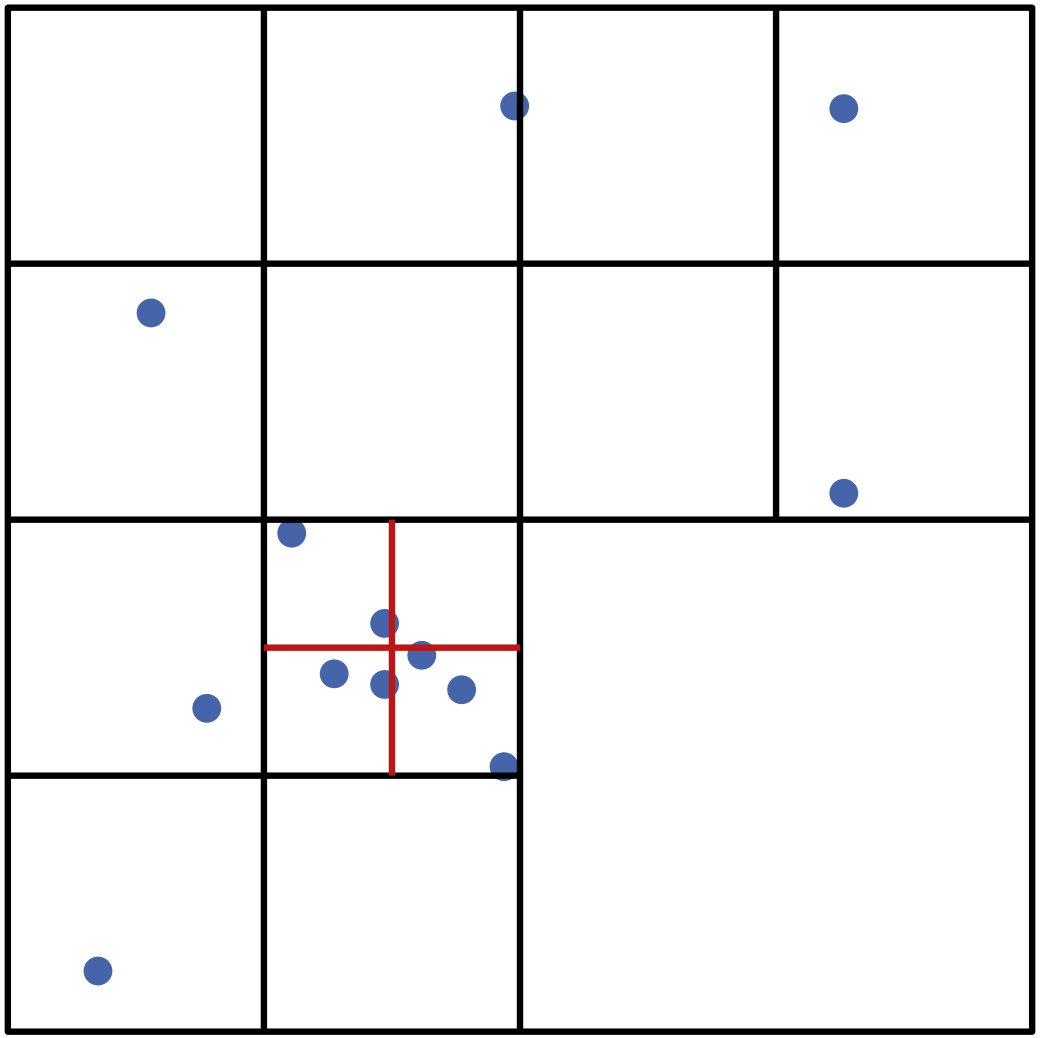
Beispiel



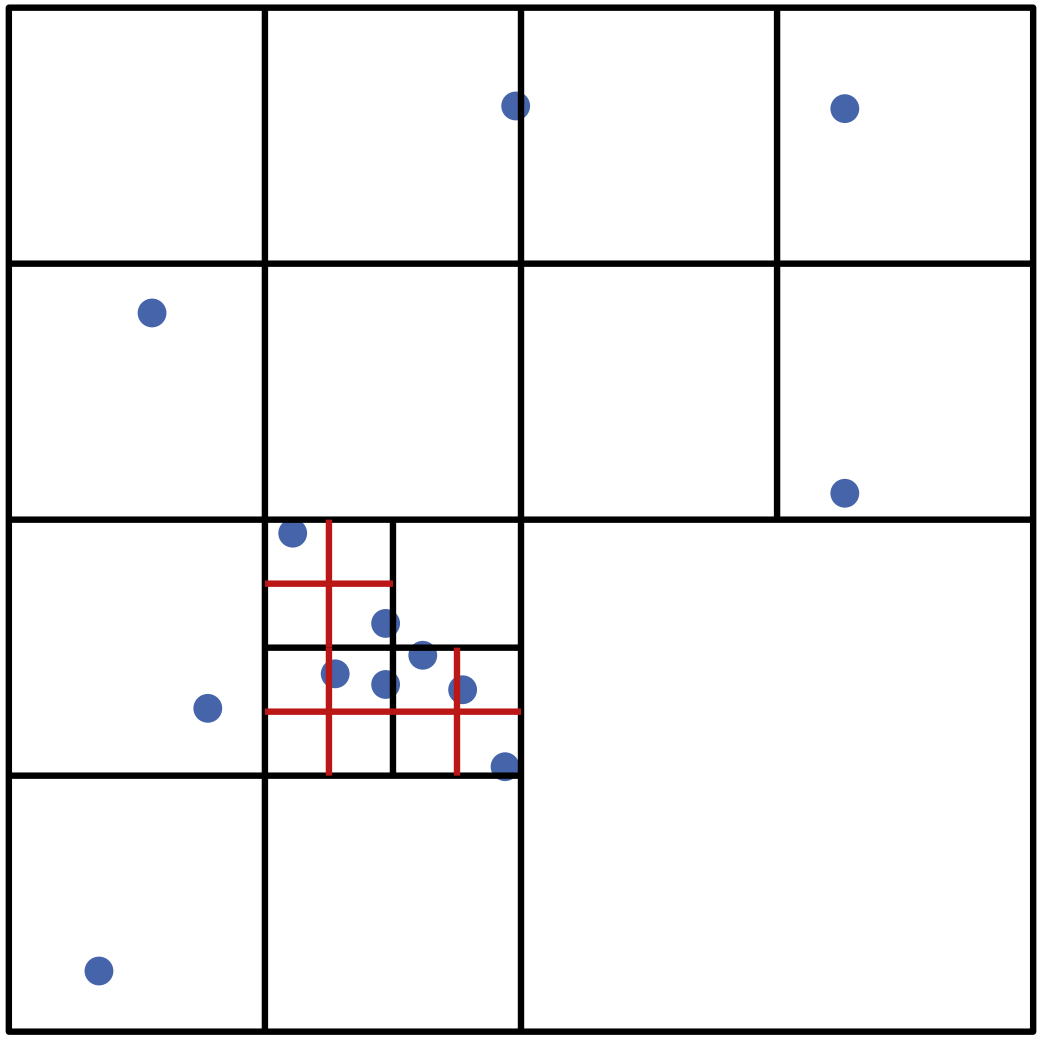
Beispiel



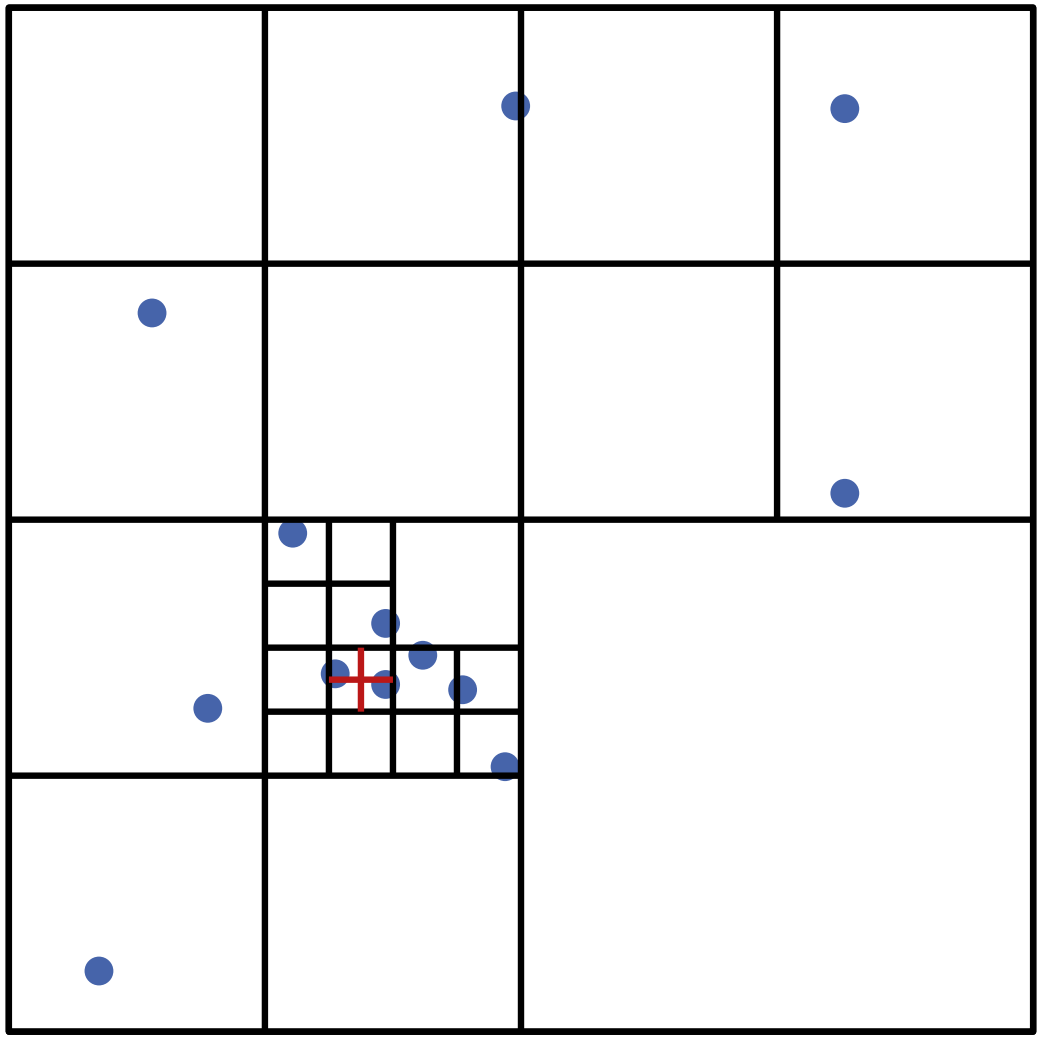
Beispiel



Beispiel



Beispiel



Quadtree Eigenschaften

Die rekursive Definition eines Quadtrees lässt sich direkt in einen Algorithmus zur Konstruktion umsetzen.

Quadtree Eigenschaften

Die rekursive Definition eines Quadtrees lässt sich direkt in einen Algorithmus zur Konstruktion umsetzen.

Welche Tiefe hat ein Quadtree für n Punkte?

Quadtree Eigenschaften

Die rekursive Definition eines Quadtrees lässt sich direkt in einen Algorithmus zur Konstruktion umsetzen.

Welche Tiefe hat ein Quadtree für n Punkte?

Lemma 1: Die Tiefe von $\mathcal{T}(P)$ ist höchstens $\log(s/c) + 3/2$, wobei c der kleinste Abstand in P ist und s die Seitenlänge des Wurzelquadrats Q.

Quadtree Eigenschaften

Die rekursive Definition eines Quadtrees lässt sich direkt in einen Algorithmus zur Konstruktion umsetzen.

Welche Tiefe hat ein Quadtree für n Punkte?

Lemma 1: Die Tiefe von $\mathcal{T}(P)$ ist höchstens $\log(s/c) + 3/2$, wobei c der kleinste Abstand in P ist und s die Seitenlänge des Wurzelquadrats Q.

Satz 1: Ein Quadtree $\mathcal{T}(P)$ für n Punkte und mit Tiefe d hat $O((d+1)n)$ Knoten und kann in $O((d+1)n)$ Zeit konstruiert werden.

Nachbarn finden

NorthNeighbor($v, \mathcal{T}$)

Input: Knoten v in Quadtree $\mathcal{T}$

Output: tiefster Knoten v' nicht tiefer als v mit $v'.Q$ nördl. Nachbar von $v.Q$

if $v = \text{root}(\mathcal{T})$ **then return** nil

$\pi \leftarrow \text{parent}(v)$

if $v = SW\text{-}/SO$-Kind von π **then return** $NW\text{-}/NO$-Kind von π

$\mu \leftarrow$ NorthNeighbor($\pi, \mathcal{T}$)

if $\mu =$ nil oder μ Blatt **then**

  return μ

else

  if $v = NW\text{-}/NO$-Kind von π **then return** $SW\text{-}/SO$-Kind von μ

Nachbarn finden

NorthNeighbor($v, \mathcal{T}$)

Input: Knoten v in Quadtree $\mathcal{T}$

Output: tiefster Knoten v' nicht tiefer als v mit $v'.Q$ nördl. Nachbar von $v.Q$

if $v =$ root($\mathcal{T}$) **then return** nil

$\pi \leftarrow$ parent(v)

if $v = SW$-/SO-Kind von π **then return** NW-/NO-Kind von π

$\mu \leftarrow$ NorthNeighbor($\pi, \mathcal{T}$)

if $\mu =$ nil oder μ Blatt **then**

  return μ

else

  if $v = NW$-/NO-Kind von π **then return** SW-/SO-Kind von μ

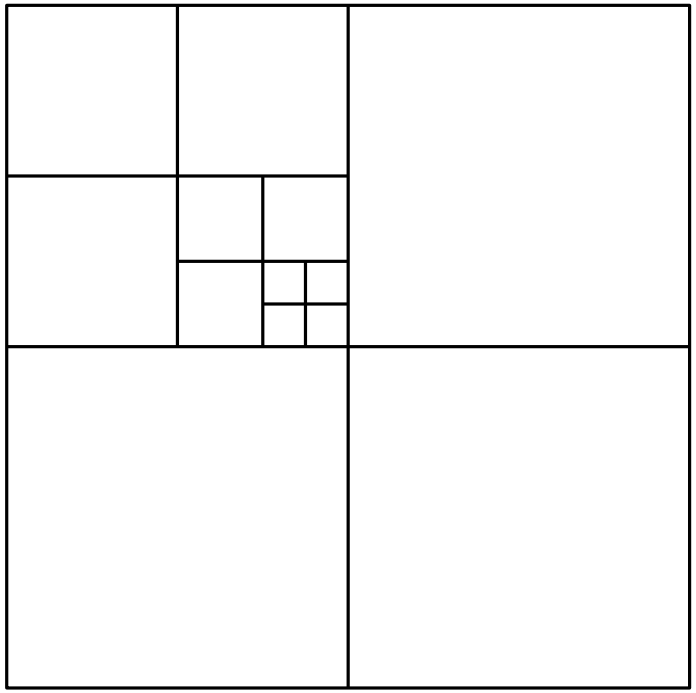
Satz 2: Sei $\mathcal{T}$ ein Quadtree mit Tiefe d. Der Nachbar eines Knotens v in beliebiger Richtung kann in $O(d+1)$ Zeit gefunden werden.

Balancierte Quadtrees

Def.: Ein Quadtree heißt **balanciert**, wenn die Seitenlänge von je zwei Nachbarquadraten in der zugeh. Unterteilung sich um einen Faktor von höchstens 2 unterscheidet.

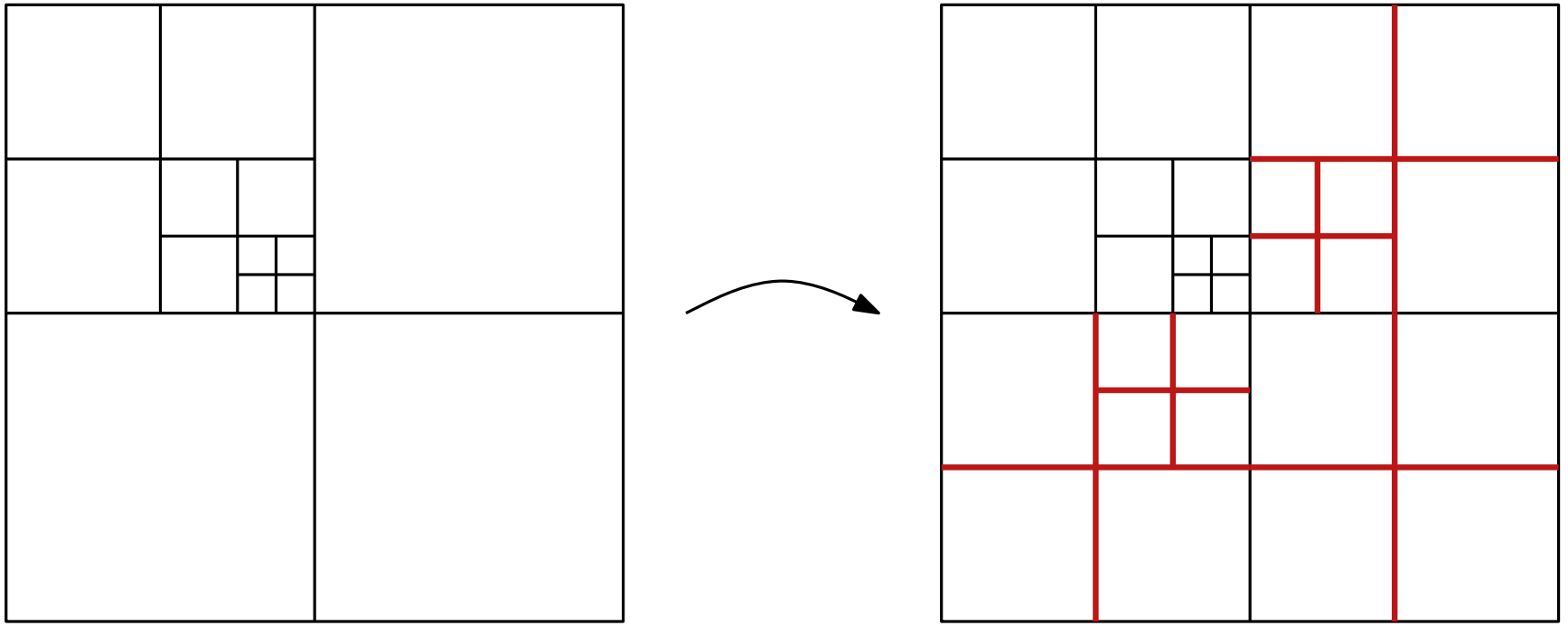
Balancierte Quadtrees

Def.: Ein Quadtree heißt **balanciert**, wenn die Seitenlänge von je zwei Nachbarquadraten in der zugeh. Unterteilung sich um einen Faktor von höchstens 2 unterscheidet.



Balancierte Quadtrees

Def.: Ein Quadtree heißt **balanciert**, wenn die Seitenlänge von je zwei Nachbarquadraten in der zugeh. Unterteilung sich um einen Faktor von höchstens 2 unterscheidet.



Quadtrees balancieren

BalanceQuadtree($\mathcal{T}$)

Input: Quadtree $\mathcal{T}$
Output: balancierter Quadtree $\mathcal{T}$
$L \leftarrow$ Liste aller Blätter von $\mathcal{T}$
while L nicht leer **do**
    $\mu \leftarrow$ extrahiere Blatt aus L
    if $\mu.Q$ zu groß **then**
        Teile $\mu.Q$ in vier Teile und lege vier Blätter in $\mathcal{T}$ an
        füge neue Blätter zu L hinzu
        if $\mu.Q$ hat jetzt zu große Nachbarn **then** füge sie zu L hinzu
    return $\mathcal{T}$

Quadtrees balancieren

BalanceQuadtree($\mathcal{T}$)

Input: Quadtree $\mathcal{T}$
Output: balancierter Quadtree $\mathcal{T}$
$L \leftarrow$ Liste aller Blätter von $\mathcal{T}$
while L nicht leer **do**
  $\mu \leftarrow$ extrahiere Blatt aus L
  if $\mu.Q$ zu groß **then**
    Teile $\mu.Q$ in vier Teile und lege vier Blätter in $\mathcal{T}$ an
    füge neue Blätter zu L hinzu
    if $\mu.Q$ hat jetzt zu große Nachbarn **then** füge sie zu L hinzu
  return $\mathcal{T}$

Wie?

Wie?

Quadtrees balancieren

BalanceQuadtree($\mathcal{T}$)

Input: Quadtree $\mathcal{T}$
Output: balancierter Quadtree $\mathcal{T}$
$L \leftarrow$ Liste aller Blätter von $\mathcal{T}$
while L nicht leer **do**
  $\mu \leftarrow$ extrahiere Blatt aus L
  if $\mu.Q$ zu groß **then**
    Teile $\mu.Q$ in vier Teile und lege vier Blätter in $\mathcal{T}$ an
    füge neue Blätter zu L hinzu
    if $\mu.Q$ hat jetzt zu große Nachbarn **then** füge sie zu L hinzu
  return $\mathcal{T}$

Wie?

Wie?

Wie groß kann ein balancierter Quadtree werden?

Quadtrees balancieren

BalanceQuadtree($\mathcal{T}$)

Input: Quadtree $\mathcal{T}$
Output: balancierter Quadtree $\mathcal{T}$
$L \leftarrow$ Liste aller Blätter von $\mathcal{T}$
while L nicht leer **do**
  $\mu \leftarrow$ extrahiere Blatt aus L
  if $\mu.Q$ zu groß **then**
    Teile $\mu.Q$ in vier Teile und lege vier Blätter in $\mathcal{T}$ an
    füge neue Blätter zu L hinzu
    if $\mu.Q$ hat jetzt zu große Nachbarn **then** füge sie zu L hinzu
  return $\mathcal{T}$

Satz 3: Sei $\mathcal{T}$ ein Quadtree mit m Knoten und Tiefe d. Die balancierte Version $\mathcal{T}_B$ von $\mathcal{T}$ hat $O(m)$ Knoten und kann in $O((d+1)m)$ Zeit konstruiert werden.

Quadtrees und adaptive Dreiecksnetze

Karlsruhe Institute of Technology

Erinnerung:

Geg.: Quadrat $Q = [0, U] \times [0, U]$ für Zweierpotenz $U = 2^j$ mit *oktilinearen*, ganzzahligen Polygonen im Inneren.

Ges.: Dreiecksnetz für Q mit folgenden Eigenschaften

gültig:
- keine Dreiecksknoten im Inneren von Kanten
- Eingabekanten müssen Teil des Dreiecksnetzes sein
- Dreieckswinkel zwischen 45° und 90°

- **adaptiv**, d.h. fein an den Polygonkanten, sonst gröber

Quadtrees und adaptive Dreiecksnetze

Karlsruhe Institute of Technology

Erinnerung:

Geg.: Quadrat $Q = [0, U] \times [0, U]$ für Zweierpotenz $U = 2^j$ mit *oktilinearen*, ganzzahligen Polygonen im Inneren.

Ges.: Dreiecksnetz für Q mit folgenden Eigenschaften

gültig:
- keine Dreiecksknoten im Inneren von Kanten
- Eingabekanten müssen Teil des Dreiecksnetzes sein
- Dreieckswinkel zwischen 45° und 90°

- **adaptiv**, d.h. fein an den Polygonkanten, sonst gröber

Idee: Nutze Quadtree als Ausgangsbasis für das Dreiecksnetz!

Quadtrees und adaptive Dreiecksnetze

Karlsruhe Institute of Technology

Erinnerung:

Geg.: Quadrat $Q = [0, U] \times [0, U]$ für Zweierpotenz $U = 2^j$ mit *oktilinearen*, ganzzahligen Polygonen im Inneren.

Ges.: Dreiecksnetz für Q mit folgenden Eigenschaften

gültig:
- keine Dreiecksknoten im Inneren von Kanten
- Eingabekanten müssen Teil des Dreiecksnetzes sein
- Dreieckswinkel zwischen 45° und 90°
- **adaptiv**, d.h. fein an den Polygonkanten, sonst gröber

Idee: Nutze Quadtree als Ausgangsbasis für das Dreiecksnetz!

Was muss angepasst werden?

Quadtrees und adaptive Dreiecksnetze

Karlsruhe Institute of Technology

Erinnerung:

Geg.: Quadrat $Q = [0, U] \times [0, U]$ für Zweierpotenz $U = 2^j$ mit *oktilinearen*, ganzzahligen Polygonen im Inneren.

Ges.: Dreiecksnetz für Q mit folgenden Eigenschaften

gültig:
- keine Dreiecksknoten im Inneren von Kanten
- Eingabekanten müssen Teil des Dreiecksnetzes sein
- Dreieckswinkel zwischen 45° und 90°
- **adaptiv**, d.h. fein an den Polygonkanten, sonst gröber

Idee: Nutze Quadtree als Ausgangsbasis für das Dreiecksnetz!

Anpassung: Zerteile Quadrate bis sie nicht mehr von Polygon geschnitten werden oder Größe 1 haben

Quadtrees und adaptive Dreiecksnetze

Karlsruhe Institute of Technology

Erinnerung:

Geg.: Quadrat $Q = [0, U] \times [0, U]$ für Zweierpotenz $U = 2^j$ mit *oktilinearen*, ganzzahligen Polygonen im Inneren.

Ges.: Dreiecksnetz für Q mit folgenden Eigenschaften

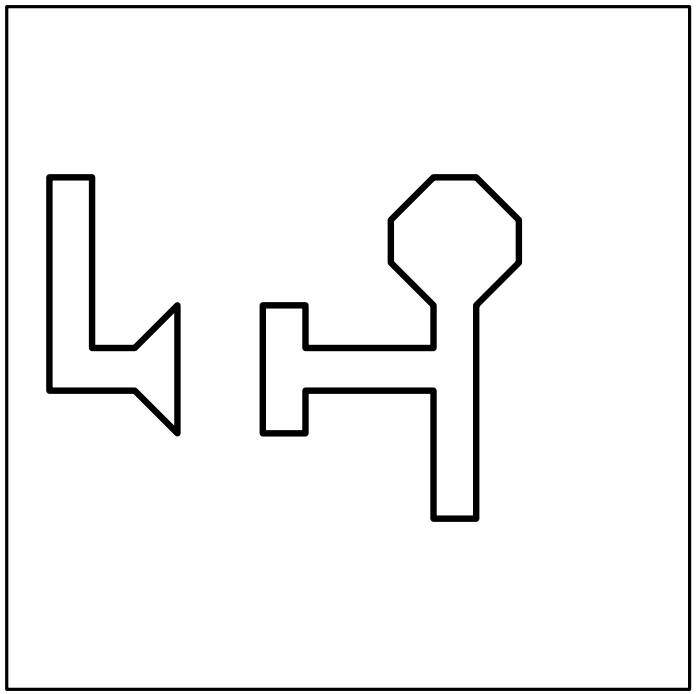
gültig:
- keine Dreiecksknoten im Inneren von Kanten
- Eingabekanten müssen Teil des Dreiecksnetzes sein
- Dreieckswinkel zwischen 45° und 90°
- **adaptiv**, d.h. fein an den Polygonkanten, sonst gröber

Quadrate und Kanten sind abgeschlossen

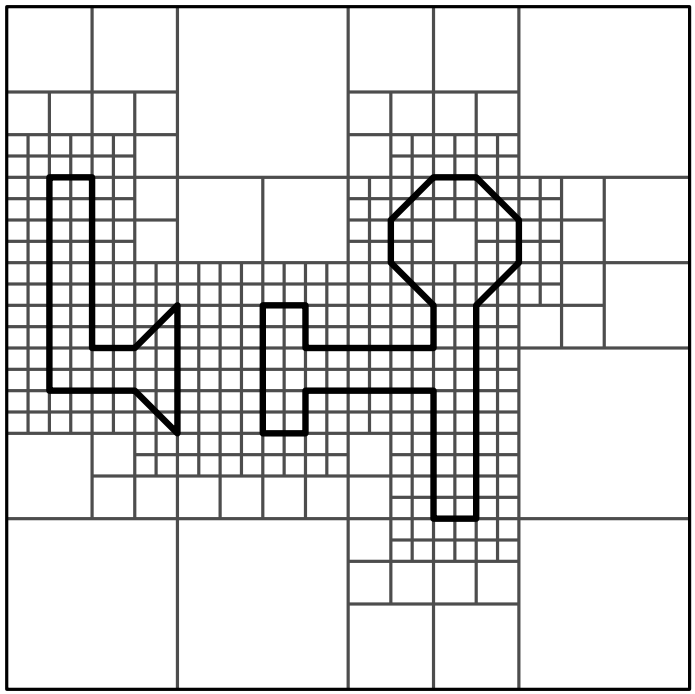
Idee: Nutze Quadtree als Ausgangsbasis für das Dreiecksnetz!

Anpassung: Zerteile Quadrate bis sie nicht mehr von Polygon geschnitten werden oder Größe 1 haben

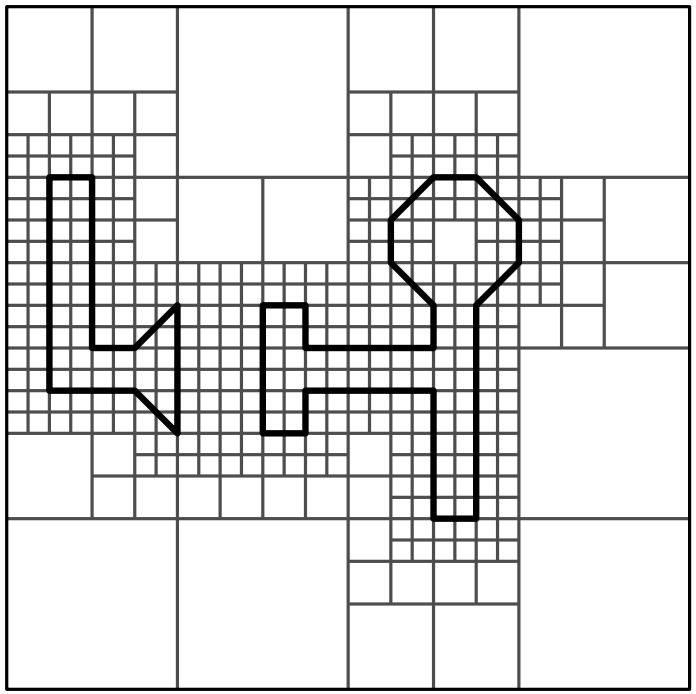
Vom Quadtree zum Dreiecksnetz



Vom Quadtree zum Dreiecksnetz

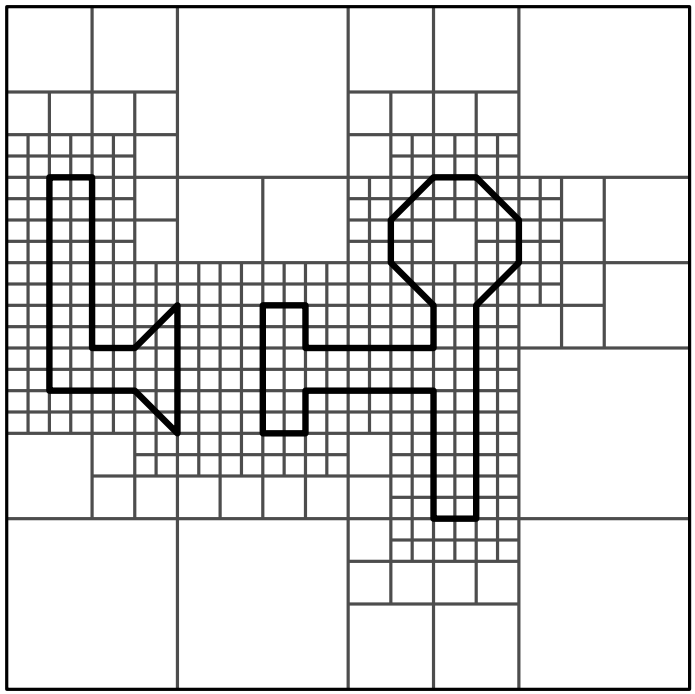


Vom Quadtree zum Dreiecksnetz



Beob.: wenn das Innere eines Quadrats im Quadtree geschnitten wird, dann durch Diagonale

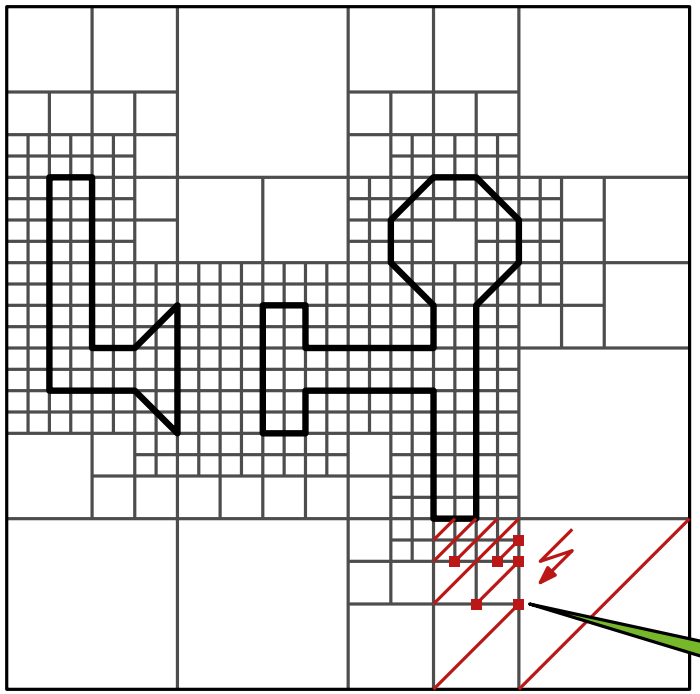
Vom Quadtree zum Dreiecksnetz



Beob.: wenn das Innere eines Quadrats im Quadtree geschnitten wird, dann durch Diagonale

Wie bekommt man ein gültiges Dreiecksnetz?

Vom Quadtree zum Dreiecksnetz



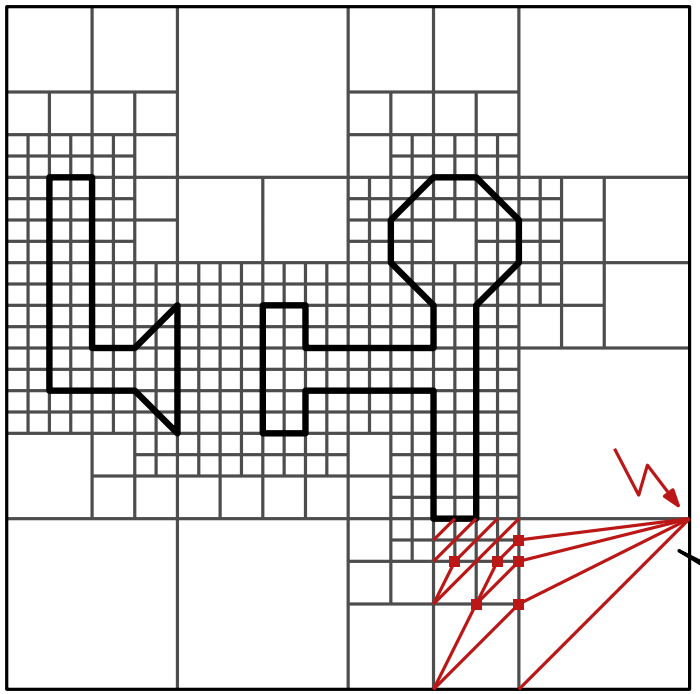
Beob.: wenn das Innere eines Quadrats im Quadtree geschnitten wird, dann durch Diagonale

Wie bekommt man ein gültiges Dreiecksnetz?

Knoten im Kanteninneren

- Diagonalen für alle verbleibenden Quadrate?

Vom Quadtree zum Dreiecksnetz



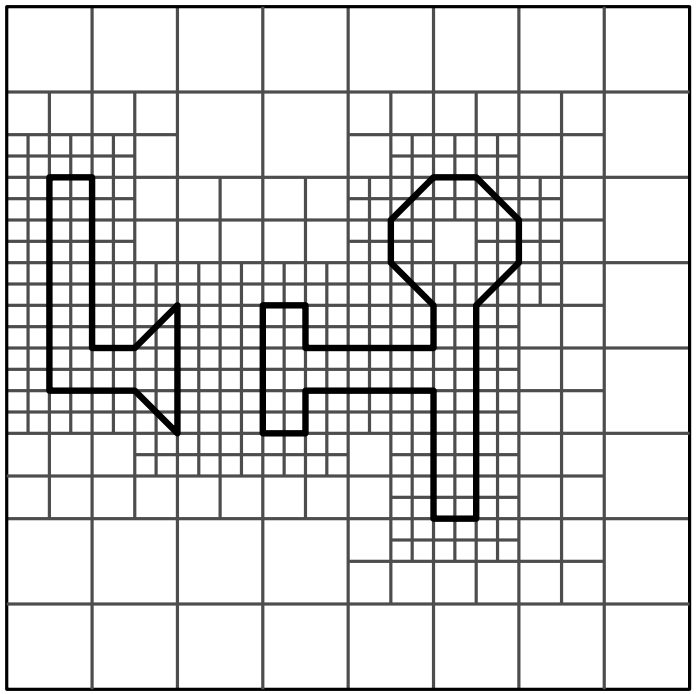
Beob.: wenn das Innere eines Quadrats im Quadtree geschnitten wird, dann durch Diagonale

Wie bekommt man ein gültiges Dreiecksnetz?

zu kleine Dreieckswinkel

- Diagonalen für alle verbleibenden Quadrate? nein!
- Nutze Unterteilungsknoten in Triangulierung?

Vom Quadtree zum Dreiecksnetz

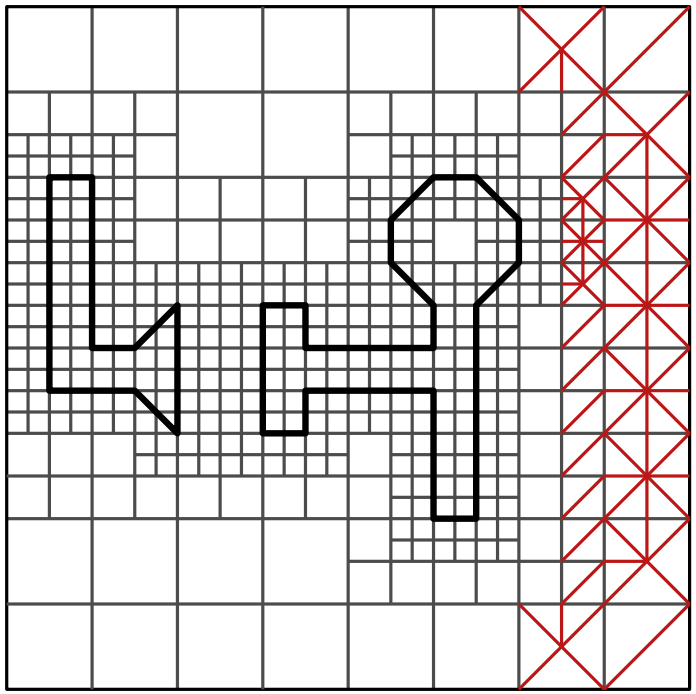


Beob.: wenn das Innere eines Quadrats im Quadtree geschnitten wird, dann durch Diagonale

Wie bekommt man ein gültiges Dreiecksnetz?

- Diagonalen für alle verbleibenden Quadrate? nein!
- Nutze Unterteilungsknoten in Triangulierung? nein!
- Balanciere Quadtree und füge ggf. einen Steinerknoten ein!

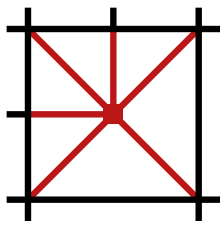
Vom Quadtree zum Dreiecksnetz



Beob.: wenn das Innere eines Quadrats im Quadtree geschnitten wird, dann durch Diagonale

Wie bekommt man ein gültiges Dreiecksnetz?

- Diagonalen für alle verbleibenden Quadrate? nein!
- Nutze Unterteilungsknoten in Triangulierung? nein!
- Balanciere Quadtree und füge ggf. einen Steinerknoten ein!



Algorithmus

ErzeugeMesh(S)

Input: Menge S oktilinearer, ganzzahliger Polygone in $Q = [0, 2^j] \times [0, 2^j]$

Output: gültiges, adaptives Dreiecksnetz für S

$\mathcal{T} \leftarrow$ CreateQuadtree

$\mathcal{T} \leftarrow$ BalanceQuadtree($\mathcal{T}$)

$\mathcal{D} \leftarrow$ DCEL für Unterteilung von Q durch $\mathcal{T}$

foreach Facette f in $\mathcal{D}$ **do**

  if int(f) $\cap S \neq \emptyset$ **then**

    füge entsprechende Diagonale in f zu $\mathcal{D}$ hinzu

  else

    if Knoten nur an Ecken von f **then**

      füge eine Diagonale in f zu $\mathcal{D}$ hinzu

    else

      erzeuge Steiner-Punkt in der Mitte von f und verbinde in $\mathcal{D}$ zu allen Knoten auf ∂f

return $\mathcal{D}$

Zusammenfassung

Satz 4: Für eine Menge S disjunkter, oktilinearer, ganzzahliger Polygone mit Gesamtumfang $p(S)$ in einem Quadrat $Q = [0, U] \times [0, U]$ kann in $O(p(S) \log^2 U)$ Zeit ein gültiges adaptives Dreiecksnetz mit $O(p(S) \log U)$ Dreiecken erzeugt werden.

Diskussion

Gibt es auch Quadtree-Varianten mit linearer Größe in n?

Diskussion

Gibt es auch Quadtree-Varianten mit linearer Größe in n?

Ja, wenn man innere Knoten mit nur einem nicht-leeren Kind kontrahiert bekommt man einen sog. *compressed Quadtree*; eine weitere Verbesserung sind *Skip Quadtrees* mit Größe $O(n)$ und Einfügen, Löschen und Suchen in $O(\log n)$ Zeit [Eppstein et al., '05]

Diskussion

Gibt es auch Quadtree-Varianten mit linearer Größe in n?

Ja, wenn man innere Knoten mit nur einem nicht-leeren Kind kontrahiert bekommt man einen sog. *compressed Quadtree*; eine weitere Verbesserung sind *Skip Quadtrees* mit Größe $O(n)$ und Einfügen, Löschen und Suchen in $O(\log n)$ Zeit [Eppstein et al., '05]

Welche weiteren Anwendungen gibt es?

Diskussion

Gibt es auch Quadtree-Varianten mit linearer Größe in n?

Ja, wenn man innere Knoten mit nur einem nicht-leeren Kind kontrahiert bekommt man einen sog. *compressed Quadtree*; eine weitere Verbesserung sind *Skip Quadtrees* mit Größe $O(n)$ und Einfügen, Löschen und Suchen in $O(\log n)$ Zeit [Eppstein et al., '05]

Welche weiteren Anwendungen gibt es?

Quadtrees werden in der Praxis häufig eingesetzt in Computergrafik, Bildverarbeitung, GIS etc., u.a. für Range Queries, obwohl sie theoretisch schlechter abschneiden als z.B. Range Trees oder kd-Trees.

Diskussion

Gibt es auch Quadtree-Varianten mit linearer Größe in n?

Ja, wenn man innere Knoten mit nur einem nicht-leeren Kind kontrahiert bekommt man einen sog. *compressed Quadtree*; eine weitere Verbesserung sind *Skip Quadtrees* mit Größe $O(n)$ und Einfügen, Löschen und Suchen in $O(\log n)$ Zeit [Eppstein et al., '05]

Welche weiteren Anwendungen gibt es?

Quadtrees werden in der Praxis häufig eingesetzt in Computergrafik, Bildverarbeitung, GIS etc., u.a. für Range Queries, obwohl sie theoretisch schlechter abschneiden als z.B. Range Trees oder kd-Trees.

Wie immer: höhere Dimensionen?

Diskussion

Gibt es auch Quadtree-Varianten mit linearer Größe in n?

Ja, wenn man innere Knoten mit nur einem nicht-leeren Kind kontrahiert bekommt man einen sog. *compressed Quadtree*; eine weitere Verbesserung sind *Skip Quadtrees* mit Größe $O(n)$ und Einfügen, Löschen und Suchen in $O(\log n)$ Zeit [Eppstein et al., '05]

Welche weiteren Anwendungen gibt es?

Quadtrees werden in der Praxis häufig eingesetzt in Computergrafik, Bildverarbeitung, GIS etc., u.a. für Range Queries, obwohl sie theoretisch schlechter abschneiden als z.B. Range Trees oder kd-Trees.

Wie immer: höhere Dimensionen?

Quadtrees lassen sich problemlos auf höhere Dimensionen verallgemeinern. Sie heißen dann auch Octrees.