Triangulierung

häufige Aufgabe
Motivation:

- Approximation komplizierter Geometrien durch einfachere
- Dreiecke oft effizienter zu bearbeiten als Polygone

Problemstellung 1:
Gegeben: ein einfaches Polygon P
Gesucht: eine Zerlegung von P in Dreiecke ohne Hinzufügen neuer Punkte

wieviele Dreiecke braucht man?

P habe n Ecken
t = gesuchte Zahl der Dreiecke

in der Triangulierung gehört jede Innenkante zu genau 2 Dreiecken und jede der n Außenkanten zu einem Dreieck
$\Rightarrow$ Gesamt-Kantenzahl $e = (3t + n)/2$

Eulersche Polyederformel (Facetten: t Dreiecke und 1 Außengebiet):
$v - e + f = 2 \Rightarrow n - (3t+n)/2 + t+1 = 2 \Rightarrow t = n - 2$

$\Rightarrow$ die Zahl der Dreiecke liegt fest!

(Formel gilt nur, wenn P keine Löcher hat)

Wie findet man eine beliebige Triangulierung?

Spezialfälle:

(a) P konvex

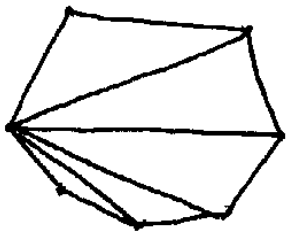
Zur Erinnerung:
Eine Teilmenge des R^n heißt *konvex*, wenn mit je 2 Punkten a, b immer auch ihre Verbindungsstrecke ab
(als Formel: $ab = \{ \lambda a + (1-\lambda) b \mid 0 \leq \lambda \leq 1 \}$)
zur Menge gehört. (Gegenteil: *konkav*.)

Eine Ecke eines Polygons heißt konvex, wenn das Polygon in einer Umgebung dieser Ecke konvex ist:

oben: konvexe Ecke, unten: konkave Ecke

Für konvexe Polygone: Triangulierungsaufgabe ist trivial

- wähle eine beliebige Ecke, verbinde sie mit allen nicht benachbarten Ecken



(b) *P* monoton

Ein einfaches Polygon heißt *monoton*, wenn es eine Richtung bzw. Gerade gibt, so dass das Polygon in zwei *monotone Ketten* bzgl. dieser Geraden zerlegbar ist.
Ein Kantenzug (Kette) heißt monoton bzgl. einer Geraden *g*, wenn alle zu *g* senkrechten Geraden den Kantenzug in genau einem Punkt schneiden.

Monotone Polygone können mit einem einfachen Algorithmus in der Zeit $O(n)$ (n = Eckenzahl) trianguliert werden:

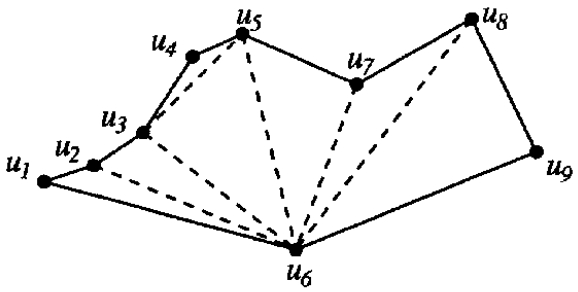
Sei o.B.d.A. die x-Achse die Gerade, bzgl. derer Monotonie vorliegt (sonst entspr. Rotation vorschalten).

- Sortiere die Eckpunkte nach aufsteigender x-Koordinate: (u_1, u_2, ..., u_n)
- Prädikat *Nachbar*(u, v): erfüllt, wenn u und v durch Kante verbunden (in P oder erst während der Triangulation)
- Verwendung eines Stacks (e_1, e_2, ..., e_i), e_i oberstes Element

Algorithmus: Triangulation monotoner Polygone

```
begin
    (* Initialisierung *)
    Lege ersten und zweiten Punkt aus P auf den Stack;
    u :=dritter Punkt in P;
    while ¬(Nachbar(u, e_1) ∧ Nachbar(u, e_i)) do begin
      if (Nachbar(u, e_1) ∧ ¬Nachbar(u, e_i))
        then begin
          Füge Kanten ue_1, ..., ue_i ein;
          Ersetze Stackinhalt durch e_i, u;
          u :=Nachfolger(u) in P;
        end;
      else
        begin
          if (i > 1) ∧ ((e_{i-1} e_i u) ist konvexe Ecke)
            then begin
              (* Fall 2a *)
              Füge Kante ue_{i-1} ein;
              Lösche e_i im Stack;
            end;
          else
            begin
              (* Fall 2b *)
              Lege u auf den Stack;
              u :=Nachfolger(u) in P;
            end;
        end;
    end; (* while *)
    (* Fall 3 : Nachbar(u, e_1) ∧ Nachbar(u, e_i) *)
    Füge Kanten ue_2, ..., ue_{i-1} ein;
end;
```

Im Algorithmus werden alle konvexen Ecken abgeschnitten (Fall 2a), so dass der Stack nur nichtkonvexe Ecken speichert. Wenn eine Ecke auf der "Gegenseite" (Nachbar von e_1) gefunden wird (Fall 1), werden durch Einfügen von Kanten und Abschneiden der entstehenden Dreiecke alle Stackelemente (bis auf das oberste) gelöscht.
Beispiel:



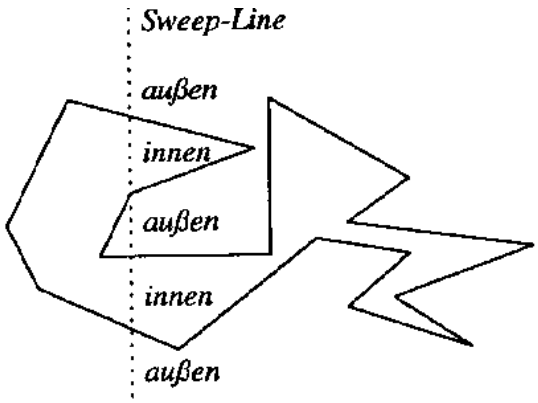
Stack	*u*	*Fall*
u_1, u_2	u_3	*2b*
u_1, u_2, u_3	u_4	*2b*
u_1, u_2, u_3, u_4	u_5	*2a*
u_1, u_2, u_3	u_5	*2b*
u_1, u_2, u_3, u_5	u_6	*1*
u_5, u_6	u_7	*1*
u_6, u_7	u_8	*2a*
u_6	u_8	*3*

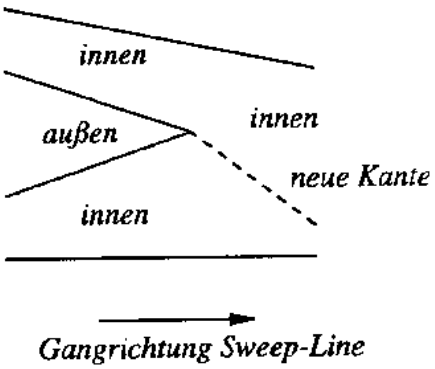
(c) Triangulierung einfacher Polygone (nicht notwendig monoton)

- Sweep-Line-Algorithmus von Garey, Johnson, Preparata und Tarjan:

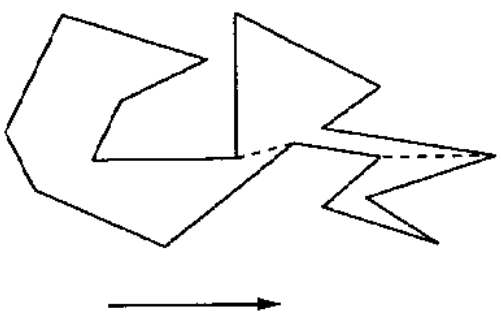
- Zerlegung des Polygons in monotone Teilpolygone (bzgl. *x*-Achse)
- diese werden dann nach obigem Verfahren trianguliert.

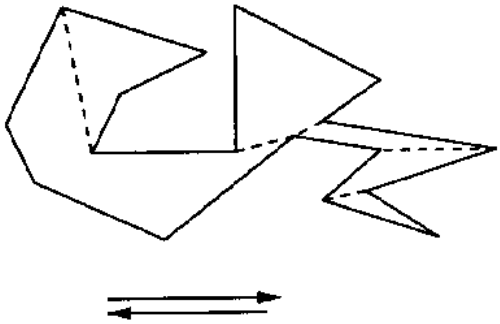
Monotone Zerlegung erfolgt durch zweimaliges Herüberschwenken einer "Sweep Line" (Gerade senkrecht zur *x*-Achse) über *P*: einmal in zunehmender, einmal in abnehmender *x*-Richtung.
Entlang der Sweep Line wird abgespeichert, ob das gerade durchlaufene Gebiet innerhalb oder außerhalb von *P* liegt.
Durch Einfügen von Kanten während des Herüberschwenkens werden monotone Teilpolygone erzeugt.





Einfügen einer neuen Kante: an den Punkten, wo ein Außengebiet durch Zusammenlaufen zweier Kanten zu einem gemeinsamen Eckpunkt verschwindet, wird eine neue Kante von diesem Eckpunkt zum in Gangrichtung nächsten Punkt des umschließenden Innengebiets gezogen (⇒ Überkreuzungsfreiheit von Trennkanten und Polygonkanten).

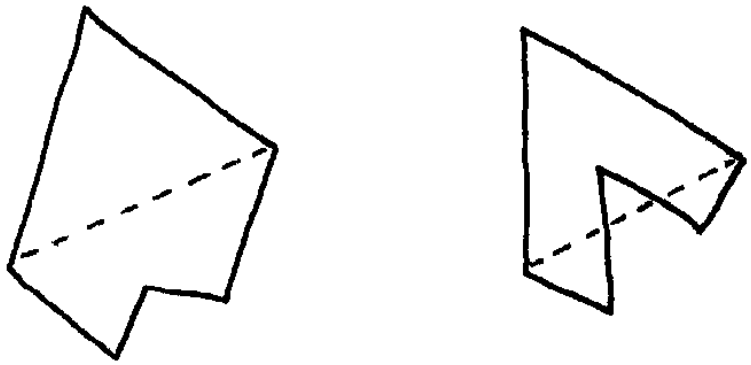




Zeitkomplexität des Verfahrens: $O(n \log n)$

- Algorithmus von Kong

- direkte Triangulierung einfacher Polygone durch sukzessives Abschneiden konvexer Ecken (vgl. Algorithmus für monotone Polygone)
- Problem: das aus einer konvexen Ecke gebildete Dreieck kann dennoch weitere Polygon-Eckpunkte enthalten
- wenn das nicht der Fall ist ("günstiger Fall"), nennen wir die Ecke ein "Ohr".



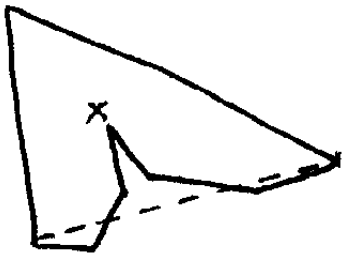
links: Ohr, rechts: konvexe Ecke, die kein Ohr ist

Vorgehensweise des Algorithmus von Kong:
dem Polygon werden sukzessive die Ohren abgeschnitten

dazu Unterprozedur: Test, ob eine Ecke *x* ein Ohr ist
P sei als doppelt verkettete Liste der Ecken gespeichert
die Liste *R* enthalte alle nichtkonvexen Ecken von *P*

```
Funktion IstOhr (x)
begin
    if R = ∅
      then wahr;        (* P ist konvex *)
      else
        if x ist konvexe Ecke then
          if Inneres von △(pred(x), x, succ(x)) enthält keinen Punkt aus R
            then wahr;
            else falsch;
        else falsch;
end;
```

dies funktioniert, weil eine konvexe Ecke, die nicht Ohr ist, immer mindestens eine nichtkonvexe Ecke enthält. Beispiel:



Essentiell für das Funktionieren des Algorithmus von Kong ist der
Satz:
In einem einfachen Polygon mit mehr als 3 Ecken existieren immer mindestens 2 disjunkte Ohren. (2 Ohren sind disjunkt, wenn sie keine gemeinsame Kante haben.)

Beweis: durch vollst. Induktion über die Eckenzahl (siehe Schmitt et al. 1996).

Algorithmus von Kong:
Preprocessing: Bestimmung der Liste *R* der nichtkonvexen Eckpunkte (für den Ohrentest)
Hauptprogramm: schneide solange Ohren ab, bis nur noch ein Dreieck übrigbleibt

- die kritische Operation ist der Ohrentest (kann durch Zellraster-Techniken beschleunigt werden)
- Gesamt-Zeitkomplexität: $O(n^2)$
- besonders geeignet für Polygone mit geringer Zahl nichtkonvexer Ecken

- Triangulierung mit Sweep-Verfahren und "Sacktechnik"

- Algorithmus für einfache Polygone, auch mit Löchern
- Sweep-Verfahren mit nur einem Durchgang
- aber komplizierte Fallunterscheidungen während des Sweep-Durchlaufs ("Sackerweiterung", "-teilung", "-schließung", "-vereinigung")
- Zeitkomplexität $O(n \log n)$

- Zeitoptimales Verfahren

es gelang lange Zeit nicht, die vermutete untere Schranke von $n \log n$ für die Triangulierung einfacher Polygone zu beweisen

Den Grund fand Chazelle 1990 nach intensiver Forschung:

Satz:
Einfache Polygone können mit einem Zeitaufwand $O(n)$ trianguliert werden, und das ist optimal.

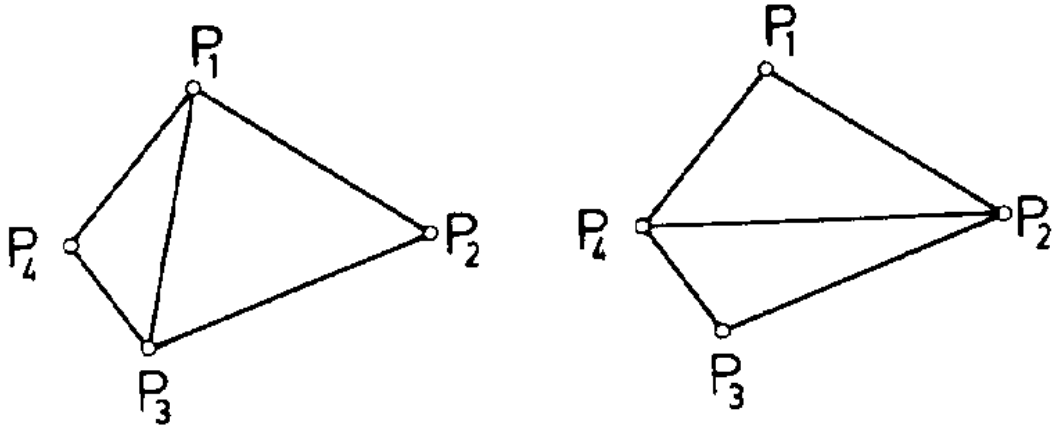
Beweis durch Angabe eines Algorithmus, der aber für die praktische Verwendung zu kompliziert ist.
Offene Frage: gibt es auch einen einfachen, zeitlinearen Algorithmus?

Häufig will man nicht 1 Polygon zerlegen, sondern Datenpunkte so durch Kanten verbinden, dass (nur) Dreiecke entstehen.

Problemstellung 2:
Gegeben: eine endliche Menge *P* von Punkten in der Ebene
Gesucht: eine Zerlegung der konvexen Hülle von *P* (kleinste konvexe Menge, die *P* enthält: konvexes "Hüllpolygon" von *P*) in Dreiecke, so dass die Elemente von *P* die Eckpunkte der Dreiecke sind.
Zusatzforderungen:
- Summe der Kantenlängen minimal
- Dreiecke "möglichst gleichschenklig"

Anwendungen: Geodäsie, Finite-Elemente-Methode (FEM).

Schon bei 4 Punkten gibt es i. allg. zwei verschiedene Lösungen:



verschiedene Optimalitätskriterien für Dreieckszerlegungen:

- minimale Kantenlängensumme

Nachteil: verhindert nicht die Erzeugung langer, dünner Dreiecke (ungünstig bei FEM u. Visualisierung)

- Max-Min-Winkelkriterium: der kleinste vorkommende Dreieckswinkel wird maximiert

- Min-Max-Winkelkriterium: der größte vorkommende Dreieckswinkel wird minimiert

- Max-Min-Radiuskriterium: der kleinste Radius der in die Dreiecke einbeschriebenen Kreise wird maximiert

- Min-Max-Radiuskriterium: der größte Radius der in die Dreiecke einbeschriebenen Kreise wird minimiert

- Max-Min-Flächenkriterium: der kleinste Flächeninhalt der Dreiecke wird maximiert

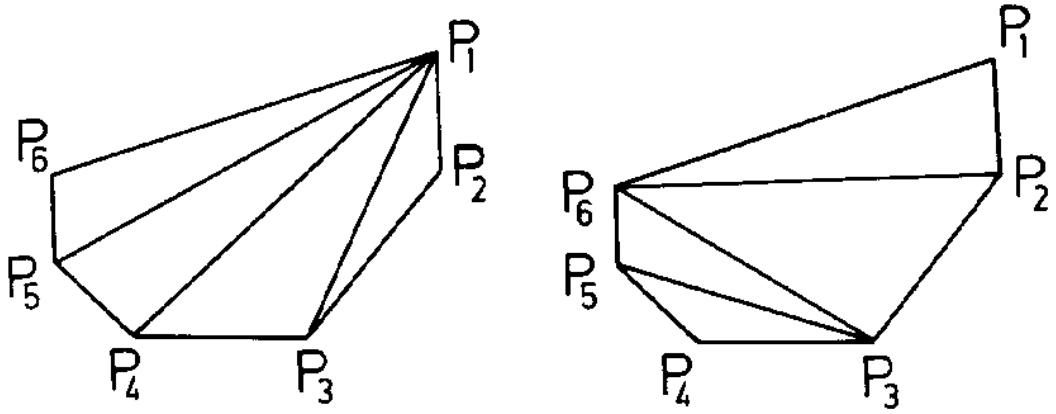
- Max-Min-Höhenkriterium: die kleinste Höhe der Dreiecke wird maximiert

Alle Kriterien können (in speziellen Fällen) unterschiedliche Triangulierungen liefern!

Eine Triangulierung *T* heißt *lokal optimal* bzgl. eines Kriteriums *K*, wenn jedes Viereck, definiert durch je 2 entlang einer gemeinsamen Kante aneinandergrenzende Dreiecke von *T*, bzgl. *K* optimal trianguliert ist.
Eine Triangulierung der Punktmenge *P* heißt *global optimal* bzgl. *K*, wenn jede andere Triangulierung von *P* ungünstiger als *T* bzgl. *K* ist.

Eine Punktmenge kann mehrere lokal optimale Triangulierungen haben:

Zwei bzgl. des Min-Max-Winkelkriteriums lokal optimale Triangulierungen von 6 Punkten



(aus Hoschek & Lasser 1992)

Das *Max-Min-Winkelkriterium* ist das einzige bekannte Kriterium, für das lokale Optima stets auch globale Optima sind!

⇒ Auffinden des globalen Optimums dann durch lokale Operationen unabh. von der Vorgehensweise möglich.

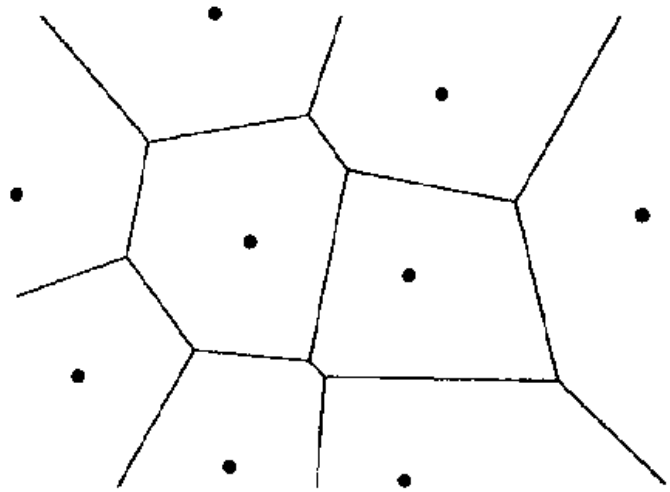
(Aber: Häufig führen das Max-Min- und das Min-Max-Winkelkriterium zur gleichen Triangulierung – vergleichende Tests mit Zufalls-Punktmengen: Abweichungen nur in ca. 10 % der Fälle.)

Die mit dem Max-Min-Winkelkriterium konstruierte Triangulierung ist identisch mit der *Delaunay-Triangulierung*.

Dazu Def.:
Das *Voronoi-Diagramm* (auch: *Dirichlet-Parkettierung*, *Thiessen-Polygon-Zerlegung*) einer *n*-elementigen Punktmenge *P* in der Ebene ist eine Zerlegung der Ebene in *n* disjunkte Gebiete, von denen jedes genau einen der Punkte aus *P* und alle Punkte der Ebene, für die dieser Punkt aus *P* der von allen Punkten aus *P* nächste Nachbar ist, enthält ("Gebiete gleicher nächster Nachbarn").
In Mengenschreibweise (wenn $P = \{ p_1 , ..., p_n \}$):

$$V_i = \{ x \in R^2 \mid d(x, p_i) \leq d(x, p_j) \text{ für alle } j \neq i \}.$$

Die Gebiete V_i sind geradlinig begrenzt und heißen *Dirichlet-Zellen* oder *Thiessen-Polygone* (beachte: V_i kann unbeschränkt sein, also kein Polygon i.e.S.).

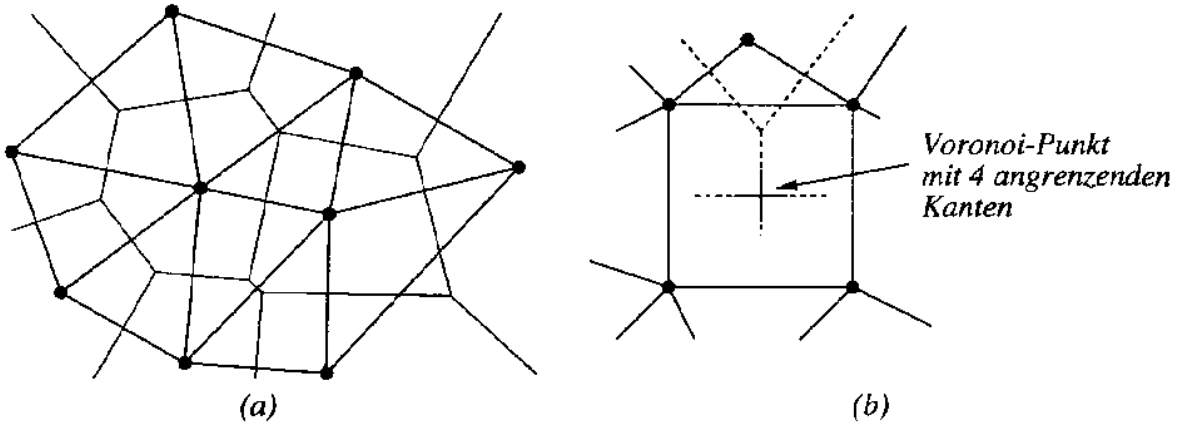


Voronoi-Diagramm von 9 Punkten

Bedeutung: wichtig bei vielen Fragestellungen der Geostatistik, Berechnung von Einzugsgebieten (Ressourcennutzung etc.), optimale Versorgung von Flächen von endlich vielen Quellpunkten aus, min. aufspannende Bäume usw.

Konstruktion:
- jedes einzelne Thiessen-Polygon durch Konstruktion der Mittelsenkrechten je zweier Punkte, dann alle Polygone zusammenfügen (Aufwand $O(n^3)$)
- mit geschickten divide-and-conquer-Algorithmen Aufwand auf $O(n^2 \log n)$ reduzierbar (s. Schmitt et al. 1996), und dies ist optimal.

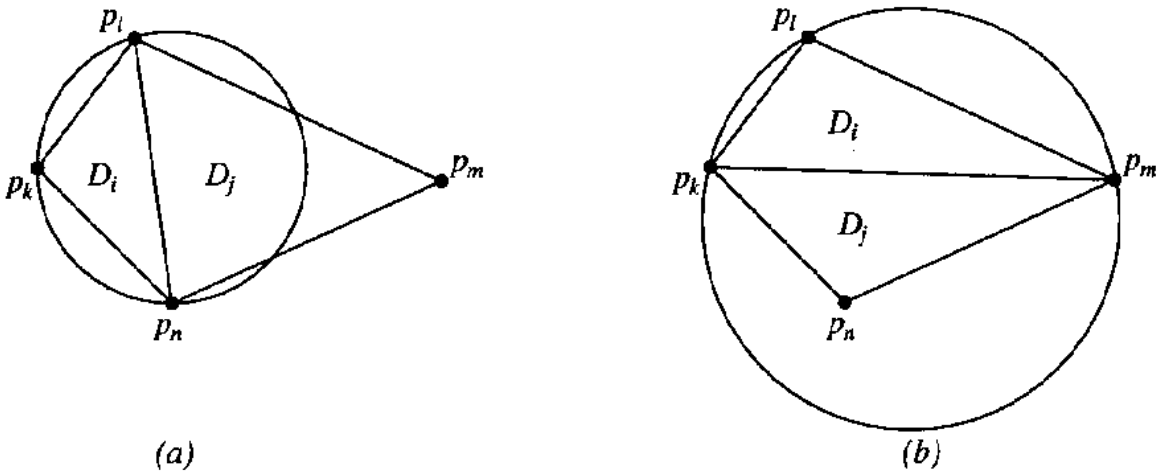
Die *duale Parkettierung* zur Voronoi-Parkettierung der Ebene heißt *Delaunay-Parkettierung*:
Ziehe zwischen all den Punktepaaren aus P Verbindungskanten, die eine gemeinsame Kante im Voronoi-Diagramm haben.



links: Delaunay-Parkettierung zu den 9 gegebenen Punkten

Die Delaunay-Parkettierung ist i.allg. eine *Triangulierung* – Ausnahme: wenn Voronoi-Punkte vom Grad > 3 auftreten (Abb. oben rechts).
In diesem Ausnahmefall müssen die entstehenden Polygone mit > 3 Ecken in der Delaunay-Parkettierung nachträglich trianguliert werden
⇒ Delaunay-Triangulierung

Lokales Umkreiskriterium für eine Triangulation:
Für je 4 Punkte, die zu 2 benachbarten Dreiecken der Triangulation gehören, enthält der Umkreis des einen Dreiecks *nicht* den vierten Punkt.



lokales Umkreiskriterium: links: erfüllt, rechts: nicht erfüllt

globales Umkreiskriterium: für sämtliche Dreiecke der Triangulierung enthält ihr Umkreis keine Punkte aus *P*.

lokales Umkreiskrit. überall erfüllt ⇒ globales Umkreiskrit. gilt.

- Bei Nichterfüllung des lokalen Umkreiskrit. wird retrianguliert durch Diagonalentausch
- durch solche lokalen Änderungen ist das globale Optimum erreichbar
- Optimum identisch zu Delaunay-Triangulation.

Übersicht:
Algorithmentypen für Erzeugung von Triangulierungen (nach Hoschek & Lasser 1992):

- *nachoptimierende Algorithmen*: erzeugen eine Initialtriangulierung, die dann mit einem lokalen Kriterium verbessert wird
- *iterativ aufbauende Algorithmen*: konstruieren die Triangulation ausgehend von einem Initialdreieck und durch schrittweise Addition je eines zusätzl. Punktes, so dass zu jedem Zeitpunkt eine lokal optimale Triangulierung vorliegt
- *divide-and-conquer-Algorithmen* unterteilen *P* in Teilmengen und konstruieren für jede Teilmenge lokal optimale Triangulierungen, die am Schluss verschmolzen werden.

Resumé:
- Ein guter Triangulierungsalgorithmus erzeugt immer eine global optimale Triangulierung.
- Aufgrund der Konfluenzeigenschaft bei der lokalen Optimierung sollten lokale Änderungen immer auf der Entscheidungsgrundlage des Max-Min-Winkelkriteriums bzw. des Umkreiskriteriums erfolgen, die zueinander äquivalent sind. Ergebnis ist dann die Delaunay-Triangulierung.

Mesh Simplification

Problem: Möglicherweise sehr detaillierte geometrische Modelle müssen dargestellt, gespeichert und übertragen werden

- hoher Detailgrad ⇒ viel Speicherplatz
- viele Operationen sind auch auf vereinfachten Modellen möglich
- der Benutzer will den gewünschten Detailgrad selbst einstellen können (LOD = *level of detail*)

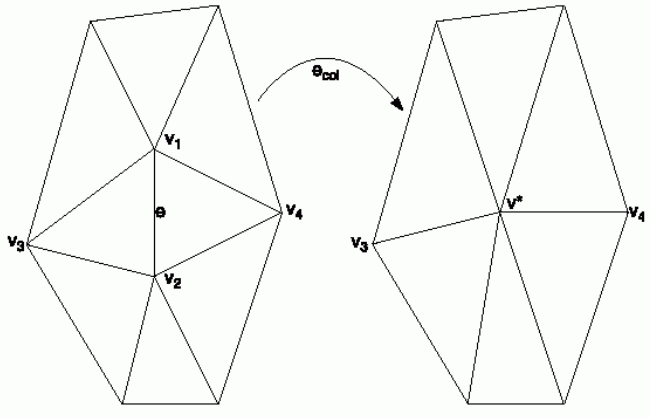
⇒ Multiresolution-Darstellungen

verschiedene Methoden in Gebrauch:

- vertex clustering
- edge collapse
- re-tiling
- Verwendung von Fourier-Transformation oder Wavelets (vgl. JPEG-Kompression)
- multiscaled graphs (z.B. in der Pflanzenmodellierung)

hier als Beispiel knapp vorgestellt:
Edge-collapse-Methode

- Hughes HOPPE: Progressive Meshes, 1996
- Grundlegede Operation: Edge Collapse



- Berechnung des neuen Vertex v^* aus v_1 und v_2 in homogenen Koordinaten: $v^* = v_1 + v_2$

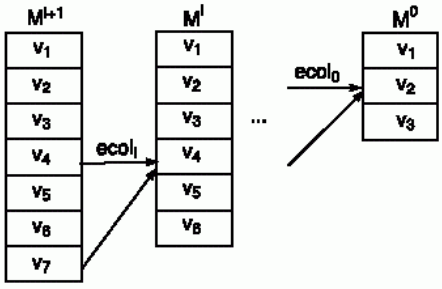
- Jede Edge Collapse Operation
  - verringert die Anzahl der Knoten (Vertices) um eins
  - verringert die Anzahl der Flächen um *mindestens* eins (üblicherweise zwei)
  - verringert die Anzahl der Kanten um *mindestens* zwei (üblicherweise drei oder mehr)

- Frage: Wie wendet man diese Operation zur Vereinfachung eines Polygonnetzes an?

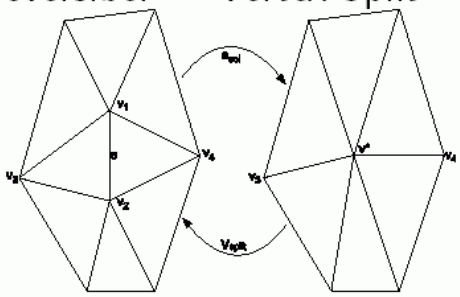
Progressive Meshes

- Idee: Modell = Basismodell + Menge von Verfeinerungsschritten
- Vorgehensweise: Vergröbere das Modell durch eine Folge von Edge Collapse Operationen (ecol)

$$M^n \xrightarrow{\text{ecol}_{n+1}} M^{n-1} \xrightarrow{\text{ecol}_n} \dots \xrightarrow{\text{ecol}_1} M^1 \xrightarrow{\text{ecol}_0} M^0$$



- Edge Collapse ist reversibel → Vertex Split



- neuen Knoten einfügen, Netz aktualisieren, alten Knoten verschieben

- Repräsentation eines Modells als Progressive Mesh:
  Basisnetz + Folge von Vertex Split Operationen

$$\text{PM} = (M_0, \{\text{vsplit}_0, \text{vsplit}_1, \ldots, \text{vsplit}_{n-1}\})$$