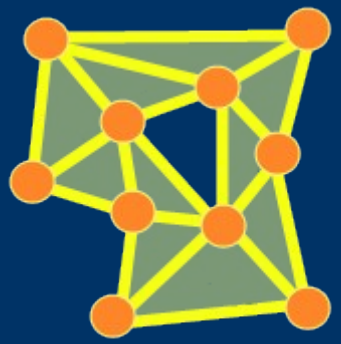
Triangulierung

Übersicht

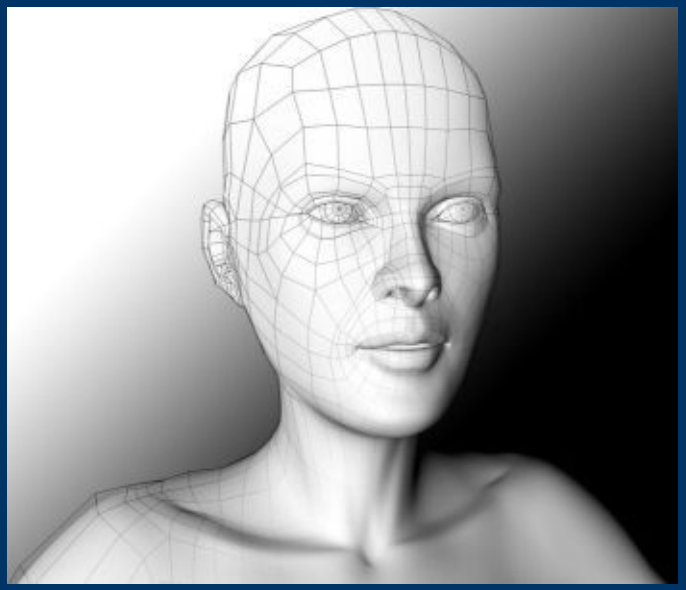
- Begriffserklärung
- Motivation / Anwendungen
- Drei Algorithmen
- Zusammenfassung
- Fragen
- Quellen

Begriffserklärung

- Ein ***Graph*** ist trianguliert, wenn jeder Kreis mit mind. vier Knoten eine Sehne (=Kante zw. zwei im Kreis nicht benachbarten Knoten) enthält.
- Ein ***Polygon*** ist trianguliert wenn es vollständig in Dreiecke zerlegt ist welche sich nicht überschneiden und deren Eckpunkte alle auch Punkte des Polygons sind.
- Eine ***Punktmenge*** ist trianguliert wenn die zugehörige konvexe Hülle vollständig trianguliert ist wobei die Punkte die Ecken der disjunkten Dreiecke bilden.



Anwendungsbeispiele



- Punktanfrageprobleme
- Visualisierung von 3D-Objekten
- Approximation komplexer Geometrie
- Simulation von Kristallwachstum
- Metallurgie (Untersuchung von Legierungen)
- Kartographie, Stadtplanung
- Netzgenerierung für Finite-Elemente-Methode, Visualisierung von Flächen, Geländedaten usw.

Anwendungsbeispiele
Museumsproblem

- Wieviele Wachen sind nötig um ein Museum zu bewachen
- Wache kann auch Lampe, Camera, Sensor sein
- Hat 360°-Blickwinkel
- Museum wird als Polygon dargestellt
- Lösungsansatz:
  - Polygon triangulieren
  - Danach 3-färben
  - Wachen auf geringste Farbe stellen
  - höchstens $\lfloor N/3 \rfloor$ Wachen nötig

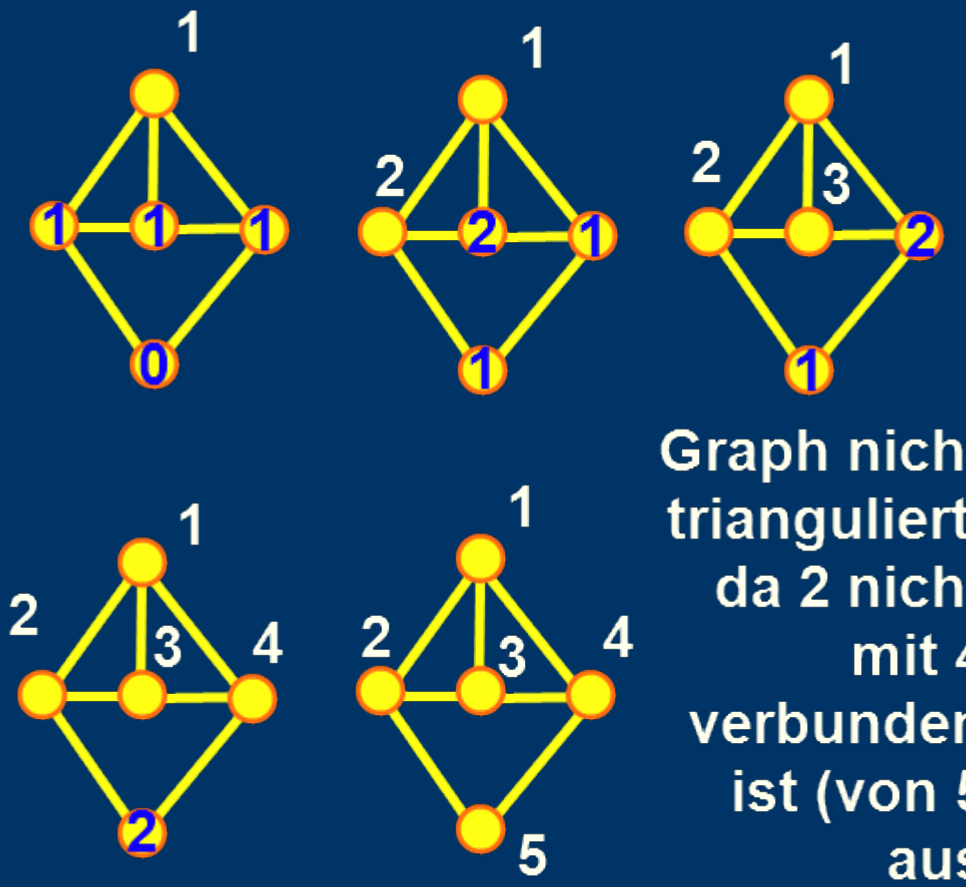
Algorithmus 1 – für Graphen

- Mit MCS kann man einfach eine Knotenreihenfolge erzeugen
- Mit dieser kann der Fill-In den Graphen triangulieren
- Benötigen je O(n+m)

Algorithmus 1 – für Graphen
Maximum Cardinality Search

von Tarjan und Yannakakis

- Zu beginn sind alle Knoten unbesucht
- Wiederhole bis alle Knoten besucht wurden
  - Gehe zu dem unbesuchten Knoten mit den meisten bereits besuchten Nachbarn
  - Wenn der Teilgraph der num. Nachbarn nicht zusammenhängend ist, ist der Graph nicht trianguliert
- Gib die Reihenfolge zurück



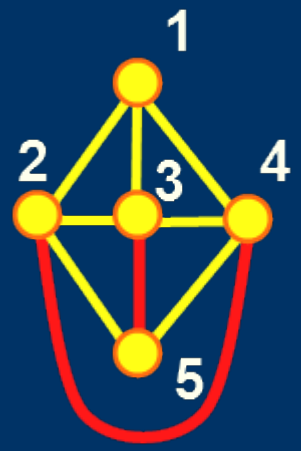
Graph nicht trianguliert, da 2 nicht mit 4 verbunden ist (von 5 aus gesehen)

Algorithmus 1 – für Graphen
Fill-In

- Für alle *vi* einer Knotenpermutation
  - Füge Kanten zwischen den Nachbarn von *vi* ein bis *vi* mit seinen Nachbarn vollständig ist
  - Seien die neuen Kanten in *Ti* gespeichert
  - Lösche *vi* und alle Kanten die *vi* enthalten aus dem Graph
- Erzeuge den trianguliertenGraphen indem die Kantenlisten *T1*...*Tn* eingefügt werden

Algorithmus 1 – für Graphen
Beispiel zum Fill-In

vi	Nachbarn	Ti
1	2,3,4	K(2,4)
2	3,4,5	K(3,5)
3	4,5	---
4	5	---
5	---	---

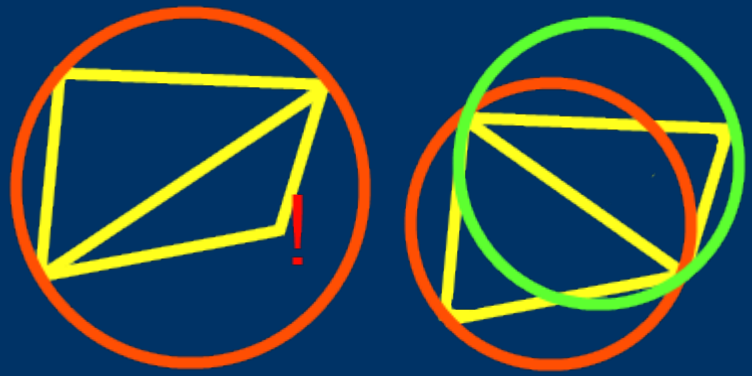


Aber: Mit diesem Algorithmus ist kein minimaler Fill-In garantiert, ohne K(3,5) gänge es auch.

Algorithmus 2 – für Punktmenge
Radial sweep

von Mirante

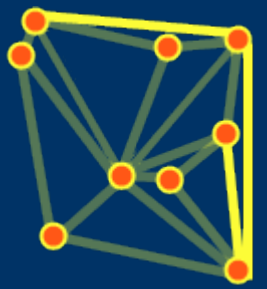
- Radial sweep gehört zur Delaunay Triangulation
- Alle Dreiecke erfüllen Umkreisbedingung
- Erzeugt größtmögliche Innenwinkel
- Rundungsfehler werden verringert
- Umkugelbedingung in 3D
- $O(n^2)$ im worst-case, aber meist schneller



Algorithmus 2 – für Punktmenge Radial sweep

von Mirante

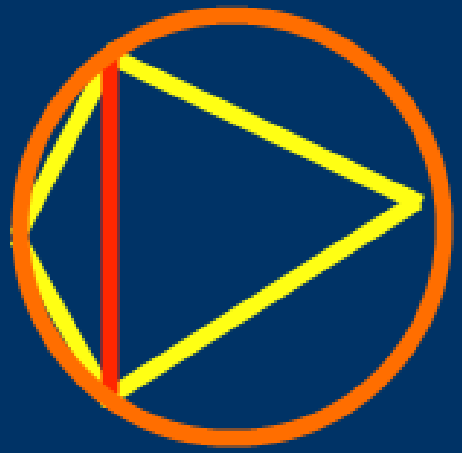
- Einen Punkt in der Mitte wählen und Entfernung und Richtung zu allen übrigen Punkten berechnen
- Der Richtung nach sortieren und Verbindungen einzeichnen
- Benachbarte Punkte am Rand verbinden
- Am Rand solange Dreiecke einfügen bis man die konvexe Hülle hat
- Gültige Triangulation aber Dreiecksform nicht optimal
- Zwei benachbarte Dreiecke bilden ein Viereck, mit der kürzeren Diagonalen „verbessert“ sich die Form -> Diagonalen tauschen bis nirgends eine kürzere gefunden wird



Algorithmus 2 – für Punktmenge Radial sweep

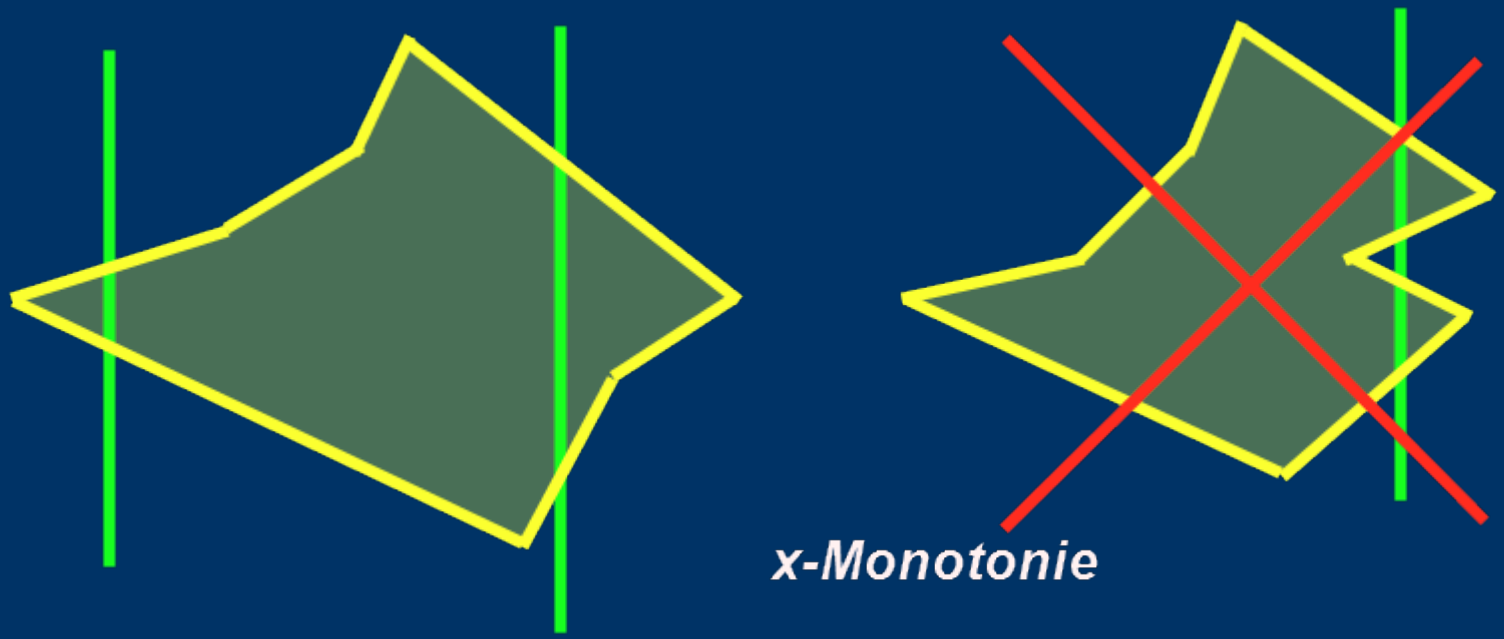
von Mirante

- Bei diesem Algorithmus erhält man nur „fast“ eine Delaunay Triangulation, das Kreiskriterium wird mit der kürzesten Diagonale nicht immer erfüllt. Wenn man die Diagonalen so tauscht dass sich die Winkel maximieren stimmt es wieder.



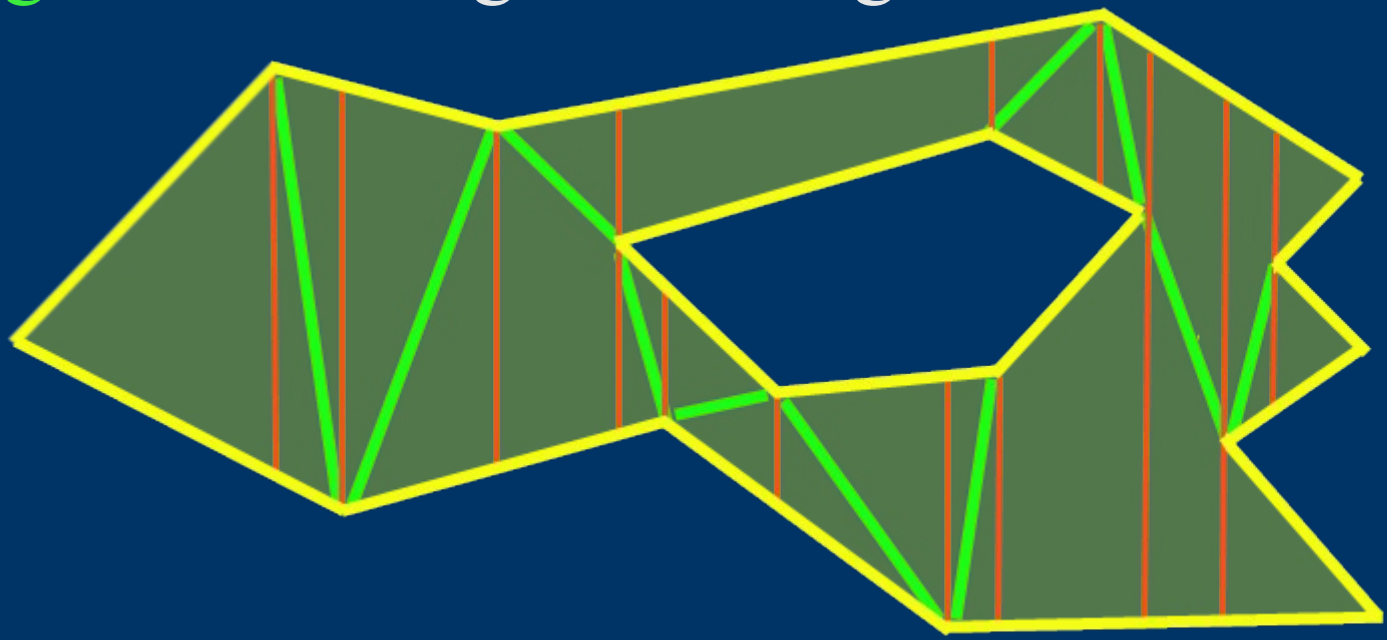
Algorithmus 3 – für Polygone

- Ein Polygon heißt monoton bzgl. einer Geraden wenn alle dazu orthogonalen Geraden das Polygon zusammenhängend schneiden.



Algorithmus 3 – für Polygone Mit Trapezzerlegung zur x-Monot.

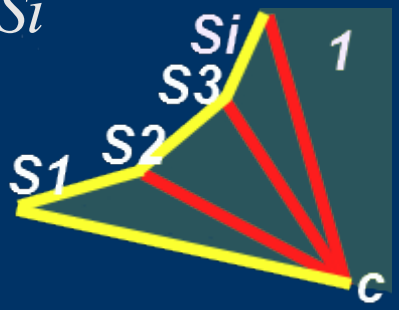
- Polygon in Trapeze zerlegen und in diese Diagonalen einfügen wo möglich.

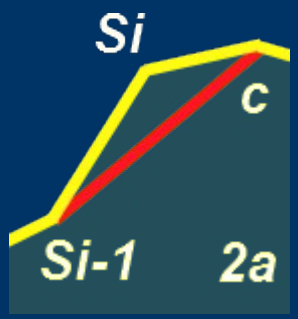


- Benötigt O(n*log n)
- Alle Teilpolygone sind x-monoton.

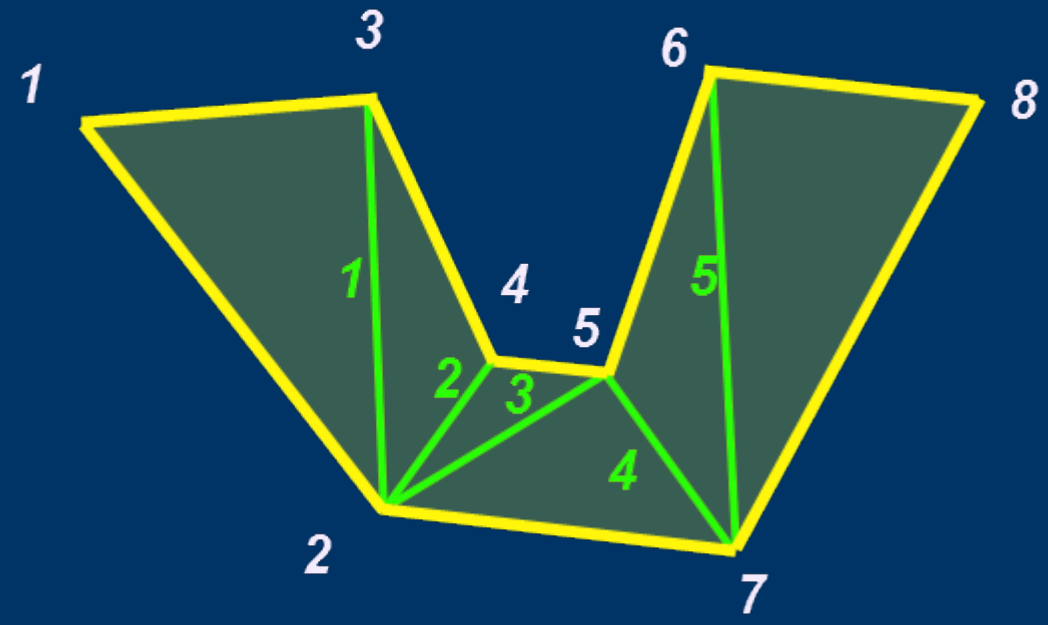
Algorithmus 3 – für Polygone x-monotone Polygone triangulieren

- Punkte ggf. aufsteigend nach x sortieren
- 1. und 2. auf Stack[*S1*...*Si*] legen, *c* = 3.
- **Solange** *S1* **<u>oder</u>** *Si* nicht mit c benachbart
  - 1. **Wenn** *c* mit *S1* benachbart **<u>aber</u>** nicht mit *Si*
    - Kanten *S2*-*c* ... *Si*-*c* einfügen
    - Stack mit *Si* und *c* überschreiben
    - *c*++
  - Sonst
    - 2a. **Wenn** *Stacksize* > 1 **<u>und</u>** (*Si-1*, *Si*, *c*) = konvexe Ecke
      - Kante (*Si*-1)-*c* einfügen
      - Oberstes Stackelement löschen
      - Falls *Stacksize* = 1: *c* auf Stack und *c*++
    - 2b. **Sonst**
      - *c* auf Stack legen
      - *c*++
- Kanten *S2*-*c* ... *Si*-*c* einfügen





Algorithmus 3 – für Polygone Beispiel



Stack	c	Fall
1,2	3	1
2,3	4	2a
2,4	5	2a
2,5	6	2b
2,5,6	7	1
6,7	8	3

- Benötigt O(n) wenn Punkte schon sortiert waren

Zusammenfassung

- Triangulierung wird in vielen Bereichen gebraucht
- Mit MCS+Fill-In kann man schnell und einfach einen Graphen triangulieren
- Radial sweep hat unter Umständen $O(n^2)$, erzeugt aber „schöne“ Dreiecke
- Mit der Trapezzerlegung kann man beliebige Dreiecke in Monotone zerlegen, diese kann man dann schnell triangulieren

Fragen ?

Quellen

- http://www.itee.uq.edu.au/.../PolygonTriangulation.ppt
- http://prl.cs.gc.cuny.edu/.../GraphTriangulation.pdf
- http://de.wikipedia.org/wiki/Delaunay-Triangulation
- http://www.stud.tu-ilmenau.de/.../radsweep.htm
- http://i11www.iti.uni-karlsruhe.de/.../folien_triang_polygone.pdf
- http://www-sst.informatik.tu-cottbus.de/.../cg_v08c.pdf