6. Polygontriangulierung:
Wie bewacht man eine Kunstgalerie?

6.1. Grundlegendes zu Polygonen

Es sei *P* ein einfaches Polygon in der Ebene; *P* habe *n* Ecken.

Definition:

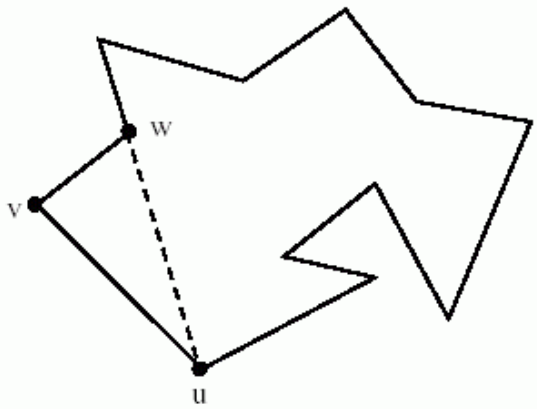
$\underline{Diagonale}$ von P = Liniensegment zwischen 2 Ecken von P,
das in P enthalten ist,
aber mit ∂P nur seine Eckpunkte gemeinsam hat
$\rightarrow$ zerlegt P in zwei einfache Teilpolygone

Hilfssatz:

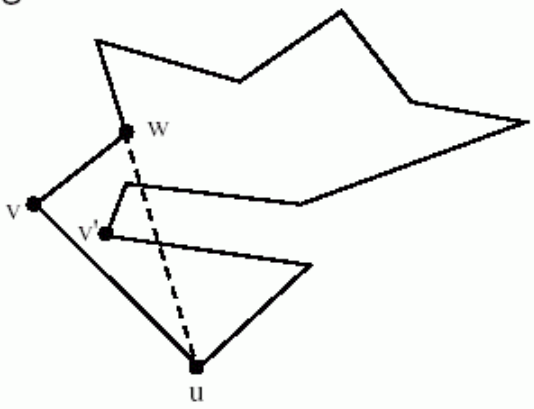
Zu jedem einfachen Polygon mit mehr als 3 Ecken existiert eine Diagonale.

Beweis:

v sei der am weitesten links liegende Eckpunkt von $\mathcal{P}$.
u und w seien die beiden benachbarten Eckpunkte von v auf $\mathcal{P}$.
Liegt das offene Segment $\overline{uw}$ im Innern von $\mathcal{P}$, so haben wir eine Diagonale gefunden:



Sonst gibt es einen oder mehrere Eckpunkte von $\mathcal{P}$, die im Innern des von u, v und w aufgespannten Dreiecks oder auf der Diagonalen $\overline{uw}$ liegen:

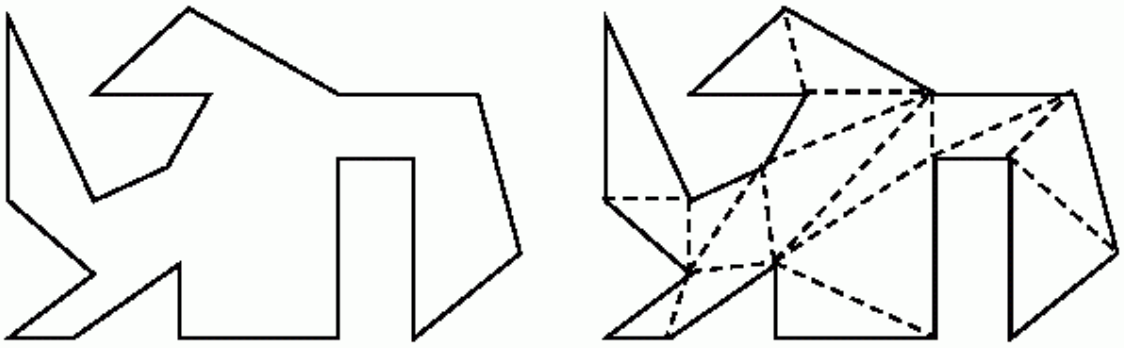


Sei v' derjenige Eckpunkt mit dem größten Abstand von $\overline{uw}$. $\overline{vv'}$ kann keine Kante von $\mathcal{P}$ schneiden, da solch eine Kante einen Endpunkt innerhalb des Dreiecks besitzen würde, der weiter als v' von $\overline{uw}$ entfernt wäre, was der Definition von v' widersprechen würde $\Rightarrow$ $\overline{vv'}$ eine Diagonale $\Rightarrow \exists$ Diagonale

(aus Hinrichs 2001)

Definition:

- Eine Zerlegung eines Polygons in Dreiecke durch eine maximale Menge von sich nicht schneidenden Diagonalen nennt man eine Triangulation des Polygons:



- Triangulation gewöhnlich nicht eindeutig!

alternative Def.: Triangulation von $P = \partial P \cup$ maximale Menge sich nicht kreuzender Diagonalen von P.

Satz:
Jedes einfache Polygon lässt sich triangulieren.

Beweis:
Vollständige Induktion über n.
$n = 3$: trivial.
Sei $n > 3$, und wir nehmen an, die Aussage sei wahr für jedes einfache Polygon mit Eckenzahl $m < n$. Sei P einf. Polygon mit n Ecken.
Nach dem Hilfssatz ex. eine Diagonale von P.

Diagonale schneidet $\mathcal{P}$ in zwei einfache Teilpolygone $\mathcal{P}_1$ und $\mathcal{P}_2$.
m_1: Anzahl Eckpunkte von $\mathcal{P}_1$
m_2: Anzahl Eckpunkte von $\mathcal{P}_2$
$m_1 < n,\ m_2 < n$
$\Rightarrow \mathcal{P}_1$ und $\mathcal{P}_2$ können trianguliert werden (Ind.vor.)
$\Rightarrow \mathcal{P}$ kann trianguliert werden.

Satz:
Jede Triangulation eines einfachen Polygons mit n Ecken besitzt genau n–2 Dreiecke und n–3 Diagonalen.

Beweis:
Induktion über n. Für $n = 3$ trivial. Sei $n > 3$. P habe n Ecken.
Sei T eine Triangulation von P.
D sei eine Diagonale aus $T \Rightarrow D$ zerlegt P in 2 Teilpolygone mit m_1 bzw. m_2 Ecken; $m_1 + m_2 = n + 2$ (da D hier doppelt gezählt).

Nach Induktionsannahme:
1. Teilpolygon hat $m_1 - 2$ Dreiecke und $m_1 - 3$ Diagonalen,
2. Teilpolygon hat $m_2 - 2$ Dreiecke und $m_2 - 3$ Diagonalen.

$\Rightarrow T$ hat $(m_1 - 2) + (m_2 - 2) = n + 2 - 4 = n - 2$ Dreiecke
und $(m_1 - 3) + (m_2 - 3) \mathbf{+ 1} = n + 2 - 6 + 1 = n - 3$ Diagonalen.

Hilfssatz: In jeder Triangulation *T* eines einfachen Polygons existiert ein Dreieck, das nur eine Diagonale als Kante hat (wo die anderen beiden Kanten also Polygonkanten sind).

Beweis:
indirekt. Annahme:
0 Dreiecke von *T* haben genau 1 Diagonale als Kante,
a Dreiecke von *T* haben genau 2 Diagonalen als Kante,
b Dreiecke von *T* haben genau 3 Diagonalen als Kante.

Nach dem vorherigen Satz gilt:
Anzahl der Dreiecke = $a + b = n - 2$,
Anzahl der Diagonalen = n – 3; da jede Diagonale zu genau 2 Dreiecken gehört, muss gelten: $2a + 3b = 2(n - 3)$.

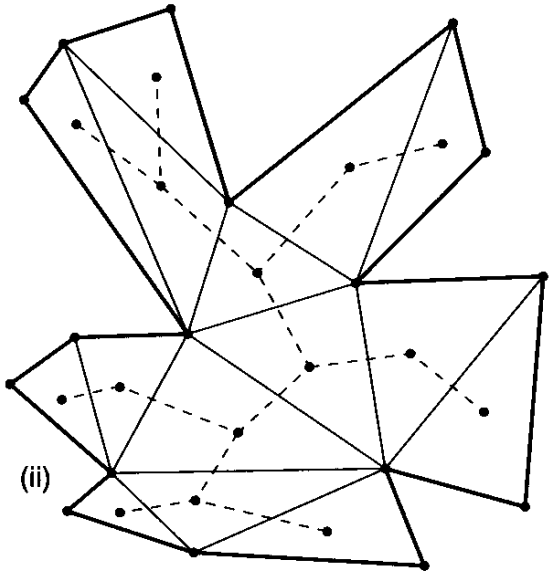
Aus beiden Gleichungen $a + b = n - 2$ und $2a + 3b = 2(n - 3)$ erhält man die absurde Konsequenz $b = -2$; Widerspruch.

Dualer Graph einer Triangulation T von P

Dualer Graph T^* zu T:

Knoten $=$ Dreiecke von T

Kante (u, v) g.d.w. Dreiecke u, v haben gemeinsame Kante d



Satz: T^* ist ein Baum mit Knotengrad ≤ 3.

Beweis:

Grad $\surd$ da ein Dreieck max. 3 Nachbardreiecke hat

Dreiecke von T füllen P lückenlos aus $\rightarrow T^*$ zusammenhängend

zeige: T^* ist azyklisch: durch Induktion:

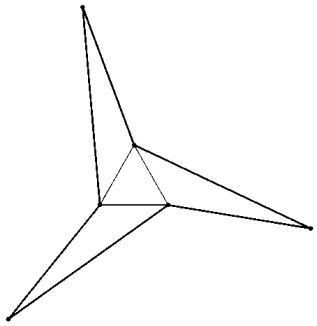
Hilfssatz $\rightarrow \exists$ Dreieck D in T, das nur eine Diagonale als Kante hat

entferne D aus P und T. $\rightarrow P'$ mit $n-1$ Ecken

Ind.–Annahme: T'^* ist azyklisch

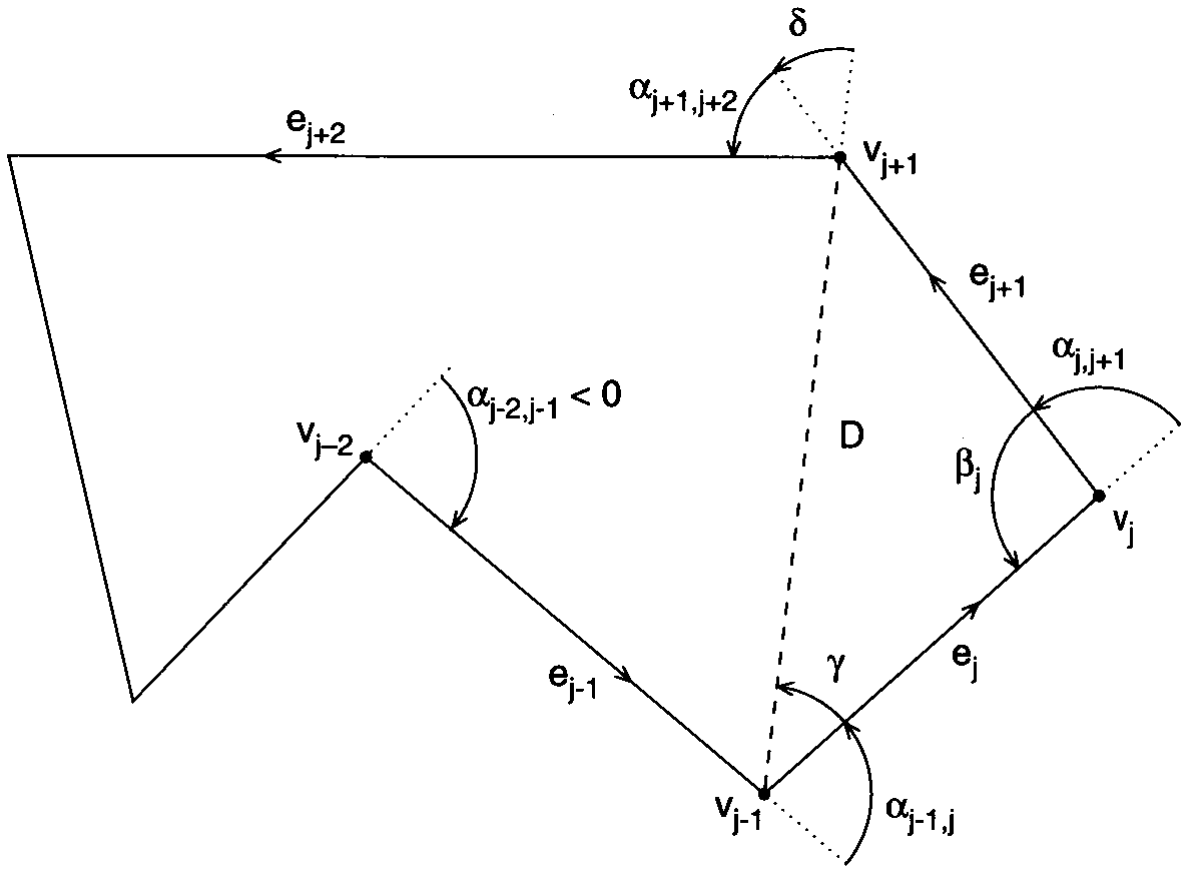
$\rightarrow T^*$ azyklisch. $\square$

Kann man den Knotengrad weiter herabdrücken, d.h. findet man immer eine Triangulation, in der jedes Dreieck höchstens **zwei** Nachbardreiecke hat?
– Nein!



Daß sich Polygone triangulieren lassen, ist oft ein nützliches Beweishilfsmittel. Betrachten wir zum Beispiel die Drehungen, die eine *Schildkröte* ausführen müßte, während sie sich gegen den Uhrzeigersinn auf dem Rand eines Polygons entlangbewegt.[11] Die Kanten werden von einer beliebigen Startkante aus mit $e_0, e_1, \ldots, e_{n-1}$ durchnumeriert. Mit $\alpha_{i,i+1}$ wird die Drehung bezeichnet, die für den Übergang von e_i nach e_{i+1} erforderlich ist; siehe Abbildung . Dabei werden Linksdrehungen positiv und Rechtsdrehungen negativ gezählt.

[11] Um diese *turtle geometry* geht es in dem gleichnamigen Buch von Abelson und diSessa [1]. In der Programmiersprache LOGO gibt es eine *turtle*, die durch Vorwärtsbewegungen und Drehungen gesteuert wird.



Für $i \neq k$ bezeichnen wir mit

$$\alpha_{i,k} = \alpha_{i,i+1} + \alpha_{i+1,i+2} + \ldots + \alpha_{k-1,k}$$

die Summe der Drehwinkel von Kante e_i bis zur Kante e_k. Die Winkelsumme bei *einem vollständigen Umlauf* von e_i bis e_i wird auch die *Gesamtdrehung* genannt und mit $\alpha_{i,i}$ bezeichnet. Es gilt folgende schöne Aussage:

Satz:
Sei *P* ein einfaches Polygon und e_i eine beliebige Kante von *P*. Dann ist $\alpha_{i,i} = 2\pi$.

Beweis. Durch Induktion über die Kantenzahl $n \geq 3$. Sei $n = 3$. Da sich der äußere Drehwinkel $\alpha_{j,j+1}$ und der Innenwinkel β_j jeweils zu π aufaddieren (vgl. Abbildung), folgt

$$\alpha_{i,i} = \sum_{j=0}^{2} \alpha_{i+j,i+j+1} = \sum_{j=0}^{2} (\pi - \beta_{i+j}) = 3\pi - \sum_{j=0}^{2} \beta_{i+j} = 2\pi,$$

denn die Summe der Innenwinkel beträgt beim Dreieck bekanntlich π.

Hat das Polygon mehr als drei Ecken, wählen wir eine Diagonale, die ein Dreieck D vom Rest des Polygons abtrennt, und wenden die Induktionsvoraussetzung auf den Rest an. Dann fügen wir das Dreieck D wieder hinzu und betrachten die Änderung in der Winkelsumme, die sich hierdurch ergibt; mit den Bezeichnungen von Abbildung 4.17 erhöht sie sich gerade um

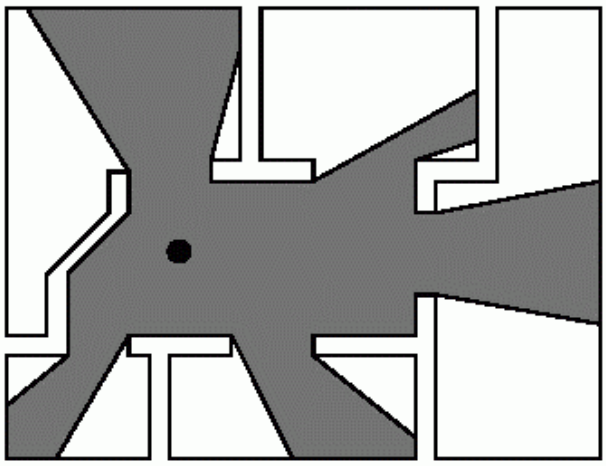
$$\alpha_{j,j+1} - (\gamma + \delta) = \pi - (\beta_j + \gamma + \delta) = 0. \qquad \square$$

(aus Klein 1997).

6.2. Das Art-Gallery-Theorem

"Art-Gallery-Problem":
wieviele Kameras werden zur Bewachung einer Kunstgalerie benötigt?

- Eine Kunstgalerie wird beschrieben durch ein einfaches Polygon, d.h. ein Polygon, dessen Kanten sich nur in den Endpunkten schneiden dürfen. Das Polygon darf keine Löcher haben. Eine Kameraposition wird gegeben durch einen Punkt in diesem Polygon, die Kamera sieht alle die Punkte, mit denen sie durch ein offenes Segment, das ganz im Innern des Polygons liegt, verbunden werden kann:



- Wie viele Kameras werden zur Bewachung der Galerie benötigt?
- Offensichtlich: Je komplexer das Polygon, um so mehr Kameras werden benötigt.

- Die Schranke für die Anzahl Kameras wird in Abhängigkeit von der Anzahl n der Eckpunkte des Polygons ausgedrückt.
- Konvexes Polygon kommt, unabhängig von n, immer mit einer Kamera aus!
- Wir betrachten den schlimmsten Fall, d.h. wir geben obere Schranke für alle Polygone mit n Eckpunkten.

Einfache (aber ineffiziente) Lösung:

- Bewachung von $\mathcal{P}$, indem in jedes Dreieck einer Triangulation $\mathcal{T}_\mathcal{P}$ von $\mathcal{P}$ eine Kamera plaziert wird.

Aus obigem Satz über Anzahl der Dreiecke in T_P folgt:

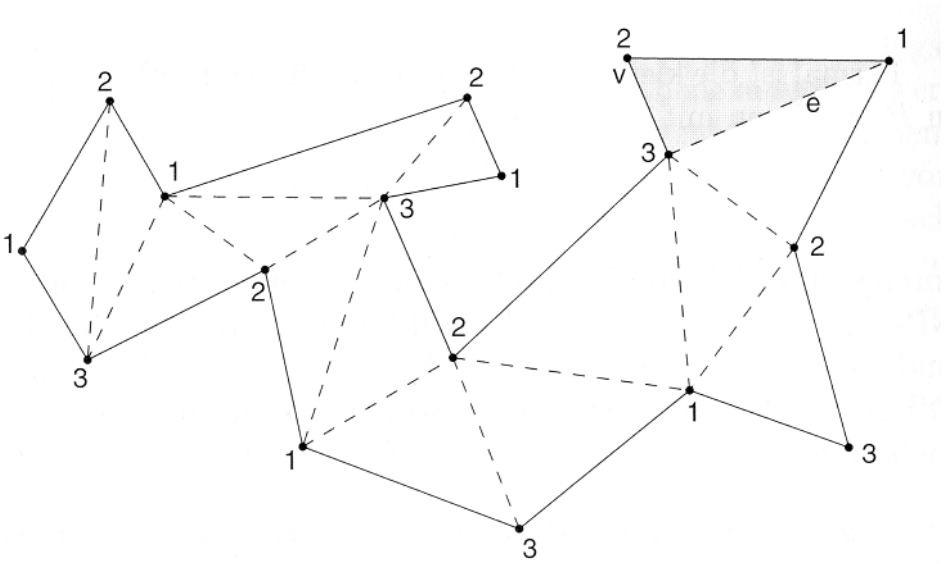
Jedes einfache Polygon mit n Eckpunkten kann durch n–2 Kameras überwacht werden.

Aber:

- Nicht notwendig, in jedem Dreieck eine Kamera zu plazieren. Eine auf einer Diagonalen plazierte Kamera kann zwei Dreiecke überwachen ⇒ durch Plazierung der Kameras auf wohlausgewählten Diagonalen kann deren Anzahl auf n/2 reduziert werden.
  Noch besser: Kameras in den Eckpunkten plazieren, da ein Eckpunkt inzident zu mehreren Dreiecken ⇒ alle diese Dreiecke können durch eine Kamera überwacht werden
- Sei $\mathcal{T}_\mathcal{P}$ eine Triangulation von $\mathcal{P}$.
  - Selektiere Teilmenge von Eckpunkten von $\mathcal{P}$, so daß jedes Dreieck wenigstens einen selektierten Eckpunkt besitzt.
  - Plaziere die Kameras in den selektierten Eckpunkten.

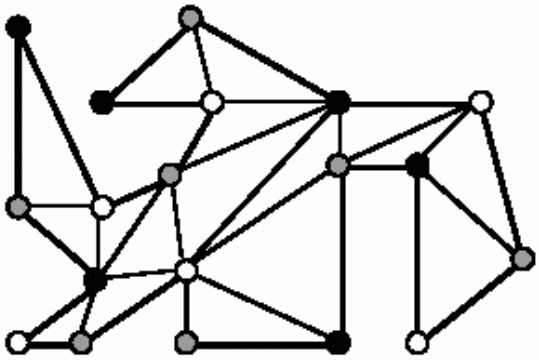
Vierfarbensatz: Jeder planare Graph ist 4-färbbar.
Wir nutzen aus: Triangulationen sind spezielle planare Graphen, die sogar 3-färbbar sind.

Beachte: Bei "Färbbarkeit" wird hier immer vorausgesetzt, dass 2 Ecken, die durch eine Kante verbunden sind, verschieden gefärbt sind.



Um dann eine geeignete Teilmenge von Ecken für die Überwachung zu finden, färbe man die Eckenmenge mit 3 Farben (z.B. weiß, grau, schwarz) und plaziere die Kameras z.B. in allen grau gefärbten Ecken ⇒ das gesamte Polygon wird überwacht.

- Wähle für die Plazierung der Kameras die Farbe mit der minimalen Anzahl Eckpunkte ⇒ Polygon kann mit höchstens $\lfloor n / 3 \rfloor$ Kameras überwacht werden.



noch zu zeigen: es existiert immer eine 3-Färbung.

Wir traversieren den dualen Graphen $G(T_P)$ der Triangulierung durch Tiefensuche (depth-first search)
(dabei wird ausgenutzt: dieser ist Baum; s.o.)

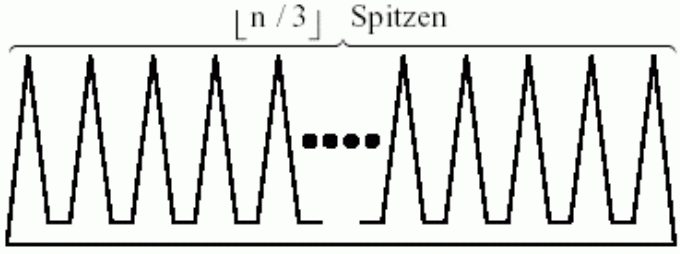
- Aufrechterhalten folgender Invariante während der Tiefensuche: Alle Eckpunkte der bereits angetroffenen Dreiecke sind weiß, grau oder schwarz eingefärbt, und keine zwei miteinander verbundenen Eckpunkte sind gleich eingefärbt worden.
- Invariante ⇒ gültige 3-Färbung, nachdem alle Dreiecke angetroffen wurden.

- Tiefensuche kann an einem beliebigen Knoten des Graphen $\mathcal{G}(\mathcal{T}_P)$ starten. Die drei Eckpunkte des entsprechenden Dreiecks werden weiß, grau und schwarz eingefärbt.
- Erreichen eines Knotens ν von einem Knoten μ kommend:
  Dreiecke $t(\nu)$ und $t(\mu)$ haben eine Diagonale gemeinsam.
  Eckpunkte von $t(\mu)$ wurden bereits eingefärbt
  $\Rightarrow$ nur noch ein Eckpunkt von $t(\nu)$ einzufärben
  Für diesen Eckpunkt gibt es nur noch eine verfügbare Farbe, nämlich die Farbe, die für die beiden Endpunkte der gemeinsamen Diagonale zwischen $t(\nu)$ und $t(\mu)$ nicht benutzt wurde.
  $\mathcal{G}(\mathcal{T}_P)$ Baum $\Rightarrow$ die anderen zu ν benachbarten Knoten noch nicht besucht $\Rightarrow$ weise dem Eckpunkt die noch verbleibende Farbe zu

Somit kann jedes einfache Polygon durch $\lfloor n/3 \rfloor$ Kameras überwacht werden.

- Geht es auch mit noch weniger Kameras? Eine an einem Eckpunkt plazierte Kamera kann ja mehr als nur die inzidenten Dreiecke überwachen!

- Nein! Für jedes n existieren einfache Polygone, die $\lfloor n / 3 \rfloor$ Kameras benötigen.
  Beispiel: kammförmiges Polygon mit $\lfloor n / 3 \rfloor$ Spitzen, so daß eine Kamera von keinem Punkt des Polygons aus gleichzeitig in zwei Spitzen hineinsehen kann:



Damit gezeigt:

Satz ("Art Gallery Theorem"):

Für ein einfaches Polygon $\mathcal{P}$ sind $\lfloor n / 3 \rfloor$ Kameras gelegentlich notwendig und immer ausreichend, um jeden Punkt in $\mathcal{P}$ von wenigstens einer Kamera sehen zu können.

Bemerkung:
Eine geeignete Menge von $\lfloor n/3 \rfloor$ Kamerapositionen kann in der Zeit $O(n \log n)$ gefunden werden.

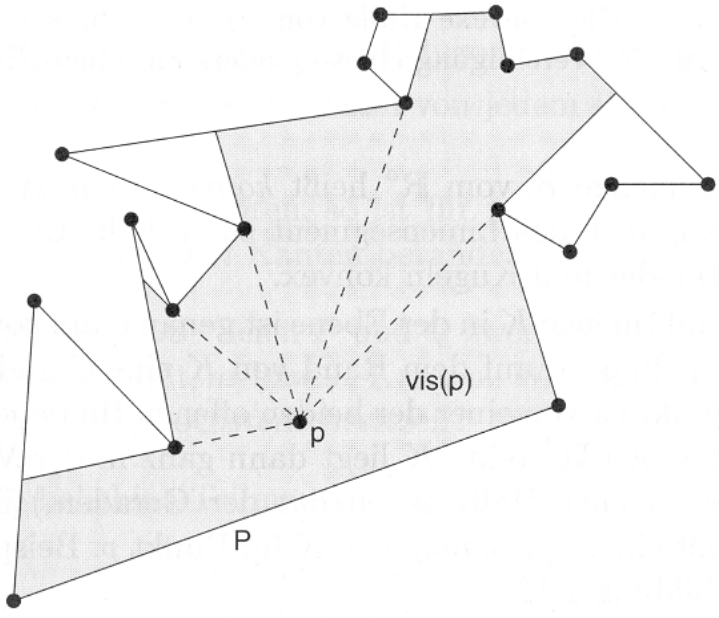
(Hinrichs 2001)

6.3. Das Sichtbarkeitspolygon

Das "Art-Gallery-Problem" als Spezialfall der Fragestellung:

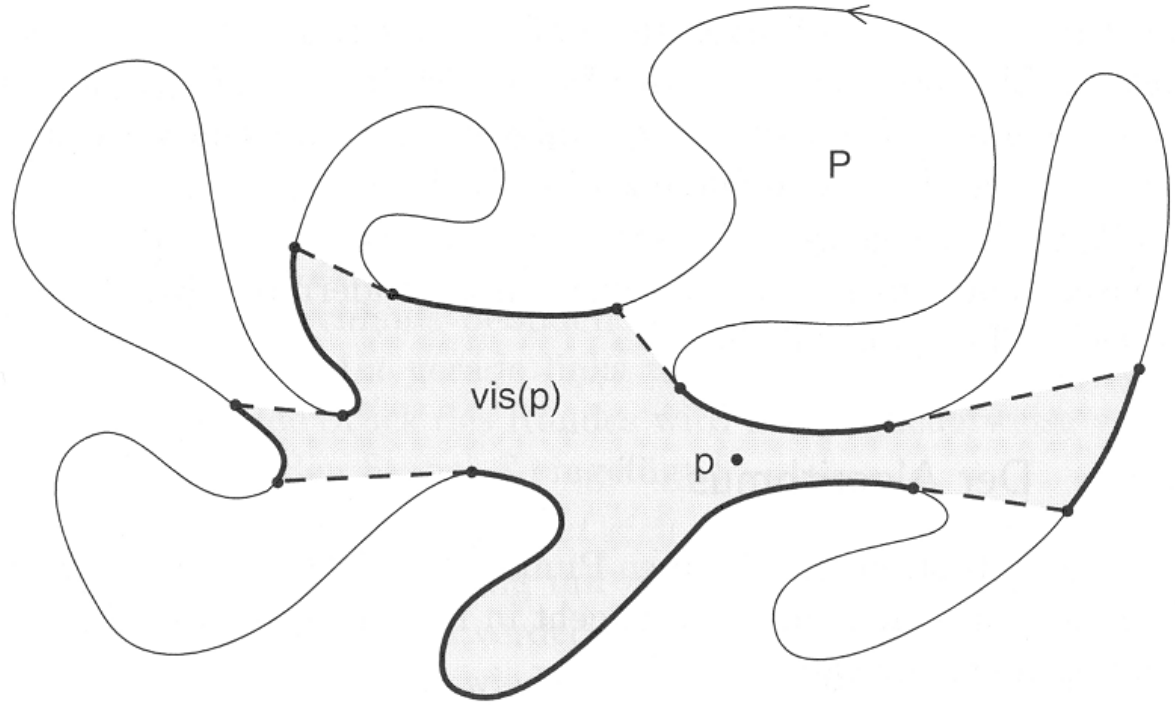
Welche Punkte innerhalb eines Polygons sind von einem gegebenen Punkt aus sichtbar?

Sei *p* ein Punkt im Inneren des einfachen Polygons *P*.
Sichtbarkeitspolygon *vis*(*p*) = der Teil von *P*, der von *p* aus sichtbar ist



Anwendung: Bewegungsplanung für Roboter

was kann ein Roboter von einem gegebenen Punkt aus sehen?



Kanten von *vis*(*p*):

- Teile von Kanten von *P* (dick gezeichnet)
- "künstliche Kanten" (gestrichelt) = Teile von Tangenten von *p* an den Rand von *P*

zur Konstruktion von *vis*(*p*) genügt es, die Kanten des ersten Typs zu bestimmen.

gegeben: Rand von *P* als zyklisch verkettete Liste von Strecken und Punkt *p* in *P*

Algorithmus zur Bestimmung von *vis*(*p*) in linearer Zeit:

- starte mit sichtbarem Punkt auf dem Rand (auffindbar durch Schnitt von Strahl mit Rand von *P*)
- durchlaufe den Rand von *P* gegen den Uhrzeigersinn
- führe Stack *S* mit, der potenziell sichtbare Teile des Randes verwaltet

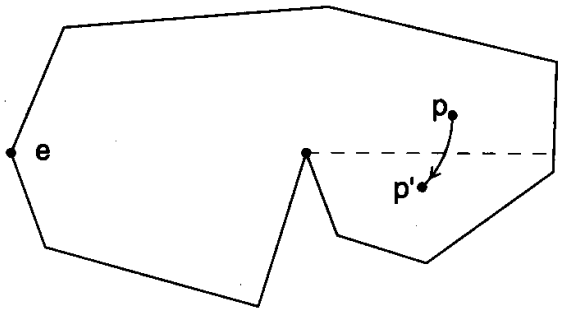
- aktualisiere *S* bei bestimmten Änderungen der Bewegungsrichtung relativ zu *p* (Fallunterscheidungen)
- am Ende des Umlaufs enthält *S* genau die von *p* sichtbaren Teile des Randes
- ergänze die künstlichen Kanten und gebe *vis*(*p*) aus

Details bei Klein (1997), S. 186-192.

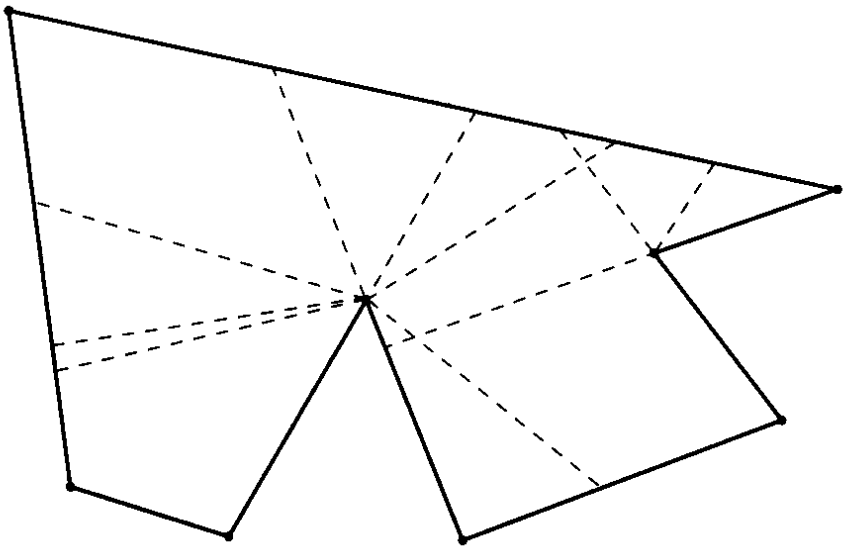
wichtig in der Robotik:
wie ändert sich *vis*(*p*), wenn sich *p* innerhalb von *P* bewegt?

bei kleinen Bewegungen: Menge der sichtbaren Ecken ändert sich zunächst nicht

- ändert sich erst, wenn *p* die Verlängerung einer Tangente von einer Ecke *e* an eine spitze Ecke (Innenwinkel > 180°) überschreitet (kritisches Tangentenstück = "Sichtsegment")



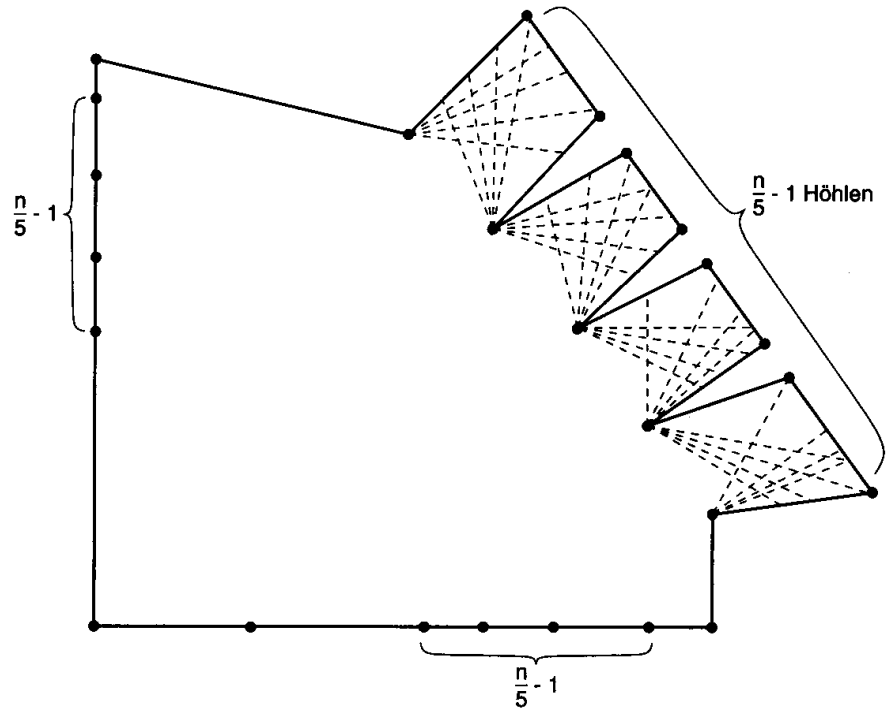
Äquivalenzrelation zwischen Punkten in *P*:
p äquivalent zu *p*' $\Leftrightarrow$ von *p* aus sind genau dieselben Ecken von *P* sichtbar wie von *p*' aus.
Äquivalenzklassen = *Sichtregionen*.



Übungsaufgabe: Man zeige, dass die Sichtregionen stets konvex sind. ($\Rightarrow$ je 2 äquivalente Punkte können einander sehen.)

Die Anzahl der Sichtregionen ist $\Theta(n^3)$ (Beweis: s. Klein 1997).

Beispiel für die untere Schranke (Polygon mit Anzahl Sichtregionen kubisch in der Eckenzahl n):



6.4. Sternpolygone

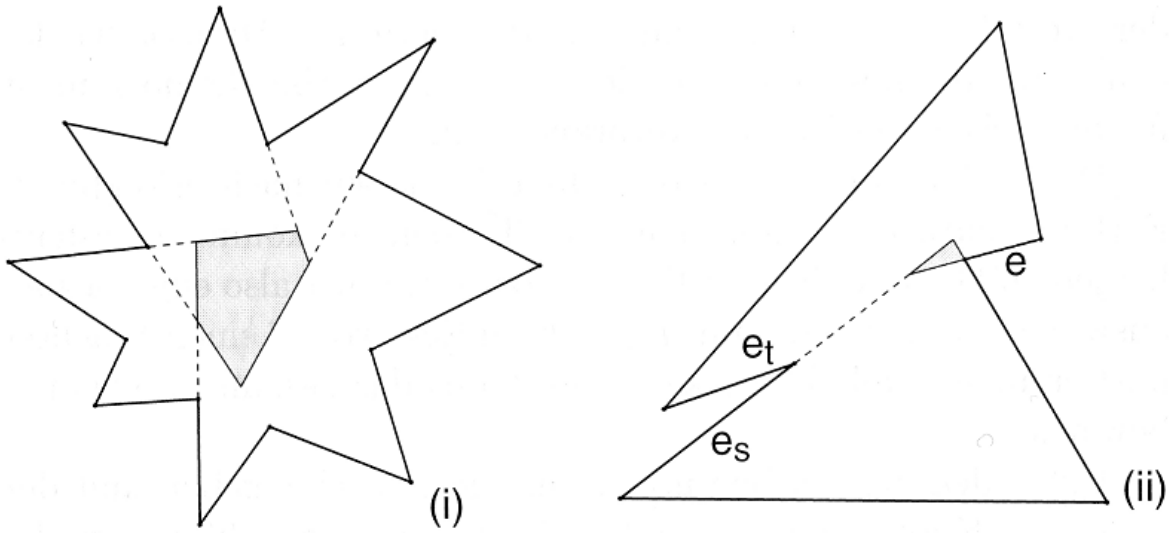
Definition:
Ein einfaches Polygon P heißt *sternförmig* oder *Sternpolygon*, wenn es einen Punkt p in P gibt, so dass $vis(p) = P$.

Gesamtheit aller Punkte in P mit dieser Eigenschaft:
Kern von P

$$ker(P) = \{ p \in P \mid vis(p) = P \}.$$

Es gilt: P sternförmig $\Leftrightarrow$ $ker(P) \neq \emptyset$.

Beispiele für Sternpolygone (Kern schraffiert):

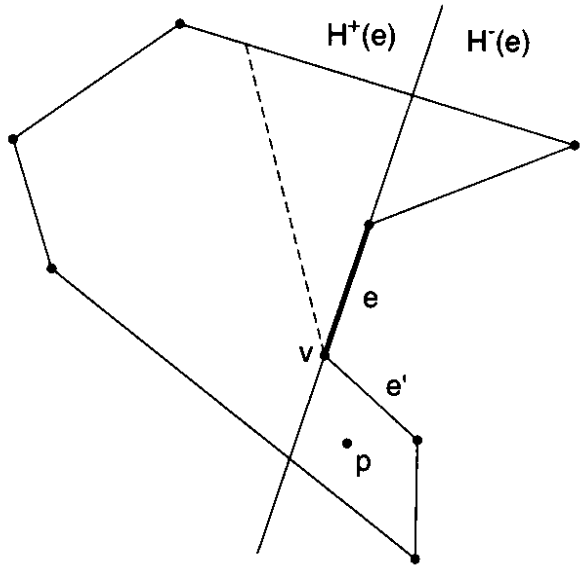


Es gilt:

- P konvex $\Leftrightarrow$ $ker(P) = P$
- $\forall p \in P$: $ker(P) \subseteq ker(vis(p))$

(Beweis: Übung)

Es sei P einfaches Polygon, der Rand von P sei gegen den Uhrzeigersinn orientiert.
Für eine Kante e von P bezeichne $H^+(e)$ die abgeschlossene Halbebene links von e und $H^-(e)$ die offene Halbebene rechts von e. (In der Nähe der Kante e ist also das Innere von P in $H^+(e)$ enthalten, s. Abb.).



Satz:
Es gilt:

$$ker(P) = \bigcap_{e \text{ Kante von } P} H^+(e)$$

Beweis:

Der Kern ist in jeder Halbebene $H^+(e)$ enthalten (vgl. Abb.): wenn ein $p \in H^-(e)$ überhaupt zu P gehört, ist von p aus jedenfalls e nicht sichtbar.

Liege nun p im Durchschnitt, also in jeder Halbebene $H^+(e)$.
$\Rightarrow$ p liegt in P,
denn sonst: p könnte mindestens 1 Kante e "von außen" sehen und läge deshalb in $H^-(e)$.
Annahme: $p \notin ker(P)$
$\Rightarrow$ $vis(p)$ echte Teilmenge von P
$\Rightarrow \exists$ spitze Ecke v von P mit Kanten e und e', so dass p nur e' sehen kann, aber nicht e (siehe Abb.)
$\Rightarrow p \in H^-(e)$, Widerspruch.

Somit:
Kern von P Durchschnitt von n bekannten Halbebenen
$\Rightarrow$ Kern lässt sich in der Zeit $O(n \log n)$ berechnen

Lee & Preparata (1979) haben gezeigt, dass der Kern sogar in linearer Zeit berechnet werden kann (s. Klein 1997, S. 199-204).

Man nutzt aus: Vorsortieren der Halbebenen nach Winkeln; Berechnung von Durchschnitten von Halbebenen dann in Zeit $O(n)$ (durch Reduktion auf Berechnung der konvexen Hülle sortierter Punktmengen).

6.5. Effiziente Triangulierung von Polygonen

Sei P wieder ein einfaches Polygon mit n Ecken.

Wieviele Dreiecke braucht man für eine Triangulierung?

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):

$v - e + f = 2$

$\Rightarrow\ n - (3t+n)/2 + t+1 = 2\ \Rightarrow t = n - 2$

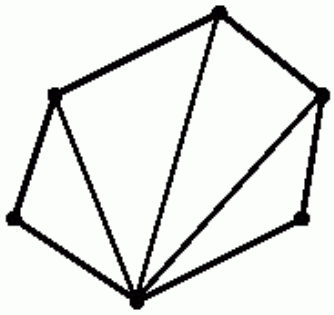
$\Rightarrow$ die Zahl der Dreiecke liegt fest!

(Formel gilt nur, wenn P keine Löcher hat – einfaches Polygon!)

Wir wissen: Triangulation von *P* existiert immer.

- konstruktiver Beweis $\Rightarrow$ rekursiver Triangulationsalgorithmus: Finde eine Diagonale und trianguliere die beiden sich ergebenden Teilpolygone rekursiv.
- Auffinden der Diagonale: nehme den am weitesten links liegenden Eckpunkt von $\mathcal{P}$ und versuche, ihn mit seinen zwei Nachbarn u und w zu verbinden; falls dies nicht klappt, verbinde v mit dem Eckpunkt, der innerhalb des durch u, v und w definierten Dreiecks am weitesten von $\overline{uw}$ entfernt liegt.
- Zeitaufwand für das Auffinden der Diagonale: O(n)
- Diagonale kann $\mathcal{P}$ in ein Dreieck und ein Polygon mit n – 1 Eckpunkten aufteilen, z.B wenn es uns gelingt, u und w miteinander zu verbinden
  $\Rightarrow$ Triangulationsalgorithmus benötigt im schlimmsten Fall $O(n^2)$ Zeit
- Geht es schneller?

- Sicherlich für einige Klassen von Polygonen, z.B. für konvexe Polygone: Wähle einen Eckpunkt des Polygons und ziehe Diagonalen von diesem Eckpunkt zu aller anderen Eckpunkten außer den direkten Nachbarn:

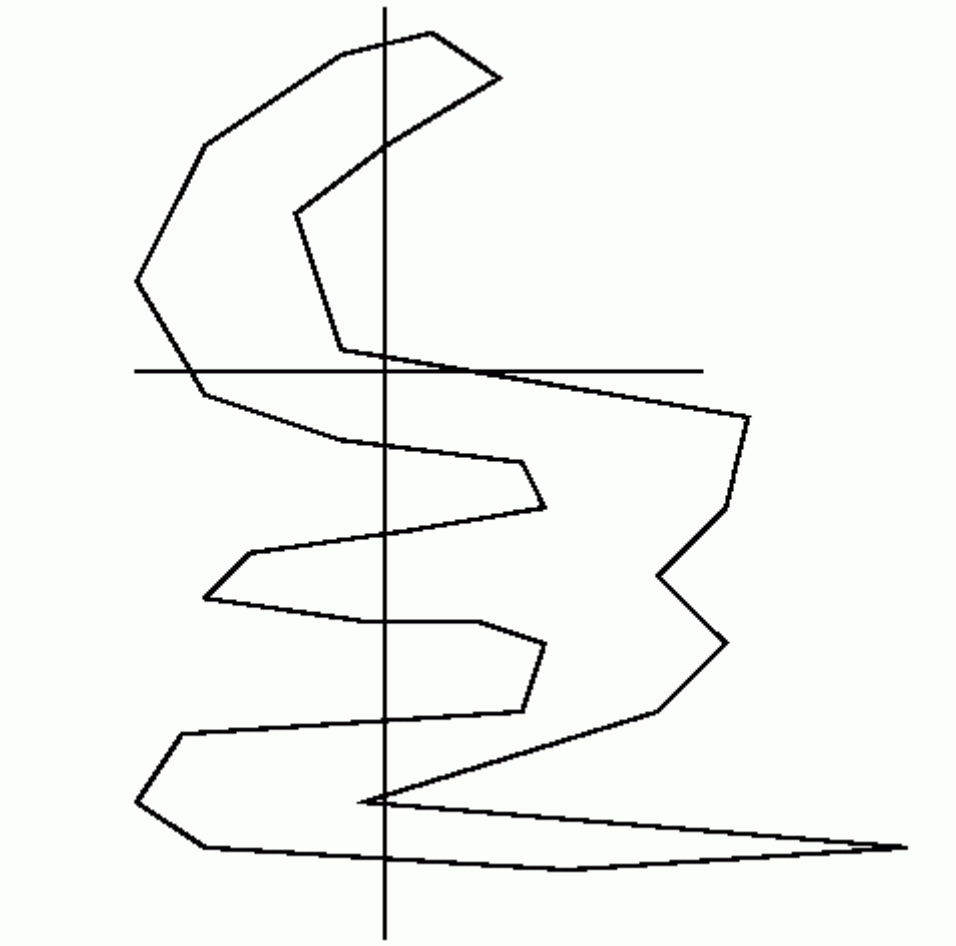


- Zeitaufwand: O(n)

- möglicher Zugang, ein nicht-konvexes Polygon $\mathcal{P}$ zu triangulieren: zerlege $\mathcal{P}$ zunächst in konvexe Teilpoygone und trianguliere dann diese Teilpolygone.
- Aber: Zerlegung eines Polygons in konvexe Teilpolygone ist genauso schwierig wie Triangulation.
- Besser: Zerlegung von $\mathcal{P}$ in sogenannte monotone Teilpolygone, was sich als sehr viel einfacher herausstellt.

Definition:

Eine einfaches Polygon $\mathcal{P}$ heißt *monoton* bzgl. einer Geraden g, falls für jede Gerade g' senkrecht zu g der Schnitt des Polygons mit g' zusammenhängend ist. Mit anderen Worten sollte der Schnitt ein Liniensegment, ein Punkt oder die leere Menge sein. Ein bzgl. der y-Achse monotones Polygon heißt *y-monoton*.



äquivalente Definition:

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 oder gar nicht schneiden.

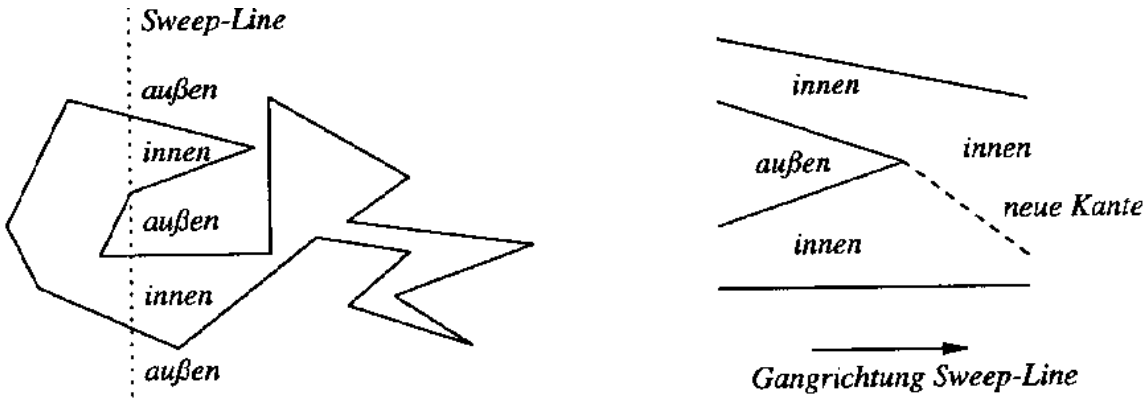
- Charakteristische Eigenschaft für y-monotone Polygone:
  Wenn man vom obersten Eckpunkt entlang des linken (oder rechten) Randes zum untersten Eckpunkt läuft, so bewegt man sich immer abwärts oder horizontal, aber nie aufwärts.
- Strategie für Triangulation von Polygon $\mathcal{P}$:
  Zerlege $\mathcal{P}$ zuerst in y-monotone Teilpolygone und trianguliere dann die Teilpolygone.

Zerlegung in monotone Polygone: in Zeit $O(n \log n)$
Triangulierung monotoner Polygone: in Zeit $O(n)$

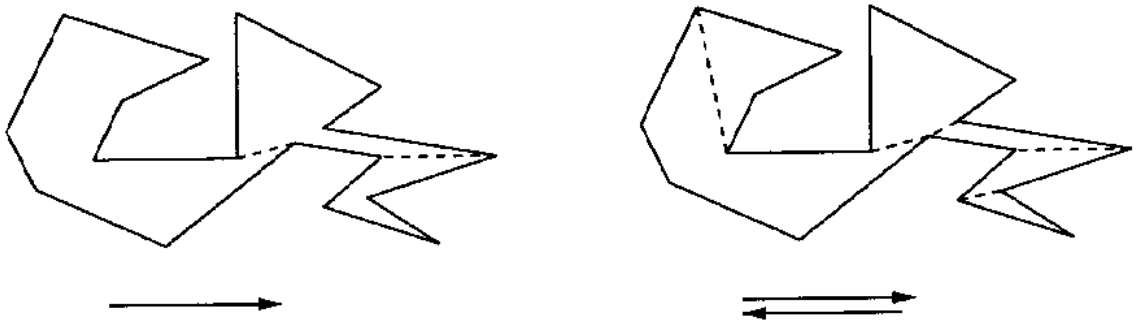
Die Zerlegung eines einfachen Polygons in monotone Polygone ist mit einem Sweep-Verfahren möglich:

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

Zerlegung in *x*-monotone Teilpolygone erfolgt durch zweimaliges Herüberschwenken der 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 ($\Rightarrow$ Überkreuzungsfreiheit von Trennkanten und Polygonkanten).



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

Verfahren zur Triangulierung monotoner Polygone in linearer Zeit:

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 $\neg$(*Nachbar*$(u, e_1) \wedge$ *Nachbar*(u, e_i)) *do begin*

if (*Nachbar*$(u, e_1) \wedge \neg$*Nachbar*(u, e_i))

then begin

Füge Kanten $\overline{ue_1}, \ldots, \overline{ue_i}$ *ein;*

Ersetze Stackinhalt durch e_i, u*;*

u *:=Nachfolger(*u*) in* P*;*

end;

else

begin

if $(i > 1) \wedge ((e_{i-1}e_iu)$ *ist konvexe Ecke*)

then begin

(Fall 2a *)*

Füge Kante $\overline{ue_{i-1}}$ *ein;*

Lösche e_i *im Stack;*

end;

else

begin

(Fall 2b *)*

Lege u *auf den Stack;*

u *:=Nachfolger(*u*) in* P*;*

end;

end;

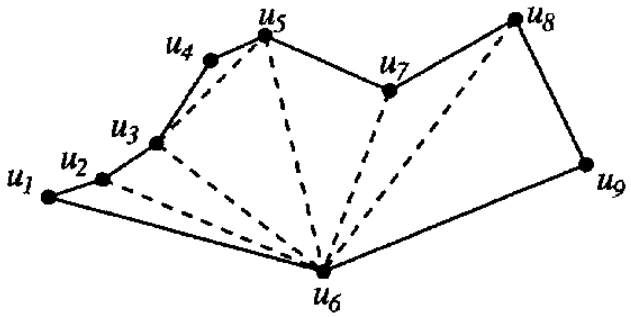
end; (while *)*

(Fall 3 :* *Nachbar*$(u, e_1) \wedge$ *Nachbar*(u, e_i) **)*

Füge Kanten $\overline{ue_2}, \ldots, \overline{ue_{i-1}}$ *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:

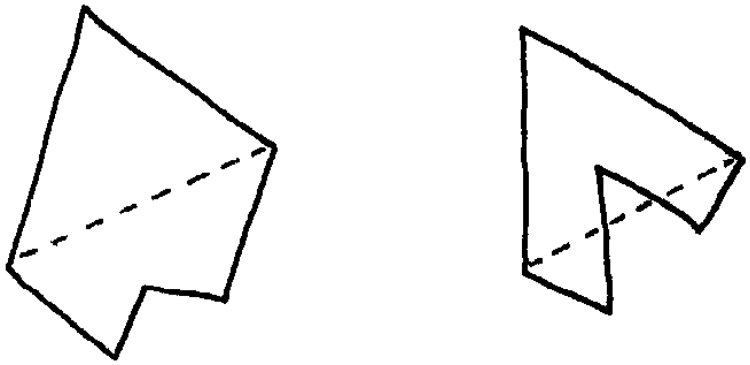


Stack	*u*	*Fall*
u_1, u_2	u_3	*2b*
u_1, u_2, u_3	u_4	*2b*
u_1, u_2, u_3, u_4	u_5	*2a*
u_1, u_2, u_3	u_5	*2b*
u_1, u_2, u_3, u_5	u_6	*1*
u_5, u_6	u_7	*1*
u_6, u_7	u_8	*2a*
u_6	u_8	*3*

Alternative:
einfache Polygone direkt triangulieren

Der 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".



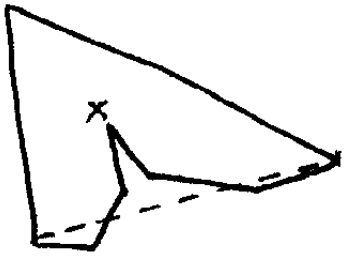
links: Ohr, rechts: konvexe Ecke, die kein Ohr ist

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

Optimales Triangulierungsverfahren? – lange offenes Problem.

Chazelle'91:

Jedes einfache Polygon kann in Zeit $O(n)$ trianguliert werden.

(aber kaum praktikabel)

Seidel'91: randomisierte inkrementelle Konstruktion

Jedes einfache Polygon kann in Zeit $O(n \log^* n)$ trianguliert werden.

(fast linear, praktikabel)

Bem.: $\log^* n$ ist die kleinste ganze Zahl m, für die die m–fache Logarithmierung $\log(\log(...\log(n) \leq 1$ wird.