Der Weiler-Atherton-Algorithmus

Flächenunterteilungsalgorithmus im Objektraum

- Minimierung der Unterteilungen durch Unterteilung an Polygongrenzen
- benötigt: Clipping von Polygonen an Polygonen (allgemeine Polygone, auch konkav, evtl. mit Löchern!)
- Ergebnis: nicht überlappende Polygone, die einfach dargestellt werden können
- kann für visible line determination (bei Drahtgittermodellen) und visible surface determination genutzt werden

Vorgehensweise:

- Vorsortierung der Polygone nach Tiefenwerten
- Kopie des ersten Polygons dieser Liste (dem Betrachter am nächsten) wird als *Clip-Polygon* (*CP*) genutzt
- alle Polygone (einschließlich des ersten) werden an diesem CP geclippt: Clipping der Projektionen der Polygone in 2D

Aufbau von zwei Listen: *inside list* und *outside list*

- Teilpolygone außerhalb des CP $\rightarrow$ outside list
- Teilpolygone innerhalb des CP $\rightarrow$ inside list
- alle vom CP verdeckten Teilpolygone werden entfernt
- wenn inside list nicht leer, dann rekursiv für inside list
- rekursiv für outside list

Der Weiler-Atherton-Algorithmus funktioniert auch mit zyklischen Überlappungen mehrerer Polygone und mit "eingeschobenen" Polygonen:
dann zusätzliche einfache Tests notwendig (i. Prinzip Markieren schon bearbeiteter Polygone)

- wichtig: als Clip-Polygone werden immer die *Original-polygone*, nie Teilpolygone verwendet! $\Rightarrow$ Verringerung der Anzahl der Clipping-Operationen

trotzdem: effektiver und genereller Polygon-Clipping-Algorithmus notwendig

Weiler-Atherton 2D-Clipping-Algorithmus

kann Polygone mit Löchern gegen Polygone mit Löchern clippen

Bezeichnungen:
- zu clippendes Polygon: *subject polygon*, SP
- Polygon, an dem geclippt werden soll: *clip polygon*, CP.

Kanten des resultierenden Polygons sind identisch mit (Teil-) Kanten des CP, d.h. es werden keine neuen Kanten erzeugt ($\Rightarrow$ minimale Anzahl resultierender Polygone).

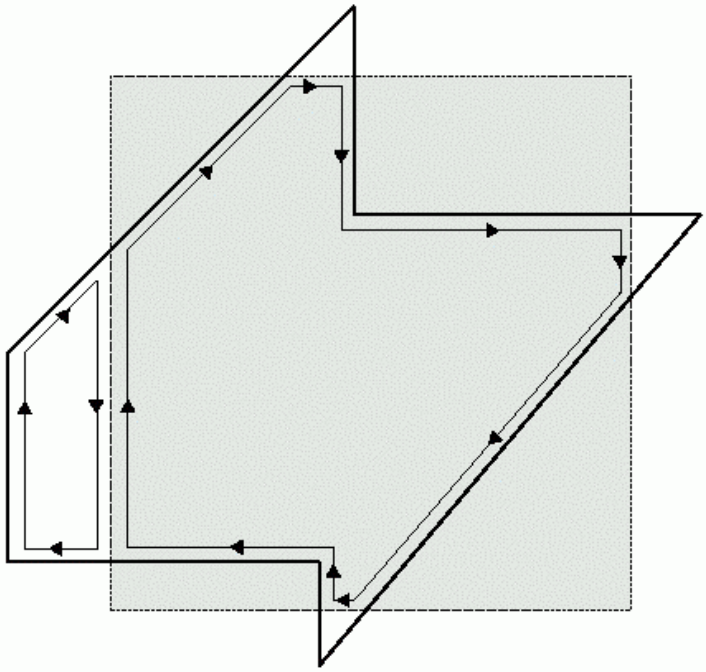
SP und CP werden als zyklische Listen von Eckpunkten beschrieben.
Dabei:
- Beschreibung äußerer Kanten im Uhrzeigersinn
- Beschreibung innerer Kanten (Löcher) entgegen dem Uhrzeigersinn

$\Rightarrow$ das Innere des Polygons ist beim Verfolgen der Kanten immer rechts von der Kante!

topologisches Prinzip: Schnittpunkte zwischen SP und CP treten immer paarweise auf (eine SP-Kante betritt das CP und eine verlässt es).

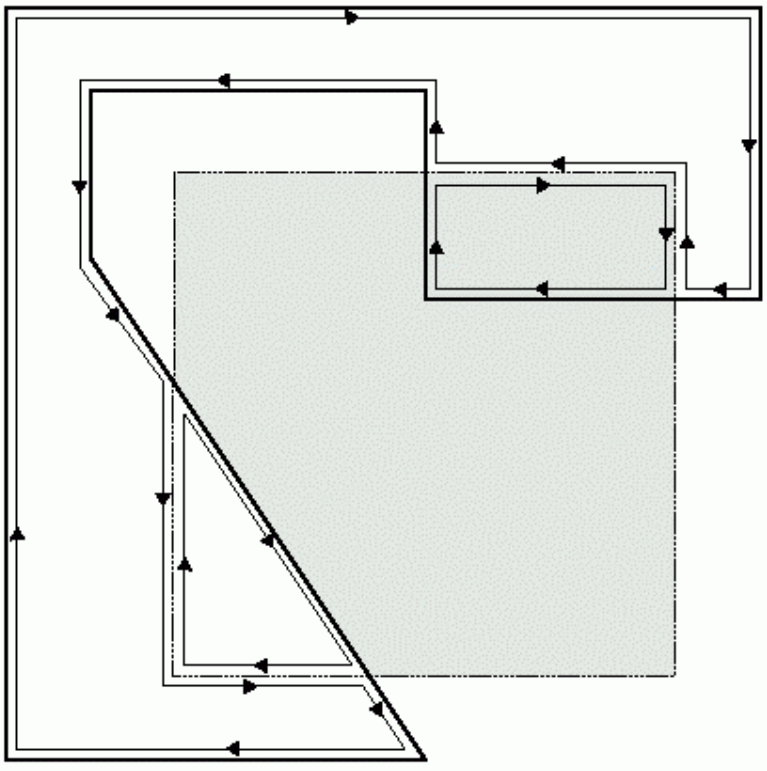
Idee des Algorithmus:
- starte an einem Schnittpunkt, wo eine SP-Außenkante in das CP eintritt
- folge der Kante des SP im Uhrzeigersinn bis zum nächsten Schnittpunkt
- drehe dort nach rechts und folge der Kante des CP im Uhrzeigersinn bis zum nächsten Schnittpunkt
- drehe wieder nach rechts und folge dem SP...
- usw. bis der Startpunkt wieder erreicht wird.
- Gleiche Vorgehensweise bei inneren Kanten, dann entgegen dem Uhrzeigersinn.

Beispiel: „Einfache“ Polygone

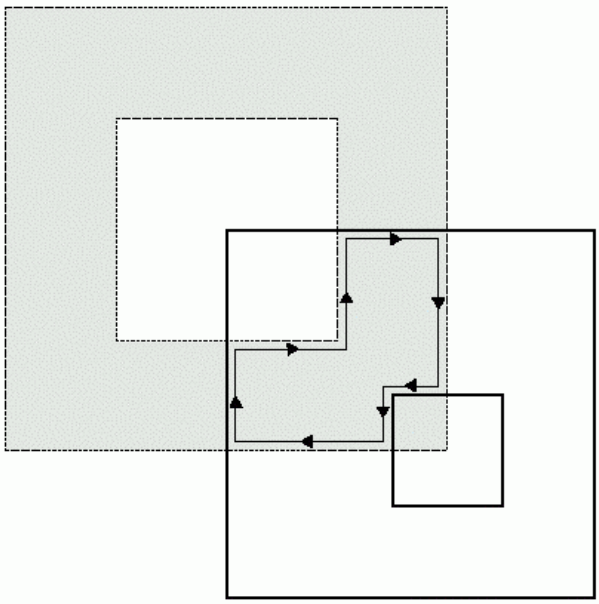


(CP: grau unterlegt)

Beispiel: Umschließendes Polygon



Beispiel: SP und CP mit Löchern



Durch Markierungen muss sichergestellt werden, dass kein Pfad zweimal durchlaufen wird.

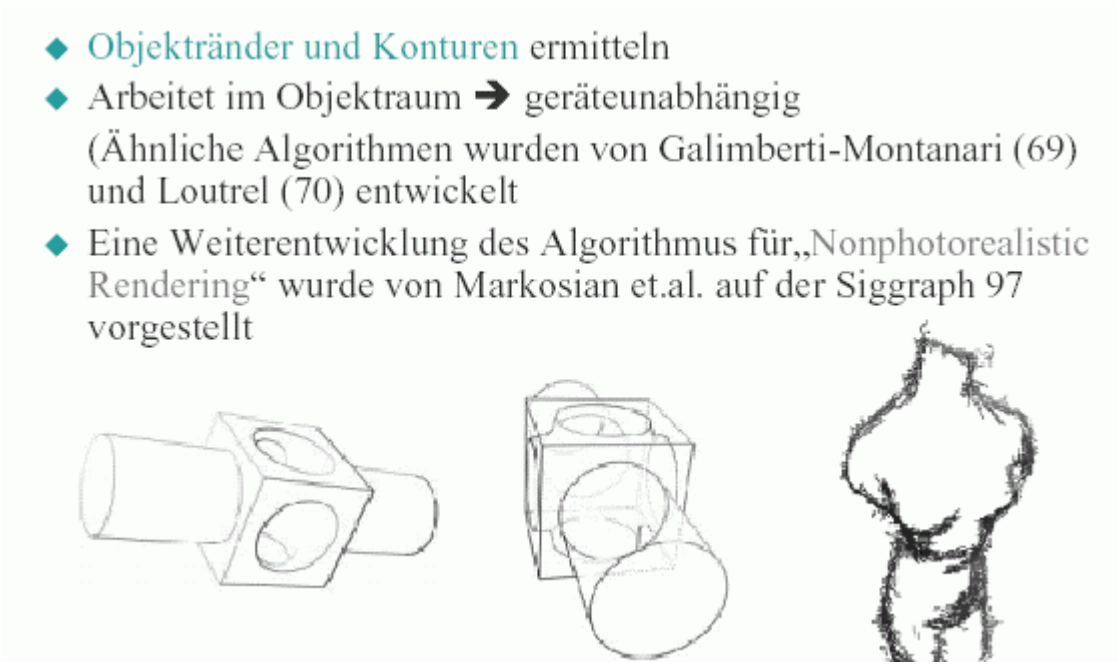
Gegenüber dem Warnock-Algorithmus sind erheblich weniger Unterteilungen vorzunehmen, allerdings sind sie mit komplexeren Operationen verbunden.

Zusammenfassung Weiler-Atherton

- Komplexität: $O((v + I) \log v)$ mit v = Anzahl Vertices und I = Anzahl Schnittpunkte (bei entsprechender Implementierung)
- Aber: Anzahl der Schnittpunkte $O(n^2)$ mit n=Anzahl Polygone
- analytischer Algorithmus im Objektraum $\rightarrow$ hohe Genauigkeit
- kein Preprocessing
- nicht in Hardware realisierbar

Der Algorithmus von Appel

sehr alter Algorithmus (Appel 1967) für visible **line** determination



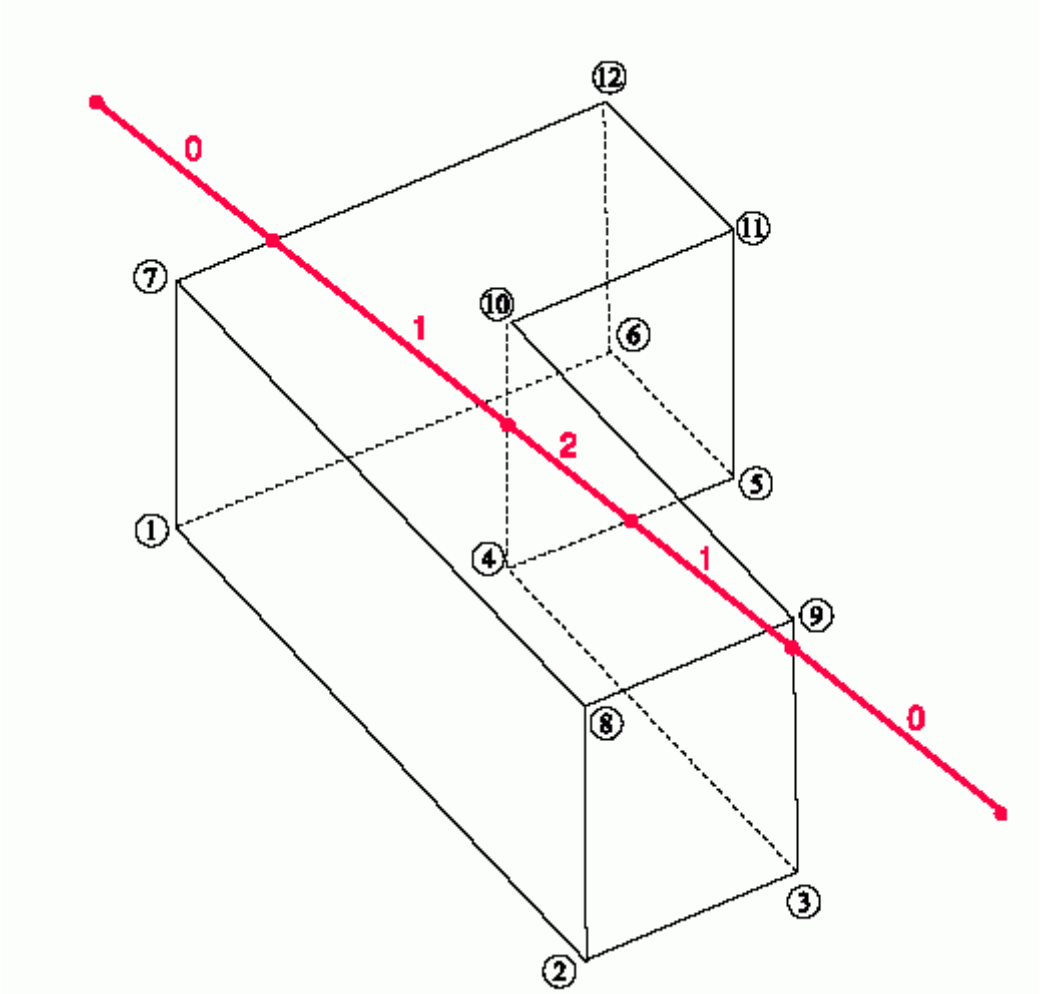
- "Quantitative Unsichtbarkeit" (*quantitative invisibility*, QI) einer Linie als Grundelement
- Erweiterungen zum Zeichnen von "haloed lines"

Quantitative Unsichtbarkeit:

Def.: Anzahl potenziell sichtbarer Flächen, die zwischen dem Liniensegment und dem Betrachterstandpunkt liegen.

- Eine Fläche ist "potenziell sichtbar", wenn der Winkel zwischen nach außen zeigender Normale und LOS größer als 90° ist, sonst unsichtbar (vgl. back-face culling).
  (Die LOS ist dabei vom Betrachter weg orientiert.)
- QI eines Liniensegments ist 0 $\Rightarrow$ Liniensegment sichtbar
- QI ändert sich nur um ± 1 und nur an Konturkanten
- "Konturkante": gemeinsame Kante einer potenziell sichtbaren und einer unsichtbaren Fläche
- QI erhöht sich um 1, wenn die Linie eine Region betritt, die von einer Konturkante begrenzt wird, und verringert sich um 1 bei deren Verlassen.

Beispiel:



Konturkanten sind hier: 2-3, 3-9, 9-10, 5-11, 11-12, 12-7, 7-1, 4-10.
Die betrachtete Linie beginnt mit einer QI von 0.

- geht hinter Linie 7-12 vorbei: QI = +1
- geht hinter Linie 4-10 vorbei: QI = +2
- geht hinter Linie 4-5 vorbei: QI = +1
- geht hinter Linie 9-3 vorbei: QI = 0

Alle Teile der Linie mit QI = 0 sind sichtbar.

Entscheidung, ob ein Liniensegment eine Region betritt oder verlässt: mittels parametrischer Beschreibung der Geraden

Entscheidung, ob Linie vor oder hinter einer Konturkante vorbeigeht:

- betrachte Dreieck mit Konturkante als Basis und Betrachterstandpunkt als Spitze
- wenn Linie die Ebene des Dreiecks durchstößt und der Schnittpunkt liegt innerhalb des Dreiecks $\Rightarrow$ Linie geht vor der Konturlinie vorbei.

Bleibt noch als Entscheidung: QI des ersten Punktes.

- Raycasting: Strahl vom Betrachterstandpunkt zum Punkt.
- Zählen der Schnittpunkte des Strahls mit Polygonen vor dem Punkt.

Haloed Lines:

- Hervorheben der Linienkreuzungen
- Projektion jeder Linie wird mit einem weißen "Halo" umgeben
- wenn Linie A näher zum Betrachter liegt als Linie B, dann wird der Teil von B, der im Halo vor A liegt, nicht gezeichnet.

Erweiterungen möglich: Zeichenstärke der Linien abhängig von ihrer Entfernung zum Betrachter (Elber 1995).

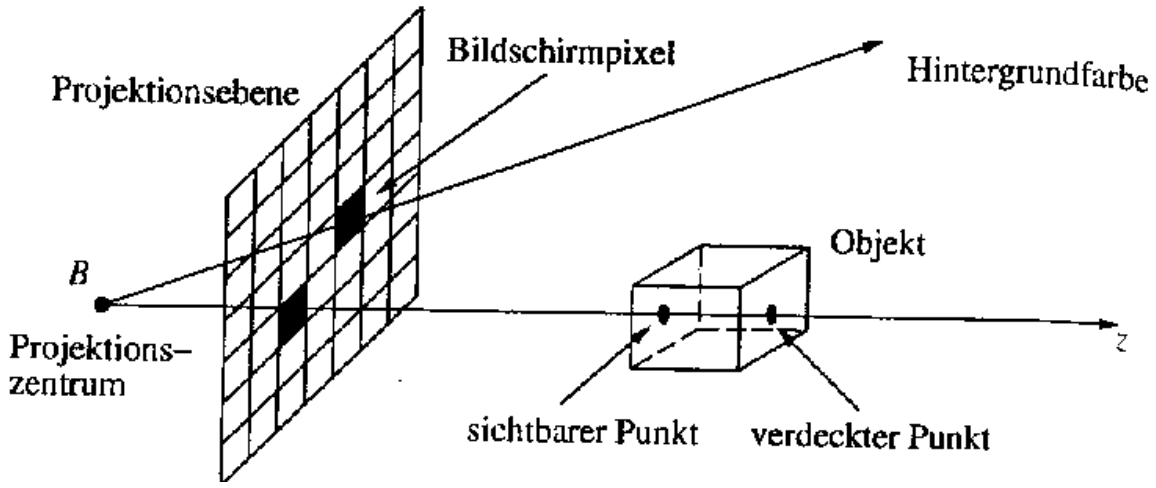
Zusammenfassung zu Appels Algorithmus:

- Objektraumalgorithmus ($\Rightarrow$ keine Abtastfehler, geräteunabhängig, Genauigkeit nicht auflösungsabhängig – z.B. wichtig für Plotterausgabe)
- liefert Silhouetten
- ist für "non-photorealistic rendering" (künstlerische Anwendungen) geeignet
- relativ aufwändige Berechnung
- für große Szenen in der derzeitigen Form nicht echtzeitfähig.

Ray-Casting-Verfahren

- Hybrid Objektraum-Bildraum
- Prinzip der Strahlverfolgung (vgl. Raytracing in der Beleuchtungsrechnung, siehe später)
- vom Betrachterstandpunkt wird je ein Strahl durch jedes Pixel in die 3D-Szene verfolgt.
- Berechnung der Schnittpunkte der Strahlen mit den Objektoberflächen

Fallunterscheidung dabei:
1. es gibt keinen Schnittpunkt: verwende die Hintergrundfarbe.
2. es existieren Schnittpunkte: bestimme den zum Betrachter nächstgelegenen, dieser ist sichtbar, die anderen verdeckt.



Nachteil: hoher Rechenaufwand durch häufige Schnittpunkt-berechnungen (ca. 95 % der Gesamtrechenzeit bei typischen Szenen)
Vorteile:

- leicht parallelisierbar (bei *p* Prozessoren teile Bildschirm in *p* disjunkte Rechtecke – allerdings muss bei diesem Ansatz die 3D-Szene auf jedem Prozessor vollständig zur Verfügung stehen)
- Effizienzverbesserung durch *boundary volume*-Techniken möglich (hierarchische Konzepte, Clustering)
- Kombination mit klassischem Raytracing (Beleuchtungs-rechnung) möglich

Abschließende Bemerkungen zur Bewertung von Visibilitätsverfahren

- Objektraumalgorithmen (analytische Verfahren): i.allg. wenige, aber komplizierte Tests
- Bildraumalgorithmen (Rasterverfahren): viele einfache Tests

optimale Verfahren liegen aus theor. Gründen in der Klasse der Objektraumalgorithmen (Hornung 1984).
Diese algorithmische Minimalität sagt aber kaum etwas über die praktische Nützlichkeit eines Verfahrens aus:

- alle derzeit gängigen Echtzeitverfahren sind Raster-algorithmen!

Faustregel:

- Ausgabe als Strichzeichnung: Objektraumalgorithmen besser
- flächenhafte Ausgabe: Bildraumalgorithmen (Raster) besser

- Anwendung bestimmt Auswahl des Algorithmus
- Tradeoff zwischen hoher Interaktionsrate und Exaktheit der Darstellung

mit weiterer Verbesserung und Verbilligung der Speicher-bausteine wird der (erweiterte) *z*-Buffer-Ansatz immer attraktiver

- mit zunehmender Szenenkomplexität (größere Anzahl von Dreiecken) wird die Zahl der überdeckten Pixel pro Dreieck geringer und verteilt sich m.o.w. gleichmäßig über die ganze Bildschirmfläche

$\Rightarrow$ Preis-Leistungs-Verhältnis des *z*-Buffers wird mit steigender Szenenkomplexität immer besser.