





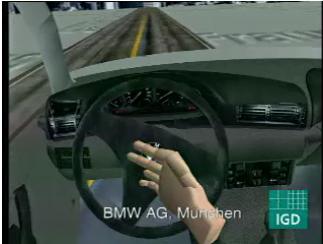

# Virtuelle Realität Kollisionserkennung

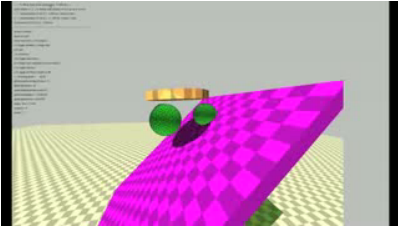
G. Zachmann  
Clausthal University, Germany  
[cg.in.tu-clausthal.de](http://cg.in.tu-clausthal.de)



## Anwendungsbeispiele

*Virtual Prototyping*





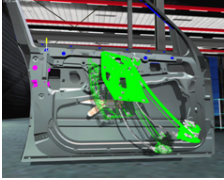
*Physikalisch basierte Simulation*

G. Zachmann Virtuelle Realität und Simulation - WS 09/10

Kollisionserkennung 2

## Einsatzgebiete von Kollisionserkennung

- Grundlegende Operation:
  - Physikalisch-basierte Simulation
  - Interaktion in VR
  - Haptisches Rendering
- Anwendungsfelder:
  - Spiele, Animation, Medizin, Virtual Prototyping, Pfadplanung, Teleoperation, Roboter-Kollisionsvermeidung, ...





*Hierarchische Kollisionserk.*

G. Zachmann Virtuelle Realität und Simulation - WS 09/10
Kollisionserkennung 3

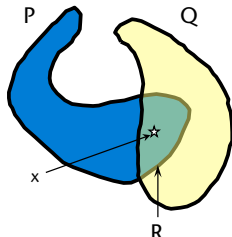
## Koll.erkennung innerhalb einer Simulation

- Main loop:
  - Objekte bewegen
  - Kollisionen checken
  - Kollisionen behandeln, z.B.: Objekte zurückbewegen, Kräfte bestimmen
- Kollisionen stellen zwei Probleme:
  - Kollisionserkennung
  - Kollisionsbehandlung
- Im folgenden: Kollisionserkennung

G. Zachmann Virtuelle Realität und Simulation - WS 09/10
Kollisionserkennung 4

## Definitionen

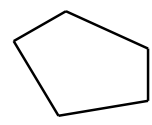
- Gegeben  $P, Q \subseteq \mathbb{R}^3$
- Erkennungsproblem (“*detection problem*”):  
 “P und Q Kollidieren”:  $\Leftrightarrow$   
 $P \cap Q \neq \emptyset \Leftrightarrow$   
 $\exists x \in \mathbb{R}^3: x \in P \wedge x \in Q$
- Konstruktionsproblem (“*construction problem*”):  
 $R := P \cap Q$
- Definition “**Kollision**” für polygonale Objekte:  
 P,Q kollidieren  $\Leftrightarrow$   
 $\exists f \in F^P \exists f' \in F^Q: f \cap f' \neq \emptyset$
- Andere Definition in der Spielebranche



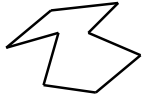
G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 5

## Objekt-Klassen


- konvex
- geschlossen, einfach  
(keine Selbstdurchdringung)
- Sack Polygone (“*polygon soup*”)  
 ▪ nicht notwendig geschlossen  
 ▪ doppelte Polygone  
 ▪ koplanare Polygone  
 ▪ Selbstdurchdringungen  
 ▪ degenerierte Polygone  
 ▪ Löcher
- starr / flexibel



konvex



einfach & geschlossen



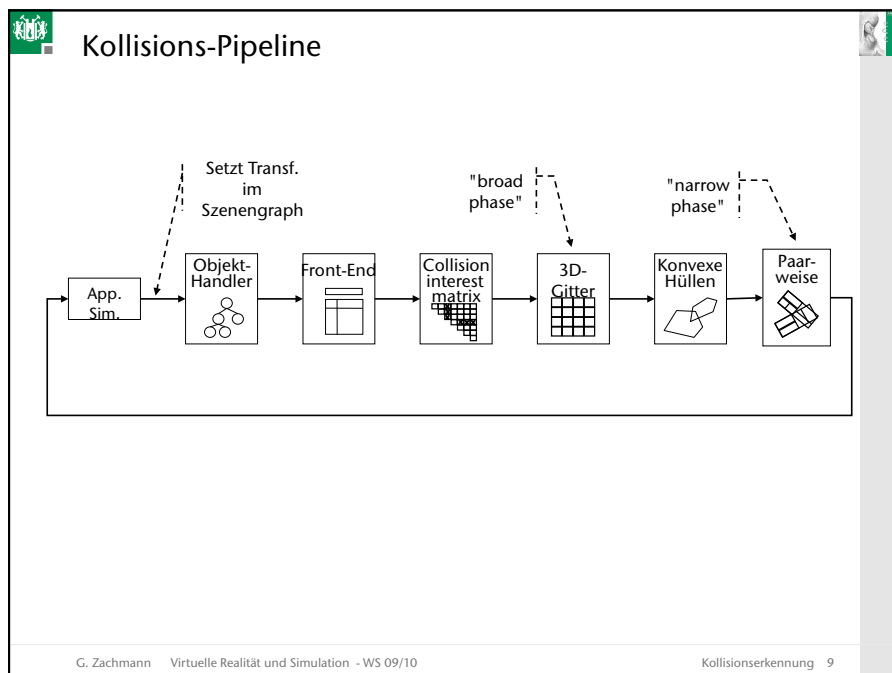
polygon soup

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 7

## Anforderungen an Kollisionserkennung

- Möglichst große Klasse von Objekten
- Viele bewegliche Objekte (einige 1000)
- Schnell, damit physikalische Simulation iterieren kann (wenigstens  $2 \times 100,000$  Polygone in  $< 1$  Millisek.)
- Kollisionspunkt ("*witness*"), falls Kollision; und optional: alle Kollisionspunkte
- Nicht zu große zusätzliche Datenstrukturen ( $< 2x$ ); der Aufbau dieser Datenstrukturen sollte nicht zu lange dauern, damit man das zur Ladezeit machen kann ( $< 5$ sec / Objekt)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 8



## Die Collision-Interest Matrix

- Anwendungsspezifisch:
  - Verschiedene Module interessieren sich für verschiedene Paare;
  - manche Objekt-Paare kollidieren immer, manche Paare können nicht kollidieren;
- Vermeide unnötige Kollisionstests  
⇒ *Collision-Interest Matrix*
- Dreiecks-Matrix, Elemente enthalten:
  - Flag, ob Kollisionserkennung
  - bei inkrementellen Algos  
Infos zum Status beim letzten Frame  
(z.B. bei Algo S die separierende Ebene)
  - *Callbacks* in die Module

Obj	1	2	3	4	5	6	7	8
1		x	x	x	x			
2					x			
3						x		x
4							x	
5								x
6								
7								x
8								

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 10

## Mehrkörper-Kollisionserkennung

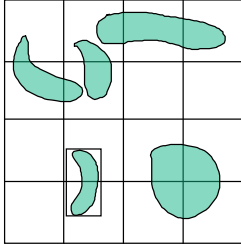
- Kollisionserkennung vieler Objekte:
  - Schließe jedes Objekt in eine Bounding-Box ein; vergleiche deren BBoxes vor dem exakten Koll.test
- n Objekte bewegen sich  
→ *brute-force* Methode muss  $O(n^2)$  BBoxes vergleichen.
- Idee:
  - versuche zum Objekt P schnell die “Nachbarn” zu finden und mache nur mit diesen Bbox-Vergleiche
  - → Raum-Gitter, *Sweep-Plane*, etc.

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 11

## Raum-Gitter

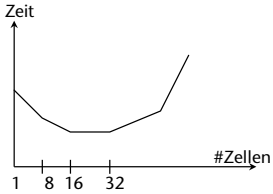
Idee:

1. Teile die "Welt" in regelmäßiges Gitter.
2. Objekte "benachbart", wenn sie gemeinsame Gitterzelle(n) belegen.
3. Bestimme Zellen-Belegung anhand der BBoxes (exakte Bestimmung zu teuer)
4. Objekt bewegt sich  
→ Gitter updaten.



Trade-Off:

- weniger Zellen = größere Zellen  
→ entfernte Objekte sind "benachbart";
- mehr Zellen = kleinere Zellen  
→ Objekte belegen mehr Zellen  
→ Aufwand zum Update wird größer



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 12

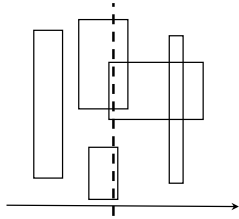
## Plane-Sweep

Idee:

Lasse eine Ebene senkrecht zur x-Achse durch den Raum streichen ("sweep")

Algo:

- sortiere x-Koord. der Box-Ränder
- starte mit der linken Box
- führe Liste "aktiver" Boxes
- springe von Box-Rand zu Box-Rand:
- if** aktueller Box-Rand ist "linke" Seite einer Box
- füge diese Box zur aktiven Liste hinzu
- checke neue Box gegen alle anderen in der aktiven Liste (2D)
- else**
- entferne diese Box aus der aktiven Liste



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 13

## Szenengraph

Idee:

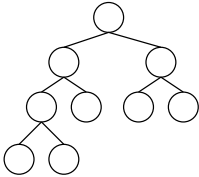
- Verwende die Hierarchie des Szenen-Graphen,
- Analog zu hierarchischer Koll.erkennung.

Unterschiede:

- Nur 1 Hierarchie statt 2;
- Blätter = Objekte (statt Blätter = Polygone);
- Alle Blätter bewegen sich.

Probleme:

- Hierarchie wird schnell ineffizient,
- Oft keine Hierarchie von außen vorgegeben.  
(Bsp.: Auto-Daten)



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 14

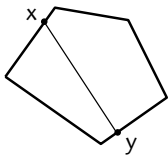
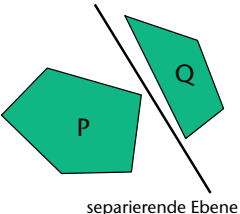
## Frame-to-Frame Coherence

- Beobachtung:  
*Zwei aufeinander folgende Bilder in einer zusammenhängenden Sequenz unterscheiden sich (meistens) wenig.*
- Beispiele
  - Kamerabewegung
  - Objektbewegung
- Anwendungen
  - Computer Vision (Bsp.: Tracking von Markern)
  - MPEG
  - Kollisionserkennung
  - Ray Tracing von Animationen (?)
- Algorithmen basierend auf frame-to-frame coherence heißen **“inkrementell”**, manchmal auch **“dynamisch”** oder **“on-line”**

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 26

## Konvexe Objekte

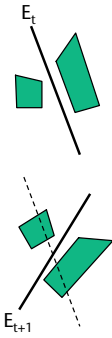
- Definition “konvexer Polyeder”:
 
$$P \subset \mathbb{R}^3 \text{ konvex} \Leftrightarrow \forall x, y \in P : \overline{xy} \subset P \Leftrightarrow P = \bigcap_{i=1, \dots, n} H_i, H_i = \text{Halbräume}$$
- Bedingung für Nicht-Kollision: “linear separierbar”
 
$$P \cap Q = \bigcap_{i=1}^{n1+n2} H_i = \emptyset \Leftrightarrow \exists i : P \subseteq H_i \wedge Q \subseteq H_i^c$$
 (“P liegt ganz auf der einen Seite von  $H_i$ , Q liegt ganz auf der anderen Seite”)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 27

## Algorithmus “separierende Ebene”

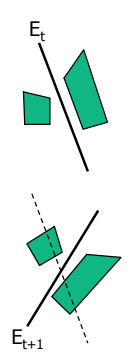
- Idee: wenn im letzten Frame E eine trennende Ebene zwischen P und Q war, dann ist in diesem Frame die trennende Ebene “in der Nähe” von E (evtl. ist es sogar dieselbe).



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 29



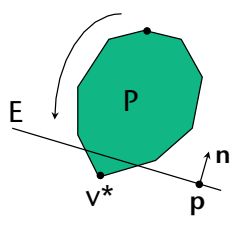
$E_t$  war separierende Ebene zu  $P, Q$  im Frame  $t$  gewesen  
 $E := E_t$   
**repeat** max n times  
 if es ex.  $v \in V^P$  auf der Rückseite von  $E$   
   drehe/verschiebe  $E$ , so daß  $v$  auf der Vorderseite  
 if es ex.  $v \in V^Q$  auf der Vorderseite von  $E$   
   drehe/verschiebe  $E$ , so daß  $v$  auf der Rückseite  
 if kein  $v \in V^P$  und kein  $v \in V^Q$  auf der “falschen” Seite  
   **return** “keine Kollision”  
 es gibt immer noch  $v$ 's auf der “falschen” Seite  
   **return** “Kollision” {kann manchmal falsch sein}  
 speichere  $E_{t+1} := E$  für's nächste Frame



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 30

### Finde schnell eine Ecke auf der “falschen” Seite

- Brute-Force: teste für alle  $v$  ob
 
$$f(v) = (v - p) \cdot n > 0$$
- Beobachtungen:
  1.  $f$  ist linear,
  2.  $\exists^1 v^* : f(v^*) = \min$
  3.  $P$  konvex  $\Rightarrow f(x)$  hat genau ein lokales Minimum über allen Punkten  $x$  auf der Oberfläche von  $P$  (oder mehrere und diese liegen parallel zu  $E$ )
- Algo (minimales  $v$  bzgl.  $f$  suchen)
  - starte mit irgendeiner Ecke  $v$
  - gehe zu demjenigen Nachbarn  $v'$  von  $v$ , für das  $f$  am kleinsten ist
  - fertig, falls es kein “kleineres”  $v'$  gibt



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 31

## Eigenschaften des Algorithmus'

- + Erwartete Laufzeit ist  $O(1)$ !  
 Der Algo nützt *frame-to-frame coherence* aus:  
 wenn sich die Objekte wenig bewegt haben, dann muss man  
 nur überprüfen, dass die separierende Ebene immer noch eine ist;  
 wenn die trennende Ebene ein bisschen verschoben werden  
 muss, dann ist man meistens nach wenigen Iterationen fertig.
- + Funktioniert auch für sich verformende Objekte,  
 solange sie konvex bleiben
- Funktioniert nur für konvexe Objekte
- Terminiert nicht notwendigerweise;  
 also bricht man die Schleife nach **max** erfolglosen Versuchen ab;  
 dann kann der Algo ein falsches Ergebnis liefern.
- *Frage: Gibt es eine deterministische Variante ?!*

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 32

## Closest Feature Tracking

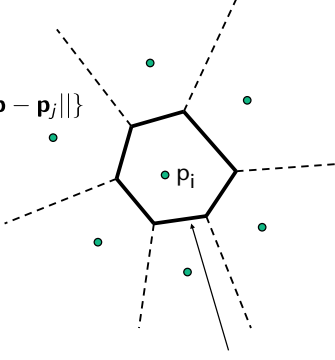
- Vorgestellt von Lin & Canny 1992  
 ( → Lin-Canny-Algorithmus)
- Idee
  - Verfolge Minimalabstand zwischen beiden Objekten
  - Wird realisiert durch je ein Punkt auf der Oberfläche
  - Bei kontinuierlicher Bewegung der Objekte wandern diese Punkte  
 kontinuierlich über die Oberfläche
- Zugrunde liegende Verfahren
  - Voronoi-Diagramme
  - "closest features"

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 33

## Voronoi-Diagramme zu Punkten

- Gegeben eine Menge Punkte  $S = \{p_i\}$
- Definition Voronoi-Region :  

$$V_i := \{p \in \mathbb{R}^2 \mid \forall j \neq i : \|p - p_i\| < \|p - p_j\|\}$$
- Definition Voronoi-Diagramm :  
Menge aller Voronoi-Regionen zu den Punkten in  $S$ .
- Partition der Ebene in Kanten und Voronoi-Regionen
- Interaktive Demo: <http://web.cs.uni-bonn.de/l/GeomLab/VoroGlide/>

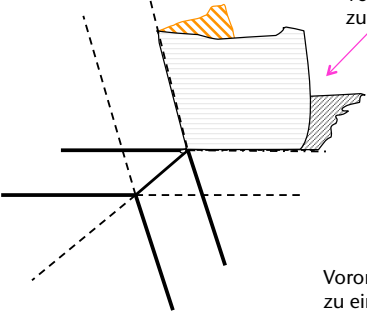


Voronoi-Region zu einem Punkt

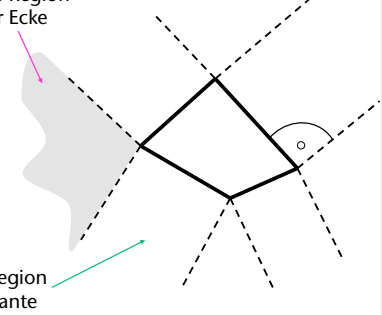
G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 34

## Voronoi-Diagramme zu Polyedern

### Voronoi-Regionen in 3D



### Voronoi-Regionen in 2D



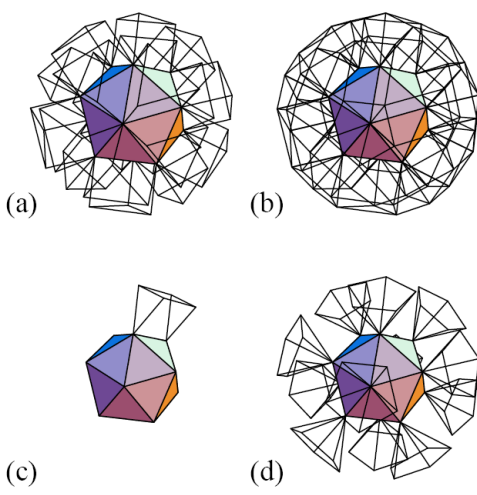
Voronoi-Region zu einer Ecke

Voronoi-Region zu einer Kante

Äußere Voronoi-Regionen sind für konvexe Objekte sehr einfach zu konstruieren!  
(Innere Voronoi-Regionen brauchen wir nicht.)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 35

## Äußere Voronoi-Regionen eines Polyeders



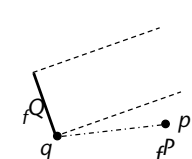
The external Voronoi regions of ...

- (a) faces
- (b) edges
- (c) a single edge
- (d) vertices

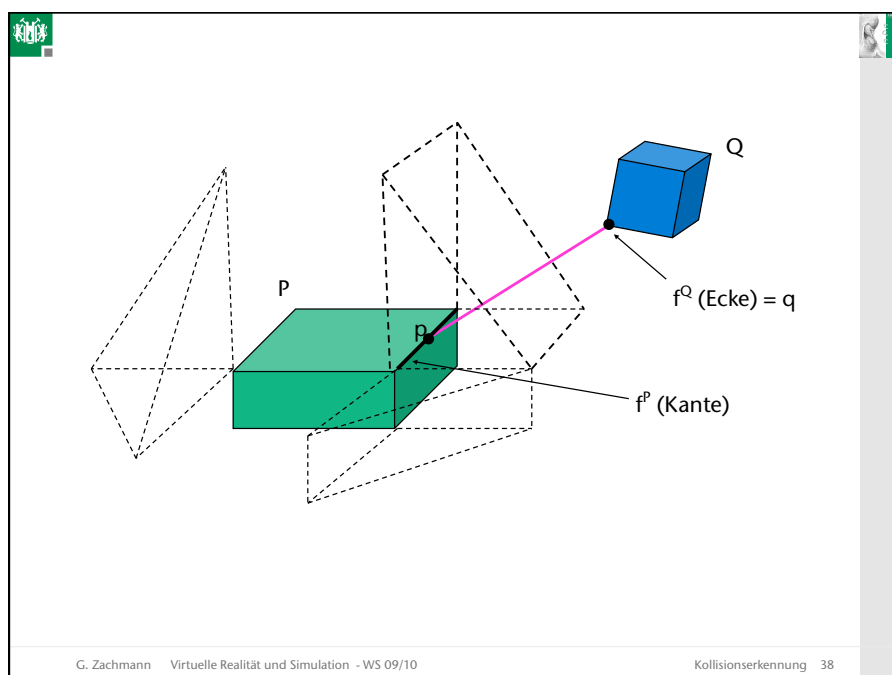
G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 36

## Closest Features

- Definition *Feature*  $f^P$  :=  
Ecke, Kante oder Polygon eines Polyeders  $P$ .
- Definition "*Closest Feature*":  
Seien  $f^P$  und  $f^Q$  zwei Features auf  $P$  bzw.  $Q$ , und seien  $p, q$  Punkte auf  $f^P$  bzw.  $f^Q$  die den minimalen Abstand von  $P$  und  $Q$  realisieren, d.h.,  $d(P, Q) = d(f^P, f^Q) = \|p - q\|$ .  
Dann heißen "*closest features*".
- Lemma:  
Sei  $V(f)$  die Voronoi-Region zu einem Feature  $f$ ;  
 $f^P, f^Q$  sind "*closest features*"  $\Leftrightarrow$   
 $p$  liegt in  $V(f^Q)$ ,  $q$  liegt in  $V(f^P)$ .



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 37

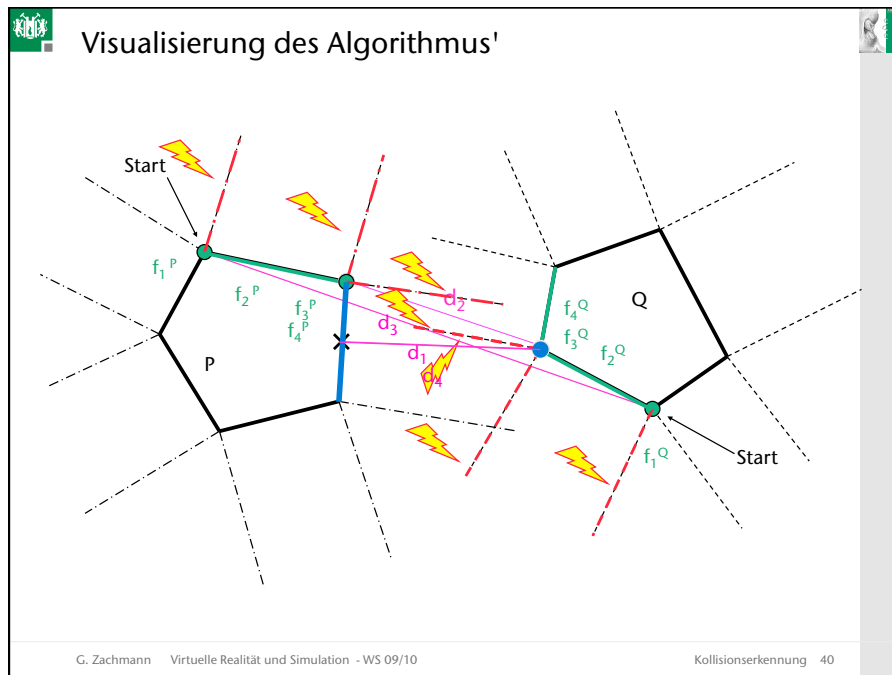


### Algorithmus

starte mit zwei beliebigen Features  $f^P, f^Q$  auf  $P$  bzw.  $Q$   
**while** ( $f^P, f^Q$ ) sind noch nicht closest features &&  $d(f^P, f^Q) > 0$   
     **if** ( $f^P, f^Q$ ) wurde schon einmal betrachtet  
         **return** "Kollision" (weil Zyklus)  
     bestimme  $p$  und  $q$ , die den Abstand zwischen  $f^P, f^Q$  realisieren  
     **if**  $p \in V_q$  und  $q \in V_p$   
         **return** "keine Kollision", ( $f^P, f^Q$ ) sind closest features  
     **if** ex. eine Seite von  $V_q$  bzgl. der  $p$  auf der falschen Seite liegt  
      $f^P \leftarrow$  das Feature der "dahinter" liegenden Voronoi-Region  
     analog für  $q$ , falls  $q \notin V_p$   
**if**  $d(f^P, f^Q) > 0$   
     **return** "keine Kollision"  
**else**  
     **return** "Kollision"

**Achtung:** bei Kollision befinden sich einige Features im Innern des anderen Objektes, aber im Innern ex. keine Voronoi-Regionen!

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 39



### Anmerkungen

- Kleine Denkaufgabe:  
Das *Voronoi-Diagramm* braucht man eigentlich nicht!  
(aber mit *Voronoi-Diagramm* ist der Algo schneller)
- Berechnungsdauer hängt ab vom "Maß" der zeitlichen Kohärenz
- Verbesserung durch *Lookup-Table*:  
trage sphärische Koordinaten der Features  
in Tabelle ein

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 41

Movie

## Incremental Collision Detection for Polygonal Models

Madhav K. Ponamgi  
Jonathan D. Cohen  
Ming C. Lin  
Dinesh Manocha

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 42

Hierarchische Kollisionserkennung

- Für "Polygon soups"
- Algorithmentechnik:  
Divide & Conquer

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 43

### Bounding Volume Hierarchy (BVH)

- Schließe alle Polygone aus P in ein Hüllvolumen (*bounding volume*)  $BV(P)$  ein
- Teile P auf in  $P_1, P_2, P_3, \dots, P_n$  mit  $P_1 \cup P_2 \cup P_3 \cup \dots \cup P_n = P$
- Rekursiv für die  $P_i$ .

→ *bounding volume hierarchy*

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 44

### Simultane Traversierung

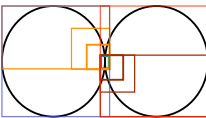
traverse(X, Y)  
 if X, Y do not overlap then  
     return  
 if X, Y are leaves then  
     check polygons  
 else  
     for all children pairs do  
         traverse(  $X_i, Y_j$  )

Bounding Volume Test Tree (BVTT)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 45

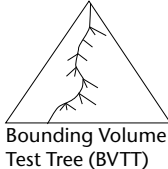


## Einfache Laufzeit-Abschätzung

- Best-case:  $O(\log n)$ 

- Einfache *average-case* Abschätzung:
  - $P[k]$  = Wahrsch.keit daß genau  $k$  Kinderpaare überlappen,  $k \in [0, \dots, 4]$ 

$$P[k] = \frac{1}{16} \binom{4}{k}$$
  - Annahme: alle Ereignisse sind gleich wahrscheinlich
  - Erwartete Laufzeit :
 
$$T(n) = \frac{1}{16} \cdot 0 + \frac{4}{16} \cdot T\left(\frac{n}{2}\right) + \frac{6}{16} \cdot 2T\left(\frac{n}{2}\right) + \frac{4}{16} \cdot 3T\left(\frac{n}{2}\right) + \frac{1}{16} \cdot 4T\left(\frac{n}{2}\right)$$

$$T(n) = 2T\left(\frac{n}{2}\right) \in O(n)$$
- In der Praxis besser



Bounding Volume Test Tree (BVTT)






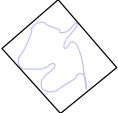


G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 46

## Bounding Volumes

Anforderungen:

- sehr schneller Überlappungstest
- auch dann, wenn die *Bounding Volumes* rotiert oder transl. sind!  
 → "einfache" *Bounding Volumes*
- eine Überdeckung des ganzen Raumes sollte möglichst wenig mehrfach belegten Raum haben → "tight BVs"

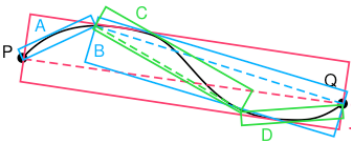
Einige mögliche *Bounding Volumes*:

 Box (AABB)	 k-DOP hier z.B. 8-DOP	 Kugelschale (spherical shell)	 Prisma
 Kugel	 OBB (oriented bounding box)	 Konvexe Hülle	 Zylinder

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 47

## The Wheel of Re-Invention

- OBB-Trees: wurden für 2D-Kurven in 1981 schon von Dana Ballard vorgeschlagen, nannten diese "strip trees"



- AABB-Hierarchien: wurden in den 80er Jahren auch in der Datenbank-Gemeinde erfunden, heißen dort "R-Tree", "R\*-Tree", "X-Tree", etc.

G. Zachmann Virtuelle Realität und Simulation - WS 09/10
Kollisionserkennung 48

## Exkurs: Das Rad der Fortuna




Boccaccio De Casibus Virorum Illustrium Paris: 1467



Codex Buranus

G. Zachmann Virtuelle Realität und Simulation - WS 09/10
Kollisionserkennung 49

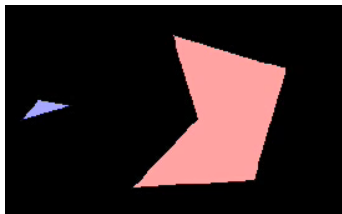
## Schnitttest für konvexe BVs (Polyeder)



- Hermann Minkowski (1864 – 1909), deutscher Mathematiker und Physiker
- Definition (*Minkowski-Summe*):  
Seien  $A$  und  $B$  Teilmengen eines Vektorraums;  
die Minkowski-Summe von  $A$  und  $B$  ist
 
$$A \oplus B = \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in A, \mathbf{b} \in B\}$$
- Entsprechend die *Minkowski-Differenz*:
 
$$A \ominus B = \{\mathbf{a} - \mathbf{b} \mid \mathbf{a} \in A, \mathbf{b} \in B\}$$
- Zusammenhang zwischen *Minkowski-Summe* und *-Differenz*:
 
$$A \ominus B = A \oplus (-B)$$
- Anwendungen: Computergraphik, Bildverarbeitung, Lineare Optimierung, Roboter-Pfadplanung, ...

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 51

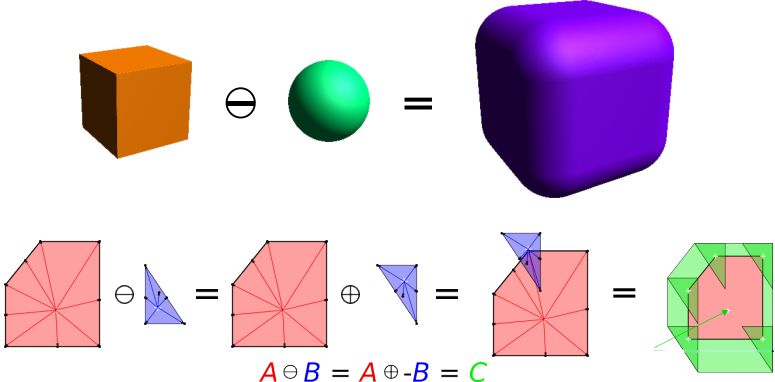
- Intuitive "Berechnung" der Minkowski-Summe:



- Achtung: das gelbe Polygon zeigt die Minkowski-Summe **modulo(!)** eventueller Translationen!

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 52

▪ Weitere Beispiele:



$A \ominus B = A \oplus -B = C$

▪ Dieses Objekt nennt man auch das *Configuration Space Obstacle (CSO)*

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 53

▪ Eigenschaften

▪ *Minkowski-Summen* sind:

- Kommutativ:  $A \oplus B = B \oplus A$
- Assoziativ:  $A \oplus (B \oplus C) = (A \oplus B) \oplus C$
- Distributiv bzgl. Vereinigung:  $A \oplus (B \cup C) = (A \cup B) \oplus (A \cup C)$
- Invariant (in gewissem Sinne) gegenüber Translation:  $T(A) \oplus B = T(A \oplus B)$

▪ *Minkowski-Differenz*:


$$T(A) \ominus T(B) = A \ominus B$$

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 54

# Exact Minkowski Sums of Convex Polyhedra

Efi Fogel and Dan Halperin

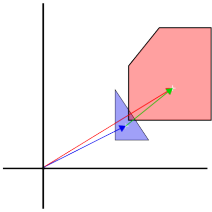
School of Computer Science  
Tel Aviv University



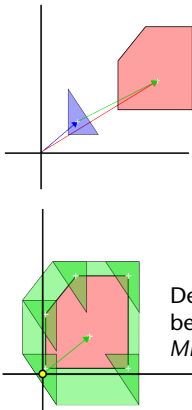
G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 55

## Schnitttest für zwei konvexe BVs

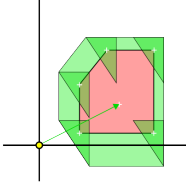
- A und B schneiden sich  $\Leftrightarrow 0 \in A \ominus B$
- Beispiel:



$A \ominus B = A \ominus B = C$



$A \ominus B = A \ominus B = C$



$A \ominus B = A \ominus B = C$

Der Koordinatenursprung befindet sich in der Minkowski-Differenz C

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 57

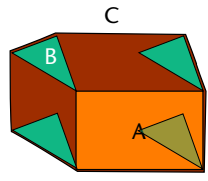
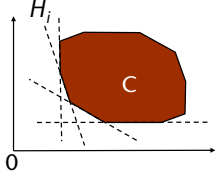
## Schnitttest für Oriented Bounding Boxes (OBB)

- Lemma "*Separating Axis Test*" (SAT):  
Seien A, B zwei konvexe Polytope (Polyeder).  
Wenn es eine separierende Ebene gibt,  
dann auch eine, die parallel zu einer Seite von A oder B ist,  
oder parallel zu mindestens einer Kante von A und einer von B.  
[Gottschalk, Lin, Manocha; 1996]
- Abwandlung des "*separating plane*" Lemmas  
("separating axis" Lemma):  
Zwei konvexe Polyeder überlappen sich nicht  $\Leftrightarrow$   
es gibt eine Gerade, so daß die Projektion der beiden  
Objekte auf dieser Geraden sich nicht überlappen.  
Diese Achse heißt "*separierende Achse*".

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 59

## Beweis des SAT-Lemmas

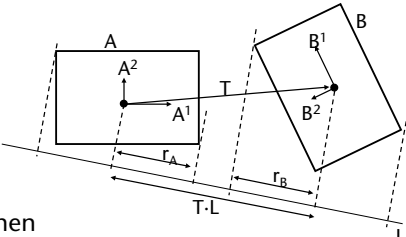
1. Annahme: A und B sind disjunkt
2. Betrachte Minkowski-Summe
3. Alle Faces von C sind entweder parallel zu einem Face von A, oder einem Face von B, oder parallel zu einer Kante von A *und* einer Kante von B
4. C ist konvex
5.  $C = \bigcap_{i=1}^m H_i$
6.  $A \cap B = \emptyset \Leftrightarrow (0, 0, 0) \notin C$
7.  $\exists i : 0 \notin H_i$  (0 liegt außerhalb eines  $H_i$ )
8. Es gibt eine separierende Ebene für A und B, die parallel zu diesem  $H_i$  ist.

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 60

### Der SAT für OBBs

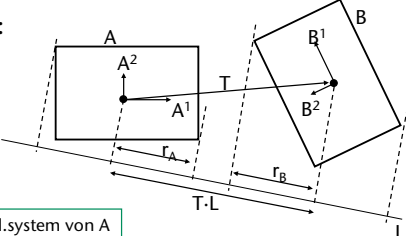
- OBdA: rechne im Koord.system von Box A
- Box A definiert durch:  $C, a^1A^1, a^2A^2, a^3A^3$
- Position von B relativ zu A ist definiert durch R & T
- Im Koord.system von A:  $B^i$  sind Spalten von R
- Gemäß Lemma müssen wir **nur einige spezielle** Ebenen betrachten, um die Separierung festzustellen
- A,B überlappen, wenn  $|T \cdot L| < r_A + r_B$  für jede dieser Ebenen
  - $L$  = Normale der Ebene
- Anzahl solcher "spezieller" Achsen bei Boxes = 15



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 61

- Bsp.:  $L = A^1 \times B^2$
- Zu berechnen:  $r_A = \sum_i a_i |A^i \cdot L|$  (und analog  $r_B$ )
- Bsp. 2-ter Term der Summe:
 
$$\begin{aligned} & a_2 A^2 \cdot (A^1 \times B^2) \\ &= a_2 B^2 \cdot (A^2 \times A^1) \\ &= a_2 B^2 \cdot A^3 \\ &= a_2 R_{32} \end{aligned}$$

Wir rechnen in Koord.system von A  
 $\rightarrow A^3$  ist 3-ter Einheitsvektor, und  
 $B^2$  ist 2-te Spalte von R

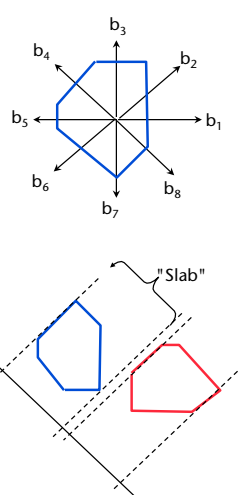


- Für jede der 15 Achsen hat man einen Test der Form
 
$$|T \cdot L| < a_2 |R_{32}| + a_3 |R_{22}| + b_1 |R_{13}| + b_3 |R_{11}|$$

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 62

### Schnitttest für Discretely Oriented Polytopes ( $k$ -DOPs)

- Definition:**  
 Wähle  $k$  Vektoren  $\mathbf{b}_i \in \mathbb{R}^3$  fest,  $k$  gerade,  
 mit  $\mathbf{b}_i$  antiparallel zu  $\mathbf{b}_{i+k/2}$ .  
 $k$ -DOPs sind als Volumen beschrieben durch
 
$$D = \bigcap_{i=1..k} H_i \quad , \quad H_i : \mathbf{b}_i \cdot \mathbf{x} - d_i \leq 0$$
- Beschreibung eines  $k$ -DOP:**  $D = (d_1 \dots d_k) \in \mathbb{R}^k$
- Überlappungstest:**  
 $D^1 \cap D^2 = \emptyset \Leftrightarrow$   
 $\forall i = 1, \dots, \frac{k}{2} : [d_i^1, d_{i+\frac{k}{2}}^1] \cap [d_i^2, d_{i+\frac{k}{2}}^2] = \emptyset$   
 →  $k/2$  Intervall-Tests

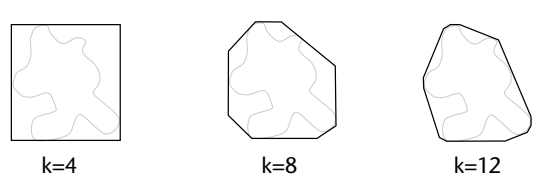


The diagram on the right shows a central point with eight vectors  $\mathbf{b}_1$  through  $\mathbf{b}_8$  pointing outwards.  $\mathbf{b}_1$  and  $\mathbf{b}_5$  are horizontal,  $\mathbf{b}_2$  and  $\mathbf{b}_6$  are at 45 degrees,  $\mathbf{b}_3$  and  $\mathbf{b}_7$  are vertical, and  $\mathbf{b}_4$  and  $\mathbf{b}_8$  are at 135 degrees. A blue polygon is formed by the intersection of half-planes defined by these vectors. Below it, two such polygons (one blue, one red) are shown overlapping. A dashed line between them is labeled 'Slab'.

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 63

### Eigenschaften

- AABBs sind spezielle DOPs
- Überlappungstest  $\in O(k)$ ,  $k$  = Anzahl Orientierungen
- Beliebig genaue Approximation der konvexen Hülle



The diagram shows three stages of approximating a complex, irregular convex shape. The first is a square with  $k=4$  sides. The second is an octagon with  $k=8$  sides. The third is a dodecagon with  $k=12$  sides. As  $k$  increases, the approximation becomes more accurate.

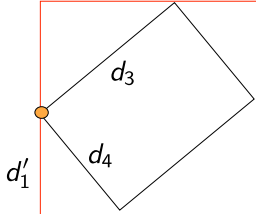
G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 64



### Overlap test of DOPs

- Algorithmus für "schiefe" DOPs:
  - Objektbewegung: Rotation R & Translation T
  - Neuer DOP nach affiner Transformation des Objektes:

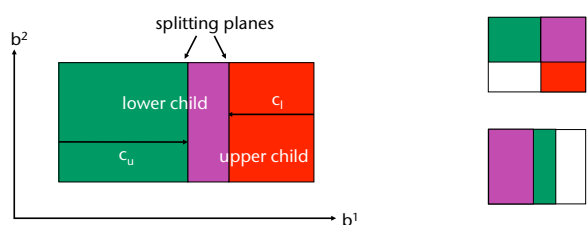
$$d'_i = \mathbf{B}_i \begin{pmatrix} \mathbf{b}_{j'_1} \\ \mathbf{b}_{j'_1} \\ \mathbf{b}_{j'_1} \end{pmatrix}^{-1} \begin{pmatrix} d_{j'_1} \\ d_{j'_1} \\ d_{j'_1} \end{pmatrix} + \mathbf{B}_i \mathbf{T},$$

$$\mathbf{b}_j = \mathbf{B}_i \mathbf{R}^{-1}$$


- Korrespondenz  $j'_i$  identisch für alle DOPs einer Hierarchie
- Aufwand:  $O(k)$ , früher  $O(k^2)$

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 65

### Restricted Boustrees (Variante von k-d- bzw. AABB-Trees)

- Kombination von k-d tree und AABB:
 
- Speicher: 1 Float, 1 Achsen-ID, 1 Pointer (= 9 Bytes)
- Weitere Namen:
  - BIH (Bounding Interval Hierarchy)
  - SKD-Tree (spatial kd-Tree)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 66

- Overlap Tests durch Re-Alignment:  
12 FLOPs (mit kleinem Trick: 8.5)
- Zum Vergleich:
  - SAT: 82 FLOPs
  - SAT lite: 24 FLOPs
  - Sphere test: 29 FLOPs
- Mehr dazu in  
<http://zach.in.tu-clausthal.de/papers/vrst02.html>

The diagram illustrates the re-alignment process for overlap testing. It shows two rectangles, X (green) and Y (yellow), with their centers marked as  $c^X$  and  $c^Y$ . A dashed line labeled 's' indicates the direction of re-alignment. The resulting configuration shows the rectangles shifted to minimize overlap, with a yellow shaded area representing the remaining overlap.

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 67

### Performance

- Bsp. *Boxtree*:

Door lock (BMW) Car (courtesy VW)

**car**

time / millisecc

# pgons / 1000

Restr. Bboxtree (red line)

DOP tree (green line)

**lock**

time / millisecc

# pgons / 1000

Restr. Bboxtree (red line)

DOP tree (green line)

The performance graphs show time in milliseconds versus the number of polygons in thousands. For the 'car' model, the Restr. Bboxtree (red) maintains a low, stable time around 0.2-0.3 ms, while the DOP tree (green) starts at 0.4 ms and increases to nearly 2.0 ms at 60,000 polygons. For the 'lock' model, the Restr. Bboxtree (red) peaks at 0.5 ms at 50,000 polygons and then drops to 0.1 ms, while the DOP tree (green) peaks at 0.5 ms and drops to 0.1 ms.

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 68

## Konstruktion von BV-Hierarchien

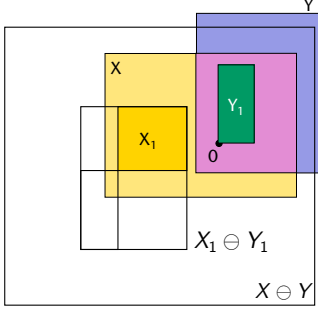
- Hierarchie schlecht → Kollisionserkennung dauert lange.
- Algorithmus: *top-down*
  1. Berechne BV um gegebene Polygon-Menge
  2. Splitte Polygon-Menge
- Split-Kriterium?
- Erwartete Traversierungskosten:
 
$$C(X, Y) = 4 + \sum_{i,j=1,2} P(X_i, Y_j) C(X_i, Y_j)$$

$$\approx 4(1 + P(X_1, Y_1) + \dots + P(X_2, Y_2))$$

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 69

- Eine Abschätzung von  $P(X_i, Y_j)$
- Hilfsmittel dabei:  
die Minkowski-Summe
- Erinnerung:
 
$$X_i \cap Y_j = \emptyset \Leftrightarrow 0 \notin X_i \ominus Y_j$$
- Die Wahrscheinlichkeit ist somit
 
$$P(X_i, Y_j) = \frac{|\text{günstige Fälle}|}{|\text{mögliche Fälle}|} = \frac{\text{vol}(X_i \ominus Y_j)}{\text{vol}(X \ominus Y)} = \frac{\text{vol}(X_i \oplus Y_j)}{\text{vol}(X \oplus Y)}$$

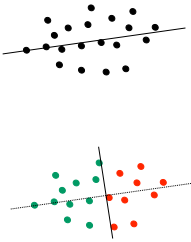
$$\approx \frac{\text{vol}(X_i) + \text{vol}(Y_j)}{\text{vol}(X) + \text{vol}(Y)}$$
- Fazit: Minimiere Summe der Volumen der Kind-BVs.



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 70

## Algorithmus

1. Orientierung für "gute" *Splitting-Ebene* aus PCA
2. Sortiere Polygone entlang dieser Achse
3. Suche Minimum mittels Plane-Sweep gemäß Volumen-Kriterium



- Komplexität:

$$T(n) = O(n \log n) + T(\alpha n) + T((1 - \alpha)n) \in O(n \log^2 n)$$

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 71

## Morphing Objects / BVs

- *Morphing*:  
Gegeben  $n$  Objekte  $O^i$ , Vertices  $v_j^i$ , Gewichte  $w_i$ ,  $\sum_i w_i = 1$   
Vertices des gemorphten Objektes  $\bar{v}_j = \sum_{i=1}^n w_i v_j^i$ ,  $j = 1, \dots, N$
- *Morphing* von DOPs:  
Gegeben  $n$  DOPs  $D^i = (S_1^i, \dots, S_{\frac{k}{2}}^i)$ ,  $S_j^i = (s_j, e_j)$   
gemorphter DOP  $\bar{D} = (\bar{S}_1, \dots, \bar{S}_{\frac{k}{2}})$ ,  $\bar{S}_j = (\sum w_i s_j^i, \sum w_i e_j^i)$
- Behauptung:  $v_j^i \in D^i \Rightarrow \bar{v}_j \in \bar{D}$
- Bew.:  $\bar{s}_j = \sum_{i=1}^n w_i s_j^i \leq \sum_{i=1}^n w_i (v^i \cdot b^j) \leq \sum_{i=1}^n w_i e_j^i = \bar{e}_j$
- Analog für Kugeln (klappt nicht für OBBs)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 73

■ Datenstruktur:  
 BV-Hierarchie mit vielen BVs pro Knoten,  
 müssen dieselben Polygone einschließen!

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 74

■ Exakte interpolierte BVs

- Betrachte im folgenden nur eine Seite des BVs (DOPs, AABBs), gegeben durch  $\mathbf{n}$
- Gesucht: Vertex wo folgendes Max angenommen wird

$$\max\{\mathbf{n}\bar{\mathbf{v}}_i \mid i = 1, \dots, N\}$$

- Sei  $i^*$  dieser Vertex, ist für einen ganzen Bereich  $\Omega_{i^*}$  "maximal"

$$\Omega_{i^*} = \bigcap_{j=i} \{\mathbf{w} \mid (\mathbf{v}_j(\mathbf{w}) - \mathbf{v}_j(\mathbf{w})) \mathbf{n} > 0\} \subset \mathbb{R}^d$$

- $\Omega_{i^*}$  ist konvexer Kegel (Apex in 0)
- Wird i.a. definiert durch kleine Anzahl Seitenflächen

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Kollisionserkennung 75

- Zelle  $\Omega_{i^*}$  hat kleine Anzahl Nachbarzellen mit je 1 gemeinsamen Begrenzungsfläche  $\rightarrow$  Adjazenzgraph

$$A = \{(i, j) \mid \Omega_i, \Omega_j \text{ haben gemeinsame Begr. seite}\}$$

- Damit wird Zelle  $\Omega_{i^*}$  zu

$$\Omega_{i^*} = \bigcap_{(i^*, j) \in A} \{\mathbf{w} \mid (\mathbf{v}_i(\mathbf{w}) - \mathbf{v}_j(\mathbf{w})) \mathbf{n} > 0\}$$

- In Animationssequenzen ist  $\mathbf{w} = \mathbf{w}_t$
- Nutze zur Berechnung der Begrenzungsseite zu n zeitliche Kohärenz:

$$N = \{j \mid (i^*, j) \in A\}$$

foreach  $j \in N$

$$\text{if } (\mathbf{v}_i(\mathbf{w}) - \mathbf{v}_j(\mathbf{w})) \mathbf{n} < 0 \text{ then}$$

$$i^* = j$$

G. Zachmann Virtuelle Realität und Simulation - WS 09/10
Kollisionserkennung 76