

Computergraphik II

Winter 2011/2012

11 Kollisionserkennung und -reaktion

Versionsdatum: 13. Dezember 2011



Prof. Dr. Andreas Kolb
Computer Graphics & Multimedia Systems

-Folie 11-1-

Computergraphik II

11 Kollisionserkennung und -reaktion ...



Klassifikation von Systemen aus Bewegungssicht: Man unterscheidet

Statische Systeme: Geometrien & Beobachter sind in Lage & Größe fix

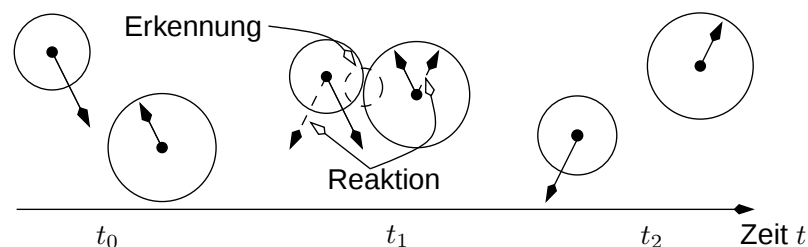
Deterministische Systeme: Lage/Größe variabel, aber a-priori festgelegt (Animation)

Dynamische Systeme: Freie, interaktive Objekte- & Beobachterbewegung

Kollision in deterministischen und dynamischen Systemen muss real wirken

Kollisionsverarbeitung: 1. *Kollisionserkennung:* Ermittle Koll.-Information

2. *Kollisionsreaktion* der (dynamischen) Objekte auf Kollisionsereignis



Kollisionsinformation: 1. Boolesche Aussage: Einfache Kollisionsreaktion



2. Kollisionspunkte und -normalen bei komplexer Kollisionsreaktion

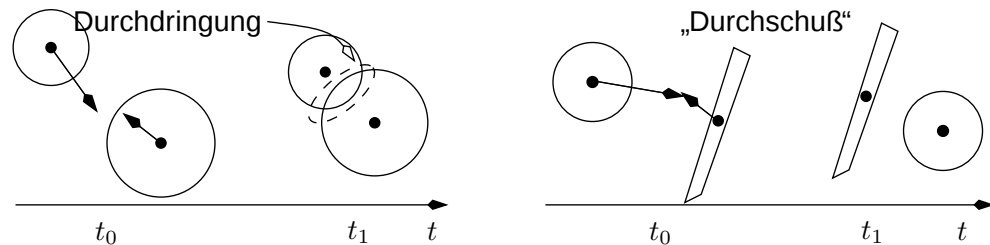
Prof. Dr. Andreas Kolb
Computer Graphics & Multimedia Systems

-Folie 11-2-

Computergraphik II

Grundsätzliche Konzepte zur Kollisionserkennung

3D, zeitdiskret: Test zu festen Zeitpunkten; exakter Kollisionszeitpunkt wird nicht erkannt!



4D: Echte Überprüfung in vier Dimensionen $\{x, y, z, t\} \Rightarrow$ komplex!

Brut Force: Komplexität bei n Objekten: $n(n-1)/2 \Rightarrow \mathcal{O}(n^2)$ Kollisionstests

Beschleunigung: Nutzung von Kohärenzen (räumlich, zeitlich) oder *Bounding Volumes*

Abstrakter Algorithmus zur Kollisionserkennung:

```
forall moving objects m do
  forall objects o != m do
```



```
    if isClose(o, m) do
      checkCollision(o, m);
```

```
// schnelle, konserv. Abschätzung
// (aufwendige) Detailprüfung
```

Prof. Dr. Andreas Kolb
Computer Graphics & Multimedia Systems

-Folie 11-3-

Computergraphik II

Kollisionsgeometrie

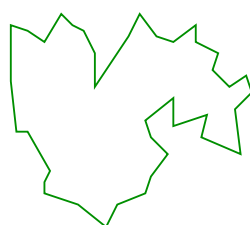
Kollisionsverarbeitung abhängig von Kollisionsinformation und Genauigkeit

1. auf Basis der *exakten Objektgeometrien*

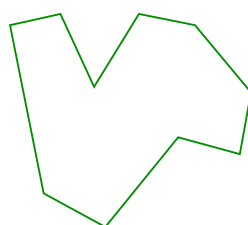
- + Exaktheit, selten Ausnutzung von spez. Geometrie, z.B. Kugel-Kollision ($\rightarrow$ schnelle Berechnung)
- Kollisionserkennung für alle Geometrie-Kombinationen, häufig sehr aufwendige Berechnung erforderlich

2. auf Basis vereinfachter, umhüllender Geometrien (*Bounding Volumes*)

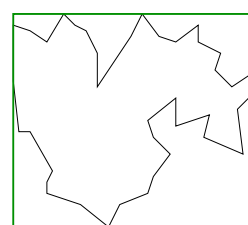
- + Einheitliches Modell der Kollisionserkennung
- Ungenauigkeit



Exakte Geometrie



Vereinfachte Geometrie



Bounding Box



Prof. Dr. Andreas Kolb
Computer Graphics & Multimedia Systems

-Folie 11-4-

Computergraphik II

11.1 Einfache Kollisionsreaktionen



Aufgabe: Anpassung der Objekt-Geschwindigkeit, so dass keine weitere Durchdringung

Randbedingungen: ○ es kollidieren genau **zwei** Objekte $\mathcal{O}_1, \mathcal{O}_2$
○ von beiden Objekte ist der *Geschwindigkeitsvektor* $\vec{v}_i$ bekannt

Ansatz Bewegungs-Unterbindung: Kollisionsinformation: Boolesch

Kollisionsreaktion: ○ behalte letzte Nichtkollisions-Position bei
○ setze aktuelle Geschwindigkeiten auf null: $\vec{v}_1 = \vec{v}_2 = 0$

Ansatz Entlanggleiten:

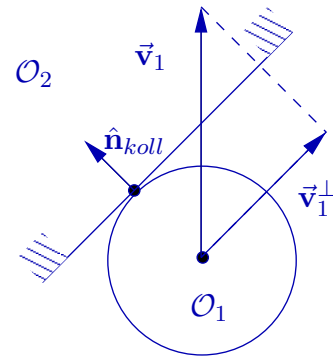
Kollisionsinformation: Normale $\hat{n}_{koll}$

Kollisionsreaktion: Entfernen des Geschwindigkeitsanteils $\vec{v}^{\parallel}$ parallel zu $\hat{n}_{koll}$:

$$\vec{v}_i = \vec{v}^{\parallel} + \vec{v}^{\perp}, \quad (\vec{v}^{\perp} \cdot \hat{n}_{koll}) = 0$$

$$\vec{v}^{\parallel} = (\vec{v} \cdot \hat{n}_{koll}) \cdot \hat{n}_{koll}$$

$$\Rightarrow \vec{v}_i^{\perp} = \vec{v}_i - (\vec{v}_i \cdot \hat{n}_{koll}) \hat{n}_{koll}$$



11.1 Einfache Kollisionsreaktionen ...



Elastischer Stoß („Reflexion“)

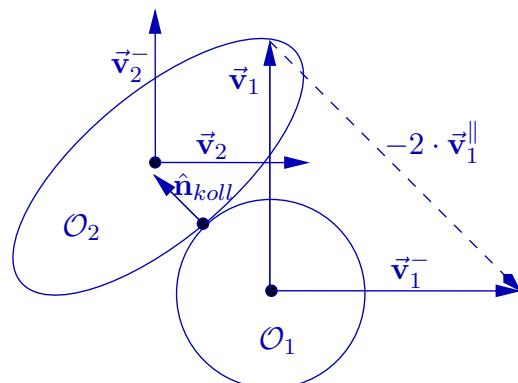
Vereinfachung: Vernachlässigung

- der Objektmasse
- der Umwandlung von linearer Bewegung in Rotationsbewegung

Idee: Reflexion des Geschwindigkeitsvektors an Kollisionsnormale:

$$\begin{aligned} \vec{v}_i^- &= \vec{v}_i - 2 \cdot \vec{v}_i^{\parallel} \\ &= \vec{v}_i - 2(\vec{v}_i \cdot \hat{n}_{koll}) \hat{n}_{koll} \end{aligned}$$

Berücksichtigung von Masse und Rotation (siehe VR)



Beachte: Alle Ansätze sind physikalisch nicht korrekt, da keine Energieerhaltung



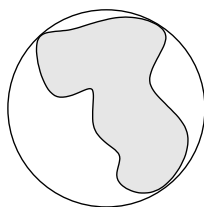
Definition: V ist ein *Bounding Volume* einer Geometrie G falls

1. V ist konvex
2. V „umfaßt“ G , d.h. $V \supseteq G$

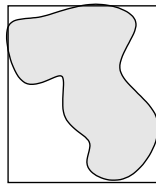
Zielsetzung zur Wahl von Bounding Volumen

1. Effiziente Kollisionsberechnung
2. Optimale Objektanpassung, d.h. $V \setminus G$ soll minimal sein
3. Effiziente Ermittlung für gegebene Objekte

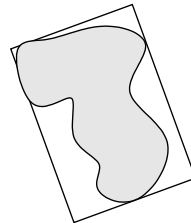
Beispiele:



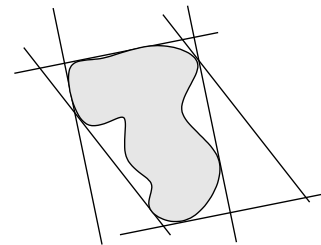
Sphäre



Axis-Aligned
Bounding Box



Object Aligned
Bounding Box



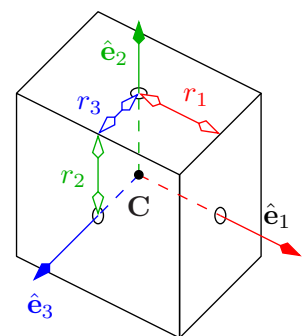
Discrete Orientation
Polytop (k -dop)



11.2.1 Bounding Boxes

Gegeben: Quader Q^1, Q^2 mit Mittelpunkt C^i , orthonormalen Achsen $\hat{e}_j^i$ und Ausdehnungen r_j^i , $j = 1, 2, 3$:

$$Q^i = \left\{ C^i + \sum_{j=1}^3 \alpha_j \hat{e}_j^i : \alpha_j \in [-r_j^i, r_j^i] \right\} \quad i = 1, 2$$



Gesucht: Boolesche Aussage, ob die Quader sich durchdringen

Axis Aligned Box (AAB): $\hat{e}_j$ sind Weltkoordinatenachsen
Achtung: AAB's ändern sich bei Objekt-Rotation!

Object Aligned Box (OAB): $\hat{e}_j$ sind am Objekt ausgerichtet

Separierende Ebene: Konvexe Geometrien durchdringen sich nicht, wenn es eine *separierende Ebene* zwischen ihnen gibt (Skizze!)

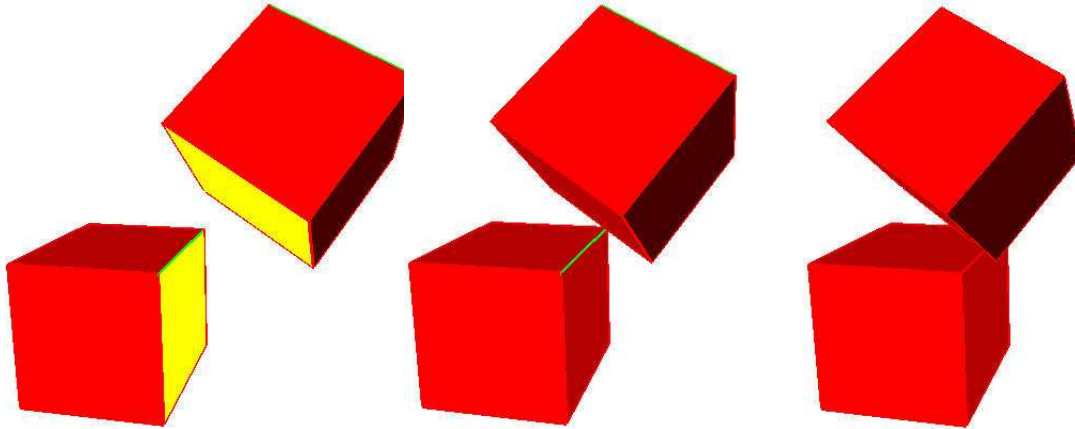
Speziell: Für OABs genügt es folgende Flächen auf Separation zu testen:

1. Seitenfläche einer Box
2. Ebene parallel zu zwei Kantenvektoren



Separierende Ebenen für OABs

- Demo-Programm:



Kantenpaar & Seiten- fläche separieren Kantenpaar separiert Durchdringung

- Es genügt **eine** separierende Ebene zu finden, um eine Kollision auszuschließen.



11.2.1 Bounding Boxes ...

Separierende Ebenen für AABs

Beobachtung: Kantenvektorpaare erzeugen keine neuen Ebenen $\implies$ teste nur Seitenflächen

Kollisions-Erkennung: Für die Seitenflächen ergibt sich:

$$\text{wobei } (x_{max}^1 < x_{min}^2) \vee (x_{max}^2 < x_{min}^1) \dots$$
$$x_{min}^i = c_x^i - r_1^i, \quad x_{max}^i = c_x^i + r_1^i \quad \text{mit } \mathbf{C}^i = (c_x^i, c_y^i, c_z^i)$$

Bemerkung:

- für zwei AABs sind maximal sechs Bedingungen zu prüfen (zwei pro Koordinatenachse)
- für OABs kommen zusätzlich maximal neun für die 3×3 Kantenvektorpaare hinzu.
- Der **genaue Kollisionszeitpunkt** kann nicht ermittelt werden $\implies$ arbeite mit ε -Umgebungen



11.2.1 Bounding Boxes ...



Separierende Ebenen für OABs

Idee: Projiziere Quader auf die Normale $\hat{n}$ der potentiell separierenden Fläche

$$\text{Ebene separiert} \Leftrightarrow \Delta > p^1 + p^2$$

Projektion eines Quaders: Für $\vec{d} = C^1 - C^2$ gilt:

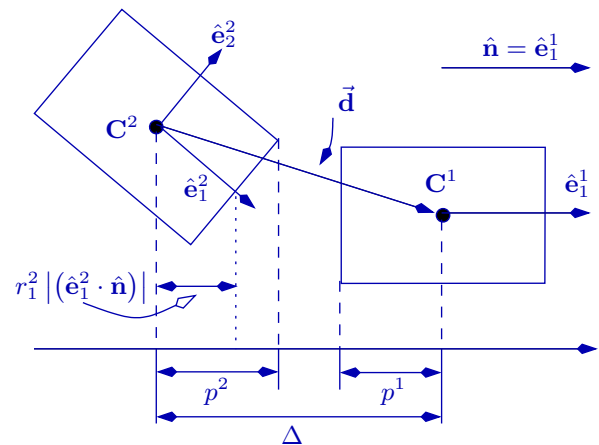
$$\Delta = \left| (\vec{d} \cdot \hat{n}) \right|, \quad \text{hier } \hat{n} = \hat{e}_1^1$$

Fall 1: $\hat{n}$ ist Seitenflächennormale:

$$\text{z.B. } \hat{n} = \hat{e}_1^1 \Rightarrow p^1 = r_1^1$$

Fall 2: $\hat{n}$ liegt beliebig zum Quader:

$$p^i = \sum_{j=1}^3 r_j^i |(\hat{e}_j^i \cdot \hat{n})|$$



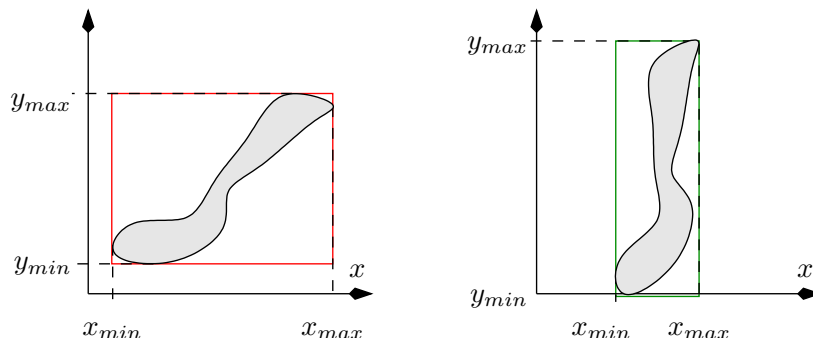
11.2.1 Bounding Boxes ...



Bewertung von AAB

Kollisionsberechnung: Sehr effizient über 6 Vergleiche

Objektanpassung: Schlecht, da AAB von der Lage der Objekte im Raum abhängen



Ermittlung für gegebene Polygonsmodelle mit Vertices $\{V^i\}_{i=1}^n$ sehr einfach:

$$x_{min} = \min_{i=1, \dots, n} \{v_x^i\}, \quad x_{max} = \max_{i=1, \dots, n} \{v_x^i\}, \quad \dots, \quad z_{max} = \max_{i=1, \dots, n} \{v_z^i\}$$



11.2.1 Bounding Boxes ...



Bewertung von OAB

Kollisionsberechnung: Max. 15 effiziente Tests auf separierende Ebenen

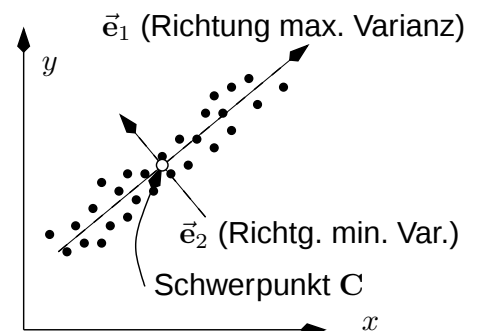
Objektanpassung: Gut, da OAB den Objekten lokal angepasst werden

Ermittlung für gegebene Polygonmodelle aufwendiger

Idee der Ermittlung: Für Polygonmodell mit Vertices $\{\mathbf{V}_i\}_{i=1}^n$:

1. Bestimme Schwerpunkt $\mathbf{C} = \frac{1}{n} \sum_{i=1}^n \mathbf{V}_i$
(besser: Schwerpunkt der konvexen Hülle)
2. Bestimme Richtungen der *max.* bzw. *min. Varianz* der Punkte $\rightarrow$ Achsen $\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2$
(besser: nur Punkte der konvexen Hülle)
3. Wähle max. Ausdehnung z.B. für Achse $\hat{\mathbf{e}}_j$:

$$r_j := \max_i |(\hat{\mathbf{e}}_j \cdot (\mathbf{V}_i - \mathbf{C}))|$$



11.2.2 Bounding Volumen Hierarchien

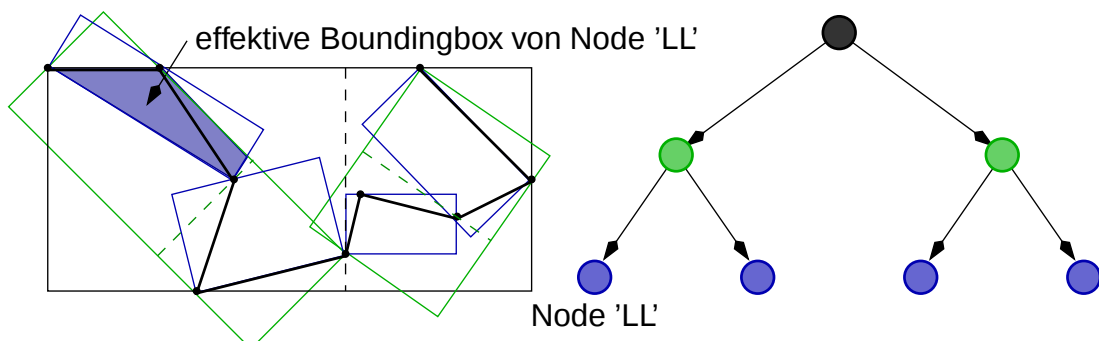


Ziel: Bessere Anpassung an Objekt durch mehrere, kleine Bounding Volumen
 $\Rightarrow$ Hierarchie von Bounding Volumen

Einschränkung: Balance zwischen Kosten für

- Kollisionstest für Geometrien in einem Volumen
- Abarbeitung/Traversierung der Hierarchie inkl. Kollisions-Test für Bounding Volumen

Beispiel OAB Hierarchie: Sukzessive Unterteilung in aktuell max. Ausdehnungsrichtung



Beachte: Effektiv wird *Schnittmenge* der Volumen im Pfad betrachtet

11.2.2 Bounding Volumen Hierarchien ...



Bounding Boxen und mehrgliedrige Modelle

Ziel: Koll.-Erkennung für mehrgliedrige Modelle inkl. zusätzl. Geometrie

Ansatz: Nutze AAB's in *lokalen Koord.*

Beachte: 1. Kollisionstest immer in *einem* Koordinatensystem (hier: VC)

2. Translation & Rotation erhalten $\hat{e}_1 \perp \hat{e}_2$, Skalierung i.a. nicht!

Transformation der lokalen AAB in VC mit Curr. Transformationmatrix M

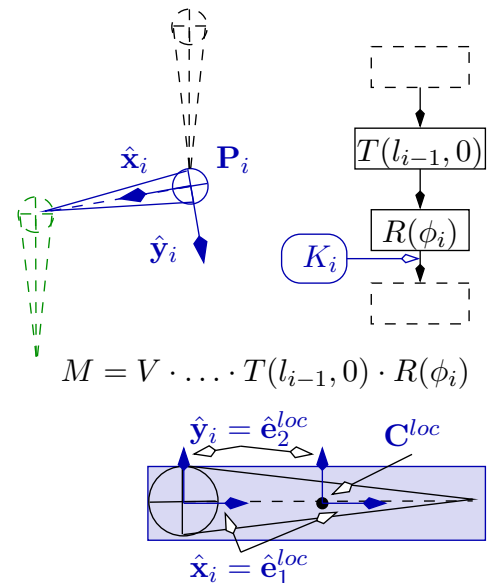
Lokale Koord.: $C^{loc}, \hat{e}_1^{loc}, \hat{e}_2^{loc}, \hat{e}_3^{loc}$

View-Koordinaten falls M ohne Skalierung:

$$C^{VC} = M \cdot C^{loc}$$

$$\hat{e}_i^{VC} = (M^T)^{-1} \cdot \hat{e}_i^{loc}, i = 1, 2, 3$$

2D-Beispiel:



11.3 Berechnung von Kollisionspunkten



Beachte: 1. zur Ermittlung von Kollisionspunkten und -normalen eignen sich *einfache implizite Geometrien* i.a. besser als Polygone

2. Kollisionspunktberechnung beruht auf *Abstandberechnung*

Beispiel: Kollisionspunkte zweier Kugeln

Gegeben: Kugeln S_i mit Basisgeometrie Punkt M_i und Radius s_i :

$$S_i = \{P : \|P - M_i\| = s_i\}, i = 1, 2$$

Gesucht: Kollisionserkennung, Kollisionspunkt(e) und Kollisionsnormale

Lösung: 1. Abstand der beiden Kugeln:

$$d(S_1, S_2) = \|M_1 - M_2\| - (s_1 + s_2)$$

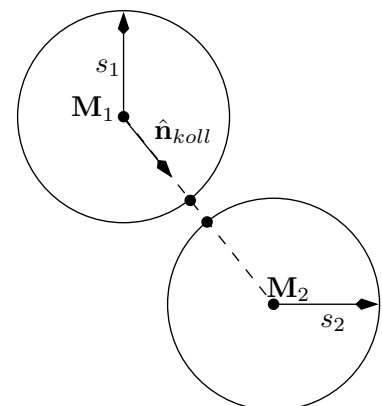
2. Kollisionserkennung: $d(S_1, S_2) \stackrel{?}{\approx} 0$

3. Kollisionsnormale:

$$\hat{n}_{koll} = (M_2 - M_1) / \|M_2 - M_1\|$$

4. Kollisionspunkt:

$$K = M_1 + s_1 \hat{n}_{koll} \approx M_2 - s_2 \hat{n}_{koll}$$



11.3.1 Abstand zweier linearer Komponenten



Aufgabe: Bestimme Abstand zweier *linearer Komponenten*, d.h. Gerade, Strecke, Strahl:

$$L_i : \mathbf{L}_i(\alpha_i) = \mathbf{V}_i + \alpha_i \vec{\mathbf{l}}_i, \quad \alpha_i \in I_i \subset \mathbb{R}, \quad I_i \in \{\mathbb{R}, \mathbb{R}_+, [0, 1]\}$$

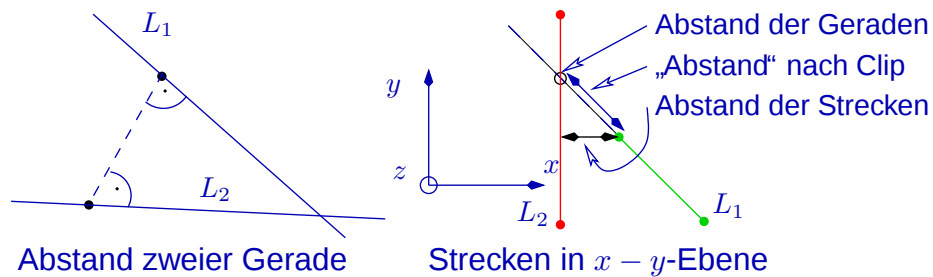
Gegeben: Lineare Komponenten L_1, L_2

Gesucht: Punkte mit kürzestem Abstand:

$$(\alpha_1^P, \alpha_2^P) \in I_1 \times I_2 \text{ mit } \|\mathbf{L}_1(\alpha_1^P) - \mathbf{L}_2(\alpha_2^P)\| = \min_{\alpha_1 \in I_1, \alpha_2 \in I_2} \{\|\mathbf{L}_1(\alpha_1) - \mathbf{L}_2(\alpha_2)\|\}$$

Beobachtung: $\bigcirc$ L_1, L_2 Geraden ($I_1 = I_2 = \mathbb{R}$) $\Rightarrow (\mathbf{L}_1(\alpha_1^P) - \mathbf{L}_2(\alpha_2^P)) \perp L_i$

$\bigcirc$ L_1 oder L_2 keine Gerade: Einfaches Clipping liefert i.a. falsches Ergebnis



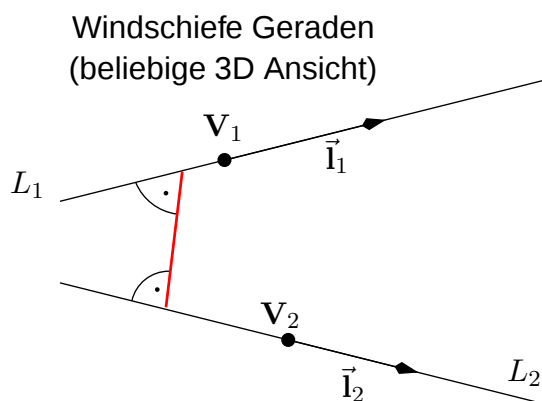
11.3.1 Abstand zweier linearer Komponenten ...



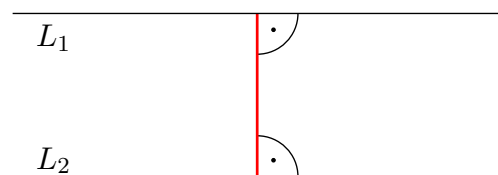
Ansichten der 3D Situation

Betrachte Punkte minimalen Abstands für zwei Geraden

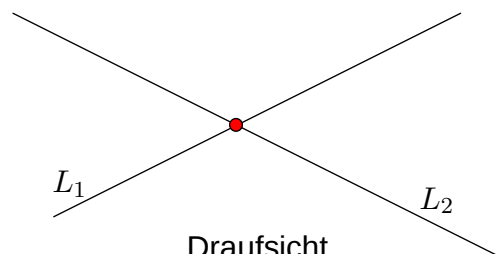
$$L_i : \mathbf{L}_i(\alpha) = \mathbf{V}_i + \alpha \vec{\mathbf{l}}_i, \quad \alpha_i \in \mathbb{R}$$



Windschiefe Geraden
(beliebige 3D Ansicht)



Seitenansicht



Draufsicht



11.3.1 Abstand zweier linearer Komponenten ...



Abstand zweier Geraden

Gegeben: Lineare Komponenten L_1, L_2 mit $I_1 = I_2 = \mathbb{R}$

Lösung: Die beiden „Senkrecht“-Bedingungen liefern für $i = 1, 2$:

$$\begin{aligned} \left((\mathbf{L}_1(\alpha_1^P) - \mathbf{L}_2(\alpha_2^P)) \cdot \vec{\mathbf{l}}_i \right) &= \left((\mathbf{V}_1 + \alpha_1^P \vec{\mathbf{l}}_1 - \mathbf{V}_2 - \alpha_2^P \vec{\mathbf{l}}_2) \cdot \vec{\mathbf{l}}_i \right) = 0, \\ \Leftrightarrow \alpha_1^P (\vec{\mathbf{l}}_1 \cdot \vec{\mathbf{l}}_i) - \alpha_2^P (\vec{\mathbf{l}}_2 \cdot \vec{\mathbf{l}}_i) &= ((\mathbf{V}_2 - \mathbf{V}_1) \cdot \vec{\mathbf{l}}_i). \end{aligned}$$

Dies ergibt ein *lineares Gleichungssystem* für die α_i^P :

$$\text{Mit } b_{ij} = (\vec{\mathbf{l}}_i \cdot \vec{\mathbf{l}}_j) : \quad \begin{pmatrix} b_{11} & -b_{12} \\ -b_{12} & b_{22} \end{pmatrix} \begin{pmatrix} \alpha_1^P \\ \alpha_2^P \end{pmatrix} = \begin{pmatrix} ((\mathbf{V}_2 - \mathbf{V}_1) \cdot \vec{\mathbf{l}}_1) \\ ((\mathbf{V}_1 - \mathbf{V}_2) \cdot \vec{\mathbf{l}}_2) \end{pmatrix}$$

Die Matrix ist regulär, wenn $\vec{\mathbf{l}}_i$ nicht parallel sind (sonst ein α_i^P frei wählbar)

$$\text{Lösung:} \quad \begin{pmatrix} \alpha_1^P \\ \alpha_2^P \end{pmatrix} = \frac{1}{b_{11}b_{22} - b_{12}^2} \begin{pmatrix} b_{22} & b_{12} \\ b_{12} & b_{11} \end{pmatrix} \begin{pmatrix} ((\mathbf{V}_2 - \mathbf{V}_1) \cdot \vec{\mathbf{l}}_1) \\ ((\mathbf{V}_1 - \mathbf{V}_2) \cdot \vec{\mathbf{l}}_2) \end{pmatrix}$$



11.3.1 Abstand zweier linearer Komponenten ...



Abstand zweier Strecken

Gegeben: Lineare Komponenten L_1, L_2 mit $I_1 = I_2 = [0, 1]$

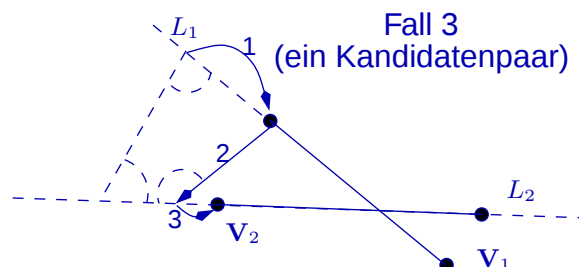
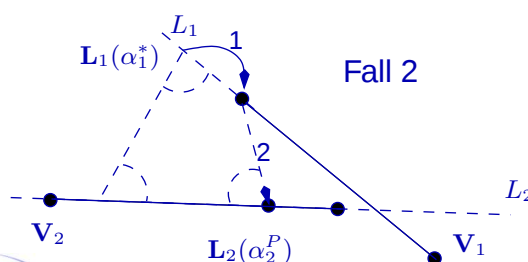
Lösung: Abstandsberechnung der zugehörigen **Geraden** liefert $\alpha_1^*, \alpha_2^* \in \mathbb{R}$

Fall 1: $\alpha_i^* \in [0, 1]$: dann $\alpha_i^P = \alpha_i^*$ und $\mathbf{L}_i(\alpha_i^*)$ sind Punkte min. Abstands.

Fall 2: nur ein α^* außerhalb: sei z.B. $\alpha_1^* \notin [0, 1]$, dann gilt:

- $\alpha_1^P = \text{clip}(\alpha_1^*, [0, 1])$
- α_2^P : Projektion von $\mathbf{L}_1(\alpha_1^P)$ auf L_2 und evtl. anschließend Clip.

Fall 3: beide α_i^* außerhalb: Bestimme zwei Kandidatenpaare entspr. Fall 2; wähle dasjenige mit kürzestem Abstand



11.3.1 Abstand zweier linearer Komponenten ...

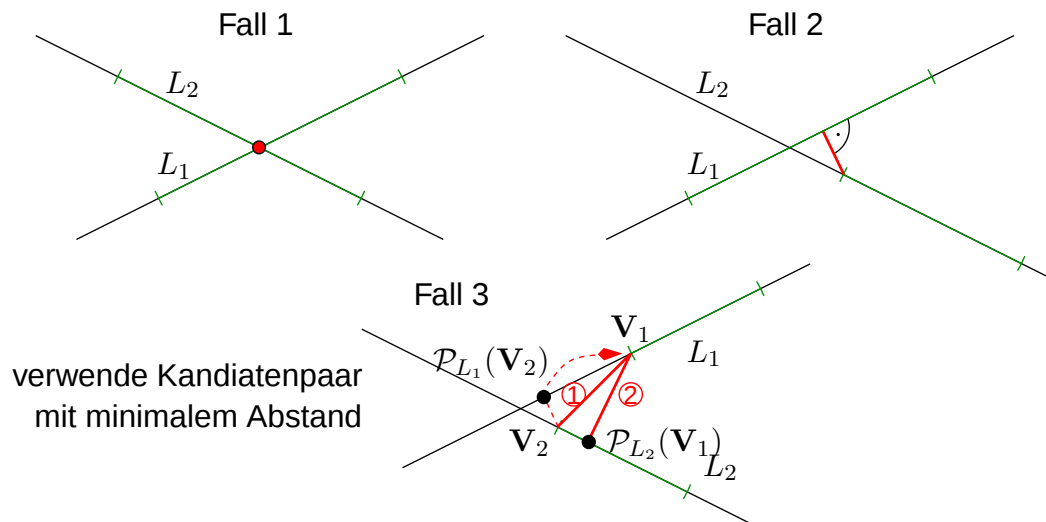


Ansichten der 3D Situation

Betrachte Punkte minimalen Abstands für zwei Strecken

$$L_i : \mathbf{L}_i(\alpha) = \mathbf{V}_i + \alpha_i \vec{\mathbf{l}}_i, \quad \alpha_i \in [0, 1]$$

Situation in der Draufsicht



verwende Kandidatenpaar
mit minimalem Abstand



11.3.1 Abstand zweier linearer Komponenten ...



Beispiel: Abstandsberechnung zweier Strecken

Gegeben: $\mathbf{L}_1(\alpha_1) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \alpha_1 \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{L}_2(\alpha_2) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}, \quad \alpha_i \in [0, 1]$

Lösung: 1. Ermittlung der nächsten Punkte auf den Geraden:

$$b_{11} = (\vec{\mathbf{l}}_1 \cdot \vec{\mathbf{l}}_1) = 4 \quad b_{22} = (\vec{\mathbf{l}}_2 \cdot \vec{\mathbf{l}}_2) = 5 \quad b_{12} = b_{21} = (\vec{\mathbf{l}}_1 \cdot \vec{\mathbf{l}}_2) = 4$$

$$((\mathbf{V}_2 - \mathbf{V}_1) \cdot \vec{\mathbf{l}}_1) = \left(\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} \right) = 2$$

$$((\mathbf{V}_1 - \mathbf{V}_2) \cdot \vec{\mathbf{l}}_2) = \left(\begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} \right) = -3$$

$$\Rightarrow \begin{pmatrix} \alpha_1^* \\ \alpha_2^* \end{pmatrix} = \frac{1}{5 \cdot 4 - 4^2} \begin{pmatrix} 5 & 4 \\ 4 & 4 \end{pmatrix} \begin{pmatrix} 2 \\ -3 \end{pmatrix} = \begin{pmatrix} -1/2 \\ -1 \end{pmatrix}$$

2. Da $\alpha_1^*, \alpha_2^* \notin [0, 1]$, müssen beide Kandidatenpaare erzeugt werden:



11.3.1 Abstand zweier linearer Komponenten ...



Beispiel: Abstandsberechnung zweier Strecken (Forts.)

2.1. Clippe α_1^* : $\alpha_1^P = \text{clip}(\alpha_1^*, [0, 1]) = 0$. Projiziere $\mathbf{L}_1(0) = \mathbf{V}_1$ auf L_2 :

$$\text{Muss gelten: } ((\mathbf{V}_2 + \alpha_2 \vec{\mathbf{l}}_2 - \mathbf{L}_1(0)) \cdot \vec{\mathbf{l}}_2) = 0 \Leftrightarrow \alpha_2 = \frac{((\mathbf{V}_1 - \mathbf{V}_2) \cdot \vec{\mathbf{l}}_2)}{(\vec{\mathbf{l}}_2 \cdot \vec{\mathbf{l}}_2)} = \frac{-3}{5}$$

$$\text{damit: } \alpha_2^P = 0 \text{ und } \|\mathbf{L}_2(0) - \mathbf{L}_1(0)\| = \left\| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\| = \sqrt{3}$$

2.2. Clippe α_2^* : $\alpha_2^P = \text{clip}(\alpha_2^*, [0, 1]) = 0$. Projiziere $\mathbf{L}_2(0) = \mathbf{V}_2$ auf L_1 :

$$\text{Muss gelten: } ((\mathbf{V}_1 + \alpha_1 \vec{\mathbf{l}}_1 - \mathbf{L}_2(0)) \cdot \vec{\mathbf{l}}_1) = 0 \Leftrightarrow \alpha_1 = \frac{((\mathbf{V}_2 - \mathbf{V}_1) \cdot \vec{\mathbf{l}}_1)}{(\vec{\mathbf{l}}_1 \cdot \vec{\mathbf{l}}_1)} = \frac{2}{4}$$

$$\text{damit: } \alpha_1^P = \frac{1}{2} \text{ und } \|\mathbf{L}_2(0) - \mathbf{L}_1(1/2)\| = \left\| \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\| = \sqrt{2}.$$



Also sind $\mathbf{L}_1(1/2)$ und $\mathbf{L}_2(0)$ die nächstliegenden Punkte beider Strecken

11.3.1 Abstand zweier linearer Komponenten ...



Quadratische Abstandsfunktion

Erinnerung: Suche Punkte minimalen Abstandes, also im Falle von Geraden:

$$(\alpha_1^P, \alpha_2^P) \text{ so dass: } \|\mathbf{L}_1(\alpha_1^P) - \mathbf{L}_2(\alpha_2^P)\| = \min_{\alpha_1 \in I_1, \alpha_2 \in I_2} \{\|\mathbf{L}_1(\alpha_1) - \mathbf{L}_2(\alpha_2)\|\}$$

Beachte: Da $(a < b) \Leftrightarrow (a^2 < b^2)$, $\forall a, b \in \mathbb{R}_0^+$ kann (α_1^P, α_2^P) alternativ als Minimum der *quadratischen Abstandsfunktion*

$Q(\alpha_1, \alpha_2) = \|\mathbf{L}_1(\alpha_1) - \mathbf{L}_2(\alpha_2)\|^2$ gesehen werden:

$$\begin{aligned} \|\mathbf{L}_1(\alpha_1^P) - \mathbf{L}_2(\alpha_2^P)\|^2 &= ((\mathbf{L}_1(\alpha_1^P) - \mathbf{L}_2(\alpha_2^P)) \cdot (\mathbf{L}_1(\alpha_1^P) - \mathbf{L}_2(\alpha_2^P))) \\ &= \min_{\alpha_1 \in I_1, \alpha_2 \in I_2} \{\|\mathbf{L}_1(\alpha_1) - \mathbf{L}_2(\alpha_2)\|^2\} = \min_{\alpha_1 \in I_1, \alpha_2 \in I_2} \{Q(\alpha_1, \alpha_2)\} \end{aligned}$$

Eigenschaften von Q :

- Graph von Q über der (α_1, α_2) -Parameterebene ist ein *Paraboloid*
- $Q(\alpha_1, \alpha_2)$ hat eindeutiges Minimum $\mathbf{M} \in \mathbb{R}^2$ falls $\vec{\mathbf{l}}_1 \nparallel \vec{\mathbf{l}}_2$
- Isolinien mit Schwellwert s $\{(\alpha_1, \alpha_2) : Q(\alpha_1, \alpha_2) = s\}$ bilden Ellipsen in (α_1, α_2) -Ebene



11.3.1 Abstand zweier linearer Komponenten ...

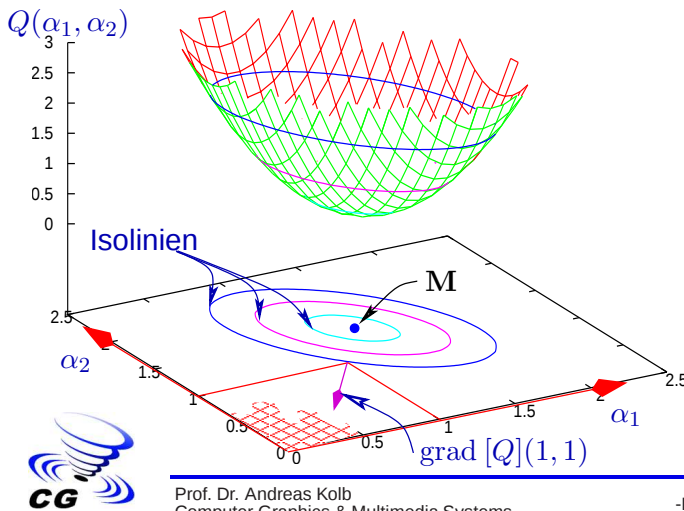


Quadratische Abstandsfunktion (Forts.)

Ziel: Finde $(\alpha_1^P, \alpha_2^P) \in I_1 \times I_2$ mit minimalem Q ; (α_1^P, α_2^P) liegt auf kleinstmöglicher Ellipse

Gradient mittels partieller Ableitungen:

$$\text{grad}[Q](\alpha_1, \alpha_2) = \left(\frac{\partial Q}{\partial \alpha_1}(\alpha_1, \alpha_2), \frac{\partial Q}{\partial \alpha_2}(\alpha_1, \alpha_2) \right)$$



Nutzung des Gradienten:

1. $M = (\alpha_1^*, \alpha_2^*) \Leftrightarrow \text{grad}[Q](M) = \vec{0}$
2. $\text{grad}[Q](\alpha_1, \alpha_2)$ weist in Richtung des größten Anstiegs von Q im Parameter (α_1, α_2) (siehe auch *Implizite Flächen*)



Prof. Dr. Andreas Kolb
Computer Graphics & Multimedia Systems

-Folie 11-25-

Computergraphik II

11.3.1 Abstand zweier linearer Komponenten ...



Berechnung des Gradienten von $Q(\alpha_1, \alpha_2)$

Gegeben:

$$Q(\alpha_1, \alpha_2) = \|\mathbf{L}_1(\alpha_1) - \mathbf{L}_2(\alpha_2)\|^2 = ((\mathbf{L}_1(\alpha_1) - \mathbf{L}_2(\alpha_2)) \cdot (\mathbf{L}_1(\alpha_1) - \mathbf{L}_2(\alpha_2)))$$

Gradient: Mit der Kettenregel $(f(x)^2)' = (f(x) \cdot f(x))' = 2 \cdot f(x) \cdot f'(x)$ gilt

$$\begin{aligned} \frac{\partial Q}{\partial \alpha_1}(\alpha_1, \alpha_2) &= 2 \cdot \left((\mathbf{L}_1(\alpha_1) - \mathbf{L}_2(\alpha_2)) \cdot \frac{\partial}{\partial \alpha_1} (\mathbf{L}_1(\alpha_1) - \mathbf{L}_2(\alpha_2)) \right) \\ &= 2 \cdot \left((\mathbf{L}_1(\alpha_1) - \mathbf{L}_2(\alpha_2)) \cdot \vec{l}_1 \right) \\ &= 2 \cdot (\alpha_1(l_1 \cdot l_1) - \alpha_2(l_1 \cdot l_2) + ((\mathbf{V}_1 - \mathbf{V}_2) \cdot l_1)) \end{aligned}$$

Das Minimum $M = (\alpha_1^*, \alpha_2^*)$ als Lösung von $\frac{\partial Q}{\partial \alpha_1}(\alpha_1^*, \alpha_2^*) = \frac{\partial Q}{\partial \alpha_2}(\alpha_1^*, \alpha_2^*) = 0$ entspricht natürlich dem Ergebnis von oben

Je größer der Abstand d , desto weiter liegt die Ellipse von M entfernt.



Prof. Dr. Andreas Kolb
Computer Graphics & Multimedia Systems

-Folie 11-26-

Computergraphik II



Abstand zweier Strecken mittels Quadratischer Abstandsfunktion

Ziel: Effiziente Behandlung der Fälle 2+3

Suche: Kontaktpunkt von Region 0
mit kleinster Ellipse um M

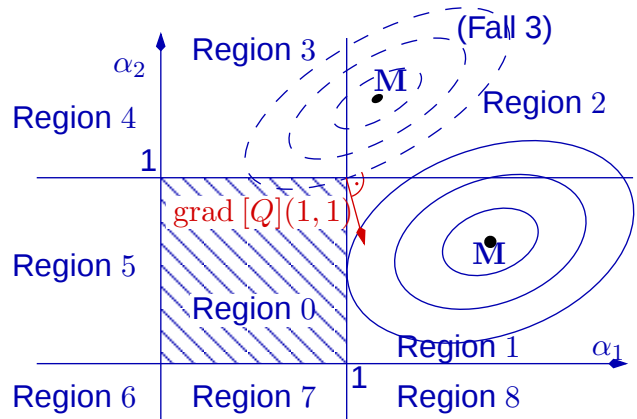
Fall 2: M liegt in Region 1, 3, 5 oder 7.

Bsp. Region 1: Optimum liegt auf $\alpha_1 = 1$

Fall 3: M liegt in Region 2, 4, 6 oder 8
Bsp. Region 2: Optimum auf

$$\{(\alpha_1, \alpha_2) : \alpha_1 = 1 \vee \alpha_2 = 1\}$$

grad [Q] bei (1, 1) liefert Richtung:



Aufteilung der (α_1, α_2) -Ebene

$$\text{Optimum liegt } \left\{ \begin{array}{l} \text{bei } \alpha_1 = \alpha_2 = 1 \\ \text{auf } \alpha_1 = 1 \\ \text{auf } \alpha_2 = 1 \end{array} \right\} \Leftrightarrow \left\{ \begin{array}{l} \text{sgn}(\text{grad } [Q](1, 1)) = (-, -) \\ \text{sgn}(\text{grad } [Q](1, 1)) = (-, +) \\ \text{sgn}(\text{grad } [Q](1, 1)) = (+, -) \end{array} \right\}$$



Beispiel: (siehe Daten von oben)

Die Fallunterscheidung von oben sieht jetzt folgendermaßen aus:

$$1. \text{ Gegeben: } \mathbf{L}_1(\alpha_1) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \alpha_1 \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{L}_2(\alpha_2) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}, \quad \alpha_i \in [0, 1]$$

$$2. \mathbf{M} = (\alpha_1^*, \alpha_2^*) = (-1/2, -1) \Rightarrow \mathbf{M} \text{ liegt in Region 6}$$

3. Auswertung des Gradienten beim Eckpunkt des Parametergebietes (0, 0):

$$\begin{aligned} \text{grad } [Q](0, 0) &= 2 \left(\left((\mathbf{L}_1(0) - \mathbf{L}_2(0)) \cdot \vec{\mathbf{i}}_1 \right), - \left((\mathbf{L}_1(0) - \mathbf{L}_2(0)) \cdot \vec{\mathbf{i}}_2 \right) \right) \\ &= 2 \left(\left(\begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} \right), \left(\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} \right) \right) = (-4, 6) \end{aligned}$$

damit liegt Optimum auf $\alpha_2 = 0$

4. Ermittle den Abstand von $\mathbf{L}_2(0)$ zu L_1 (siehe oben)



11.3.2 Weitere Abstandsmaße



Abstand Punkt-Rechteck

Vorteil der quadratischen Abstandsfunktion: Allgemeingültigkeit für ähnliche Probleme

Gegeben: Punkt P und Rechteck R :

$$R(\alpha_1, \alpha_2) = V + \alpha_1 \vec{e}_1 + \alpha_2 \vec{e}_2, \quad \alpha_1, \alpha_2 \in [0, 1]$$

Gesucht: $d_R(P) = \min_{Q \in R} \|P - Q\|$

Quadratische Abstandsfunktion:

$$Q(\alpha_1, \alpha_2) = \|R(\alpha_1, \alpha_2) - P\|^2 = ((R(\alpha_1, \alpha_2) - P) \cdot (R(\alpha_1, \alpha_2) - P))$$

Globales Minimum: $M = (\alpha_1^*, \alpha_2^*)$ ohne Rücksicht auf Einschränkung für α_i

Regionen: Unterteile (α_1, α_2) -Ebene in Regionen; es entsteht dieselbe Unterteilung wie Strecke-Strecke

Fälle: analog wie Strecke-Strecke



11.3.2 Weitere Abstandsmaße ...



Weitere Objekte

Punkt/Dreieck: Definition des Dreiecks:

$$D(\alpha_1, \alpha_2) = V + \alpha_1 \vec{e}_1 + \alpha_2 \vec{e}_2, \quad \alpha_1, \alpha_2, \alpha_1 + \alpha_2 \in [0, 1]$$

Abstand Lineare Komponente/Dreieck:

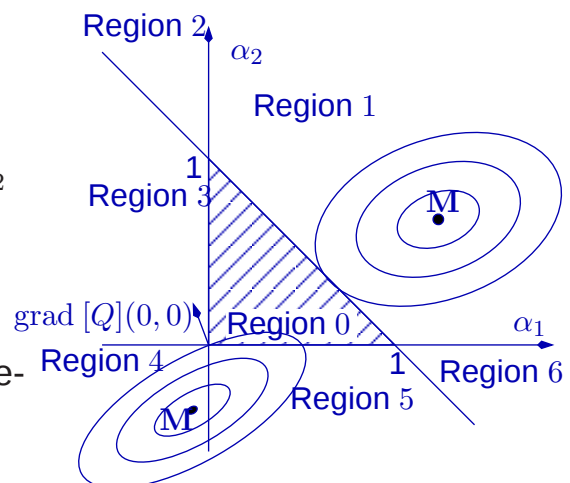
Drei Parameter für Abstandsfunktion:

$$Q(\alpha_1, \alpha_2, \alpha_3) = \|D(\alpha_1, \alpha_2) - L(\alpha_3)\|^2$$

L ist Gerade/Strahl/Strecke

$\Rightarrow$ 7/14/21 Regionen im $\mathbb{R}^3$

Dreieck/Dreieck: Insgesamt vier Parameter und 49 Regionen.



11.3.3 Kollisionspunkt und -normale



Ansatz:

Ziel: ○ Koll.-Punkt und -normale für einfache implizite Objekte effizient berechnen

○ Ermittlung unabhängig von der konkreter Basisgeometrie

Gegeben: Implizite Geometrie M_1, M_2 mit Basisgeometrien B_1, B_2 und Schwellwerten s_1, s_2 .

Abstand: Ermittlung der Punkte kürzesten Abstands auf den Basisgeometrien:

$$\begin{aligned} & \| \mathbf{B}_1(\alpha_{1,1}^P, \dots, \alpha_{1,n}^P) - \mathbf{B}_2(a_{2,1}^P, \dots, \alpha_{2,m}^P) \| \\ & = \min \{ \| \mathbf{B}_1(\alpha_{1,1}, \dots, \alpha_{1,n}) - \mathbf{B}_2(a_{2,1}, \dots, \alpha_{2,m}) \| \} \end{aligned}$$

$$\text{Abstand: } d(M_1, M_2) = \| \mathbf{B}_1(\alpha_{1,1}^P, \dots, \alpha_{1,n}^P) - \mathbf{B}_2(a_{2,1}^P, \dots, \alpha_{2,m}^P) \| - (s_1 + s_2)$$

Kollisionsinformation: Falls $d(M_1, M_2) \approx 0$:

$$\text{Normale: } \vec{n}_{koll} = \mathbf{B}_2(\alpha_{2,1}^P, \dots, \alpha_{2,m}^P) - \mathbf{B}_1(\alpha_{1,1}^P, \dots, \alpha_{1,n}^P), \quad \hat{n}_{koll} = \vec{n}_{koll} / \|\vec{n}_{koll}\|$$

$$\text{Punkte: } \mathbf{K}_1 = \mathbf{B}_1(\alpha_{1,1}^P, \dots, \alpha_{1,n}^P) + s_1 \hat{n}_{koll}$$

$$\mathbf{K}_2 = \mathbf{B}_2(\alpha_{2,1}^P, \dots, \alpha_{2,m}^P) - s_2 \hat{n}_{koll}, \quad \mathbf{K}_1 \approx \mathbf{K}_2$$



11.4 Kollision für Dreiecksnetze



Idee: Verwende Abstandsberechnung für Basisgeometrien zur Kollisionserkennung, z.B. für Dreiecken:

$$\text{Dreiecke: } \mathbf{D}_i(\alpha_1, \alpha_2) = \mathbf{V}_i + \alpha_{i,1} \vec{e}_{i,1} + \alpha_{i,2} \vec{e}_{i,2},$$

$$\text{mit } \alpha_{i,1}, \alpha_{i,2}, \alpha_{i,1} + \alpha_{i,2} \in [0, 1], \quad i = 1, 2$$

$$\text{Schwellwert: } s_i = 0, \quad i = 1, 2$$

$$\text{Kollision} \Leftrightarrow d(D_1, D_2) \approx 0$$

Problem: Relativ aufwendige Berechnung für Dreiecke (vier α -Parameter)

Alternativer Ansatz: Betrachte paarweise Kollision zwischen Teilprimitiven Eckpunkt, Kante und Dreieck(Fläche)

	Punkt	Kante	Dreieck
Punkt	degeneriert	→ Kante-Kante	!!
Kante	–	!!	→ Punkt-Dreieck → Kante-Kante
Dreieck	–	–	→ Punkt-Dreieck → Kante-Kante

Relevante Fälle: Kante-Kante und Punkt-Dreieck



11.4 Kollision für Dreiecksnetze ...



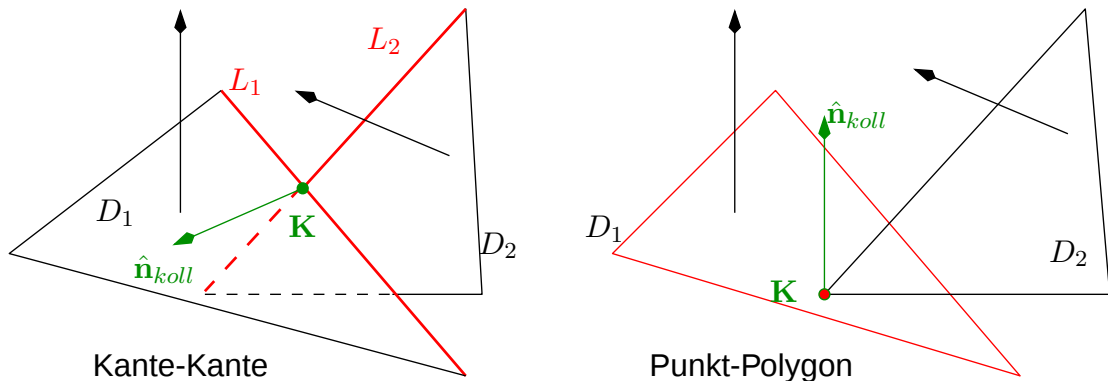
Alternativer Ansatz für Dreiecke/Polygone (Forts.)

Kollisionspunkt: Gegeben durch Schnittpunkt der Kanten bzw. durch den Punkt bei Punkt-Dreieck

Kollisionsnormale wird aus geometrischen Primitiven ermittelt:

Kante-Kante: Verwende das Kreuzprodukt der Kantenrichtungen

Punkt-Fläche: Normale des Polygons



11.4 Kollision für Dreiecksnetze ...



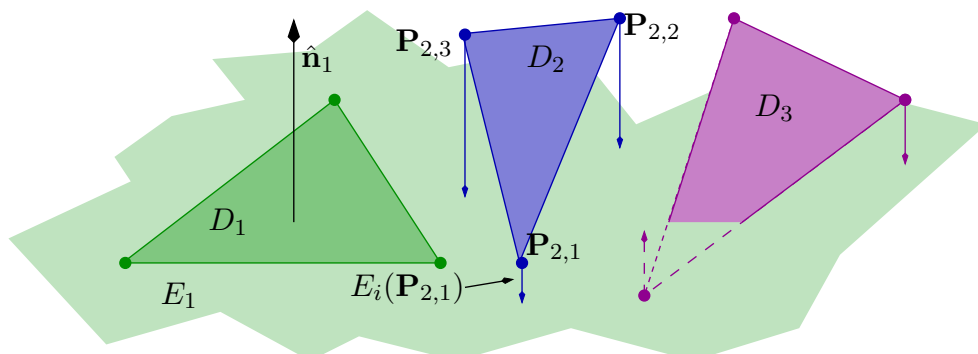
Grobe Abschätzung für Dreieck-Dreieck-Kollision

Beobachtung: Keine Kollision, wenn Eckpunkte eines Dreiecks *auf einer Seite* der anderen Dreiecksebene sind

Hessesche Normalform: $E_i(\mathbf{P}) = (\mathbf{P} \cdot \hat{\mathbf{n}}_i) - d_i$

Eckpunkte des j -ten Dreiecks $\mathbf{P}_{j,k}$, $k = 1, \dots, 3$ auf einer Seite von E_i , wenn

$$\text{sgn}(E_i(\mathbf{P}_{j,1})) = \text{sgn}(E_i(\mathbf{P}_{j,2})) = \text{sgn}(E_i(\mathbf{P}_{j,3}))$$





Einsatzbereiche von Raumunterteilungen

Visibilitätstest: Welches Objekt liegt einem Beobachterpunkt am nächsten?

Level of Detail: Wie detailliert muss ein Objekt für einen Beobachterpunkt dargestellt werden (Objektentfernung)?

Kollisionstest: Welche (statischen) Objekte sind nahe am bewegten Objekt?

Allgemeines Ziel: Effizienter „räumlicher“ Zugriff auf Objekte

Generelle Eigenschaften

Raumunterteilung: *Disjunkte* Zerlegung eines (*Beobachtungs*-)Raumes R in Teilräume U_i , d.h. $U_i \cap U_j = \emptyset$ falls $i \neq j$:

$$R = \dot{\bigcup}_{i=1}^n U_i \text{ in konvexe Teilräume } U_i, i = 1, \dots, n$$

Disjunkte heißt hier: $\forall \mathbf{P} \in R \exists_1 i : \mathbf{P} \in U_i$

Hierarchie von Raumunterteilung durch Rekursion ($\Rightarrow$ Baumstruktur)

Zuordnung von (meist statischen) Objekten zu den Teilräumen



11.5.1 2D-Quadrees und 3D-Octrees



Ausgangslage: Bounding-Box der gesamten Szene U^0

Rekursive Unterteilung am Box-Mittelpunkt in *gleichgroße Teilbereiche*:

Quadtree:

$$\text{initial: } U^0 = \dot{\bigcup}_{k=1}^4 U_k^1$$

$$\text{rekursiv: } U_I^i = \dot{\bigcup}_{k=1}^4 U_{(I,k)}^{i+1}$$

Octree

$$\text{initial: } U^0 = \dot{\bigcup}_{k=1}^8 U_k^1$$

$$\text{rekursiv: } U_I^i = \dot{\bigcup}_{k=1}^8 U_{(I,k)}^{i+1}$$

mit Multi-Index $I = (k_1, \dots, k_i)$, $k_j \in \{1, 2, 3, 4\}$ bzw. $k_j \in \{1, \dots, 8\}$.

Abbruchkriterium: Zwei gängige Alternativen

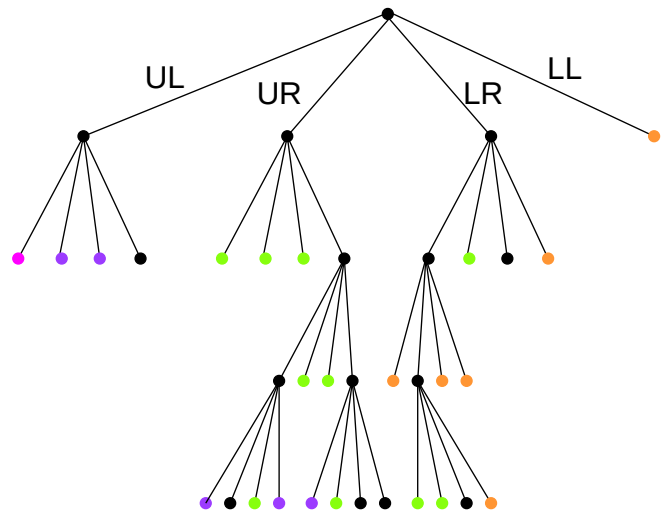
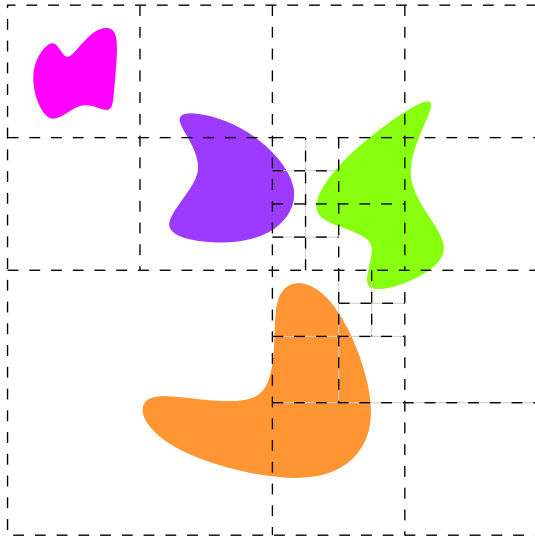
1. Teilbereich enthält (auch teilweise) max. eine Obergrenze n_{max} von Objekten
2. max. Unterteilungstiefe erreicht

Blattknoten können auch keine oder weniger als n_{max} Objekte (teilweise) enthalten





Beispiel: Quadtree (2D)



Benennung der Teilräume:

UL=„upper left“, UR=„upper right“, LR=„lower right“, LL=„lower left“



Quadtree Search

Ziel: Suche Zelle, in der Punkt $P = (p_x, p_y)$ liegt

Pseudo-Code:

```
NodeID searchQuadtree(Node root, Point P)
{
    if ( root.hasChildren == false ) return root.id;

    Point midPoint = 0.5 * (root.min + root.max); // subdiv. point
    if ( P.x < midPoint.x ) {
        if ( P.y < midPoint.y ) return searchQuadtree(root.ll, P);
        else return searchQuadtree(root.ul, P);
    }
    else { // includes the case "P.x == midPoint.x"
        if ( P.y < midPoint.y ) return searchQuadtree(root.lr, P);
        else return searchQuadtree(root.ur, P);
    }
}
```



11.5.2 Binary Space Partition (BSP)



Ansatz: Rekursive, binäre Unterteilung

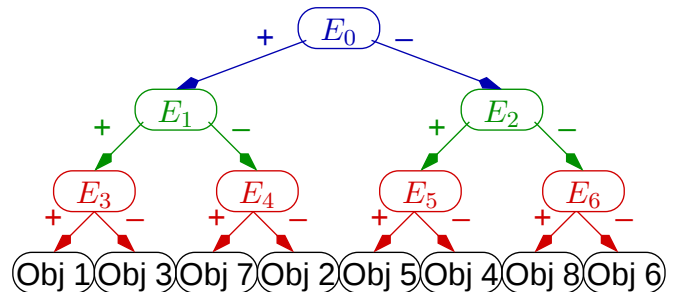
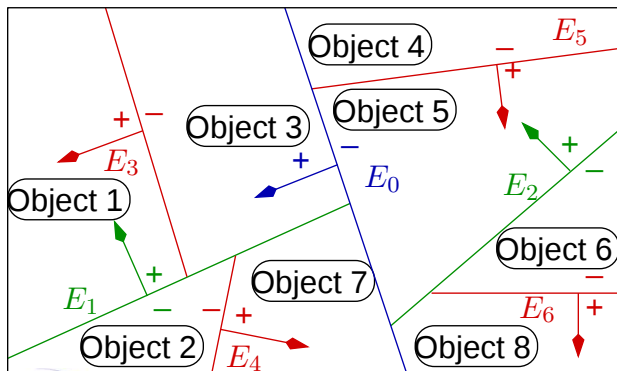
$n - 1$ -dim. *Hyper-Ebenen* unterteilen aktuellen Bereich in zwei Teilbereiche:

$$\text{Ebenengleichung: } E(\mathbf{P}) = (\mathbf{P} \cdot \hat{\mathbf{n}}) - d = 0$$

$$\text{Front-Bereich: } \{\mathbf{P} : E(\mathbf{P}) > 0\} \quad \text{Back-Bereich: } \{\mathbf{P} : E(\mathbf{P}) < 0\}$$

Ziele: ○ Objekte *vollständig* in einem Teilbereich ($\Rightarrow$ Objekte notfalls teilen!)

○ Ziel: Anzahl Objekte auf jeder Seite gleich ($\Rightarrow$ balancierter Baum)



11.5.2 Binary Space Partition (BSP) ...



BSP-Tree Search

Ziel: Suche Blattknoten, in der Punkt $\mathbf{P} = (p_x, p_y)$ liegt

Pseudo-Code:

```
NodeID searchBSPTree(Node root, Point P)
{
    if ( root.hasChildren == false ) return root.id;

    if ( evalPlane(root.plane, P) > 0 ) { // positive region
        return searchBSPTree(root.posSubtree, P);
    }
    else { // negative region
        return searchBSPTree(root.negSubtree, P);
    }
}
```



11.5.2 Binary Space Partition (BSP) ...

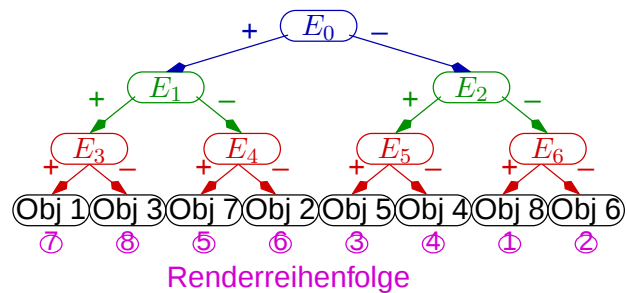
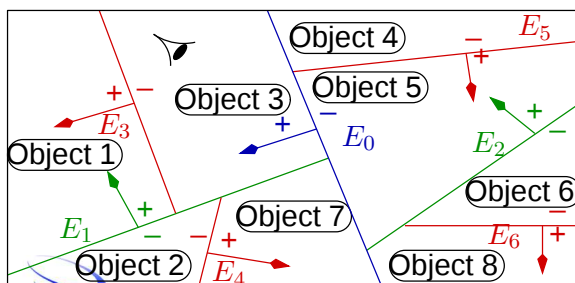


Hidden-Surface Removal mit BSP-Tree

Ziel: „Back-to-Front“-Rendering von Objekten (korrekte Verdeckung!)

Algorithmus:

```
void renderBSPTree(Node root, Camera cam) {
    if ( root.hasChildren == false ) { /* render objects */ }
    if ( evalPlane(root.plane, cam) > 0 ) { // cam in pos. region
        renderBSPTree(root.negSubtree, cam);
        renderBSPTree(root.posSubtree, cam);
    } else {
        renderBSPTree(root.posSubtree, cam); // cam in neg. region
        renderBSPTree(root.negSubtree, cam);
    }
}
```



11.5.3 k -Dimensional Tree (kd-Tree)

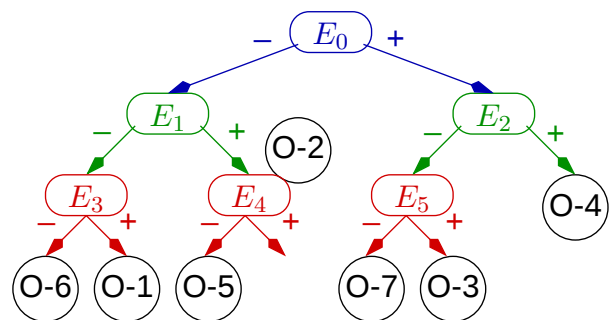
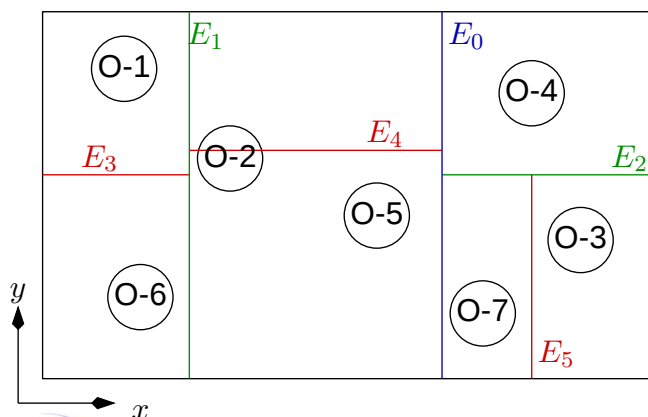


Ansatz: ○ Wie BSP-Tree, nur mit achsenparallelen Teilungsebenen

- Anzahl Objekte auf jeder Seite gleich ($\Rightarrow$ balancierter Baum)
- Unterteilung stets entlang der längsten Achse des aktuellen Teilquaders

Unterschied: ○ Objekte häufiger *nicht vollständig* in einem Teilbereich $\Rightarrow$ Objekte teilen oder übergeordneten Knoten zuordnen

- Effizientere Ebenenfindung als beim BSP-Tree





Vergleich Quadtree/Octree und BSP-Tree

Grundsätzlich: Alle Baumtypen für dynamische Szenen schwer nutzbar (Anpassung?, Neuaufbau?)

Octree: + effiziente Entscheidung auf Knotenebene (zwei Vergleiche)

- Baum i.a. nicht ausbalanciert
- Objekte u.U. in mehreren Blattknoten (Problem bei einigen Anwendungen)

BSP-Tree: + Baum i.a. gut balanciert $\Rightarrow$ konstanter Aufwand für Zugriff

- + effizienterer Baum, da Objekte genau in einem Blattknoten
- aufwendige Konstruktion durch Objekt-Splitting

kd-Tree: (Option ohne Objekt-Splitting)

- + Baum i.a. gut balanciert $\Rightarrow$ konstanter Aufwand für Zugriff
- +/- effizienter Baum, wenn Objekte genau in einem Blattknoten
- +/- effiziente Konstruktion (kein Objekt-Splitting)

