Computergraphik II

Winter 2012/2013

11 Kollisionserkennung und -reaktion

Versionsdatum: 3. Januar 2013

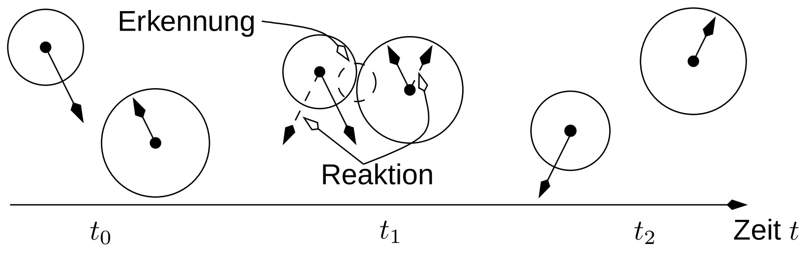
11 Kollisionserkennung und -reaktion ...

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

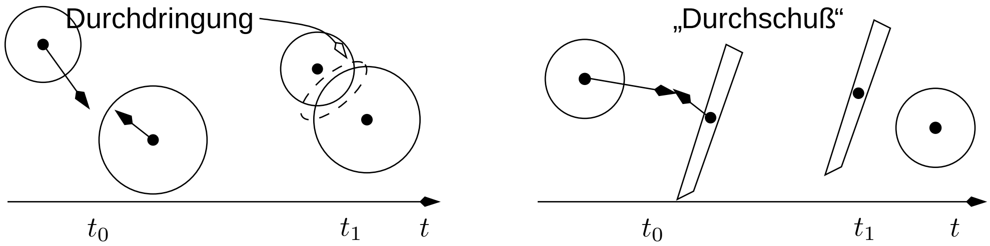
Kollisionsverarbeitung: 1. *Kollisionserkennung*: Ermittle Koll.-Information
2. *Kollisionsreaktion* der (dynamischen) Objekte auf Kollisionsereignis



Kollisionsinformation: 1. Boolsche Aussage: Einfache Kollisionsreaktion
2. Kollisionspunkte und -normalen bei komplexer Kollisionsreaktion

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 Koherenzen (räumlich, zeitlich) oder *Bounding Volumes*

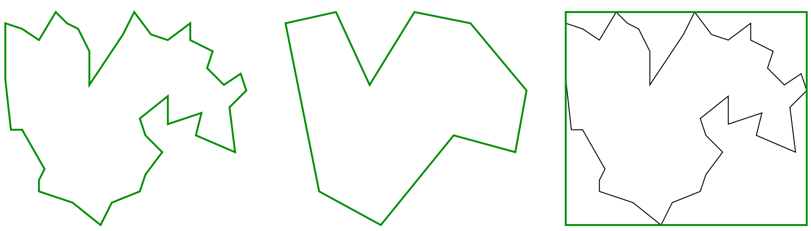
Abstrakter Algorithmus zur Kollisionserkennung:

```
forall moving objects m do
  forall objects o != m do
    if isClose(o, m) do          // schnelle, konserv. Abschätzung
      checkCollision(o, m);      // (aufwendige) Detailprüfung
```


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

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* $\vec{\mathbf{v}}_i$ bekannt

Ansatz Bewegungs-Unterbindung: Kollisionsinformation: Boolsch

Kollisionsreaktion: ❍ behalte letzte Nichtkollisions-Position bei

❍ setze aktuelle Geschwindigkeiten auf null: $\vec{\mathbf{v}}_1 = \vec{\mathbf{v}}_2 = 0$

Ansatz Entlanggleiten:

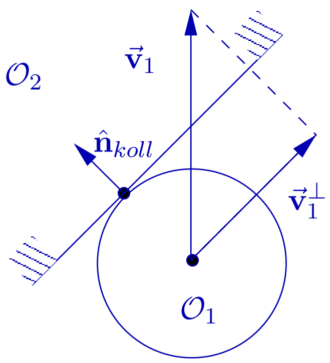
Kollisionsinformation: Normale $\hat{\mathbf{n}}_{koll}$

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

$$\vec{\mathbf{v}}_i = \vec{\mathbf{v}}^{||} + \vec{\mathbf{v}}^{\perp}, \ \left(\vec{\mathbf{v}}^{\perp} \cdot \hat{\mathbf{n}}_{koll}\right) = 0$$

$$\vec{\mathbf{v}}^{||} = \left(\vec{\mathbf{v}} \cdot \hat{\mathbf{n}}_{koll}\right) \cdot \hat{\mathbf{n}}_{koll}$$

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



11.1 Einfache Kollisionsreaktionen ...

Elastischer Stoß („Reflexion")

Vereinfachung: Vernachlässigung

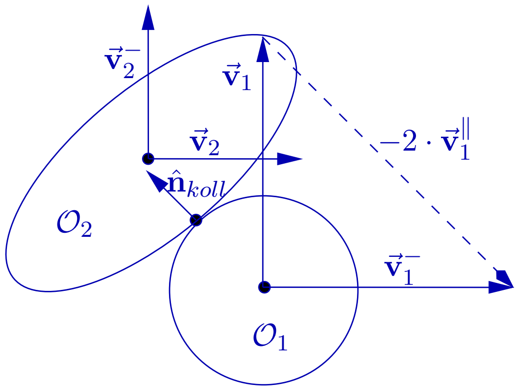
❍ der Objektmasse

❍ der Umwandlung von linearer Bewegung in Rotationsbewegung

Idee: Reflexion des Geschwindigkeitsvektors an Kollisionsnormale:

$$\vec{\mathbf{v}}_i^{-} = \vec{\mathbf{v}}_i - 2 \cdot \vec{\mathbf{v}}_i^{||}$$
$$= \vec{\mathbf{v}}_i - 2(\vec{\mathbf{v}}_i \cdot \hat{\mathbf{n}}_{koll})\hat{\mathbf{n}}_{koll}$$

Berücksichtigung von Masse und Rotation (siehe VR)



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

11.2 Boolsche Koll.-Erkennung mit Bounding Volumes

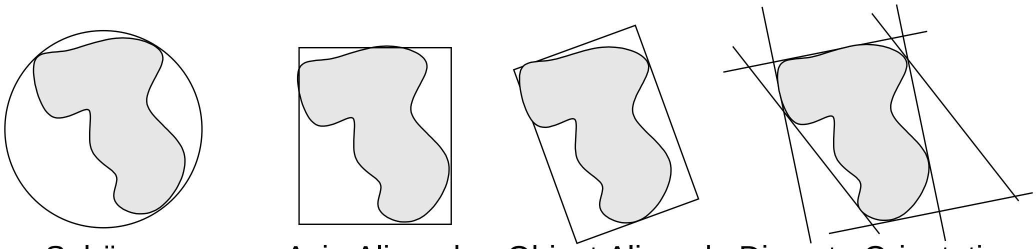
Definition: V ist ein *Bounding Volume* einer Geometrie G falls

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. $V \backslash G$ soll minimal sein
3. Effiziente Ermittlung für gegebene Objekte

Beispiele:

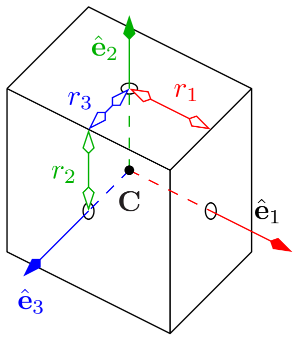


Sphäre — Axis-Aligned Bounding Box — Object Aligned Bounding Box — Discrete Orientation Polytop (k-dop)

11.2.1 Bounding Boxes

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

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



Gesucht: Boolsche Aussage, ob die Quader sich durchdringen

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

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

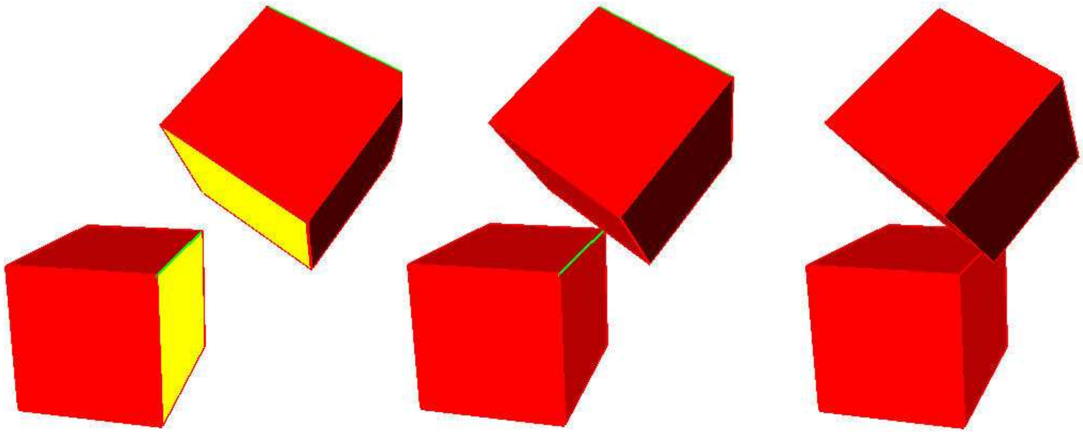
Separierende Ebene: Konvexe Geometrien durchdringen sich nicht, wenn es eine *separierende Ebene* zwischen ihnen gibt (Skizze!)

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 & Seitenfläche separieren | Kantenpaar separiert | Durchdringung

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

Separierende Ebenen für AABs

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

Kollisions-Erkennung: Für die Seitenflächen ergibt sich:

$$(x^1_{max} < x^2_{min}) \vee (x^2_{max} < x^1_{min}) \ldots$$

wobei $x^i_{min} = c^i_x - r^i_1,\ x^i_{max} = c^i_x + r^i_1$ mit $\mathbf{C}^i = (c^i_x, c^i_y, c^i_z)$

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 $\Longrightarrow$ arbeite mit ε-Umgebungen

$$(x^1_{max} - x^2_{min} < \varepsilon) \vee (x^2_{max} - x^1_{min} < \varepsilon) \ldots$$

Separierende Ebenen für OABs

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

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

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

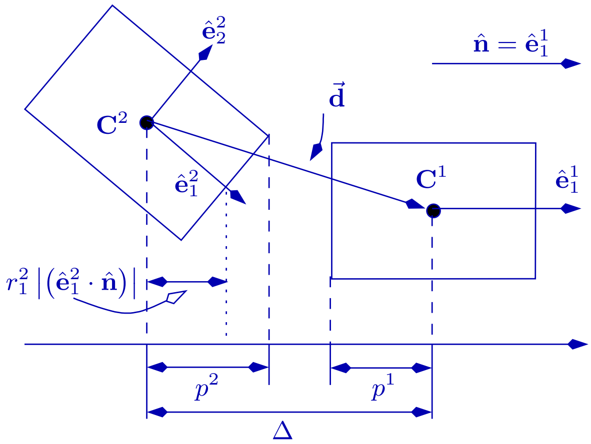
$$\Delta = \left|\left(\vec{\mathbf{d}} \cdot \hat{\mathbf{n}}\right)\right|, \quad \text{hier } \hat{\mathbf{n}} = \hat{\mathbf{e}}_1^1$$

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

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

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

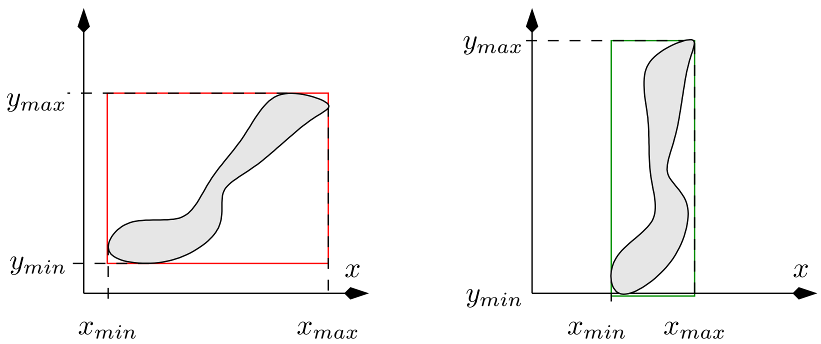
$$p^i = \sum_{j=1}^{3} r_j^i \left|\left(\hat{\mathbf{e}}_j^i \cdot \hat{\mathbf{n}}\right)\right|$$



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 $\{\mathbf{V}^i\}_{i=1}^n$ sehr einfach:

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

Bewertung von OAB

Kollisionsberechnung: Max. 15 effiziente Tests auf separierende Ebenen

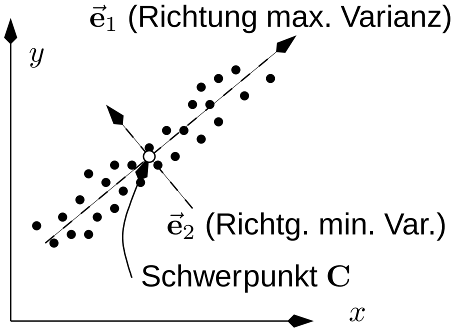
Objektanpassung: Gut, da OAB den Objekten lokal angepasst werden

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 $\hat{\mathbf{e}}_j$:

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



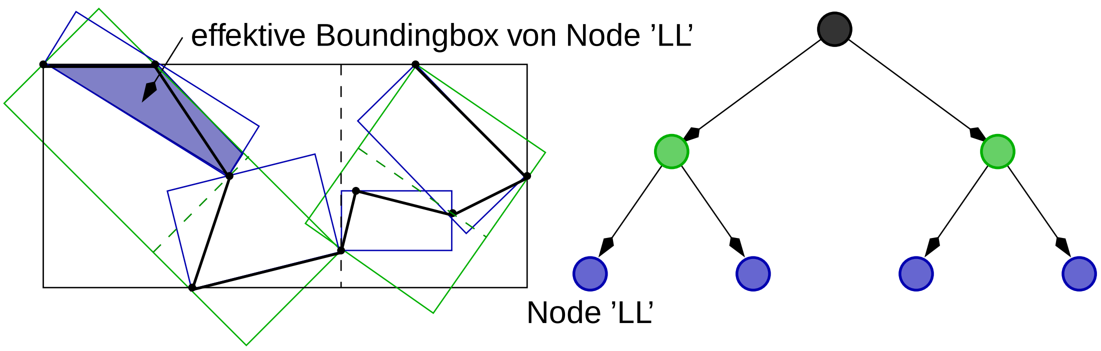
11.2.2 Bounding Volumen Hierarchien

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

Einschränkung: Ballance 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

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 $\hat{\mathbf{e}}_1 \perp \hat{\mathbf{e}}_2$, Skalierung i.a. nicht!

Transformation der lokalen AAB in VC mit Curr. Transformationmatrix M

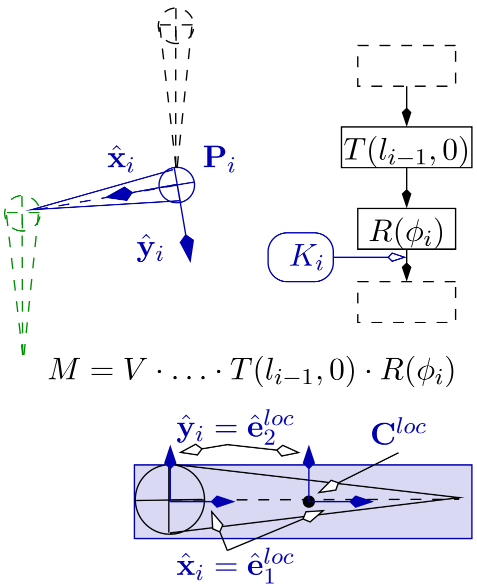
Lokale Koord.: $\mathbf{C}^{loc}, \hat{\mathbf{e}}_1^{loc}, \hat{\mathbf{e}}_2^{loc}, \hat{\mathbf{e}}_3^{loc}$

View-Koordinaten falls M ohne Skalierung:

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

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

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 $\mathbf{M}_i$ und Radius s_i:

$$S_i = \{\mathbf{P} \ : \ \|\mathbf{P} - \mathbf{M}_i\| = s_i\}, \ i = 1, 2$$

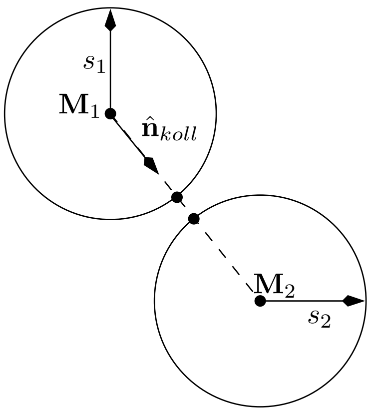
Gesucht: Kollisionserkennung, Kollisionspunkt(e) und Kollisionsnormale

Lösung: 1. Abstand der beiden Kugeln: $d(S_1, S_2) = \|\mathbf{M}_1 - \mathbf{M}_2\| - (s_1 + s_2)$

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

3. Kollisionsnormale: $\hat{\mathbf{n}}_{koll} = (\mathbf{M}_2 - \mathbf{M}_1) \,/\, \|\mathbf{M}_2 - \mathbf{M}_1\|$

4. Kollisionspunkt: $\mathbf{K} = \mathbf{M}_1 + s_1 \hat{\mathbf{n}}_{koll} \approx \mathbf{M}_2 - s_2 \hat{\mathbf{n}}_{koll}$



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

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

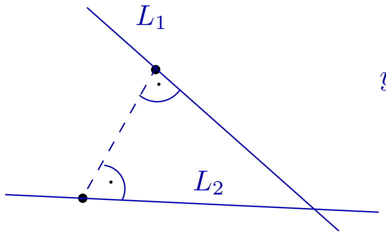
Gegeben: Lineare Komponenten L_1, L_2

Gesucht: Punkte mit kürzestem Abstand:

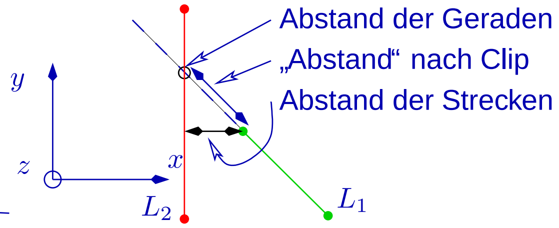
$(\alpha_1^P, \alpha_2^P) \in I_1 \times I_2$ mit $\left\|\mathbf{L}_1(\alpha_1^P) - \mathbf{L}_2(\alpha_2^P)\right\| = \min_{\alpha_1 \in I_1, \alpha_2 \in I_2} \{\|\mathbf{L}_1(\alpha_1) - \mathbf{L}_2(\alpha_2)\|\}$

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

❍ L_1 oder L_2 keine Gerade: Einfaches Clipping liefert i.a. falsches Ergebnis



Abstand zweier Gerade



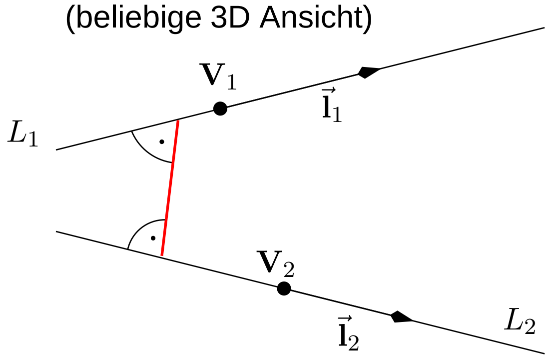
Strecken in $x-y$-Ebene

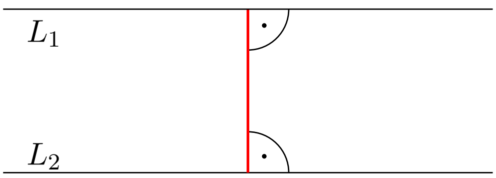
Ansichten der 3D Situation

Betrachte Punkte minimalen Abstands für zwei Geraden

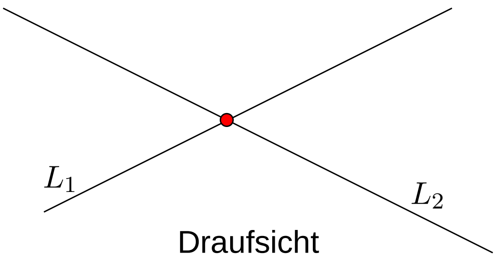
$$L_i : \quad \mathbf{L}_i(\alpha) = \mathbf{V}_i + \alpha_i \vec{\mathbf{l}}_i, \quad \alpha_i \in \mathbb{R}$$

Windschiefe Geraden (beliebige 3D Ansicht)





Seitenansicht



Draufsicht

Abstand zweier Geraden

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

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

$$\Big((\mathbf{L}_1(\alpha_1^P) - \mathbf{L}_2(\alpha_2^P)) \cdot \vec{\mathbf{l}}_i \Big) = \Big(\Big(\mathbf{V}_1 + \alpha_1^P \vec{\mathbf{l}}_1 - \mathbf{V}_2 - \alpha_2^P \vec{\mathbf{l}}_2 \Big) \cdot \vec{\mathbf{l}}_i \Big) = 0,$$

$$\Leftrightarrow \quad \alpha_1^P \Big(\vec{\mathbf{l}}_1 \cdot \vec{\mathbf{l}}_i \Big) - \alpha_2^P \Big(\vec{\mathbf{l}}_2 \cdot \vec{\mathbf{l}}_i \Big) = \Big((\mathbf{V}_2 - \mathbf{V}_1) \cdot \vec{\mathbf{l}}_i \Big).$$

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

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

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

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

Abstand zweier Strecken

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

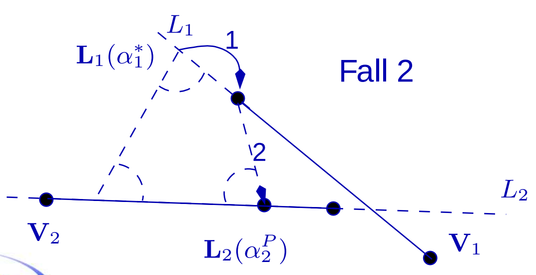
Lösung: Abstandsberechnung der zugehörigen **Geraden** liefert $\alpha_1^*, \alpha_2^* \in \mathbb{R}$

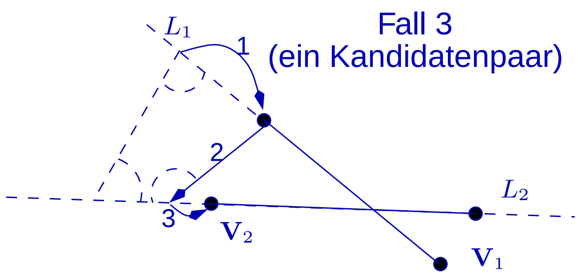
Fall 1: $\alpha_i^* \in [0, 1]$: dann $\alpha_i^P = \alpha_i^*$ und $\mathbf{L}_i(\alpha_i^*)$ sind Punkte min. Abstands.

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

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

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



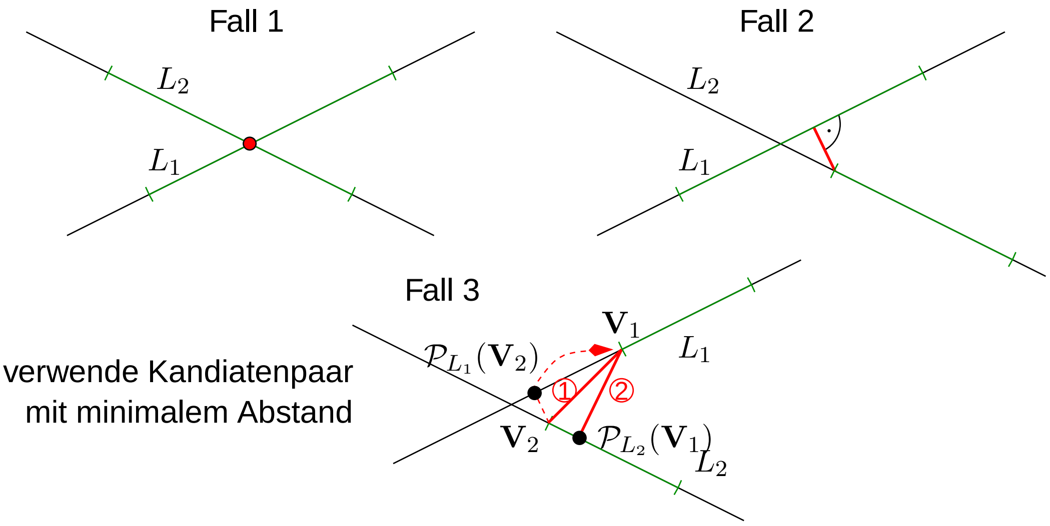


Ansichten der 3D Situation

Betrachte Punkte minimalen Abstands für zwei Strecken

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

Situation in der Draufsicht



verwende Kandiatenpaar mit minimalem Abstand

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:

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

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

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

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

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

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:

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

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

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:

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

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

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

Quadratische Abstandsfunktion

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

(α_1^P, α_2^P) so dass: $\left\|\mathbf{L}_1(\alpha_1^P) - \mathbf{L}_2(\alpha_2^P)\right\| = \min\limits_{\alpha_1 \in I_1, \alpha_2 \in I_2} \{\|\mathbf{L}_1(\alpha_1) - \mathbf{L}_2(\alpha_2)\|\}$

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:

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

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

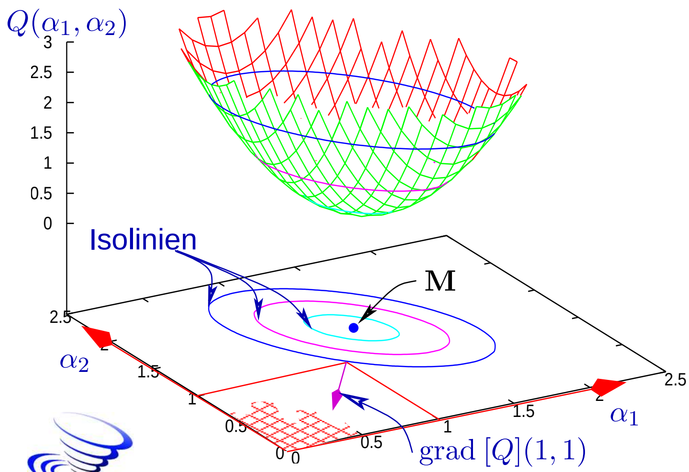
11.3.1 Abstand zweier linearer Komponenten ...

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:

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



Nutzung des Gradienten:

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

11.3.1 Abstand zweier linearer Komponenten ...

Berechnung des Gradienten von $Q(a_1, a_2)$

Gegeben:

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

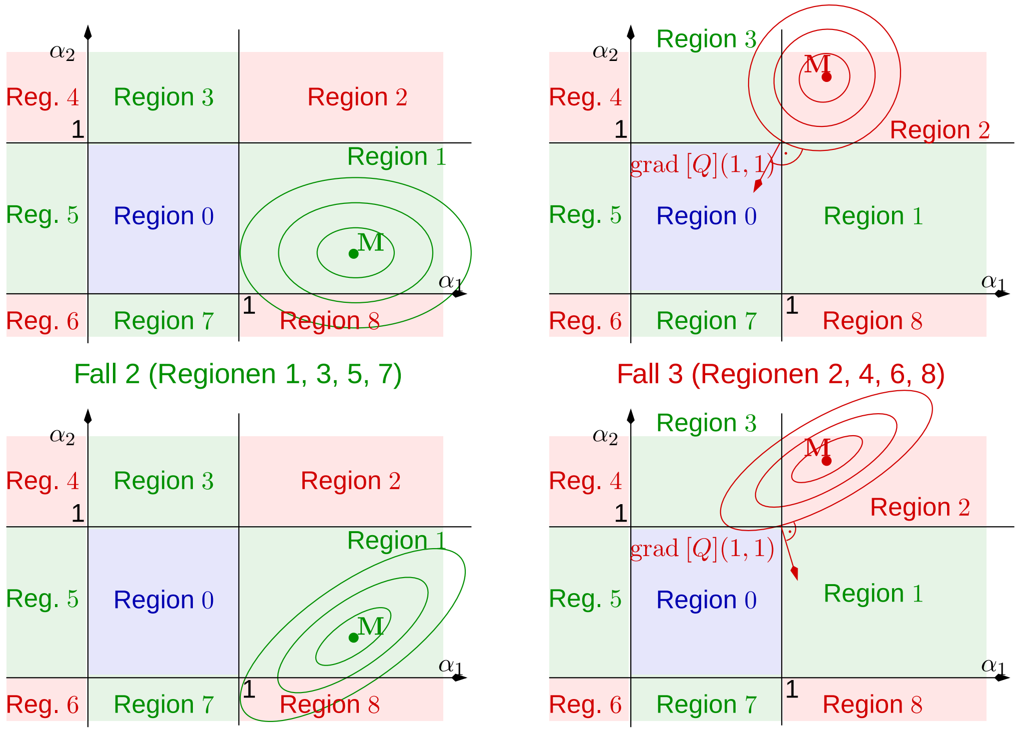
Gradient: Mit der Kettenregel $(f(x)^2)' = (f(x) \cdot f(x))' = 2 \cdot f(x) \cdot f'(x)$ gilt

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

Das Minimum $\mathbf{M} = (\alpha_1^*, \alpha_2^*)$ 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 $\mathbf{M}$ entfernt.

Isolinien in der (α_1, α_2)-Ebene



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.

Beispiel Region 1: Optimum liegt auf $\alpha_1 = 1$

Fall 3: M liegt in Region $2, 4, 6$ oder 8

Beispiel Region 2: Optimum auf

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

$\mathrm{grad}\,[Q]$ bei $(1,1)$ liefert Richtung für Region 2:

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

Beispiel: (siehe Daten von oben)

Die Fallunterscheidung von oben sieht jetzt folgendermaßen aus:

1. 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. $\mathbf{M} = (\alpha_1^*, \alpha_2^*) = (-1/2, -1) \Rightarrow \mathbf{M}$ liegt in Region 6
3. Auswertung des Gradienten beim Eckpunkt des Parametergebietes $(0,0)$:

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

   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 $\mathbf{P}$ und Rechteck R:

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

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

Quadratische Abstandsfunktion:

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

Globales Minimum: $\mathbf{M} = (\alpha_1^*, \alpha_2^*)$ 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:

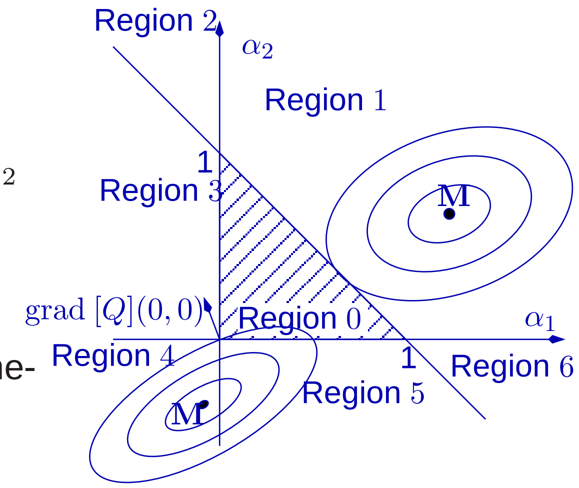
$$\mathbf{D}(\alpha_1, \alpha_2) = \mathbf{V} + \alpha_1 \vec{\mathbf{e}}_1 + \alpha_2 \vec{\mathbf{e}}_2, \quad \alpha_1, \alpha_2, \alpha_1 + \alpha_2 \in [0, 1]$$

Abstand Lineare Komponente/Dreieck: Drei Parameter für Abstandsfunktion:

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

L ist Gerade/Strahl/Strecke
$\Longrightarrow$ 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:

$$\left\|\mathbf{B}_1(\alpha^P_{1,1}, \ldots, \alpha^P_{1,n}) - \mathbf{B}_2(a^P_{2,1}, \ldots, \alpha^P_{2,m})\right\| = \min\{\|\mathbf{B}_1(\alpha_{1,1}, \ldots, \alpha_{1,n}) - \mathbf{B}_2(a_{2,1}, \ldots, \alpha_{2,m})\|\}$$

Abstand: $d(M_1, M_2) = \left\|\mathbf{B}_1(\alpha^P_{1,1}, \ldots, \alpha^P_{1,n}) - \mathbf{B}_2(a^P_{2,1}, \ldots, \alpha^P_{2,m})\right\| - (s_1 + s_2)$

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

Normale: $\vec{\mathbf{n}}_{koll} = \mathbf{B}_2(\alpha^P_{2,1}, \ldots, \alpha^P_{2,m}) - \mathbf{B}_1(\alpha^P_{1,1}, \ldots, \alpha^P_{1,n}),\ \hat{\mathbf{n}}_{koll} = \vec{\mathbf{n}}_{koll} / \|\vec{\mathbf{n}}_{koll}\|$

Punkte: $\mathbf{K}_1 = \mathbf{B}_1(\alpha^P_{1,1}, \ldots, \alpha^P_{1,n}) + s_1 \hat{\mathbf{n}}_{koll}$

$\mathbf{K}_2 = \mathbf{B}_2(\alpha^P_{2,1}, \ldots, \alpha^P_{2,m}) - s_2 \hat{\mathbf{n}}_{koll}, \qquad \mathbf{K}_1 \approx \mathbf{K}_2$

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

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

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

Schwellwert: $s_i = 0,\ i = 1, 2$

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	$\rightarrow$ Kante-Kante	**!!**
Kante	–	**!!**	$\rightarrow$ Punkt-Dreieck $\rightarrow$ Kante-Kante
Dreieck	–	–	$\rightarrow$ Punkt-Dreieck $\rightarrow$ Kante-Kante

Relavante Fälle: Kante-Kante und Punkt-Dreieck

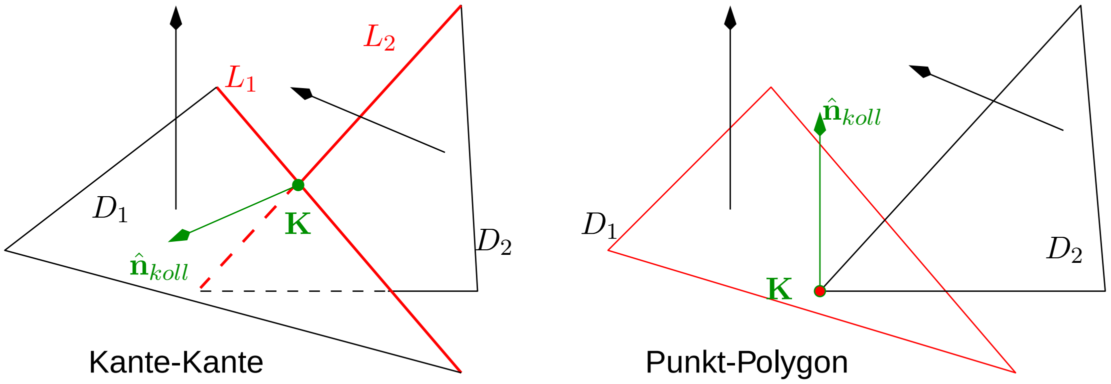
Alternativer Ansatz für Dreiecke/Polygone (Forts.)

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



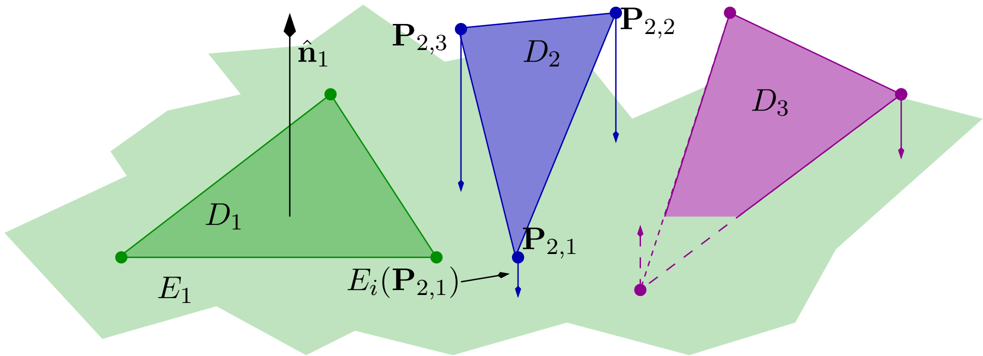
Grobe Abschätzung für Dreieck-Dreieck-Kollision

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 $\mathbf{P}_{j,k},\ k = 1, \ldots, 3$ auf einer Seite von E_i, wenn

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



Einsatzbereiche von Raumunterteilungen

Visibilitätstest: Welches Objekt liegt einem Beobachterpunkt am nächsten?

Level of Detail: Wie detailiert muss ein Objekt für einen Beobachterpunkt darstellt 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 $i \neq j$:

$$R = \dot{\bigcup_{i=1}^{n}} U_i \text{ in } \textit{konvexe Teilräume } U_i,\ i = 1, \ldots, n$$

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-Quadtrees und 3D-Octrees

Ausgangslage: Bounding-Box der gesamten Szene U^0

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

Quadtree:	Octree
initial: $U^0 = \dot{\bigcup}_{k=1}^{4} U_k^1$	initial: $U^0 = \dot{\bigcup}_{k=1}^{8} U_k^1$
rekursiv: $U_I^i = \dot{\bigcup}_{k=1}^{4} U_{(I,k)}^{i+1}$	rekursiv: $U_I^i = \dot{\bigcup}_{k=1}^{8} U_{(I,k)}^{i+1}$

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

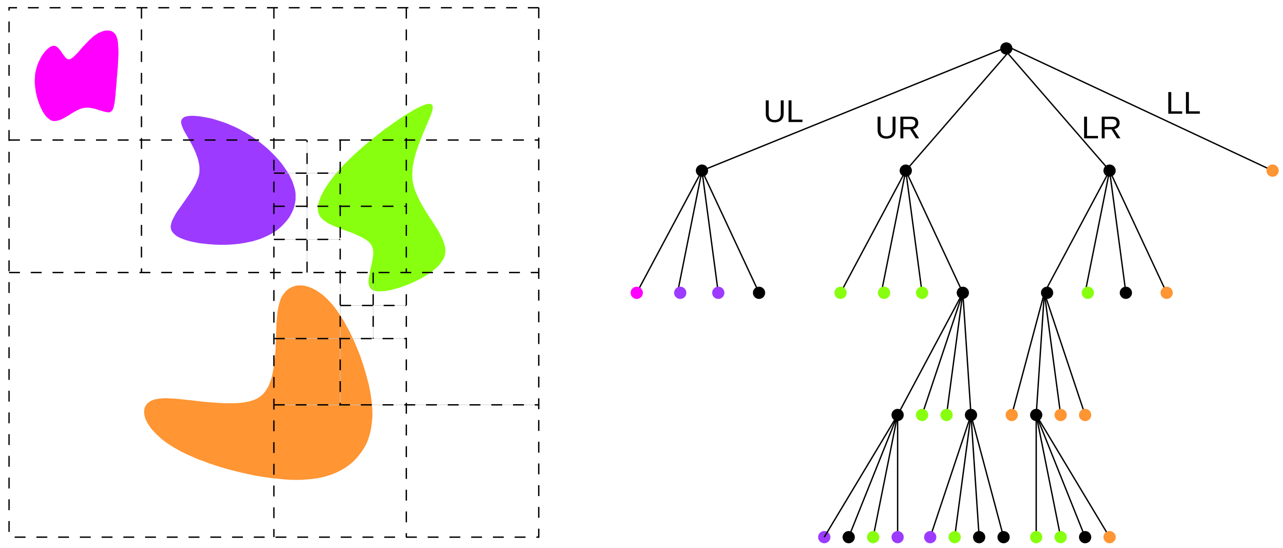
Abbruchkriterium: Zwei gängige Alternativen

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

11.5.1 2D-Quadtrees und 3D-Octrees ...

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 $\mathbf{P} = (p_x, p_y)$ 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)

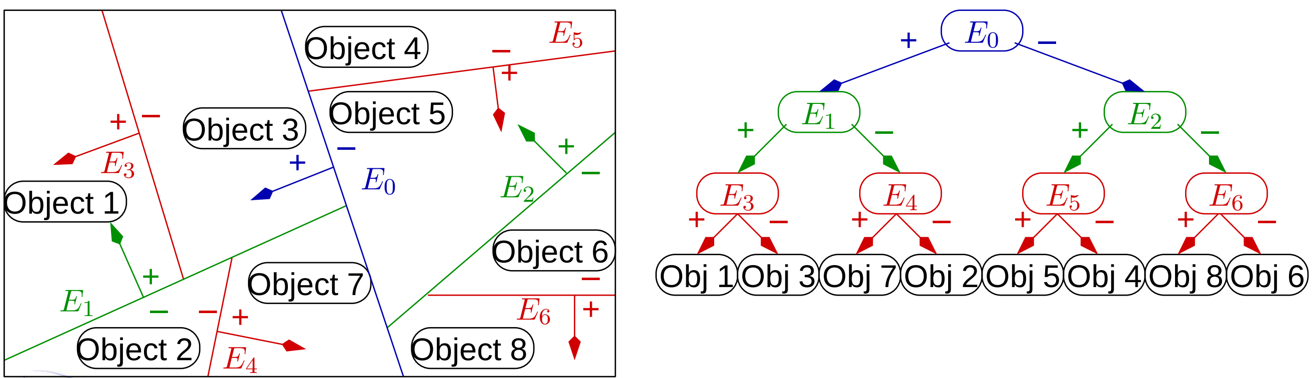
Ansatz: Rekursive, binäre Unterteilung

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

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

Front-Bereich: $\{\mathbf{P} \;:\; E(\mathbf{P}) > 0\}$ 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);
  }
}
```

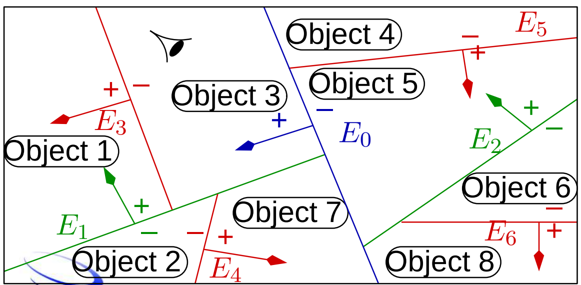

11.5.2 Binary Space Partition (BSP) ...

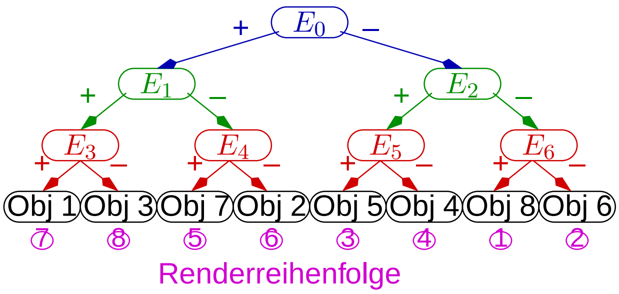
Hidden-Surface Removal mit BSP-Tree

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 {                                 // cam in neg. region
    renderBSPTree(root.posSubtree, cam);
    renderBSPTree(root.negSubtree, cam);
  }
}
```





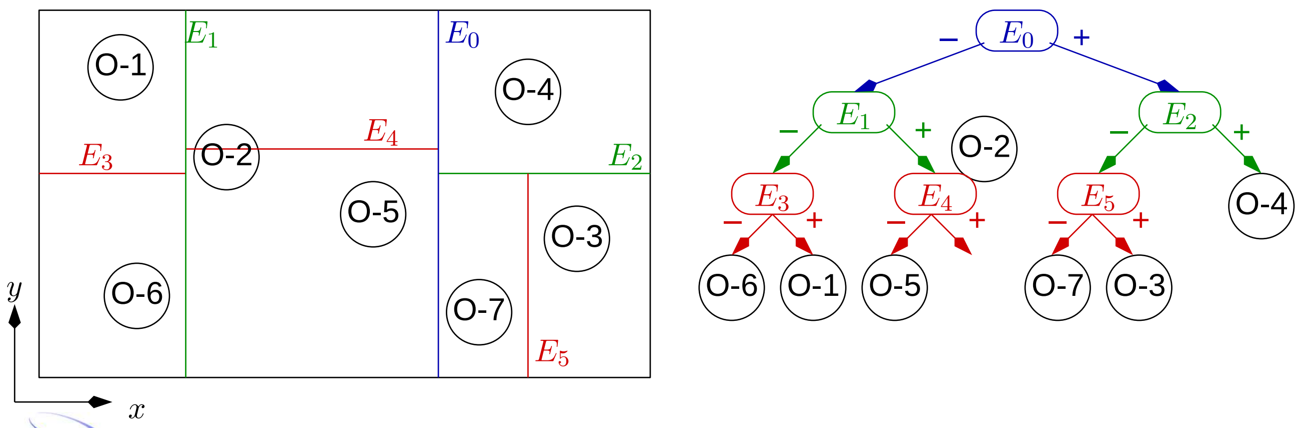
11.5.3 k-Dimensional Tree (kd-Tree)

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



11.5.3 k-Dimensional Tree (kd-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)