

§1 Hardwaregrundlagen

§2 Transformationen und Projektionen

§3 Repräsentation und Modellierung von Objekten

§4 Rasterung

§5 Visibilität und Verdeckung

§6 Rendering

§7 Abbildungsverfahren (Texturen, etc.)

§8 Freiformmodellierung

Anhang: Graphiksprachen und Graphikstandards

Anhang: Einführung in OpenGL

Weitere Themen: Netze, Fraktale, Animation, ...

- Ziel dreidimensionaler Computergraphik:
Erzeugung (zweidimensionaler) Darstellungen einer Szene oder eines Objektes ausgehend von Beschreibungen oder Modellen.
- Die Art und die Verwendung der Computer-internen Repräsentation eines Objektes hängt dabei an vielen Einflussfaktoren:
 - Das Objekt kann real oder nur in der Computerdarstellung existieren.
 - Die Herstellung des Objektes ist eng mit seiner Visualisierung verknüpft.
 - ➔ Interaktive CAD-Systeme
 - ➔ Modellierung und Visualisierung als Werkzeuge beim Herstellungsprozess
 - ➔ Mehr als 2D-Output möglich!

Einflussfaktoren (Fortsetzung)

- Die Genauigkeit der computerinternen Repräsentation orientiert sich an der Anwendung:
z. B. **exakte Beschreibung** von Geometrie und Form in CAD-Applikationen vs. einer für einen Renderer **ausreichenden approximativen** Beschreibung.
- Bei interaktiven Anwendungen existieren für ein Objekt oft sogar gleichzeitig **mehrere interne Repräsentationen** oder werden je nach Bedarf **dynamisch erzeugt**.
⇒ LOD (Level-of-Detail) Verfahren

Die Modellierung und Repräsentation von Objekten betrifft insbesondere die folgenden Aspekte

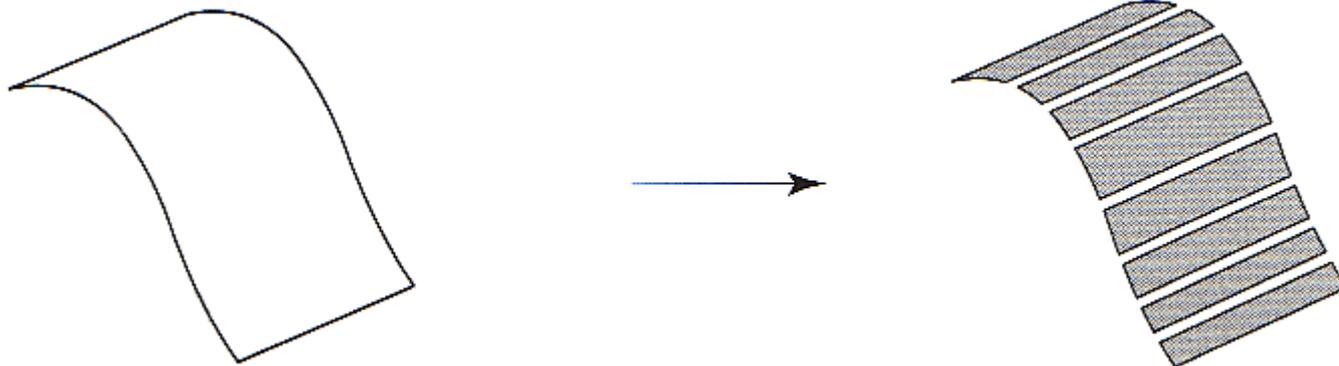
- Erzeugung von 3D **Geometrie**daten
- CAD-Interface, Digitizer, Laser-Scanner (Reverse Eng.), analytische Techniken (z.B. Sweeping), Bild- (2D) und Video(3D)-Analyse
- Repräsentation, effizienter Zugriff und Konvertierung
- Polygonnetze (z.B. Triangulierung) häufigste Repräsentation für Rendering.
Andere Darstellungen: Finite Elemente (FEM), Constructive Solid Geometry (CSG), B-Rep. („Boundary-Representation“ für CAD-Modelle), implizit (Isoflächen), Surface Elements (Surfels = Punkte + Normalen), etc.
- Manipulation \Rightarrow Formänderung der Objekte (Editing)
z.B. boolesche Operationen, lokale Glättung, Interpolation bestimmter Features (Randkurven), „Eingravieren“ geometrischer Details, etc.

Im Folgenden

- 3.2 Polygonaler Repräsentation dreidimensionaler Objekte**
- 3.3 CSG-Repräsentation von Objekten
- 3.4 Raumteilungsverfahren für die Objektrepräsentation
- 3.5 Objektrepräsentation mittels impliziter Funktionen
- 3.6 Szenenmanagement

Eigenschaften/Merkmale

- klassische Repräsentationsform dreidimensionaler Objekte in der Computergraphik
- Objekt wird durch ein **Netz polygonaler Facetten** (oft Dreiecke) repräsentiert \Rightarrow stückweise lineare Approximation.
- Die polygonalen Facetten stellen i. A. eine Approximation gekrümmter Flächen dar, die das Objekt begrenzen.



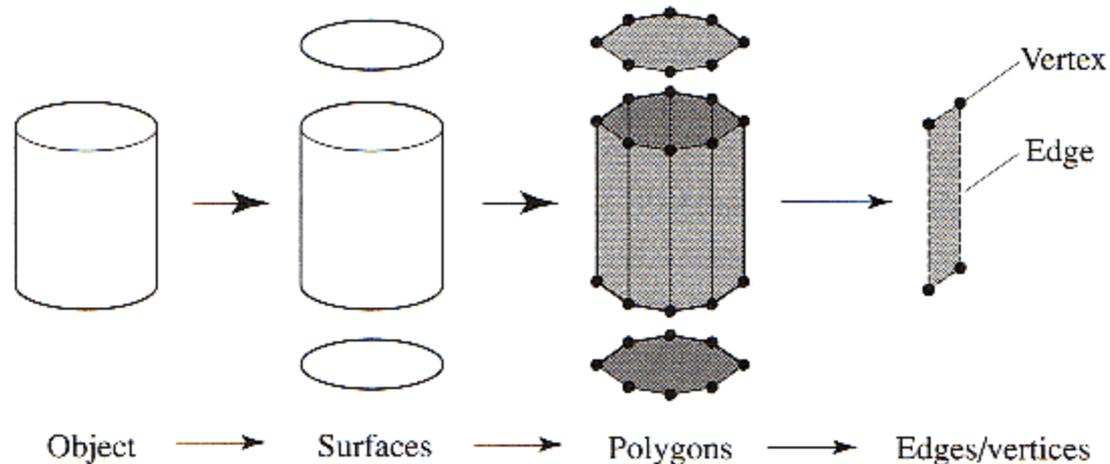
Eigenschaften/Merkmale (Fortsetzung)

Genauigkeit der Approximation (Anzahl und Größe der Polygone) kann gewählt werden, wirft aber oft Fragen und extreme Probleme auf, zB.:

- Welche Polygonauflösung benötigt man für **genaue Darstellung**?
- Welche Polygonauflösung benötigt ein Renderer, um die stückweise lineare Approximation **glatt erscheinen** zu lassen?
- Wie ist der **Zusammenhang** zwischen Polygonanzahl des Objektes und seiner Größe in der finalen Darstellung?
⇒ oft verwendete Grundregel: Polygonauflösung an die **lokale Krümmung der Fläche** binden

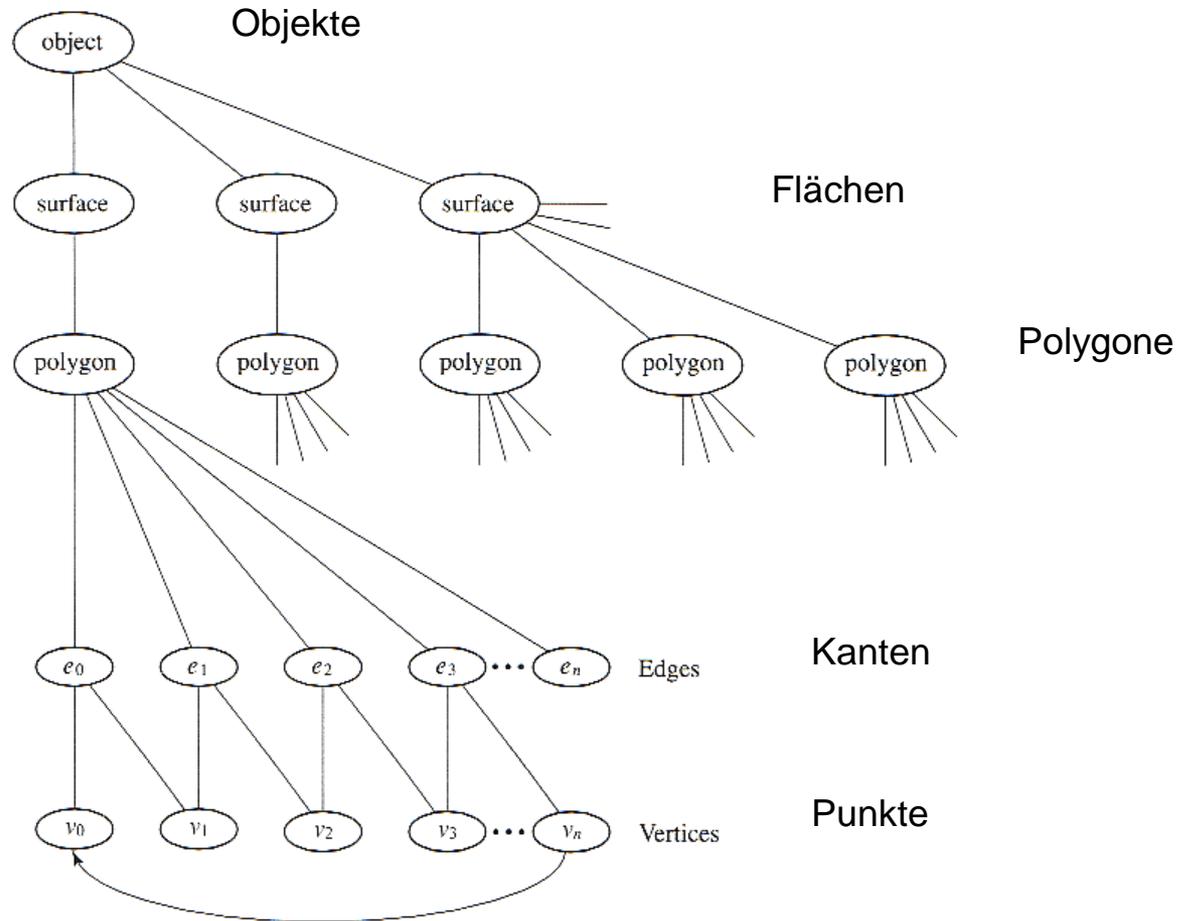
Repräsentationshierarchie

- **Konzeptuell**
 - Objekt setzt sich aus Oberflächen zusammen.
 - Oberfläche setzt sich aus Polygonen zusammen.
 - Polygon besteht aus Eckpunkten (vertices) und Kanten (edges).



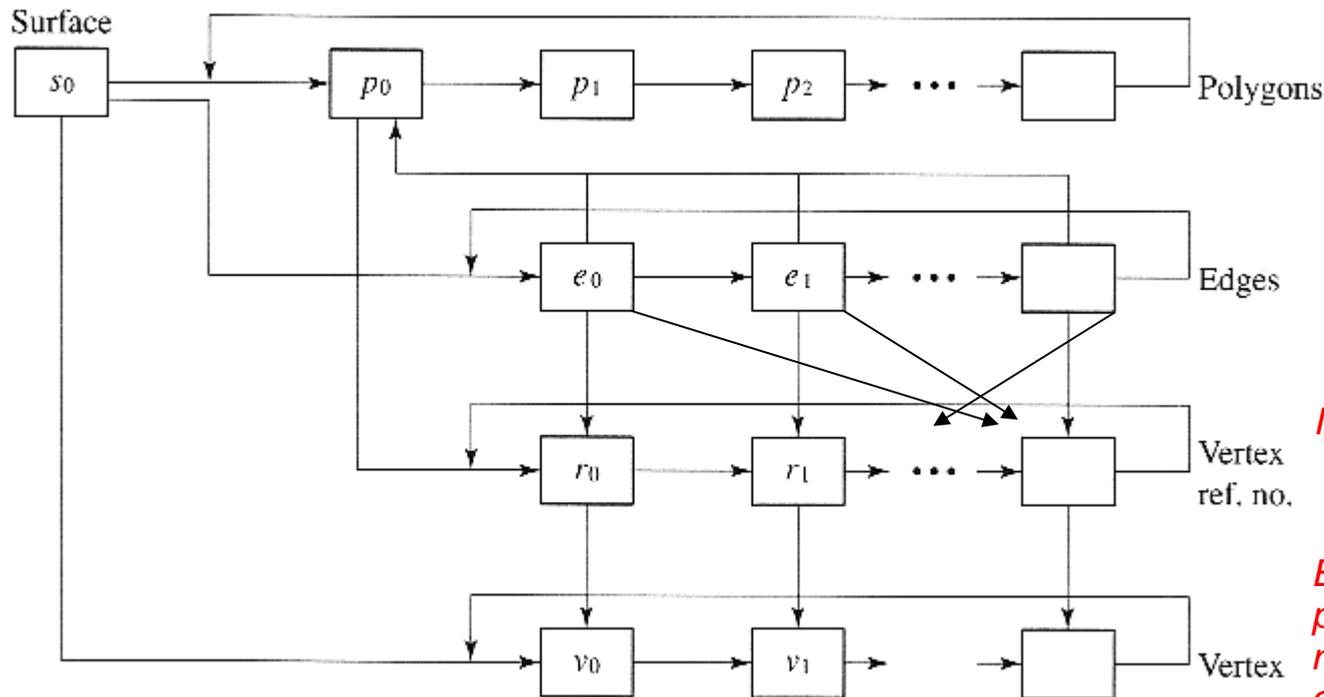
Repräsentationshierarchie

- Topologisch



Repräsentationshierarchie

- Datenstruktur

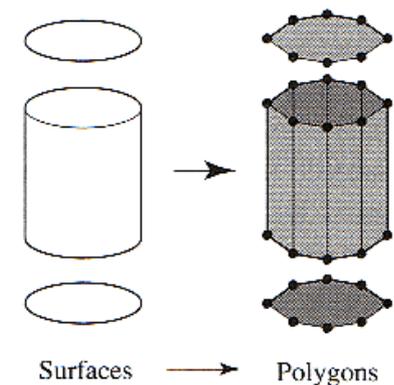


Indizes auf Eckpunkte

Eckpunkte werden pro Oberfläche nur einmal gespeichert!

Bemerkung zu Kanten

- Offensichtlich existieren in der approximierenden polygonalen Darstellung zwei Arten von Kanten:
 - Scharfe Kanten (Feature Lines)
 - ⇒ diese sollen als Kanten sichtbar bleiben
 - virtuelle Kanten (im Inneren glatter Flächen)
 - ⇒ diese sollte der Renderer „verschwinden“ lassen (HSR)
- 70er Jahre: Schattierungsalgorithmen (Interpolative Shading Algorithms)
 - ⇒ Flat/Uniform, Gouraud, Phong Shading
- Die Art der Kanten bestimmt in der Datenstruktur z. B. die Mehrfachspeicherung von Ecken und Kanten. (siehe auch Bild)



Bemerkung zu Datenstruktur

Praktische Datenstrukturen beinhalten neben der Geometrie spezielle Attribute für Anwendungen und Rendering:

- Flächenattribute:
Repräsentation(Dreieck, Polygon, Freiformfläche), Koeffizienten, Polygonnormalen, Eigenschaften (planar, konvex, hat Löcher), Verweis auf Eckpunkte (und ggf. Kanten)
- Kantenattribute:
Länge, Art (Randkante, Feature Line, virtuelle Kante), ggf. Verweis auf zugehörige Polygone und Eckpunkte
- Eckpunktattribute:
Eckpunktnormale, Farbe, Texturkoordinaten, ggf. Verweis auf Polygone und Kanten

gemittelte Polygonnormalen



Erzeugung polygonaler Objekte

Manuelle und halbautomatische Verfahren

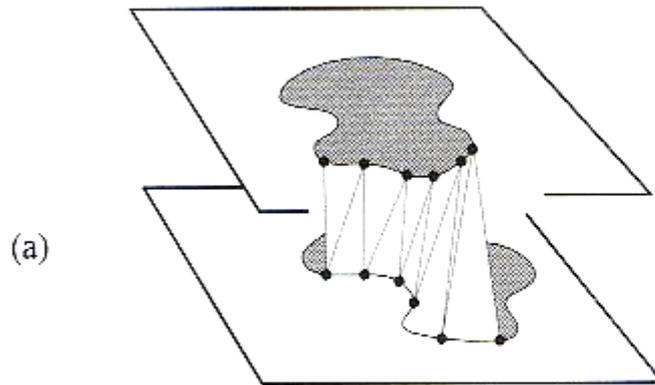
- „Manuelles“ Verschieben von (Gruppen von) Eckpunkten mittels dreidimensionaler Eingabegeräte oder Schnittstellen
 - ⇒ komplex, schwer handhabbar
 - ⇒ nur für einfache Objekte bzw. für einfache „Manipulationen“ geeignet
- 3D-Digitizer - manuelles Anbringen von Punkten auf Objekten, die mittels Digitizer zu Polygon-Eckpunkten werden sollen
Beispiel: Netze über Objektoberflächen „ziehen“
 - ⇒ erste 3D-Darstellungen von Karosserien (1974)

Erzeugung polygonaler Objekte (Fortsetzung)

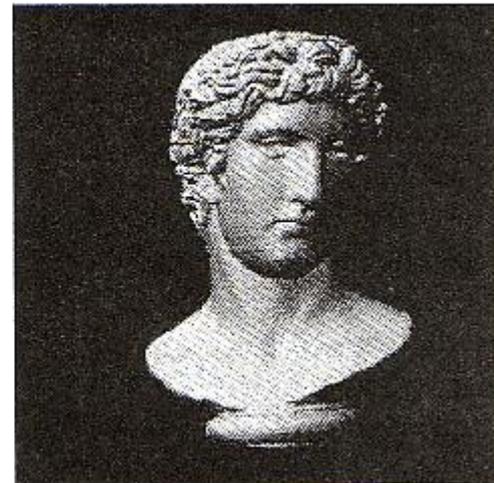
- Automatische Verfahren
- hier: Laserscanner
 - Objekt wird rundherum scheibchenweise mit einem Laserstrahl abgetastet; dieser misst den Abstand zur Objektoberfläche
 - aus den gemessenen 2D-Konturen werden mittels eines „skinning“-Algorithmus, der geeignet benachbarte Punkte verbindet, Dreiecksflächen erzeugt (Abb. (a))
 - dieser Ansatz tendiert dazu, (zu) viele Dreiecke zu erzeugen! (Abb. (b): 400.000 Dreiecke)
 - Anwendungen: Reverse Engineering, virtuelle Bekleidung, etc.

Erzeugung polygonaler Objekte: (Fortsetzung)

- Automatische Verfahren (Fortsetzung)



(b)



Problem: Ist das Objekt stellenweise „zu konkav“, gibt es Flächen, die vom Laserstrahl nicht erfasst werden können

Erzeugung polygonaler Objekte

Mathematische Verfahren

- Erzeugung von polygonalen Darstellungen aus analytischen Kurven und Flächen \Rightarrow CAD-Anwendungen
- Vorteile:
 - Benutzer arbeitet mit high-level Objektbeschreibung
 - Objektform ist direkt mit mathematisch exakter Objektbeschreibung gekoppelt
- Beispiele: Parameterflächen (stückweise Polynome), Rotationsflächen, Sweep-Flächen, ...

Erzeugung polygonaler Objekte

Prozedurale Verfahren

- Erzeugung polygonaler Objekte durch **Fraktale**
- Fraktale (Fractals) gehen in ihrem theoretischen Ansatz auf die **Mandelbrot-Geometrie** zurück
- Werden ua. für die Modellierungen von geographischen **Höhenfeldern** (Terrain Models) eingesetzt.
- Fraktale finden aufgrund ihrer Effizienz z.B. Anwendung in professionellen Flugsimulatoren für das Pilotentraining.

Erzeugung polygonaler Objekte

Prozedurale Verfahren

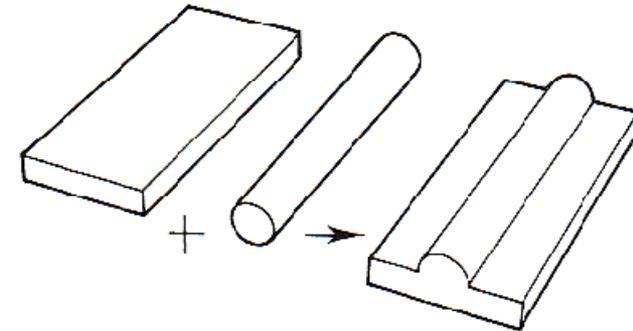
- Erzeugung polygonaler Objekte durch **Ersetzungssysteme**, zB. **Grammatiken**
- **Lindenmayer-/L-Systeme** zur Beschreibung von biologischen Entwicklungen
- Werden ua. für die Modellierungen von **Bäumen**, **Pflanzen**, etc. eingesetzt.

Eigenschaften der Constructive Solid Geometry (CSG)

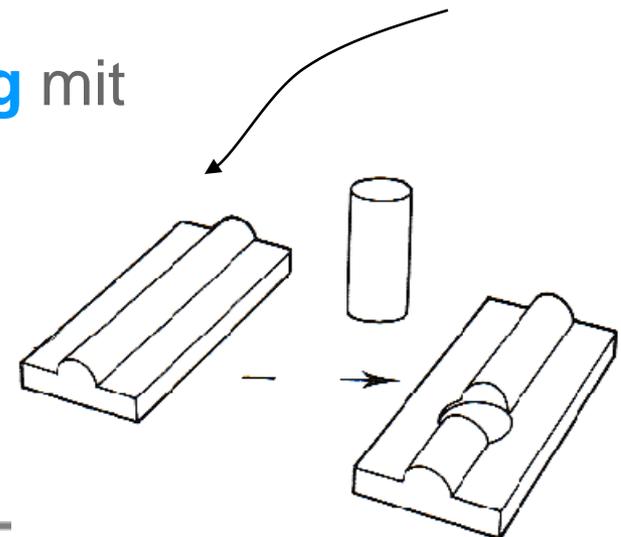
- Volumenrepräsentation dreidimensionaler Objekte
- ermöglicht dem Designer die **interaktive Konstruktion**
- Repräsentation komplexer Objekte durch **Zusammensetzen einfacher Grundobjekte** (Primitive) mittels **boolescher Mengenoperationen** und linearer Transformationen
- Geometrische Primitive: Kugel, Kegel, Zylinder, Quader, ...
- Nachteil: Darstellung von CSG-Objekten erfordert spezielle Rendering-Techniken (z.B. Raytracing) oder Umwandlung in eine polygonale Repräsentation (Boundary Evaluation)

Beispiel: Boolesche Operationen

- **Vereinigung** von Quader und Zylinder ...



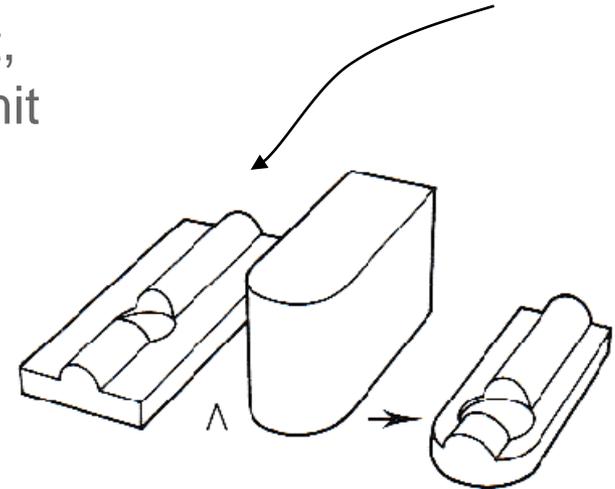
- ... anschließend **Differenzbildung** mit einem weiteren Zylinder ...



Beispiel: Boolesche Operationen (Fortsetzung)

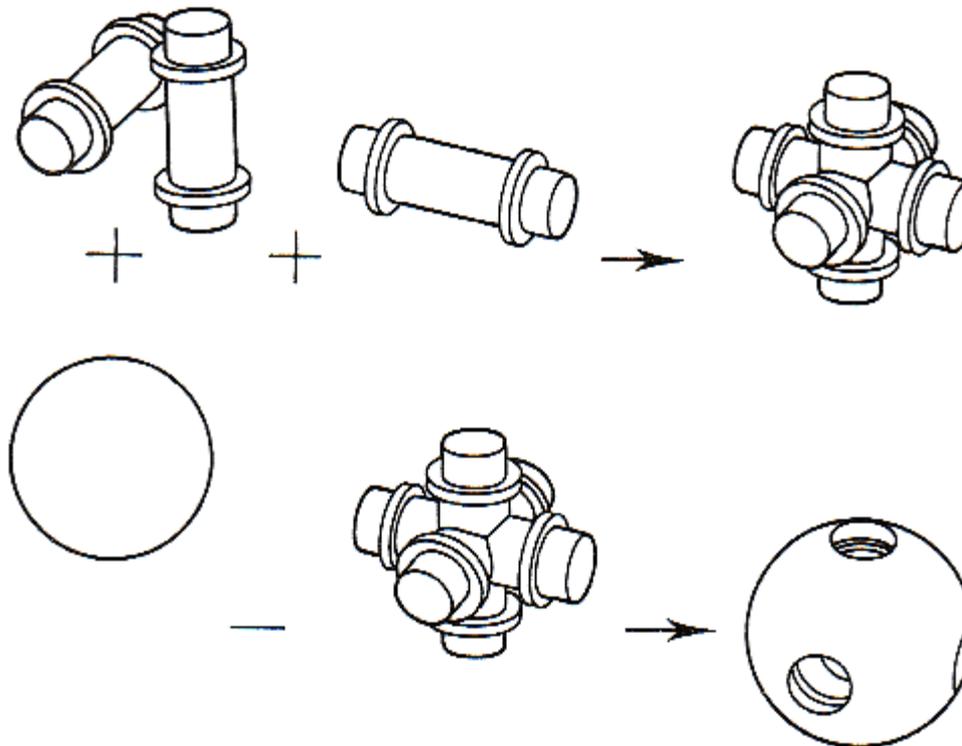
- ... anschließend **Schnitt** mit einem Objekt, das aus der Vereinigung eines Quaders mit einem weiteren Zylinder entstanden ist.

Bemerkung: Ein bestimmtes Objekt kann durch verschiedenste CSG-Operationen definiert werden.
HIER: Schritte 2 und 3 könnten auch vertauscht werden!



- Intern wird eine CSG-Repräsentation durch einen Baum verwaltet
 - Innere Knoten enthalten Information zu verwendenden booleschen Operator und die räumliche Beziehung zwischen ihren Kindern
 - Blätter des Baumes enthalten den Namen eines Primitivs und dessen Dimension/Größe

Beispiel: Konstruktion komplexer Objekte aus Primitiven



- Auch Space-Subdivision-Techniques
- Bei Repräsentation eines Objekts wird der Objektraum in **einzelne Elemente zerlegt**.
- Für jedes Element vermerken, ob der zugehörige Raum durch das Objekt **belegt ist oder nicht**.
- Standardverfahren
 - **Unterteilung des Raumes** durch ein festes, regelmäßiges Gitter in Zellen identischer Geometrie
 - Im 3D-Raum erhält man **würfelförmige Zellen**, die als Voxel (Volume Element) bezeichnet werden.
⇒ Analogie zu Pixel (Picture Element) im 2D

Vorteile

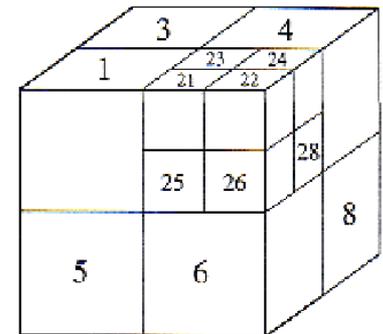
- Es ist sehr einfach zu bestimmen, ob ein gegebener Punkt **innerhalb oder außerhalb** eines Objekts liegt.
- Es ist einfach zu ermitteln, ob zwei Objekte **einander berühren**.
- Die Repräsentation eines gegebenen Objektes ist **eindeutig**.

Nachteile

- Es können **keine nur teilweise** belegten Zellen existieren.
- Objekte können im Allgemeinen **nur approximiert** werden.
- Für eine Auflösung von n Voxeln in jeder Dimension werden **n^3 Voxel benötigt**.
- Art der Objektrepräsentation ist extrem speicheraufwändig
⇒ günstigere Repräsentation durch **Octrees**

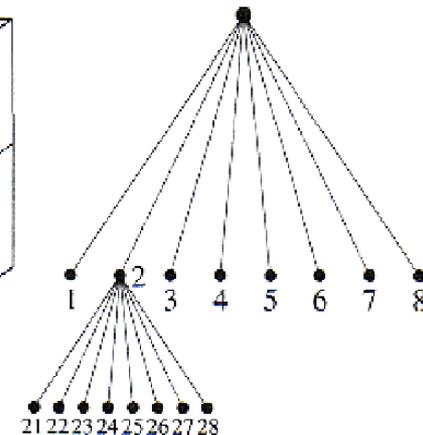
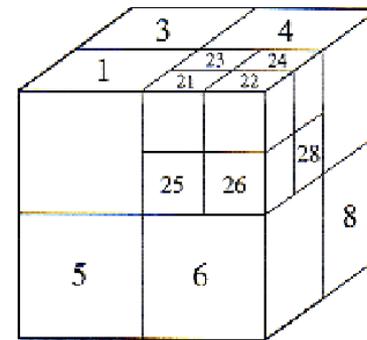
Octrees

- Ein Octree ist eine **hierarchische Datenstruktur** zur effizienten Speicherung einer ungleichmäßigen Unterteilung des 3D-Raums.
- **Prinzip**
 - Initiales Element ist ein Würfel, der den **gesamten Objektraum** umfasst und die Zustände belegt / nicht belegt annehmen kann.
 - Der Belegungszustand des Würfels wird bestimmt. Sollte er nur teilweise vom Objekt belegt sein, so wird er **entlang jeder Dimension halbiert** (Oktanten)
 - Auf jeden entstehenden (Teil-)Würfel wird dieses Verfahren **rekursiv** so lange angewandt, bis es stoppt oder die gewünschte Auflösung erreicht ist.



Octrees (Fortsetzung)

- In einem Octree besitzen alle inneren Knoten genau **acht direkte Nachfolger**.
- Die **Wurzel des Baumes** repräsentiert den initialen Würfel.
- Bei einer Unterteilung wird für jeden entstehenden Teilwürfel nach einem **festen Nummerierungsschema** ein neuer Knoten als Sohn eingefügt.
- Jedes Blatt **speichert Zustand** des zugehörigen (Teil-)Würfels.
- Jeder **innere Knoten** repräsentiert einen nur teilweise belegten Würfel.



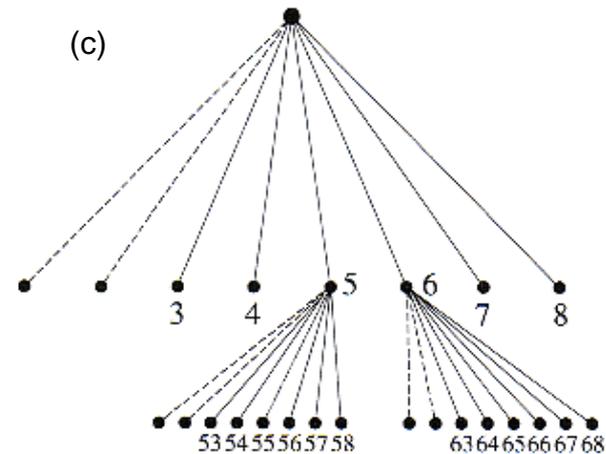
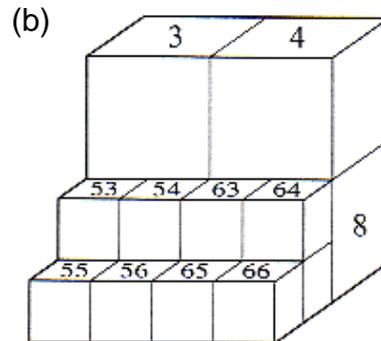
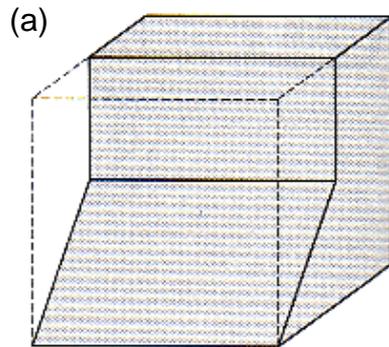
Octrees (Fortsetzung)

Beispiel: Repräsentation eines 3D-Objekts durch einen Octree

(a) Objekt, eingebettet in den initialen Würfel

(b) Repräsentation des Objektes bei maximal zweimaliger Unterteilung des Raumes

(c) zugehörige Octree-Datenstruktur



Octrees (Fortsetzung)

- Wesentlich häufiger als zur direkten Repräsentation von Objekten werden Octrees zur **räumlichen Unterteilung** der Objekte innerhalb einer Szene eingesetzt.
- Die einzelnen Objekte werden hierbei durch Standard-Datenstrukturen repräsentiert (zB. polygonal).
- Der Informationsgehalt der Zellen des Octrees wird von dem binären „belegt“-Zustand auf eine Liste von Objekten (bzw. Polygone, ...), die in der Zelle enthalten sind, erweitert.
- Dies führt zu einer wesentlichen **Beschleunigung** von Algorithmen, die lokal auf einzelnen Regionen des Raumes arbeiten (zB. Ray Tracing):
 - Leere Regionen können **übersprungen** werden

n-ary trees

- Das beim Octree realisierte Prinzip der Unterteilung des dreidimensionalen Raumes kann allgemein auch auf den **n-dimensionalen Raum** angewandt werden.
- Für den Fall $n=2$
 - I.A. Unterteilung der **Ebene**
 - Datenstruktur heißt **Quadtree**
 - Jeder innere Knoten des Baumes besitzt genau vier Kinder.
- Für den Fall $n=1$
 - **kD-Tree**
 - Elterknoten hat bis zu **zwei Kinder**
- Muss **keine feste** Unterteilungsstrategie repräsentieren

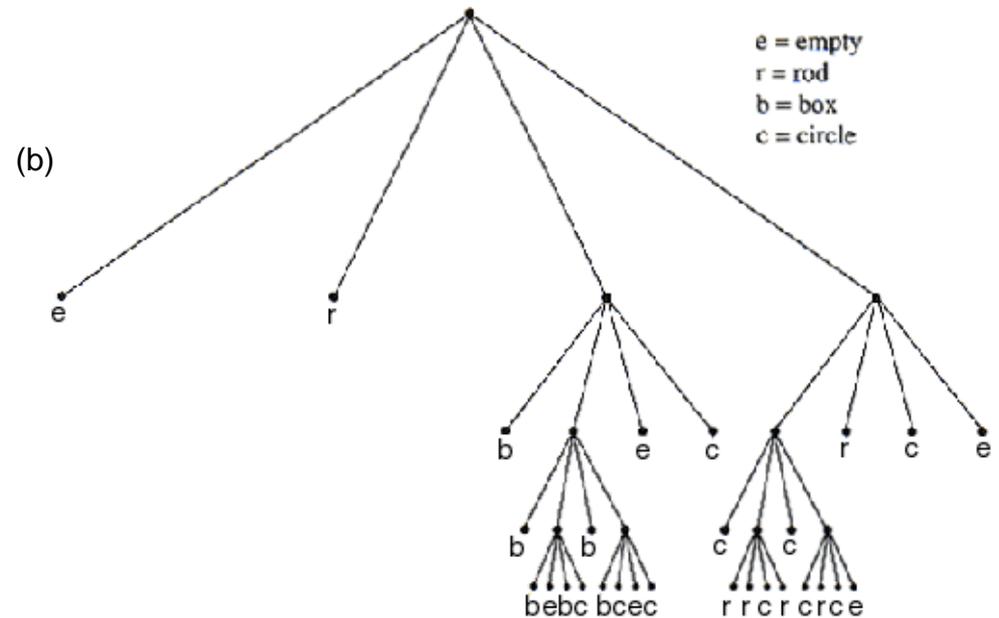
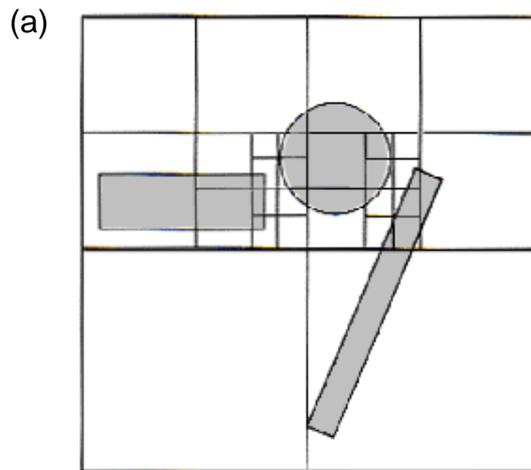
Quadrees

- I.A. Unterteilung der Ebene
- jeder innere Knoten des Baumes besitzt genau vier Kinder.
- Historisch gesehen sind Quadrees die **älteren Bäume**.
- Sie wurden bereits in den späten 60er Jahren des vorigen Jahrhunderts erstmalig verwendet.
- Octrees wurden von den Quadrees abgeleitet und kamen erst ab Ende der 70er, Anfang der 80er Jahre zum Einsatz.

Quadrees (Fortsetzung)

Beispiel: Unterteilung einer 2D-Szene durch einen Quadtree

- (a) Unterteilung des Raumes, bis die Zellen maximal eine Objektreferenz enthalten
- (b) zugehörige Quadtree-Datenstruktur, vier Generationen

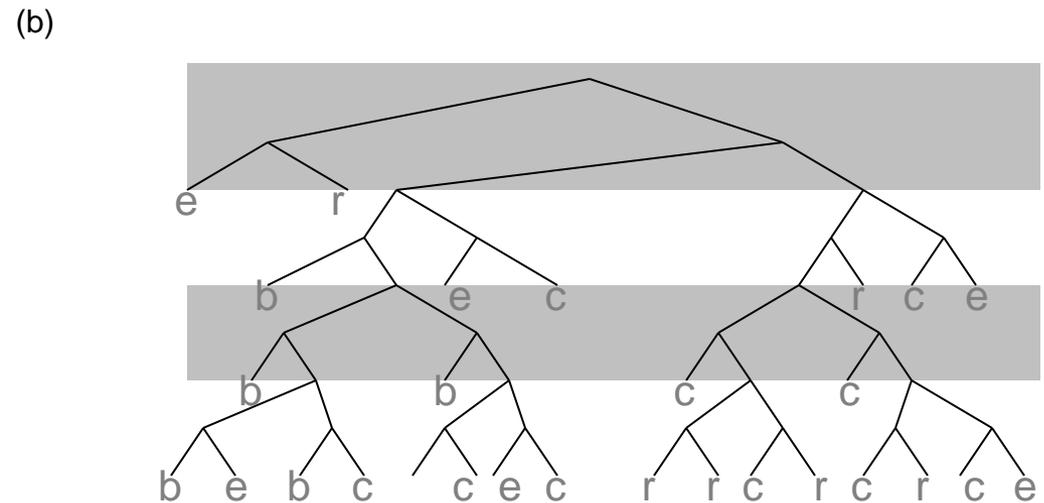
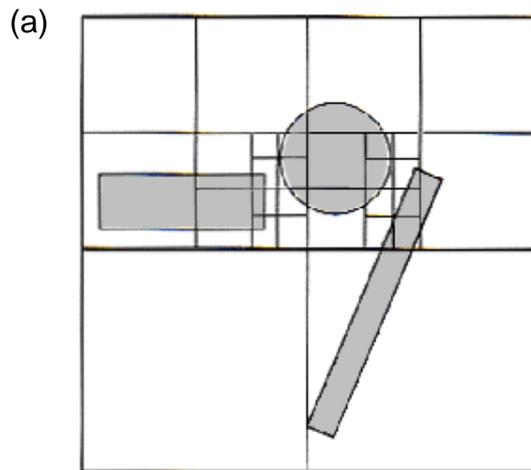


kD-Tree

Beispiel: Unterteilung einer 2D-Szene durch einen kD-Tree

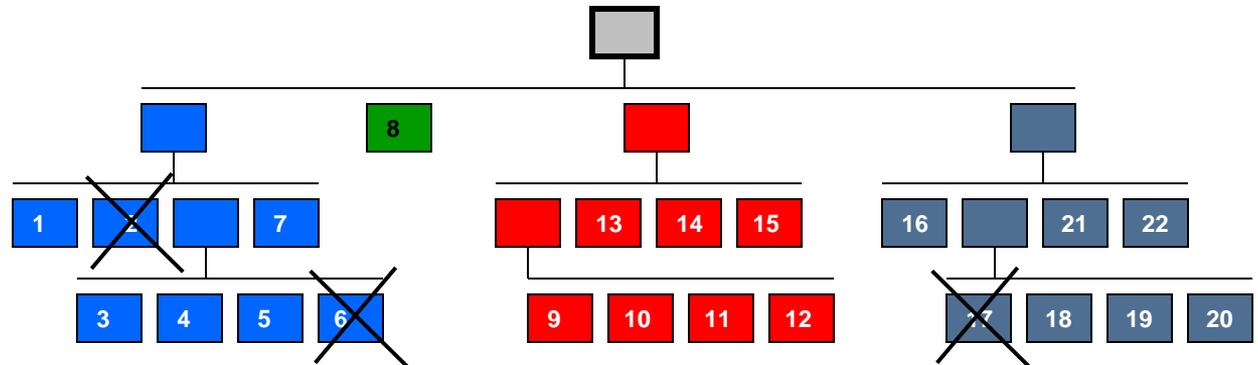
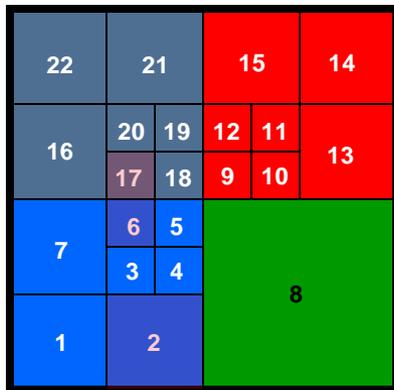
- (a) Unterteilung des Raumes, bis die Zellen maximal eine Objektreferenz enthalten
- (b) zugehörige kD-Tree-Datenstruktur, acht Generationen (Grau = Quadtree-Generationen)

e = empty
r = rod
b = box
c = circle



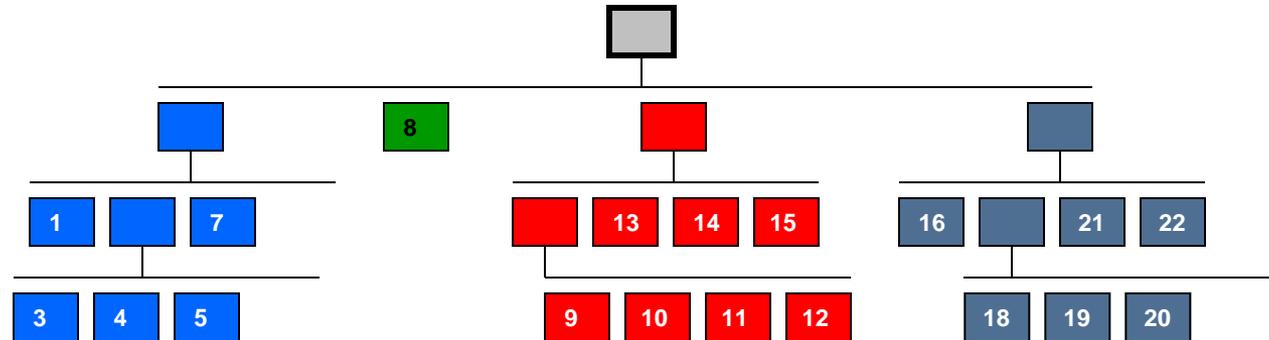
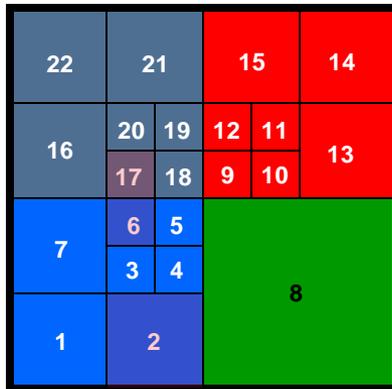
Lineare (Quad-)Trees

- **Reduzierter** (Quad-)Tree
- **Nicht alle Kinder** enthalten Information
- **Zeigerhierarchien** bis auf Voxeleneben **können sehr groß** werden
- Kompaktere Repräsentation durch **implizite Nummerierung**
- Weitere Einsparung: **nur belegte** Kinderknoten
 - Feste räumliche Zuordnung kann **verloren** gehen



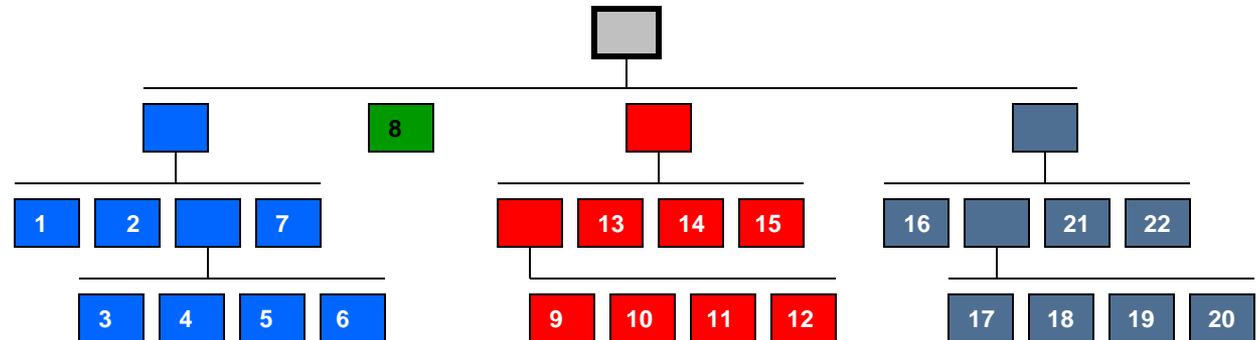
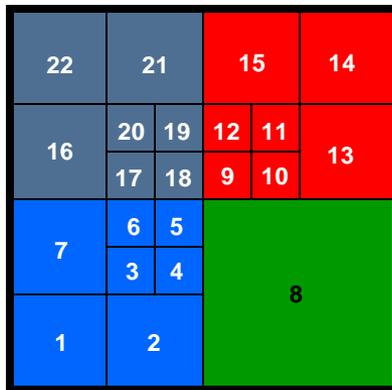
Lineare (Quad-)Trees

- **Reduzierter** (Quad-)Tree
- **Nicht alle Kinder** enthalten Information
- **Zeigerhierarchien** bis auf Voxeleneben **können sehr groß** werden
- Kompaktere Repräsentation durch **implizite Nummerierung**
- Weitere Einsparung: **nur belegte** Kinderknoten
 - Feste räumliche Zuordnung kann **verloren** gehen



Lineare (Quad-)Trees

- **Trennung** von Zeigern und Daten
- **Zeiger zeigen** auf Daten
- Ggf. **Umsortierung** entsprechend der sequentiellen Traversierungsreihenfolge zur Cache-Optimierung

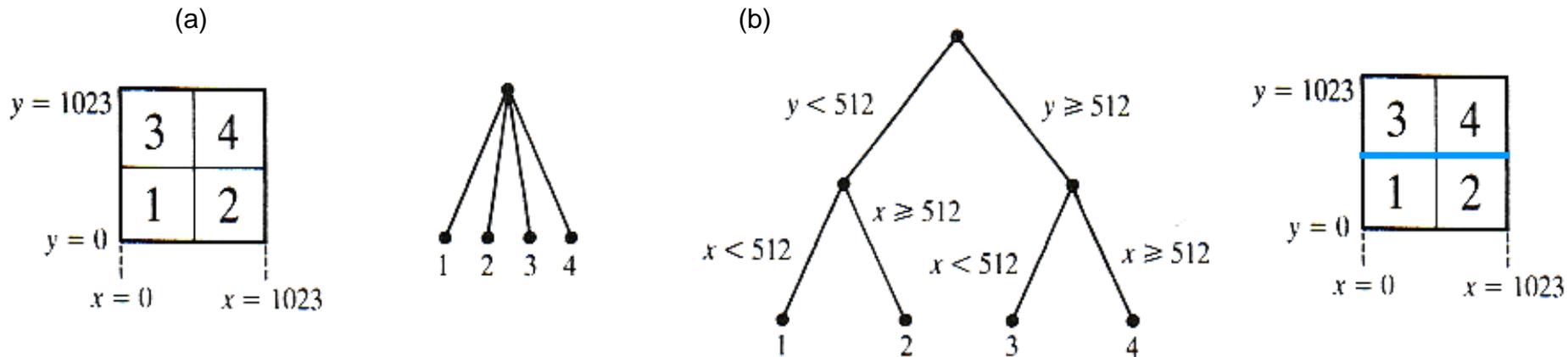


n-Bäume

Beispiel: Unterteilung einer 2D-Szene

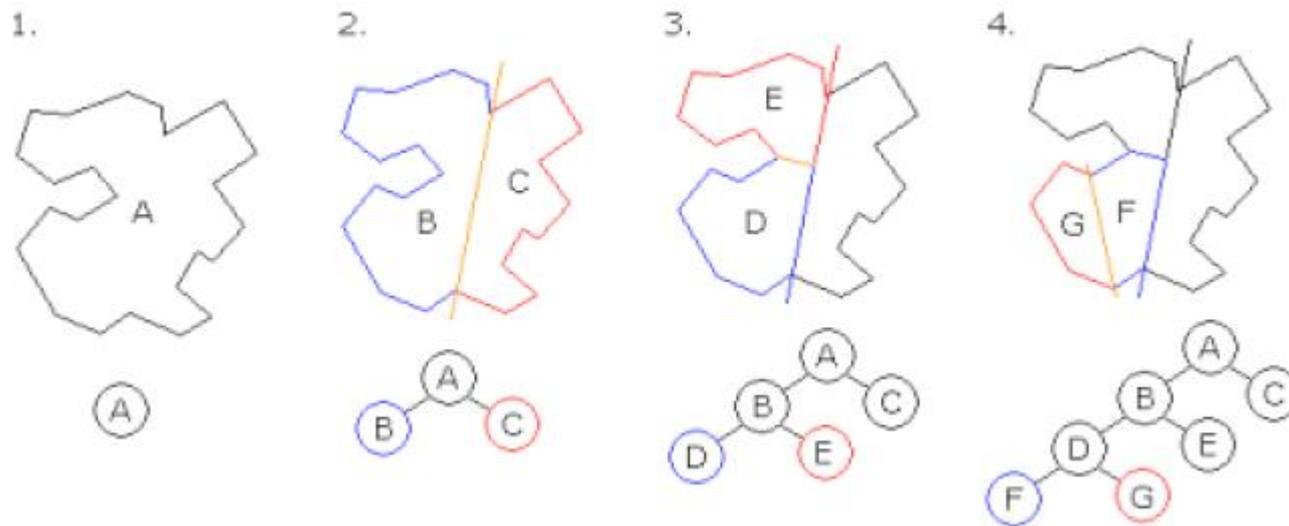
(a) durch einen Quadtree

(b) durch einen entsprechenden kD-Tree



Binary Space-Partitioning Trees (BSP-Bäume)

- Octrees, Quadtrees und kD-Trees unterteilen rekursiv für jede Stufe den Raum **wechselseitig senkrecht** zueinander in allen Dimensionen.
- Statt einer festen orthogonalen Unterteilung teilt das BSP-Verfahren den Raum rekursiv mittels einer **beliebigen (Hyper-) Ebene** in zwei Unterräume



[www.beyond3d.com]

Binary Space-Partitioning Trees (BSP-Bäume)

- Rekursive Unterteilung mittels einer **beliebigen (Hyper-)Ebene**
 - Definiert man einen Unterraum als „Innen“, den anderen als „Außen“, so können konvexe Polyeder durch geeignet orientierte, das Volumen begrenzende Ebenen modelliert werden.
 - Durch Vereinigung konvexer „Innen“-Regionen können beliebige konkave Polyeder mit Löchern definiert werden.

BSP-Bäume (Fortsetzung)

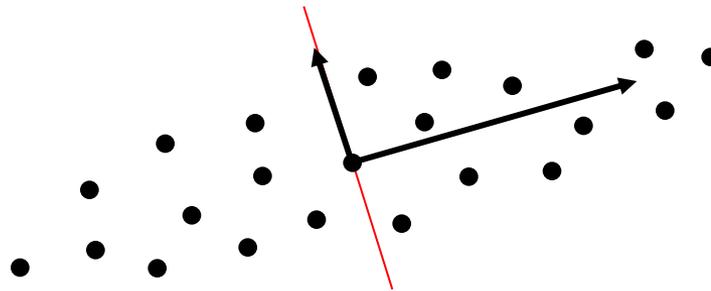
- BSP-Bäume können wie Octrees und Quadtrees zur Unterteilung von Szenen verwendet werden (siehe folgendes Beispiel).
- Dabei sind sie jedoch an **kein Raster** gebunden!
- Der Raum soll rekursiv durch Ebenen so unterteilt werden, dass in jeder Region **höchstens ein Objekt** enthalten ist.
- Über die **relative Lage der Regionen** zum Betrachter kann die Tiefenstaffelung der Objekte bestimmt werden.
(Welche Objekte können überhaupt sichtbar / verdeckt sein?)

Aufteilungsbäume

- **Dimensionalität ist Tradeoff** zwischen Anzahl der Kinder (Tests auf einer Ebene) und Hierarchietiefe (Absteigetiefe)
- iA. orientiert sich Dimensionalität an **Ausdehnungsdimension**
 - Raum (3D) \Rightarrow Octree
 - Fläche/Ebene (2D) \Rightarrow Quadtree
 - Richtung (1D) \Rightarrow kD-Tree, BSP-Tree
- Solange **feste KindID / Lokalisierungsinformation** nicht genutzt wird, kann #Kinder beliebig sein
- **Balancierte** Bäume: Gleiche Anzahl von Knoten/Blättern in den jeweiligen Verzweigungen

Principal Component Analysis (PCA)

- Eine **ideale Wahl** der Unterteilungsebene für BSP-Bäume liefert die PCA.
- **Angenommen**, eine komplexe Szene ist gegeben durch die Punktwolke $P_i \in \mathbb{R}^2$ ($i=1, \dots, n$) (zB. Objektmittelpunkte oder Eckpunkte der Polygone).
- PCA liefert ein **orthogonales Koordinatensystem** e_1, e_2, e_3 , dessen Ausrichtung der Punktwolke entspricht.



Principal Component Analysis (Fortsetzung)

- Wähle den Mittelwert c als Ursprung:
$$c = \frac{1}{n} \sum_{i=1}^n P_i$$

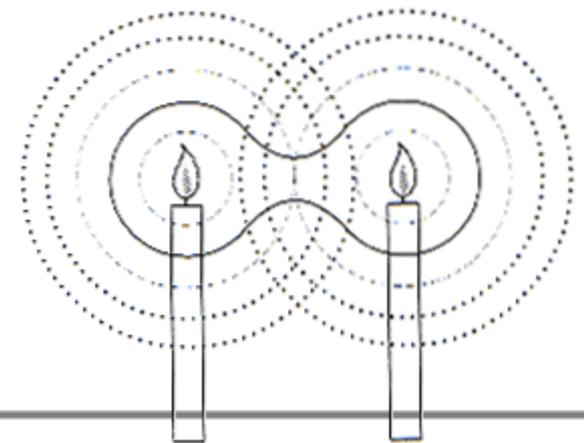
$$A = \begin{pmatrix} P_{1x} - c_x & P_{1y} - c_y & P_{1z} - c_z \\ P_{2x} - c_x & P_{2y} - c_y & P_{2z} - c_z \\ \vdots & \vdots & \vdots \\ P_{nx} - c_x & P_{ny} - c_y & P_{nz} - c_z \end{pmatrix}$$

$$B = \frac{1}{n-1} A^T A, \quad b_{ij} = \frac{1}{n-1} \sum_{k=1}^n a_{ki} a_{kj}$$

- B hat reelle Eigenwerte $\lambda_1, \lambda_2, \lambda_3$ und Eigenvektoren e_1, e_2, e_3 , dh. $\lambda_i \cdot e_i = B \cdot e_i$.
- Die Eigenvektoren mit dem Ursprung c bilden das gesuchte System.
- Die Ausdehnung der Punktwolke in Richtung e_i ist proportional zu $\sqrt{\lambda_i}$. (Analog für beliebige Dimension.)

Idee

- Beschreibung von Objekt-Flächen bzw. -Volumina als **Isoflächen in Skalarfeldern**.
- Die Skalarfelder ihrerseits entstehen kontrolliert durch **erzeugende Primitive** (Funktionen).
- Beispiel:
 - Punkthitzequellen erzeugen isoliert voneinander **sphärische Feldfunktionen**.
 - Durch Addieren beider Felder entsteht ein globales Skalarfeld.



Wir betrachten nun beispielhaft sogenannte diskrete „Blobs“

- Ausgangspunkt zur Definition eines Blobs ist eine Feldfunktion $F_d: \mathbb{R}^3 \rightarrow \mathbb{R}$ um einen Mittelpunkt $P \in \mathbb{R}_3$.
- F_d ist hierbei im wesentlichen die Komposition
 - einer **monoton fallenden Funktion** - der sogenannten Einflussfunktion
 - und eines **Abstandsmaßes** (hier euklidisches Maß) zwischen der freien Variablen $x \in \mathbb{R}^3$ und dem Mittelpunkt P .
- Der Index d deutet an, dass die Feldfunktion um einen **diskreten Mittelpunkt** definiert ist.
- Als eigentlichen Blob bezeichnet man in diesem Zusammenhang diejenige **Fläche**, die bei einem **vorgegebenen Isowert** vom Skalarfeld von F_d implizit definiert wird.

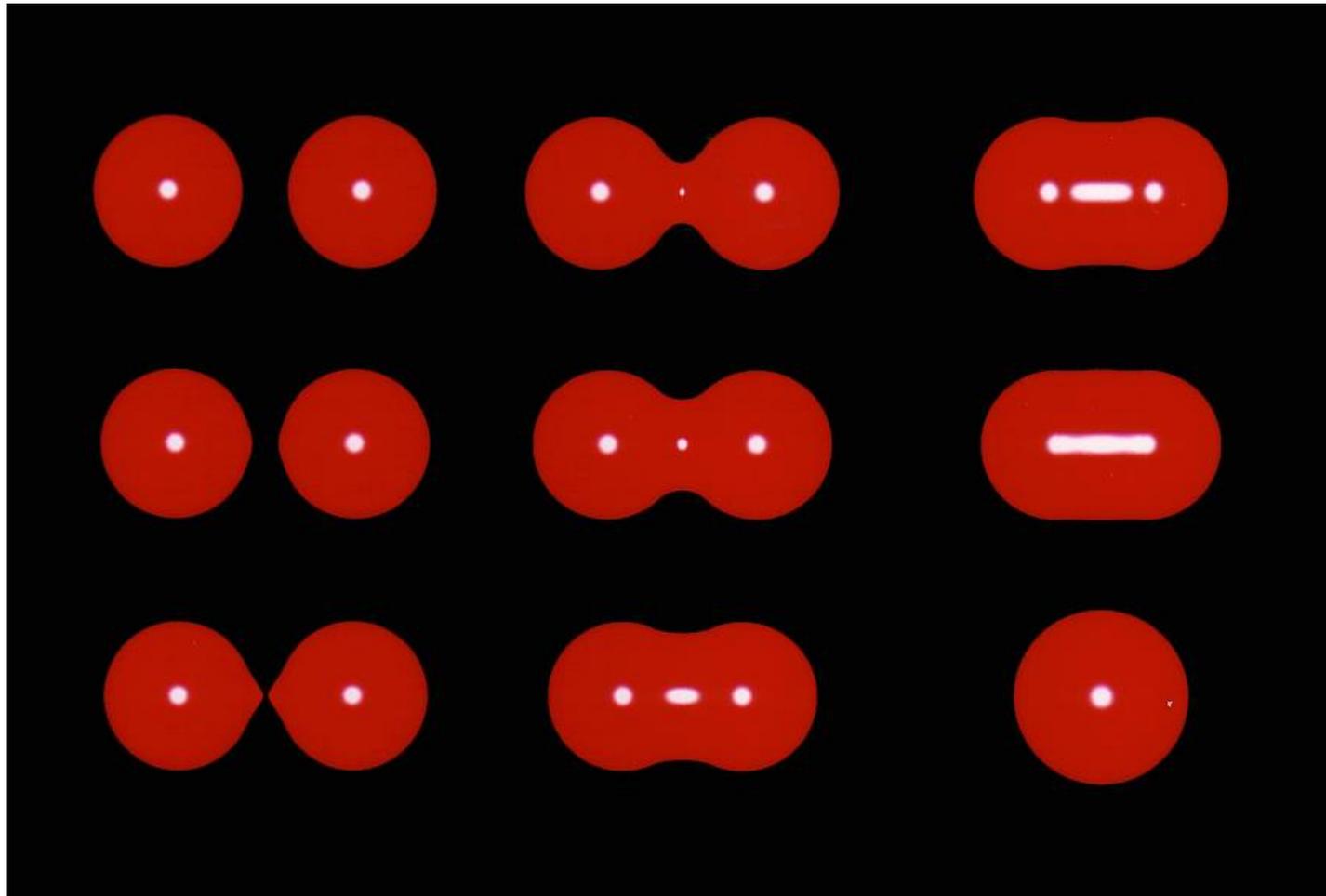
- Existieren mehrere Blobs mit Feldfunktionen $F_d(i; \mathbf{x}): \mathbb{R}^3 \rightarrow \mathbb{R}$ um Mittelpunkte $P_i \in \mathbb{R}^3$ ($i=1, \dots, n$), so überlagern sich ihre Einflüsse nach dem Superpositionsprinzip
- Die resultierende Feldfunktion des entstehenden Skalarfeldes ergibt sich dann einfach als Summe der einzelnen Feldfunktionen:

$$F : \mathbb{R}^3 \rightarrow \mathbb{R} \quad F(\mathbf{x}) = \sum_{i=1}^n F_d(i; \mathbf{x})$$

- Ist c eine vorgegebene Konstante kleiner als der maximal im Skalarfeld vorkommende Wert, so definiert die Menge aller $\mathbf{x} \in \mathbb{R}^3$ mit $F(\mathbf{x}) = c$ eine **Isofläche S** vom Niveau c .
- Die Fläche S wird durch diese Gleichung implizit definiert. **c wird auch Isowert genannt**, S auch **Niveaufläche**.

- Das **Erscheinungsbild** der zugehörigen Isofläche ist bei sinnvoller Anordnung der Mittelpunkte und Benutzung einfacher Feldfunktionen **gut kontrollierbar**.
- Die nachfolgende Abbildung zeigt Isoflächen, die von jeweils **zwei radialsymmetrischen Feldfunktionen** erzeugt werden.
 - Zugehörigen Mittelpunkte werden immer weiter **aufeinander zu bewegt** und schließlich zur Deckung gebracht.
 - **Verschmelzungseffekt: separierten Isoflächen gehen glatt** (in diesem Fall C^1 -stetig) **ineinander über**, falls der Abstand der Mittelpunkte klein genug wird.
- **Zerreißeffect** : umgekehrter Reihenfolge

3.5 Implizite Beschreibungen



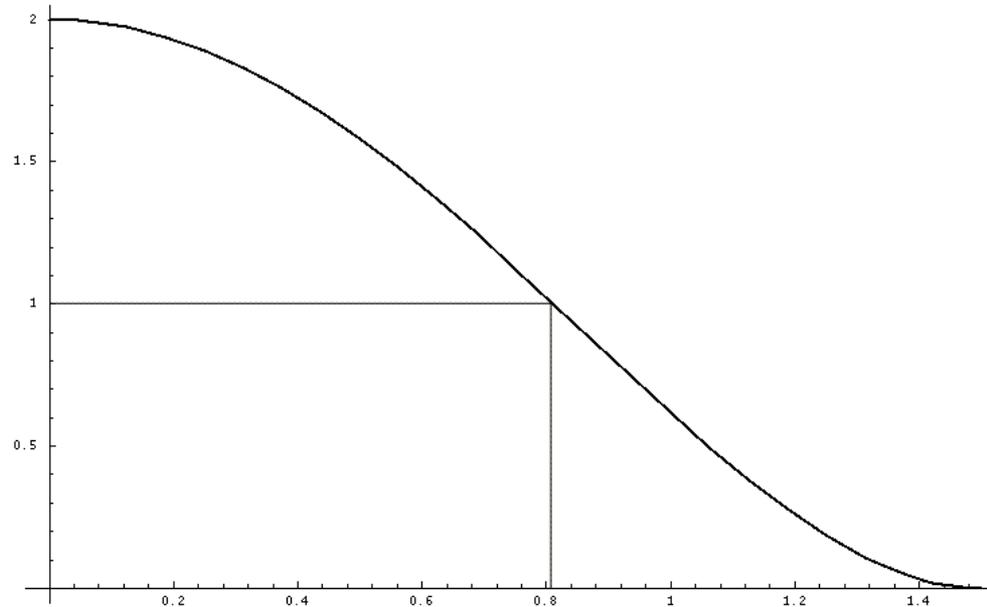
Einflussfunktion

Die häufig verwendeten Einflussfunktionen $f: \mathbb{R}_0^+ \rightarrow [0, a]$ mit $a \in \mathbb{R}$, $a > 0$ sind festgelegt durch

- Für $t \in [0, b]$ ($b \in \mathbb{R}$, $b > 0$) ist f ein gerades Polynom
- $f(t) = 0$ für $t > b$, b ist der maximale Einflussradius
- $f(0) = a$, $f(b) = 0$
- $f'(0) = 0$, $f'(b) = 0$
- f ist monoton fallend auf $[0, b]$

Folgende Abbildung zeigt den **Graph** einer so definierten **Einflussfunktion**

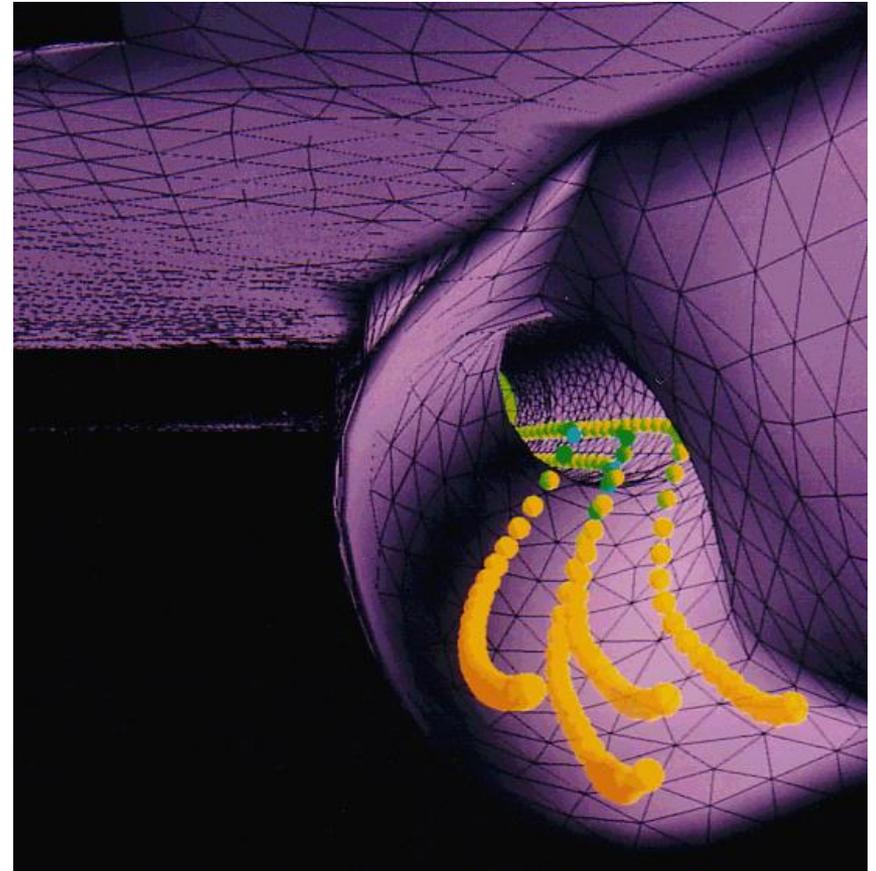
Einflussfunktion (Fortsetzung)



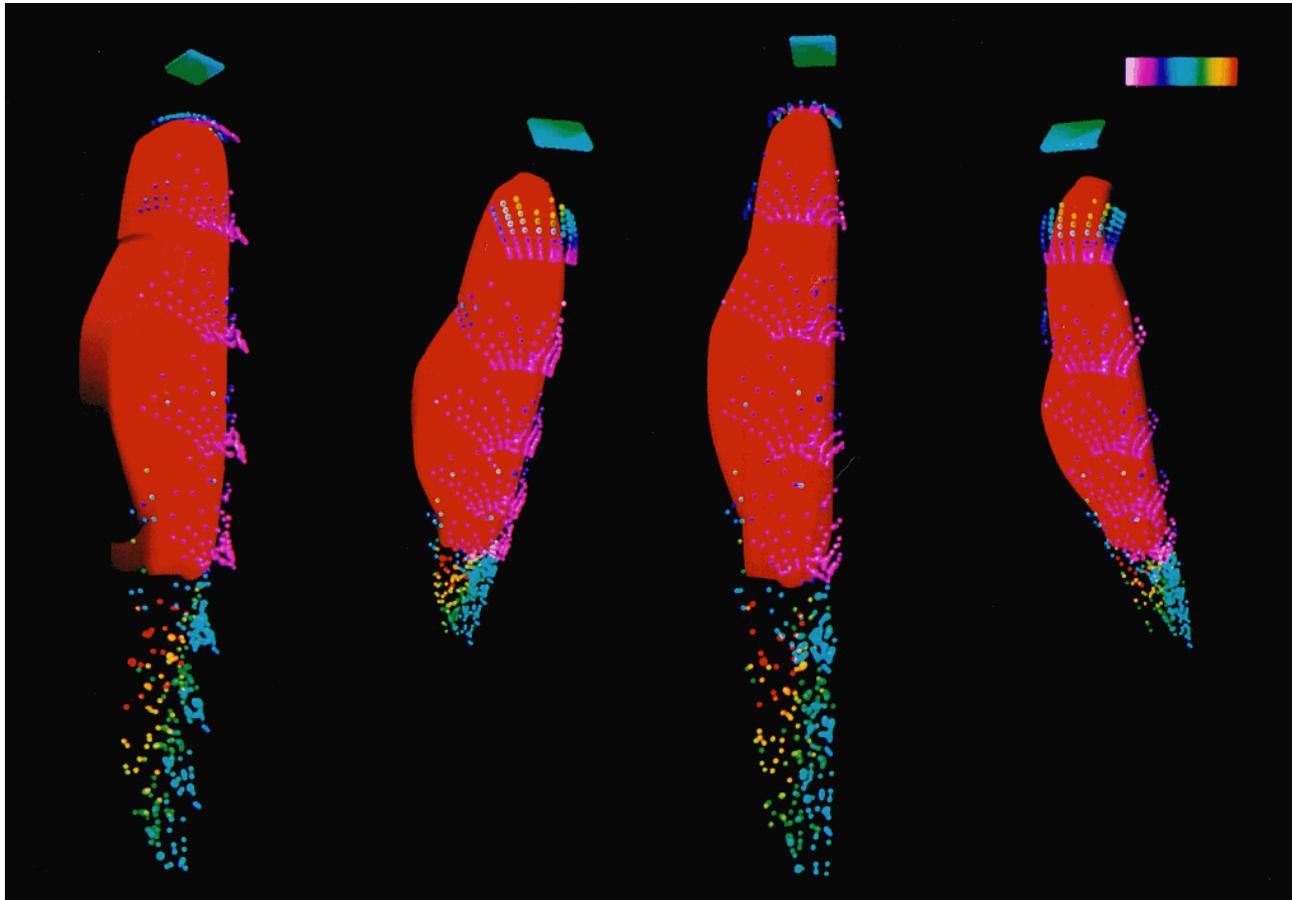
- Damit lautet schließlich die radialsymmetrische Feldfunktion eines diskreten Blobs im Punkt P:

$$F_d: \mathbb{R}^3 \rightarrow \mathbb{R} \text{ mit } F_d(x) = f(\|x-P\|^2)$$

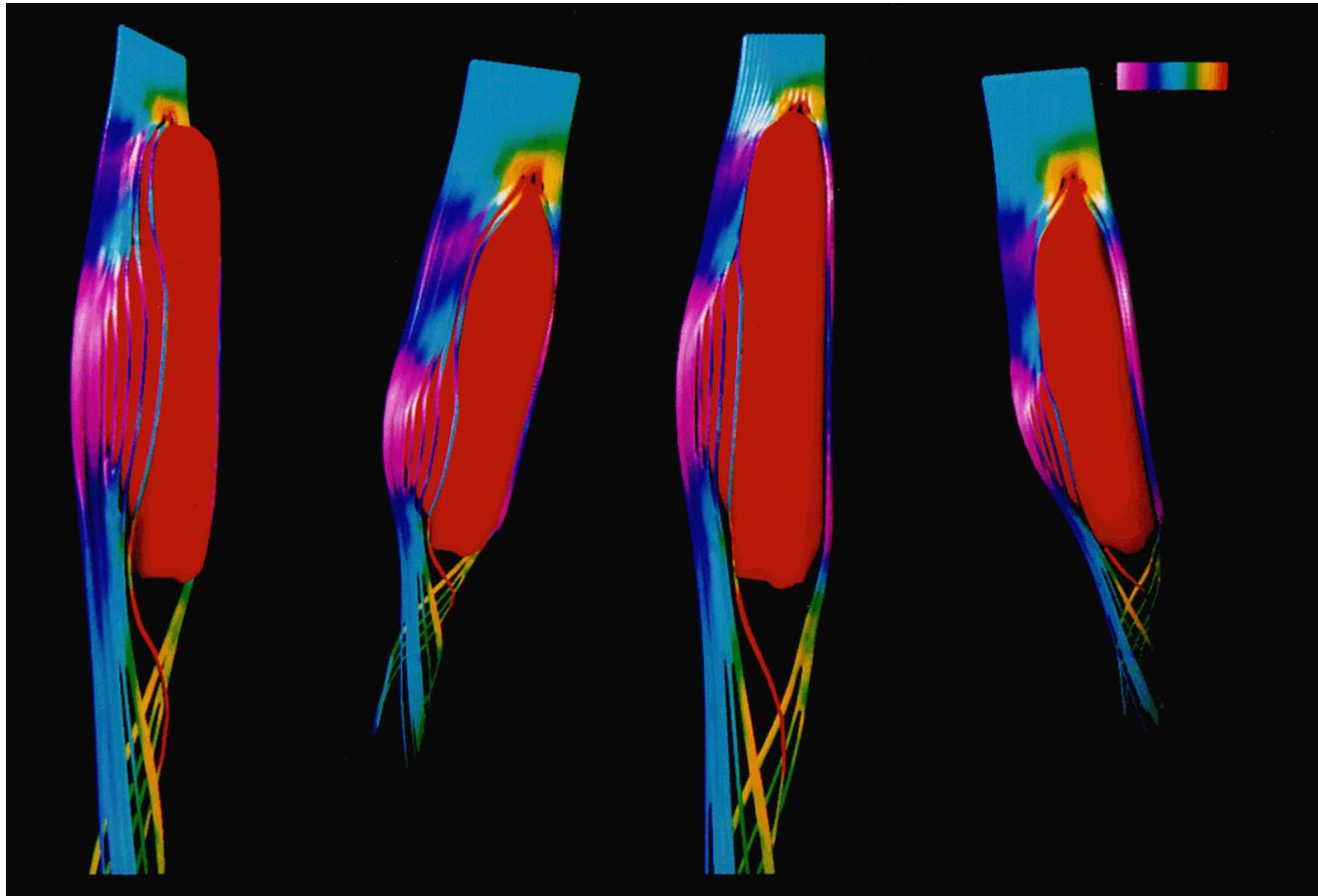
Anwendung: Strömungssimulation



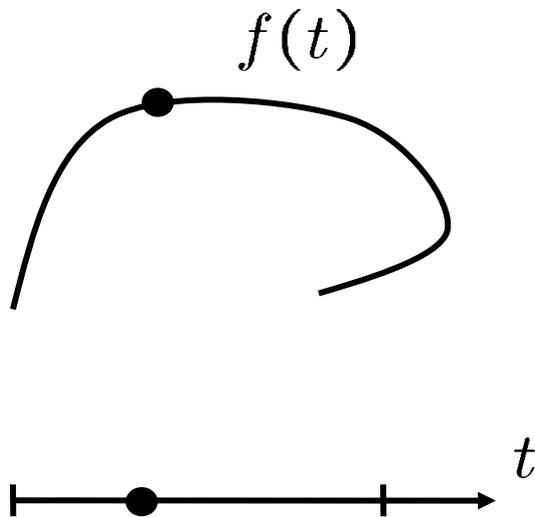
Anwendungen: Strömungssimulation



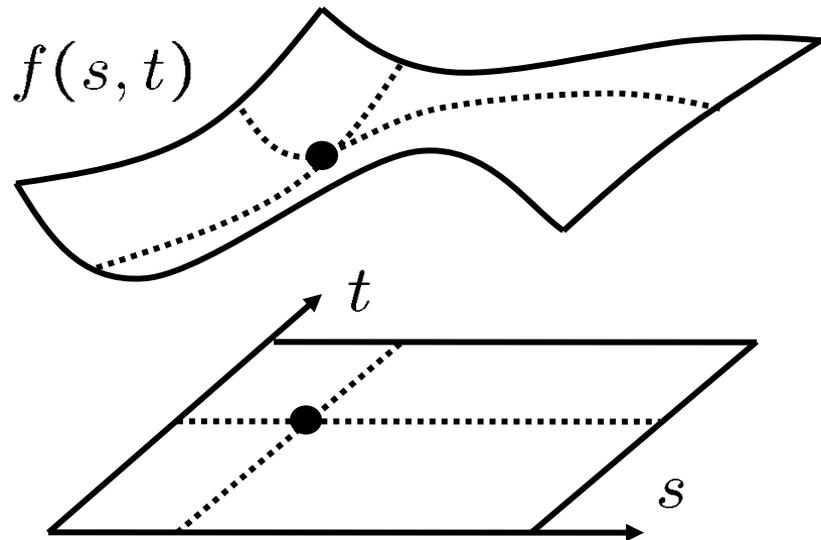
Anwendungen: Strömungssimulation



- **Parametrische Kurven und Flächen** werden neben der polygonalen Darstellung am häufigsten eingesetzt (Freiformmodellierung => Kap. 7).



Parameter-Kurve



Parameter-Fläche

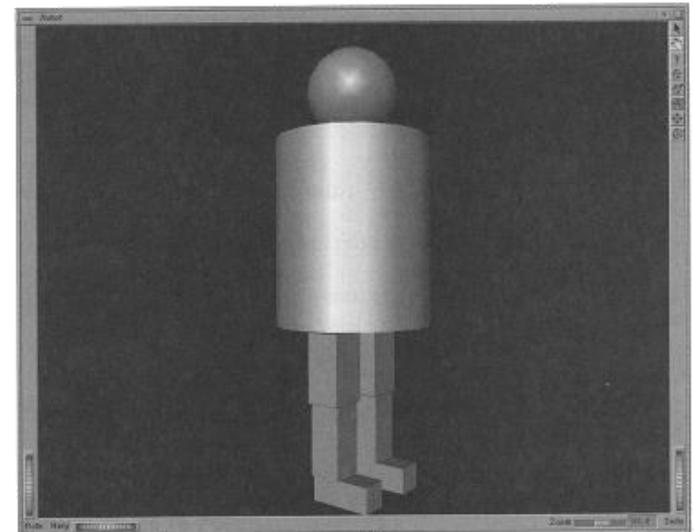
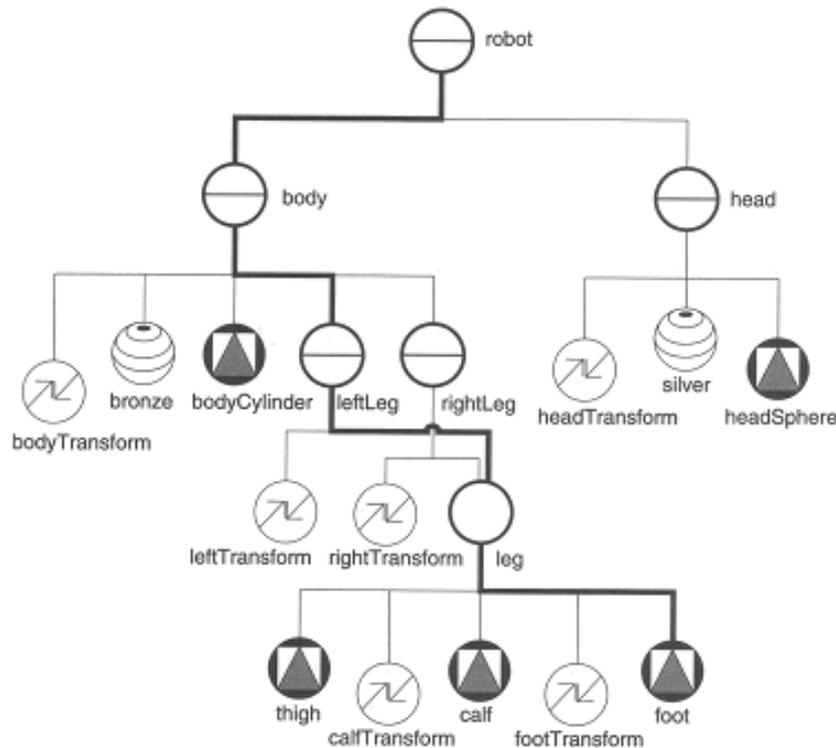
- Mit steigenden Anforderungen durch hochqualitative Realzeit-Computergraphik wird die effiziente **Organisation und Verwaltung** der Szene - das **Szenenmanagement** - immer bedeutsamer
 - Vor allem in Anwendungsbereichen Spieleindustrie und Virtual Reality
- Zugleich erfolgt das **eigentliche Rendering** einzelner Objekte in der **Hardware** (GPUs).
- Zusammenfassung und Organisation der Objekte in Szenen mit **hierarchischen Baumstrukturen** (Szenengraphen)
- Anwendung verschiedener Techniken
 - Level-of-Detail-Strukturen (LOD)
 - Verdeckungsrechnung
 - Texturparametrisierung

Szenengraph

- **Hierarchische Baumstruktur** zur Speicherung einer Szene mit ihren Modellen
- Praktisch alle heutigen Graphiksprachen bieten eine Organisation über eine Szenengraph-Struktur an
- Im Einzelnen kann der Szenengraph enthalten
 - Shapes (Formen, Geometrie, Erscheinungsbild)
 - Gruppierungen
 - Transformationen
 - Lichtquellen, Hintergrund, Nebel,...
 - Sichtdefinition
 - Verhalten (zB. bei Animationen)
 - anwendungsspezifische Attribute, akustische Ausgabe

Szenengraph (Fortsetzung)

Beispiel: Open Inventor-Szenengraph



— Path

LOD (Level of Detail) - Multiresolutionrepräsentation

Motivation: Allgemein tendieren Verfahren zur Erzeugung polygonaler Modelle dazu, „**zu viele**“ **Polygone** zu produzieren.

Problem

- In den überwiegenden Fällen ist das **Verhältnis (Polygonanzahl des Objektes) / (projizierte Fläche des Objekts)** viel zu groß.
- **Overhead** bei der Speicherung, Übertragung, Bearbeitung und Visualisierung „unnötiger“ Polygone

Lösung

- **Verschiedene polygonalen Auflösungen** der Objektrepräsentation - Level of Detail (LOD).
- Diese werden als sogenannte „Detail Pyramid“ / Multiresolutionrepräsentation verwaltet.

LOD (Level of Detail) - Multiresolutionrepräsentation

- Statische LODs: Feste, vordefinierte Auflösungsstufen
- Progressive LODs: Vordefinierte Einzeloperationen
- Dynamische LODs: Online Berechnung
- Blickpunktabhängige LODs: IdR. Variante der dynamische LODs

LOD (Fortsetzung)

Forschungsaspekte

- Mesh Simplification
Reduzierung der Polygone auf eine Anzahl, die gerade für die aktuelle **Qualitätsanforderung** ausreicht
- Level of Detail-Approximation
Vermeidung von **Popping**, dh. visuellen Sprüngen beim Umschalten zwischen verschiedenen Detaillierungsgraden
⇒ Geomorphs (sanfte visuelle Übergänge)
- Progressive Transmission
3D-Äquivalent zur progressiven Übertragung verschiedener **Detaillierungsgrade** bei 2D-Bitmap-Bildern

LOD (Fortsetzung)

Forschungsaspekte

- Mesh Compression

Minimierung des Speicherplatzes der Koordinaten / Topologie mittels Kompressionsverfahren (Transformation, Quantisierung und Kodierung der Koeffizienten)

- Selective Refinement

Dynamische kontextabhängige LOD-Technik zB.: Pilot fliegt mit Flugzeug über eine Landschaft. Diese muss nur dort detailliert dargestellt werden, wo sich das Flugzeug befindet / wo der Pilot hinschaut.

- Computergraphik, Universität Leipzig
(Prof. D. Bartz)
- Graphische Datenverarbeitung I, Universität Tübingen
(Prof. W. Straßer)
- Graphische Datenverarbeitung I, TU Darmstadt
(Prof. M. Alexa)