

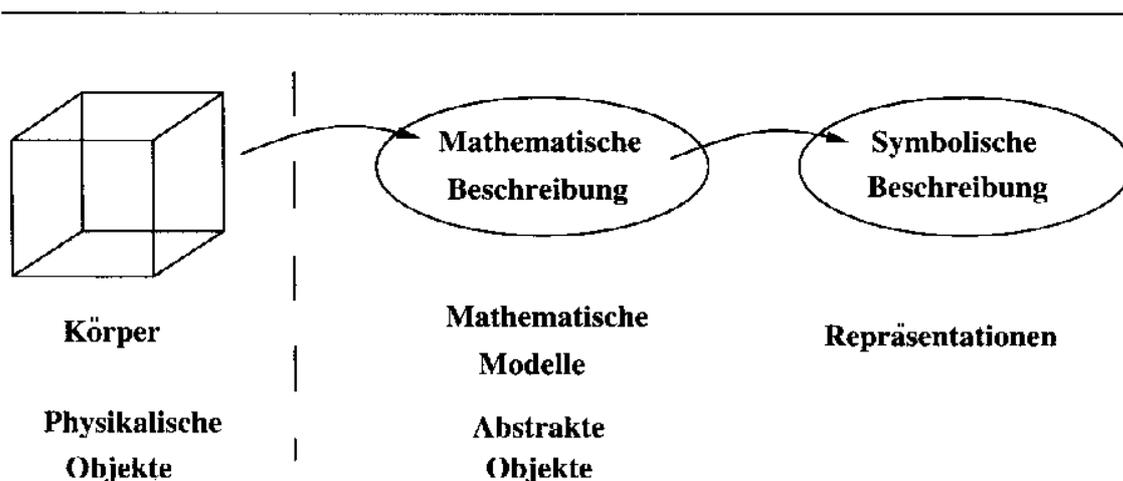
## 8. Modelle für feste Körper

*Modell*: Abbild der Realität, welches bestimmte Aspekte der Realität repräsentiert (und andere ausblendet)

- mathematische Modelle
- symbolische Modelle
- Datenmodelle
- Experimentalmodelle

*Solid Modelling* (Festkörpermodellierung):  
mathematische / datenstruktur-orientierte Darstellung  
geschlossener 3D-Körper

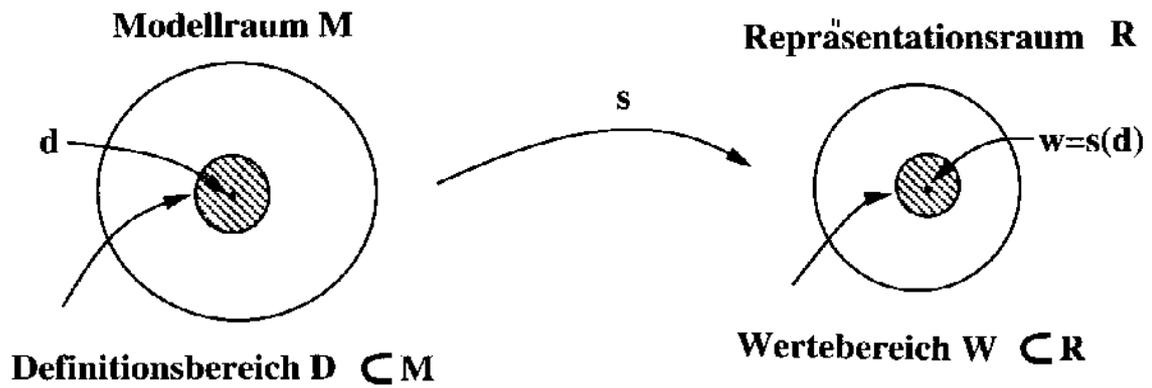
2-stufiger Prozess:



(aus Encarnação et al. 1997)

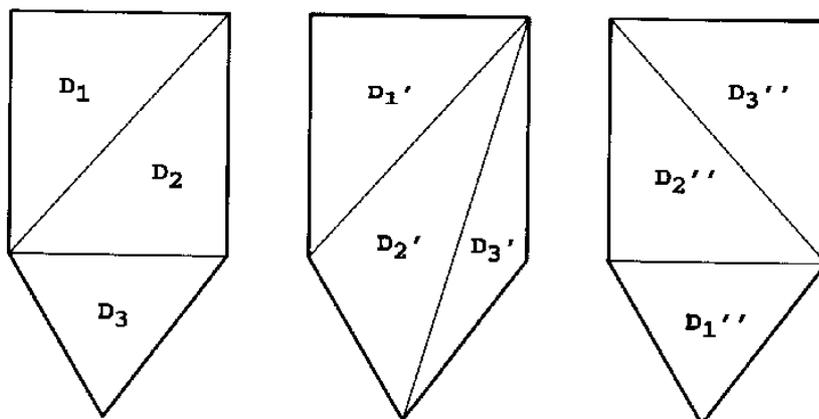
mathematischer Modellraum  $M$   
symbolischer Repräsentationsraum  $R$

Repräsentations-Relation  $s: M \rightarrow R$   
("Repräsentations-Schema")



*s* **eindeutig**: jedes Modell aus  $D$  besitzt genau eine Darstellung in  $W$  (d.h.  $s$  ist Funktion).

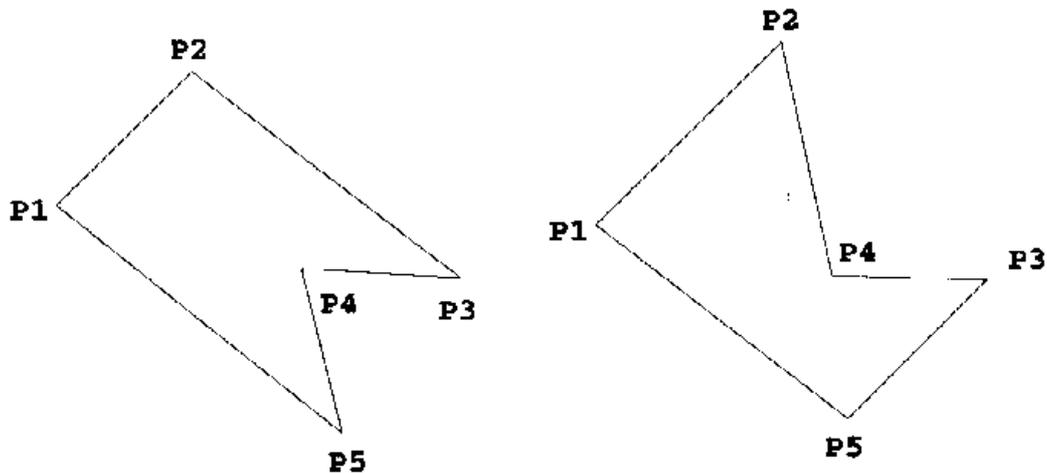
Beispiel einer nicht eindeutigen Repr.:  
Darstellung von Polygonen als endliche Vereinigung von Dreiecken



*s* **vollständig**: die Relation ist injektiv, d.h. durch jede gültige Repräsentation wird genau ein Modell aus  $D$  dargestellt.

(wenn das nicht gilt, tritt beim Übergang zur symbolischen Darstellung Informationsverlust ein.)

Beispiel einer nicht vollständigen Repräsentation:  
 Darstellung eines nicht-konvexen Polygons durch die Menge  
 seiner Eckpunkte

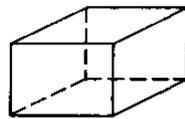
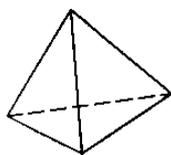


Kriterien zur Beurteilung von Geometrie-Repräsentations-  
 Schemata:

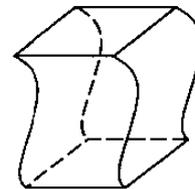
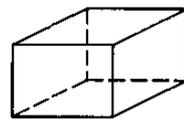
- Mächtigkeit (wie groß ist der Definitionsbereich? Wie genau können komplizierte Objekte modelliert werden?)
- Eindeutigkeit
- Vollständigkeit
- stimmt der Repräsentationsraum  $R$  mit dem Wertebereich  $W$  von  $s$  überein (oder gibt es "ungültige" symbolische Darstellungen)?
- Genauigkeit (Approximation von Objekten)
- Effizienz (Zeitaufwand für die Darstellung, Speicherplatz)
- Abgeschlossenheit (gegenüber Transformationen, Mengenoperationen)
- wie allgemein sind Operationen, die zur Manipulation von Repräsentationen verwendet werden müssen?
- formale Eleganz der Darstellung (kommt man mit wenigen Operatoren aus, gehorchen diese einfachen Gesetzen?)
- wie kompliziert sind Algorithmen zur Erzeugung und Manipulation von Objekten?

## Mathematische Grundlagen:

- Geometrie (Längen, Winkel; *Kongruenzabbildungen*)
- Topologie (Umgebungen, Inzidenzbeziehungen: welches Element hängt mit welchem zusammen? Abbildungen: *Homöomorphismen*)
- Graphentheorie
- Kombinatorik
- algebraische Strukturen
- Maßtheorie



verschiedene Topologie  
verschiedene Geometrie



gleiche Topologie  
verschiedene Geometrie

## Topologische Grundbegriffe:

Sei  $A$  eine Menge von Punkten, also eine Teilmenge von  $\mathbb{R}^3$ .

$A$  heißt *beschränkt*, wenn  $A$  ganz in einer Kugel mit endlichem Radius enthalten ist.

$A$  heißt *Umgebung* eines Punktes  $x$ , wenn es eine kleine Kugel mit Mittelpunkt  $x$  gibt, die ganz in  $A$  enthalten ist. (Die Kugeln bilden eine *Umgebungsbasis* der Standard-Topologie des  $\mathbb{R}^3$ .)

$x \in A$  heißt *innerer Punkt* von  $A$  genau dann, wenn es eine Umgebung von  $x$  gibt, die ganz in  $A$  liegt (gleichwertig: wenn es eine Kugel mit Mittelpunkt  $x$  gibt, die ganz in  $A$  liegt).

$x$  heißt *Randpunkt* von  $A$ , wenn jede Umgebung von  $x$  Punkte von  $A$  und des Komplements von  $A$  enthält. (Dabei ist auch der Fall  $x \notin A$  möglich.)

Der *Rand* von  $A$ , bezeichnet als  $\delta A$ , ist die Menge aller Randpunkte von  $A$ .

Das *Innere* von  $A$  ist def. als  $\overset{\circ}{A} = A - \delta A$  (Mengendifferenz).

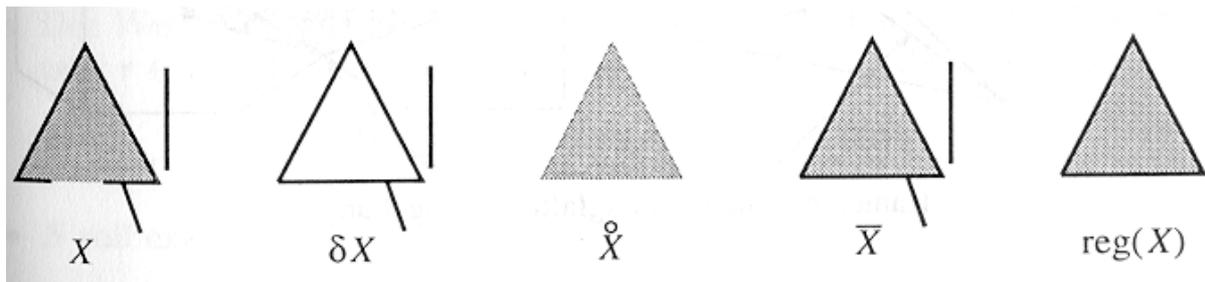
$A$  heißt *offen*, wenn  $\overset{\circ}{A} = A$ .

Der *Abschluss* von  $A$  (auch "abgeschlossene Hülle von  $A$ ") ist def. als  $\bar{A} = A \cup \delta A$  (auch:  $\text{hull}(A)$ , Hülle( $A$ )).

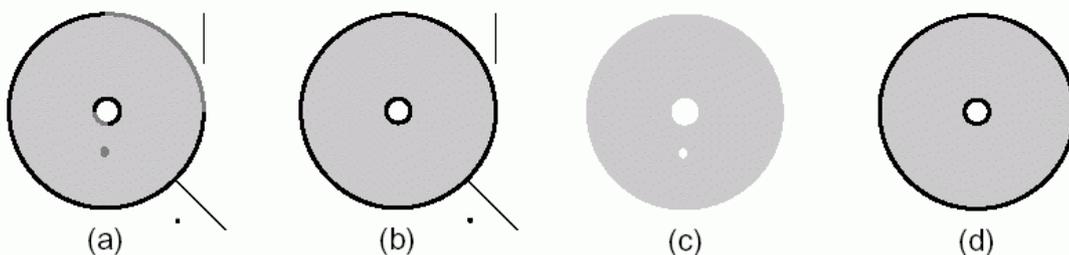
$A$  heißt *abgeschlossen*, wenn  $\bar{A} = A$ .

Die *Regularisierung* von  $A$  ist die Menge  $\text{reg}(A) = \text{Hülle}(\overset{\circ}{A})$ .

$A$  heißt *regulär*, wenn  $\text{reg}(A) = A$ .



Effekt der Regularisierung: das Objekt wird abgeschlossen, "Löcher" und "hängende Teile" niedriger Dimension werden entfernt:



(a) Ausgangsmenge  $A$

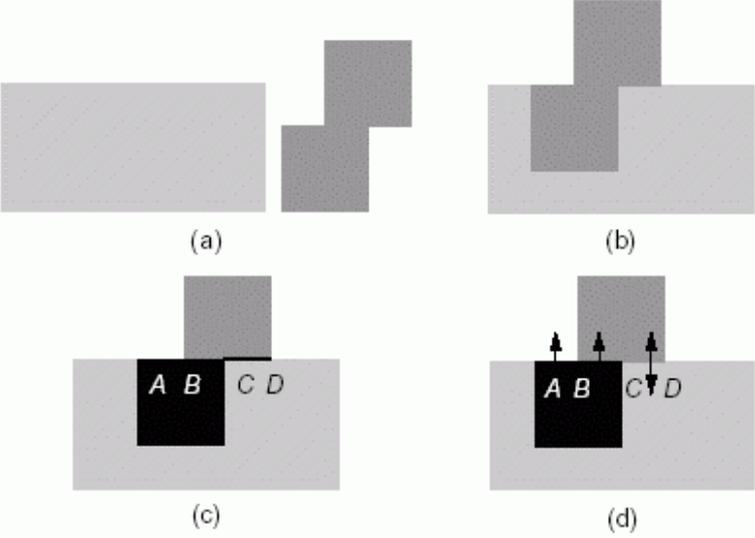
(b) Abschluss von  $A$  (ohne Regularisierung, störende Teile bleiben erhalten)

(c) Inneres von  $A$

(d) Regularisierung von  $A$ .

Regularisierte Mengenoperationen:

Definition:  $A \text{ op}^* B = \text{H\u00fclle}(\text{Inneres}(A \text{ op} B))$



M\u00f6gliche Operationen:  $\cup^*, \cap^*, -^*$

Häufig gebrauchte Grundelemente für die Modellierung:  
*Polyeder*

Definition?

dazu folgende Grundlagen:

*Graphen*  $(V, E)$  (vertices = Knoten oder Ecken, edges = Kanten): sind nicht an Geometrie gebunden, reine Inzidenzstrukturen  
( $E$  Menge ungeordneter Paare von Knoten)

Ein *geometrischer Graph* ist ein Paar  $(V, E)$ , wobei  $V$  nichtleere, endliche Teilmenge von  $\mathbb{R}^3$  und  $E$  eine Menge von Verbindungsstrecken von Punkten aus  $V$  ist.

Die Inzidenz von Ecken und Kanten im Sinne der Graphentheorie ist geometrisch erklärt durch "v ist Endpunkt der Kante e" (im geometrischen Sinne).

Ein *Polygon* ist ein geometrischer Graph  $(V, E)$  mit  $V = \{v_0, \dots, v_{n-1}\}$  und  $E = \{(v_0v_1), (v_1v_2), \dots, (v_{n-2}v_{n-1})\}$ .

Ein Polygon heißt

- *eben*, falls alle Kanten in einer Ebene liegen
- *geschlossen*, falls  $v_0 = v_{n-1}$
- *einfach*, falls gilt: Der Schnitt jeweils zweier Kanten ist entweder leer oder eine Ecke, und jede Ecke gehört zu höchstens zwei Kanten (d.h.: keine Selbstüberschneidungen des Polygons)

Es gilt der grundlegende

*Jordansche Kurvensatz für ebene, geschlossene Polygone:*

Jedes geschlossene, einfache Polygon in der Ebene unterteilt die Ebene in zwei disjunkte *Polygongebiete*, ein inneres und ein äußeres Polygongebiet.

Punkte im Inneren können dadurch charakterisiert werden, dass die Anzahl der Schnittpunkte zwischen den Kanten des Polygons und einem Strahl, der von dem Innenpunkt ausgeht, ungerade ist (bereits beim Scangeraden-Algorithmus zum Füllen von Polygonen verwendet).

Ein *Polygonnetz* ist eine Menge  $M$  von endlich vielen geschlossenen, ebenen und einfachen Polygonen mit folgenden Eigenschaften:

- die inneren Polygone von je 2 Polygonen aus  $M$  haben keine gemeinsamen Punkte
- je 2 Polygone aus  $M$  haben entweder keinen Punkt oder eine Ecke oder eine ganze Kante gemeinsam
- jede Kante eines Polygons aus  $M$  gehört zu höchstens 2 Polygonen
- die Menge aller Kanten, die nur zu einem Polygon aus  $M$  gehören, ist entweder leer ( $M$  heißt dann "geschlossen") oder bildet selbst ein einziges, geschlossenes, einfaches Polygon.

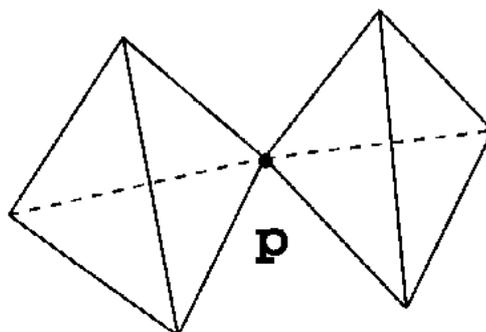
Polygonnetze spielen eine wichtige Rolle in der Modellierung (Beisp. FEM, Kontrollpunkt-Netze von Bézier- und B-Spline-Flächen, *elevation grids*)

Def. *Polyeder*:

Ein Polygonnetz  $M$  heißt *Polyeder*, wenn gilt:

- $M$  ist geschlossen (d.h. jede Kante gehört zu genau 2 Polygonen)
- $M$  ist zusammenhängend
- jede Ecke gehört zu einer endlichen, zyklisch geordneten Menge von Polygonen, in der aufeinanderfolgende Polygone jeweils eine zur Ecke gehörende Kante gemeinsam haben

Beispiel, wo die dritte Bedingung verletzt ist:



Die inneren Polygonegebiete von  $M$  heißen auch *Facetten* oder Seitenflächen des Polyeders.

Der Abschluss der Vereinigung aller Facetten heißt die *Oberfläche* (surface) des Polyeders.

Die Polyeder sind die 3D-Verallgemeinerungen der ebenen, einfachen, geschlossenen Polygone.

Insbes. gilt das 3D-Analogon des Jordanschen Kurvensatzes:

Jedes Polyeder teilt den Raum in zwei disjunkte Bereiche, das Innere und das Äußere.

Das Innere kann wieder dadurch charakterisiert werden, dass die Anzahl der Schnitte eines Strahls, der von einem Innenpunkt ausgeht, mit der Oberfläche ungerade ist.

Das Innere eines Polyeders ist also vollständig durch seine Oberfläche definiert

→ Grundlage einer wichtigen Repräsentationsform:

*boundary representation*

# Datenstrukturen für Polygonnetze

## 1. Polygonorientierte Datenstruktur

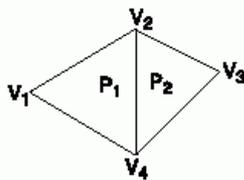
- Jedes Polygon wird durch eine Liste von Koordinaten der Knoten (Eckpunkte, Vertices) beschrieben:

$$P = ((x_1, y_1, z_1), (x_2, y_2, z_2), \dots, (x_n, y_n, z_n))$$

- Kanten werden zwischen aufeinanderfolgenden Knoten erzeugt
- Knoten werden mehrfach gespeichert
- keine explizite Kennzeichnung geteilter Ecken und Kanten
- Probleme beim Rendering (Kanten werden doppelt gezeichnet)

## 2. Knotenorientierte Datenstruktur

Beseitigung von Redundanz: Knoten werden nur jeweils einmal gespeichert und dann den einzelnen Facetten durch Zeiger zugeordnet. Auflistung der Knoten pro Facette gewöhnlich mit fester Orientierung (z.B. im Uhrzeigersinn) – nützlich für viele Algorithmen



$$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), \dots, (x_4, y_4, z_4))$$

$$P_1 = (1, 2, 4)$$

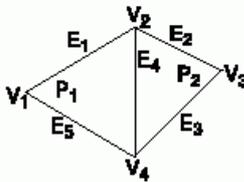
$$P_2 = (4, 2, 3)$$

- speicherplatzsparend, da Knoten nur einmal gespeichert
- einfache Transformationen
- geteilte Kanten doppelt gespeichert

### 3. Kantenorientierte Datenstruktur

Facetten als Zyklen von Kanten

Kanten als Listen von Knoten und 1 oder 2 angrenzenden Facetten; für jede Kante ist Orientierung durch Reihenfolge der Endpunkte festgelegt



$$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), \dots, (x_4, y_4, z_4))$$

$$E_1 = (V_1, V_2, P_1, \lambda)$$

$$E_2 = (V_2, V_3, P_2, \lambda)$$

$$E_3 = (V_3, V_4, P_2, \lambda)$$

$$E_4 = (V_4, V_2, P_1, P_2)$$

$$E_5 = (V_4, V_1, P_1, \lambda)$$

$$P_1 = (E_1, E_4, E_5)$$

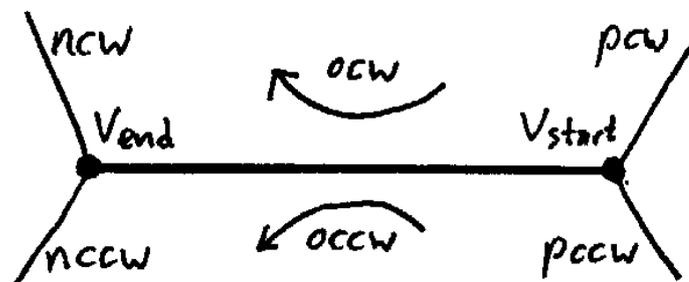
$$P_2 = (E_2, E_3, E_4)$$

Erweiterung:

es werden zusätzlich Nachbarschaftsbeziehungen unter den Kanten gespeichert

(4 Nachbarkanten: ncw = next clockwise, pcw = previous clockwise, nccw = next counterclockwise, pccw = previous counterclockwise)

für die Facetten braucht man dann nur einen Zeiger auf eine Startkante und ein Bit für die Orientierung dieser Startkante.



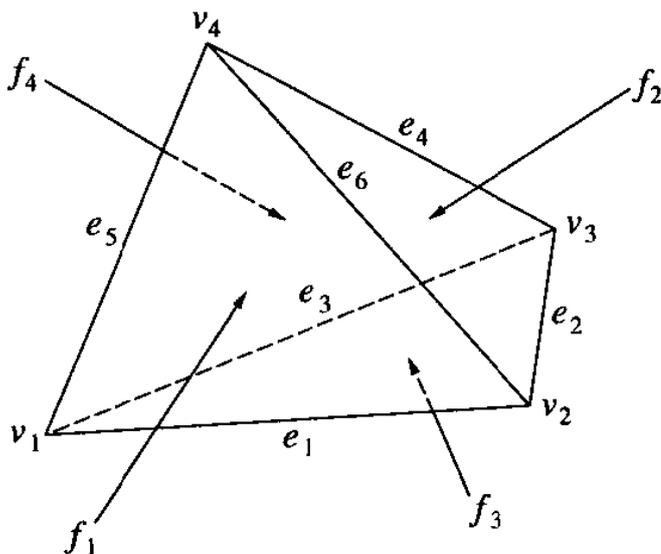
(ocw = orientation clockwise, occw = orientation counterclockwise)

→

## Winged-Edge-Repräsentation

- Datenstruktur zur Speicherung von b-reps
- Kante zeigt auf:
  - angrenzende Knoten ( $V1, V2$ )
  - angrenzende Flächen ( $F1, F2$ )
  - angrenzende Kanten ( $E2, E3, E4, E5$ )
- nur Objekte ohne Löcher werden beschrieben
- effizienter Zugriff auf Nachbarschaft (Flächen, Kanten, Knoten)

Beispiel: Tetraeder



Relationen und Listen der zugehörigen Winged-Edge-Struktur:

$v$ :

$e_1$	$v_1$	$v_2$
$e_2$	$v_2$	$v_3$
$e_3$	$v_1$	$v_3$
$e_4$	$v_3$	$v_4$
$e_5$	$v_1$	$v_4$
$e_6$	$v_2$	$v_4$

$ef$ :

$e_1$	$f_1$	$f_3$
$e_2$	$f_2$	$f_3$
$e_3$	$f_3$	$f_4$
$e_4$	$f_2$	$f_4$
$e_5$	$f_1$	$f_4$
$e_6$	$f_1$	$f_2$

$ee'$ :

$e_1$	$e_2$	$e_3$	$e_5$	$e_6$
$e_2$	$e_1$	$e_3$	$e_4$	$e_6$
$e_3$	$e_1$	$e_2$	$e_4$	$e_5$
$e_4$	$e_2$	$e_3$	$e_5$	$e_6$
$e_5$	$e_1$	$e_3$	$e_4$	$e_6$
$e_6$	$e_1$	$e_2$	$e_4$	$e_5$

$v$ -Liste:

$v_1$	$e_1$
$v_2$	$e_2$
$v_3$	$e_3$
$v_4$	$e_4$

$f$ -Liste:

$f_1$	$e_5$
$f_2$	$e_2$
$f_3$	$e_2$
$f_4$	$e_4$

Allgemeiner Ansatz: der  $vef$ -Graph

Knoten dieses Graphen:

Mengen der Ecken, Kanten und Facetten

Kanten des Graphen: geeignete Adjazenzrelation

10 Relationen kommen in Frage, davon wird eine im jeweiligen Datenmodell eine Auswahl getroffen:

z.B.  $vv$ : Punkte sind benachbart (haben gemeinsame Kante)

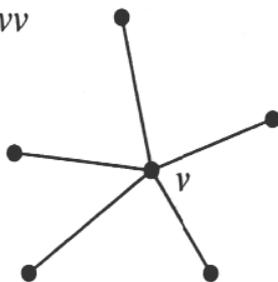
$ve$ : Punkt begrenzt Kante

$vf$ : Punkt ist Eckpunkt von Facette

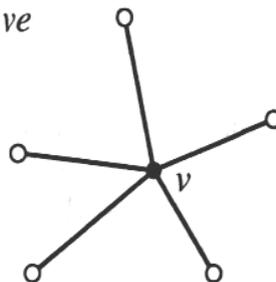
usw. bis  $ff$ : Flächen sind benachbart (haben gemeinsame Kante)

neben  $ee$  betrachtet man noch  $ee'$ : Kanten sind benachbart und begrenzen dieselbe Facette (Teilmenge von  $ee$ )

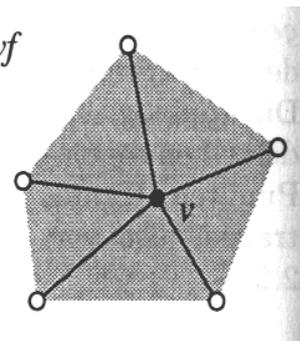
$vv$

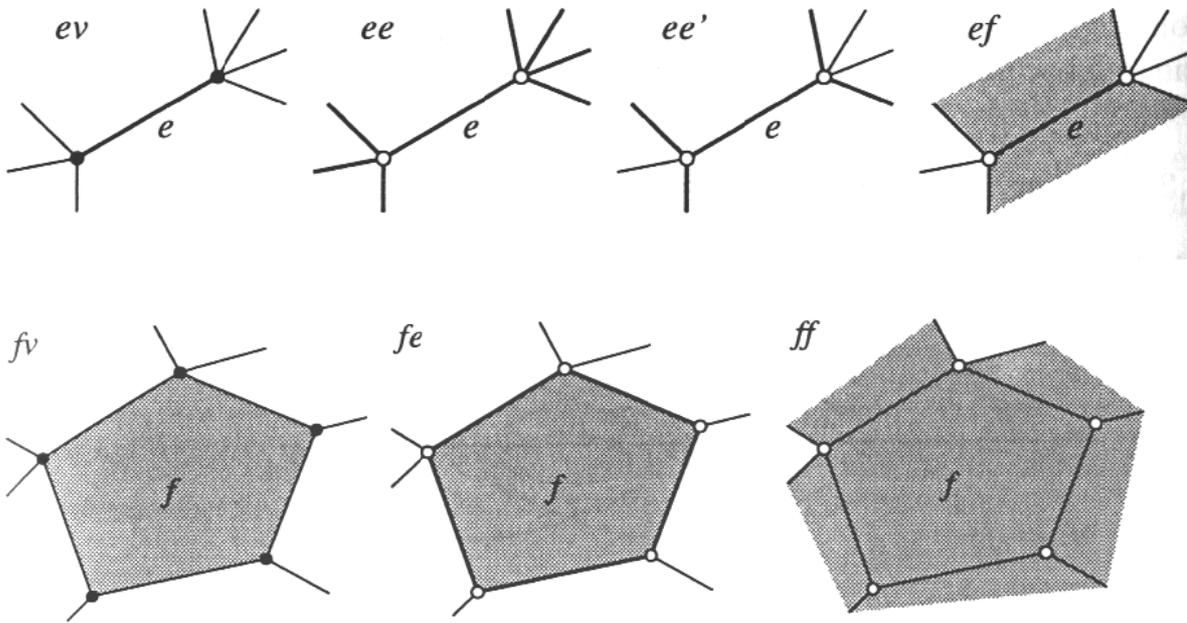


$ve$

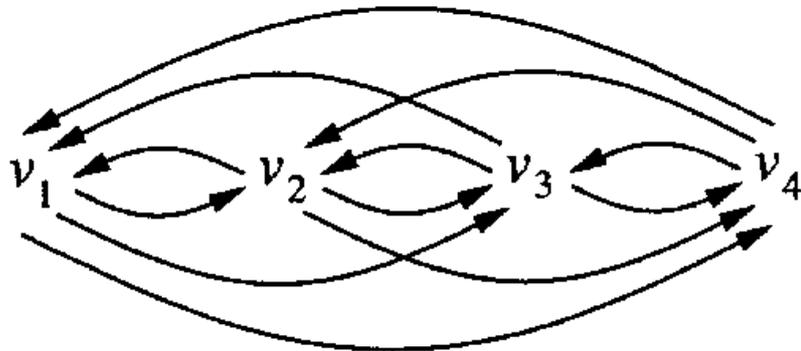


$vf$

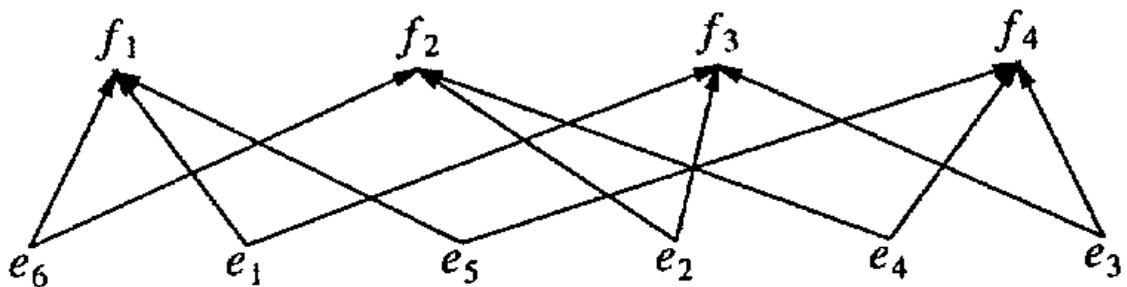




Beispiel:  $vef$ -Graph für die  $vv$ -Relation im obigen Tetraeder:



$vef$ -Graph für die  $ef$ -Relation:



bisher nur betrachtet:  
"Boundary representation"

Übersicht über die wichtigsten Repräsentationsschemata für  
3D-Modelle:

