

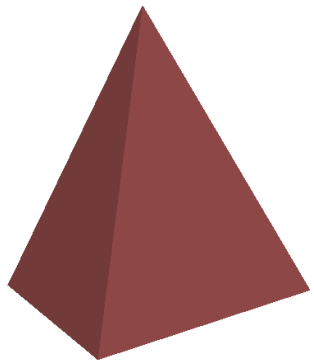
ECG - Einführung in die Geometrische Modellierung

Prof. Dr. Stefan Gumhold

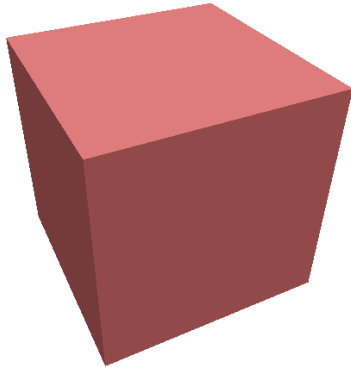


Inhalt

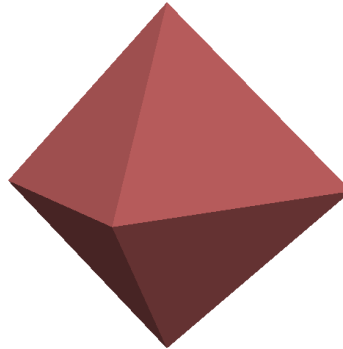
- ◆ Geometrie
 - ◆ Polygonale Netze
 - ◆ Triangulierung von Polygonen
 - ◆ Texture Mapping
 - ◆ Dateiformate
 - ◆ Szenenverwaltung
- ◆ Modellierung glatter Flächen
 - ◆ Implizite Modellierung
 - ◆ Parametrische Modellierung



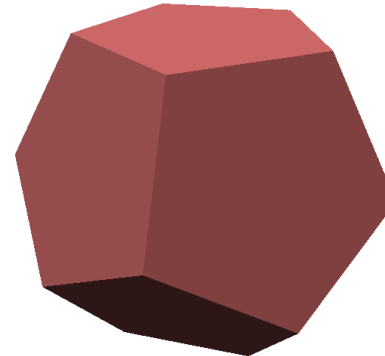
Tetraeder



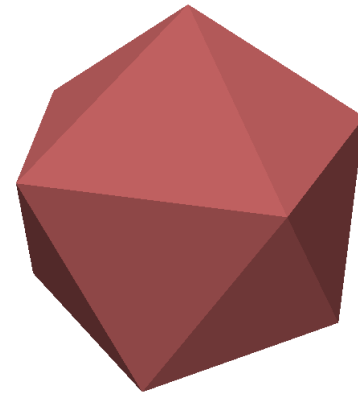
Würfel



Oktaeder



Dodekaeder



Ikosaeder

Beispiele für Körper, bei denen es wichtig ist, dass man auch eine Normale pro Facette angeben kann

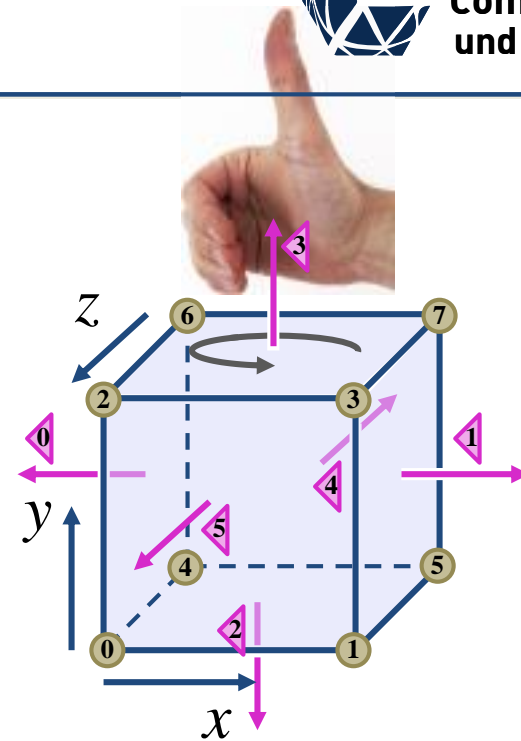
- Um sowohl pro Facetten als auch pro Knoten Attribute (z.B. Normale, Farbe) zu unterstützen, kann man Attribute pro Facettenecke angeben.
- Eine Facettenecke wird auch Keil oder englisch corner bzw. wedge genannt.
- Standardattribute sind neben der Knotenposition: Normale, Farbe und Texturkoordinate

Geometrie

Definition einer Oberfläche



- Geometrie wird facettenweise definiert
- Pro Facette werden die Polygonecken in einem konsistenten Umlaufsinn geordnet: legt man die Finger der rechten Hand so auf die Facette, dass sie dem Umlaufsinn folgen, zeigt der Daumen die Außenrichtung, die mit der Normalenrichtung konsistent sein sollte
- Der Umlaufsinn wird für das backface culling verwendet, bei dem alle Facetten nicht gezeigt werden, auf die man von Innen schaut.



Spezifikation des Würfels:

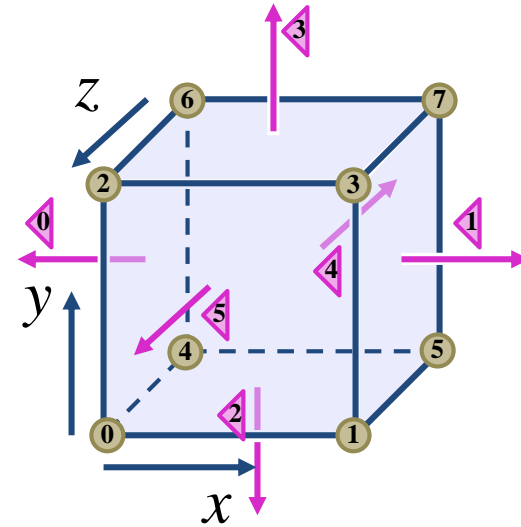
$n_0 p_0, n_0 p_2, n_0 p_6, n_0 p_4,$	(Facette 0)
$n_1 p_1, n_1 p_5, n_1 p_7, n_1 p_3,$	(Facette 1)
$n_2 p_0, n_2 p_4, n_2 p_5, n_2 p_1,$	(Facette 2)
$n_3 p_2, n_3 p_3, n_3 p_7, n_3 p_6,$	(Facette 3)
$n_4 p_4, n_4 p_6, n_4 p_7, n_4 p_5,$	(Facette 4)
$n_5 p_0, n_5 p_1, n_5 p_3, n_5 p_2,$	(Facette 5)

Geometrie

Definition mit Indizes



- Da Knotenposition und Normalen Vektoren mit drei Komponenten sind, benötigt man sehr viel Speicher
- Um die Redundanz in der keilbasierten Geometriespezifikation zu reduzieren, wurde die indizierte Darstellung eingeführt.
- Für jedes Attribut werden die verschiedenen Ausprägungen in einer separaten Liste gespeichert
- Zur Definition der Facetten werden dann in jeder Polygonecke für jedes Attribut ein Index angegeben



Indizierte Spezifikation des Würfels:

Knoten: $p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7$

Normalen: $n_0, n_1, n_2, n_3, n_4, n_5$

Facetten: $0\ 0, 0\ 2, 0\ 6, 0\ 4,$
 $1\ 1, 1\ 5, 1\ 7, 1\ 3,$
 $2\ 0, 2\ 4, 2\ 5, 2\ 1,$
 $3\ 2, 3\ 3, 3\ 7, 3\ 6,$
 $4\ 4, 4\ 6, 4\ 7, 4\ 5,$
 $5\ 0, 5\ 1, 5\ 3, 5\ 2$

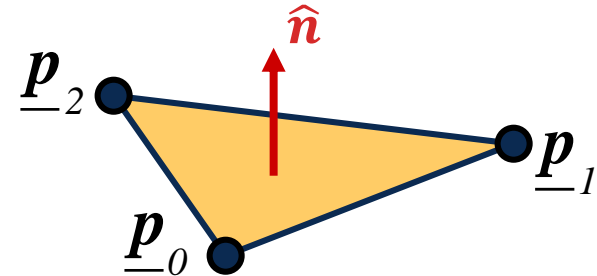
Geometrie

Berechnung von Normalen

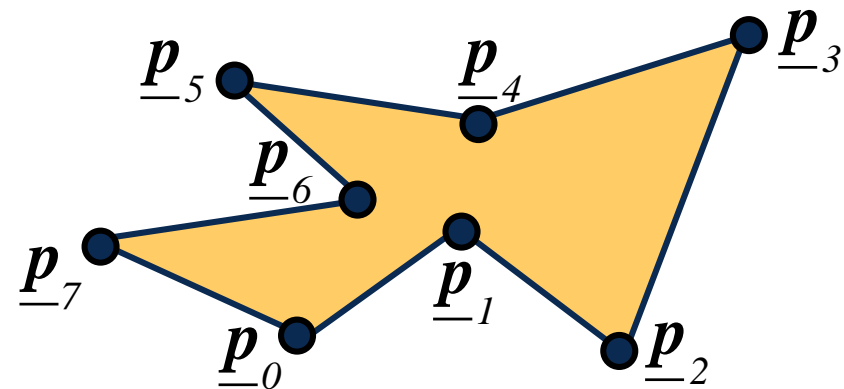


Facettennormale

- Für das Flat Shading und die Berechnung von Knotennormalen müssen Normalen von beliebigen Facetten berechnet werden
- Im Falle von Dreiecken geht das einfach mit Hilfe des Kreuzproduktes von zwei Kantenvektoren
- Die Dreiecksformel kann durch ausmultiplizieren in eine Form gebracht werden, die auf beliebige auch nicht ebene Facetten verallgemeinert werden kann.



$$\begin{aligned}\hat{n} &\propto (\underline{p}_1 - \underline{p}_0) \times (\underline{p}_2 - \underline{p}_0) \\ &= \underline{p}_0 \times \underline{p}_1 + \underline{p}_1 \times \underline{p}_2 + \underline{p}_2 \times \underline{p}_0\end{aligned}$$



$$\hat{n} \propto \underline{p}_0 \times \underline{p}_1 + \dots + \underline{p}_{d-1} \times \underline{p}_0$$

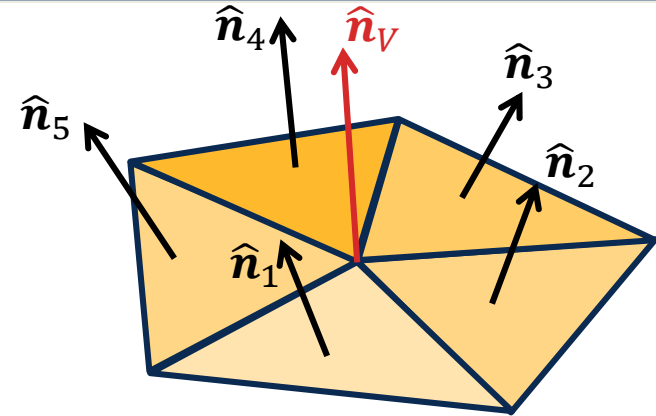
Geometrie

Berechnung von Normalen

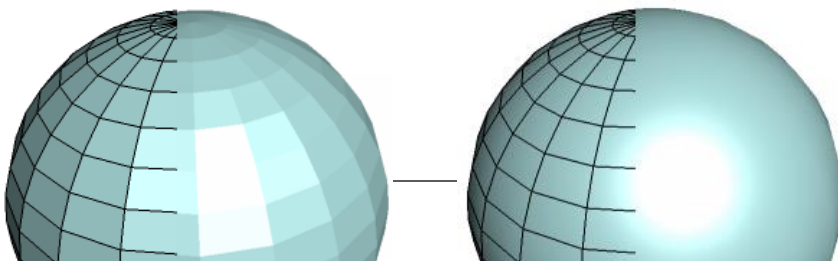
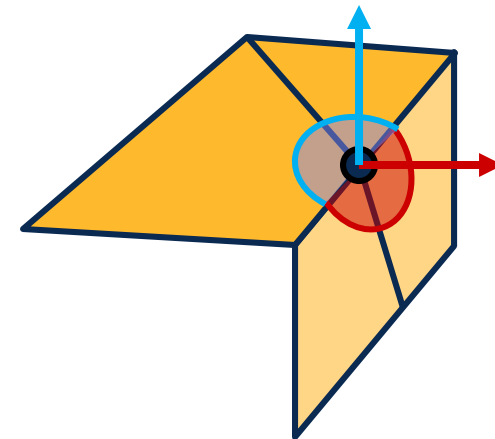


Knotennormale

- Für jeden Knoten kann man eine Normale aus den Normalen der anliegenden Facetten so schätzen, dass die Fläche bei der Beleuchtung glatt erscheint
- Dabei werden pro Knoten die anliegenden Normalen gemittelt
 - alle gleich gewichtet
 - mit Keilwinkel gewichtet
 - mit Dreiecksfläche gewichtet
- **Vorsicht:** bei Netzen mit scharfen Kanten müssen Normalen pro Keil gespeichert werden.



$$\hat{n}_v \propto \sum_i \omega_i \hat{n}_i$$

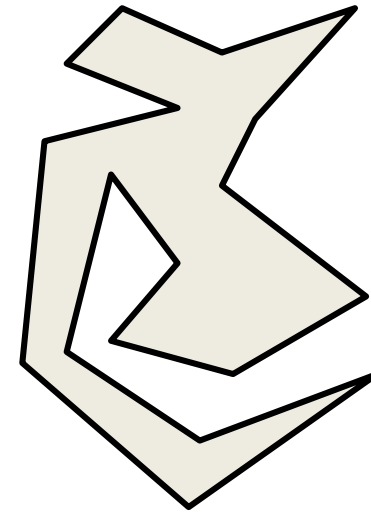
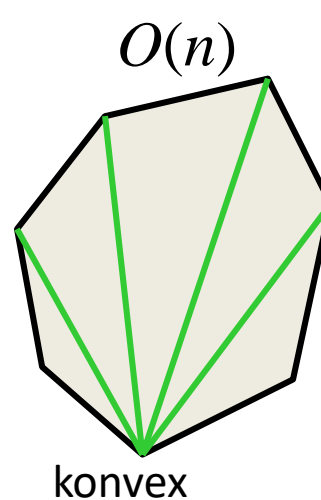


Triangulierung von Polygonen

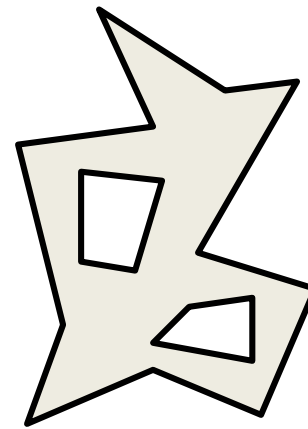
Definitionen



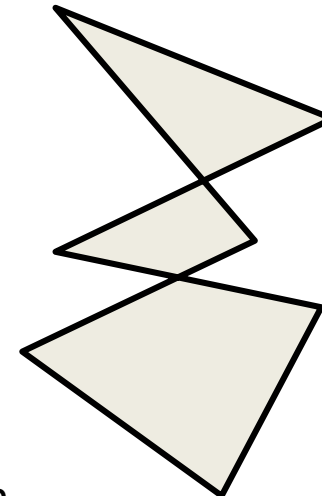
- Glatte Oberflächen müssen in Dreiecke und Vierecke zerlegt werden, bevor sie an die GPU geschickt werden können.
- Häufig finden sich in polygonalen Netzen Polygone mit mehr als 4 Knoten
- Ein Polygon besteht aus einem oder mehreren geschlossener Kantenzüge, die endlich viele Eckpunkte verbindet
- Typischerweise werden diese in Dreiecke zerlegt. Diesen Vorgang nennt man Triangulierung von Polygonen.
- Man unterscheidet verschiedene Klassen von Polygonen, deren Triangulierung unterschiedlich schwierig ist



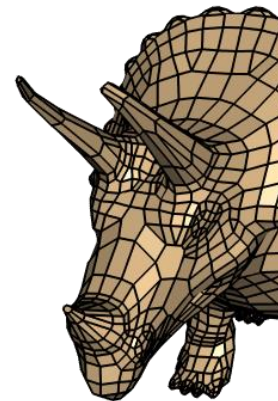
einfach \Leftrightarrow
eine Schleife,
wobei sich in
jedem Knoten
genau zwei
Kanten schnei-
den und alle
Kantenschnitt-
punkte sind
Knoten



einfach mit Löchern



allgemein



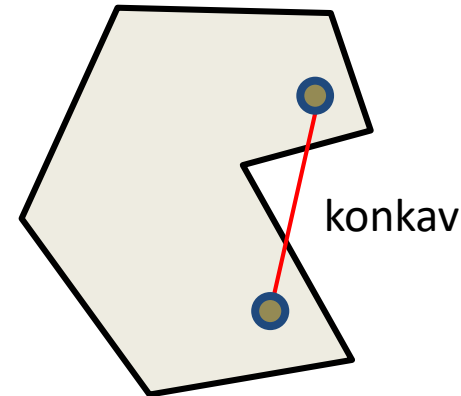
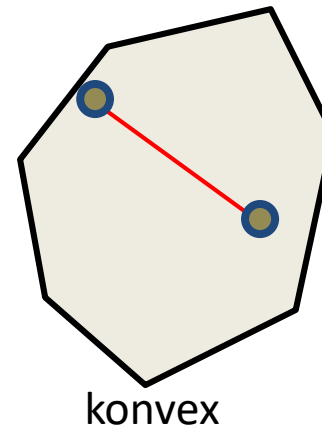
polygonales
Netz

Triangulierung von Polygonen

Eigenschaften



- Ein Polygon ist konvex, wenn die Verbindung zweier beliebiger Punkte im Polygon innerhalb des Polygons liegt.
- Ein Polygon ist konvex, wenn kein Innenwinkel größer 180 Grad ist.
- Für die Winkelberechnung mit Skalar- und Kreuzprodukt führt man konzeptionell eine dritte Dimension z ein, wobei alle z-Koordinaten auf null gesetzt werden. Beim Kreuzprodukt ist dann nur die z-Komponente ungleich null.



$$\vec{u} \times \vec{v} = \begin{pmatrix} u_x \\ u_y \\ 0 \end{pmatrix} \times \begin{pmatrix} v_x \\ v_y \\ 0 \end{pmatrix} = \begin{pmatrix} u_y v_z - u_z v_y \\ u_z v_x - u_x v_z \\ u_x v_y - u_y v_x \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ u_x v_y - u_y v_x \end{pmatrix}$$

$$\Rightarrow \|\vec{u} \times \vec{v}\| = |u_x v_y - u_y v_x|$$

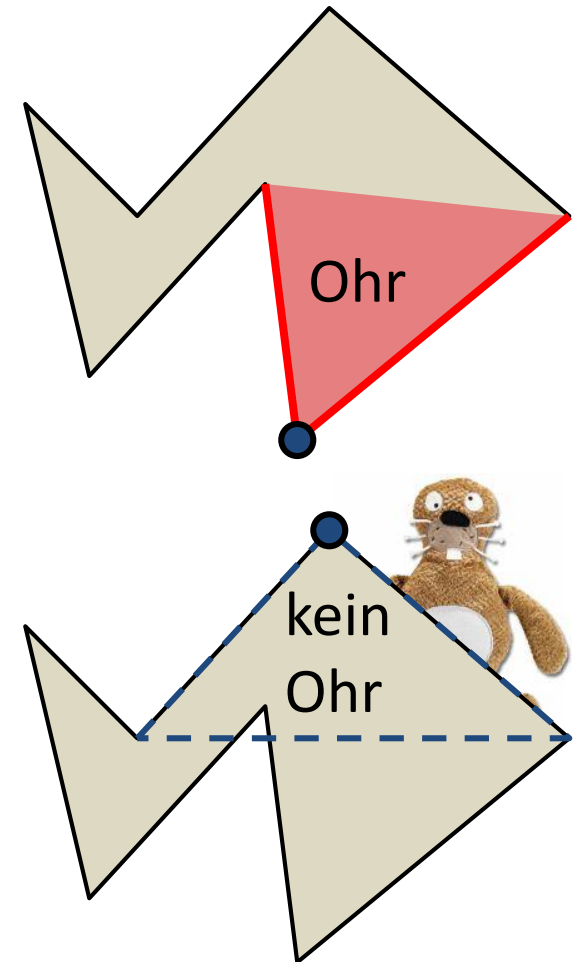
Triangulierung einfacher Polygone

Ear-Cutting Algorithmus



Ear-Cutting Algorithmus

- Algorithmus zur Triangulierung von einfachen Polygonen
- Die Polygonecke zu einem Knoten ist das aus den zwei anliegenden Kanten definierte Dreieck
- Polygonknoten und Polygonecke heißen konvex, wenn der Innenwinkel des Knotens kleiner 180 Grad ist – ansonsten konkav
- Enthält eine konvexe Polygonecke keine weiteren Knoten, so wird sie Ohr genannt
- **Satz:** Bis auf Dreiecke haben alle einfachen Polygone mindestens zwei Ohren
- **Satz:** Ist eine Ecke kein Ohr, so liegt wenigstens ein konkaver Knoten in der Ecke



Triangulierung einfacher Polygone

Ear-Cutting Beispiel und Laufzeit



Ear-Cutting Algorithmus

klassifiziere alle Knoten in konvex
oder konkav

wiederhole $n-3$ mal

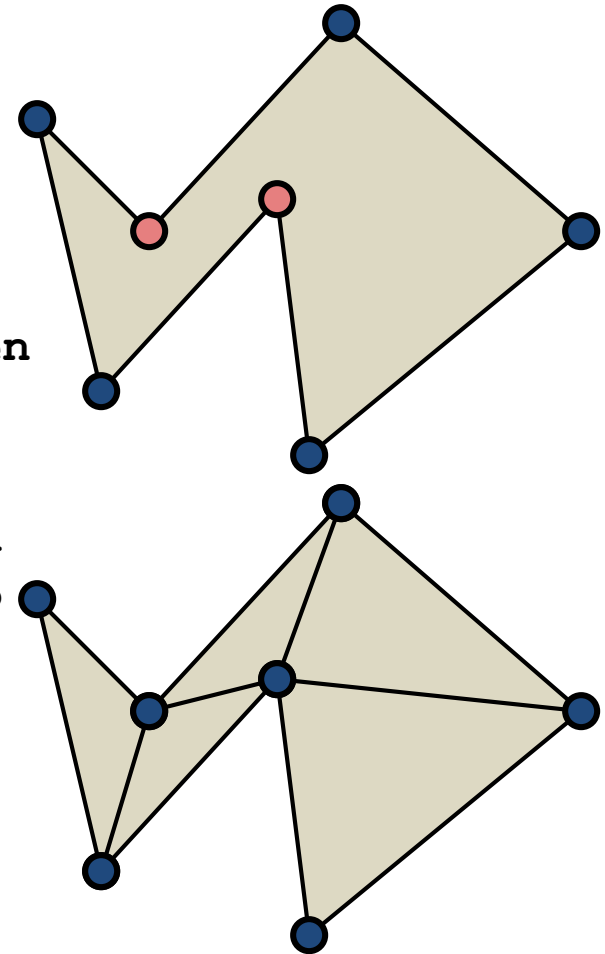
iteriere alle konvexen Knoten

prüfe, ob Ecke des Knotens ein
Ohr ist (nur gegen konkave Knoten
testen)

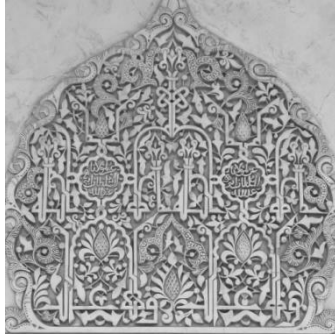
wenn ja, schneide Ohr ab,
klassifiziere benachbarte Knoten
neu & breche innere Iteration ab

ein Ear-Cut

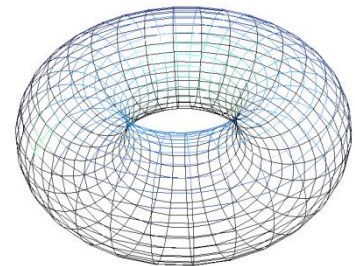
- Mit der Anzahl n der Knoten des Polygons
- Die Triangulierung eines Polygons mit n Knoten enthält grundsätzlich $n-2$ Dreiecke – deshalb müssen immer $n-3$ Ear-Cuts gemacht werden
- Bei k konvexen Knoten ist die schlechtest mögliche Laufzeit $O((n-3) \cdot k \cdot (n-k))$



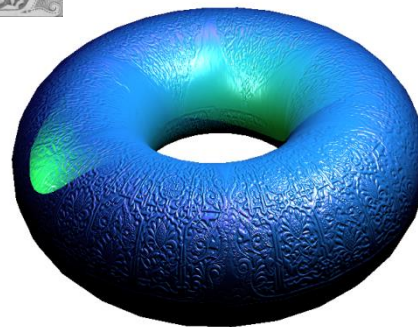
- ◆ hoch aufgelöste Bilder erlauben es, 3D Modelle mit Details anzureichern
- ◆ Bei Beeinflussung der Farbe spricht man von Texture Mapping, wenn die Normale geändert wird von Bump Mapping



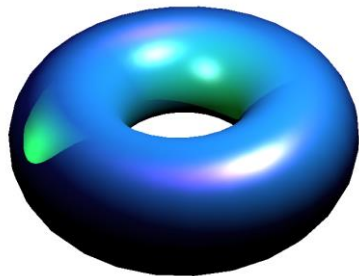
Textur



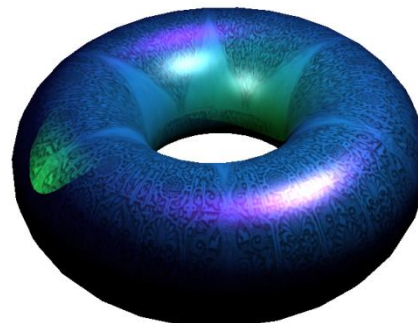
Drahtgittermodell



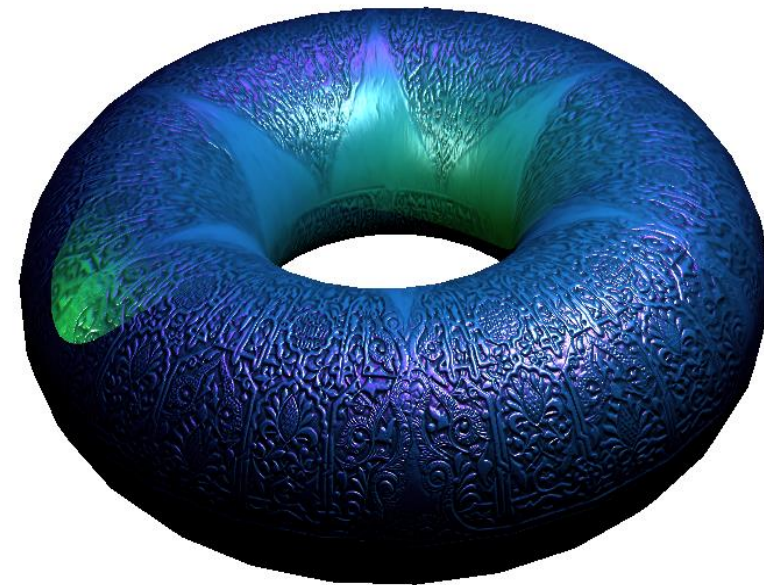
Bump Mapping



Beleuchtung mittels
pro Knoten Normalen



Texture Mapping



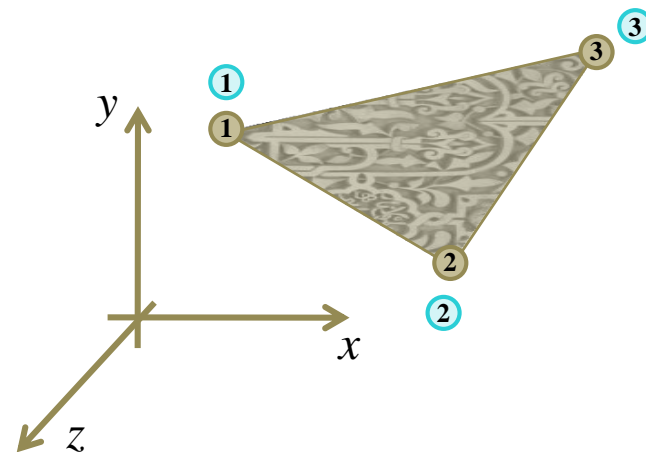
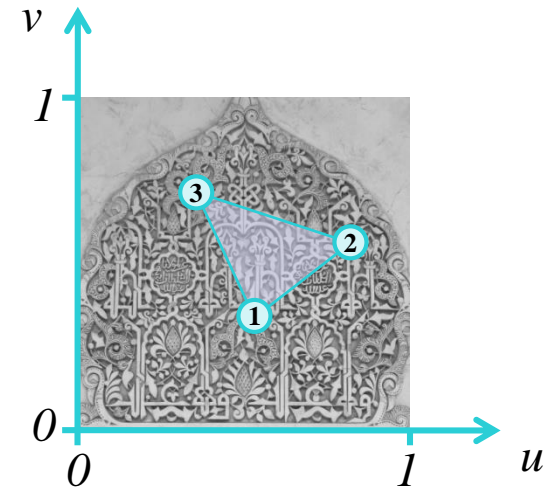
Texture and Bump Mapping

Texture Mapping

Funktionsweise



- Um Texture Mapping zu unterstützen definiert man für jede Facettenecke zusätzlich zu 3D Position und Normal eine 2D Texturposition, die meist Texturkoordinaten genannt wird.
- Die Koordinatenachsen in der Textur heißen je nach Konvention (u,v) oder (s,t) und haben jeweils einen Wertebereich von $[0,1]$
- Die Texturkoordinaten werden über Dreiecke baryzentrisch interpoliert
- Wenn ein Pixel gezeichnet wird, wird die Farbe über die Texturkoordinaten in der Textur nachgeschaut

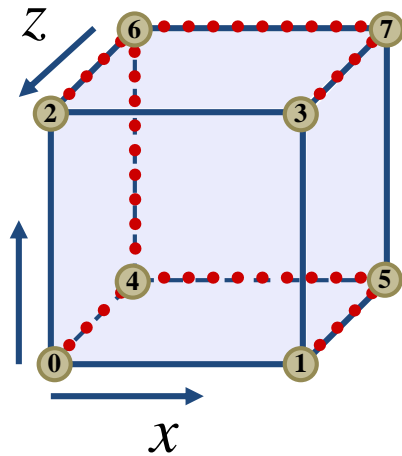


Texture Mapping

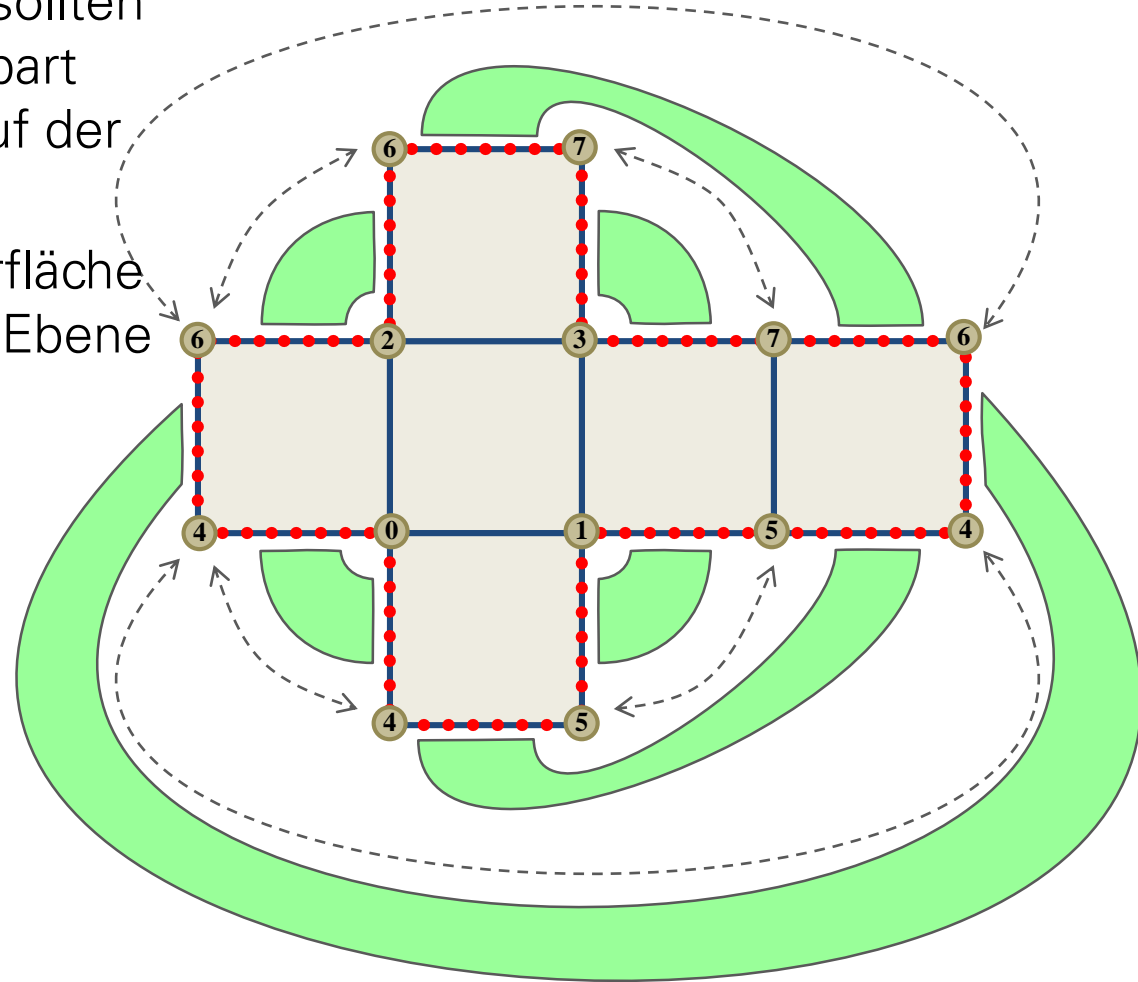
Texturraum



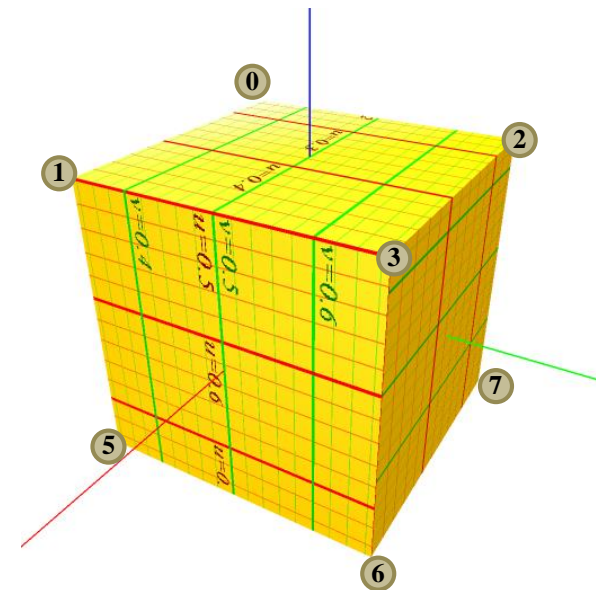
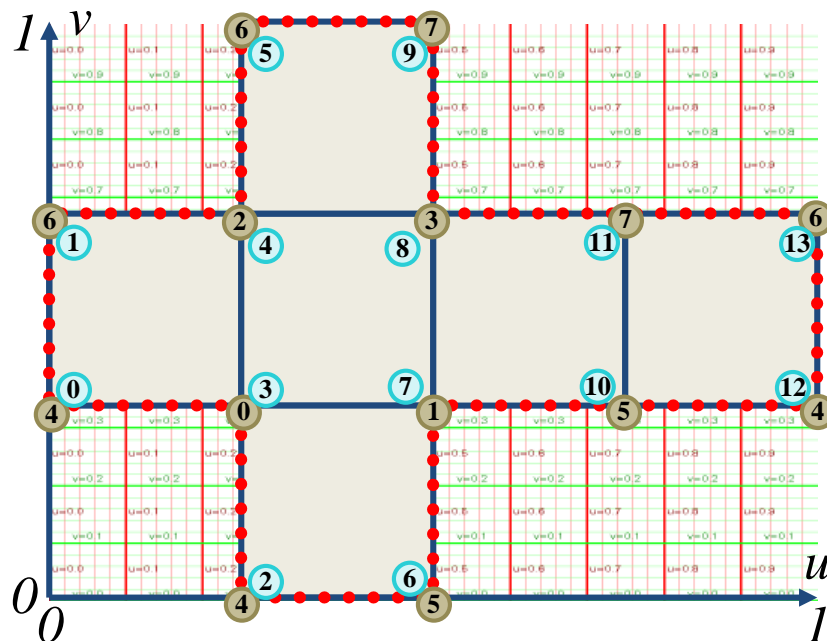
- Dreiecke im Texturraum sollten möglichst gleich benachbart sein, wie die Dreiecke auf der Oberfläche
- Dazu kann man die Oberfläche aufschneiden und in der Ebene auslegen



Schnittkanten auf dem
Würfel rot gepunktet



- Für die Texturierung eines Würfels mit der gängigen ebenen Realisierung werden 14 Texturpositionen benötigt, die wiederum indiziert pro Facettenecke gespeichert werden können.



Dateiformat

Wavefront obj

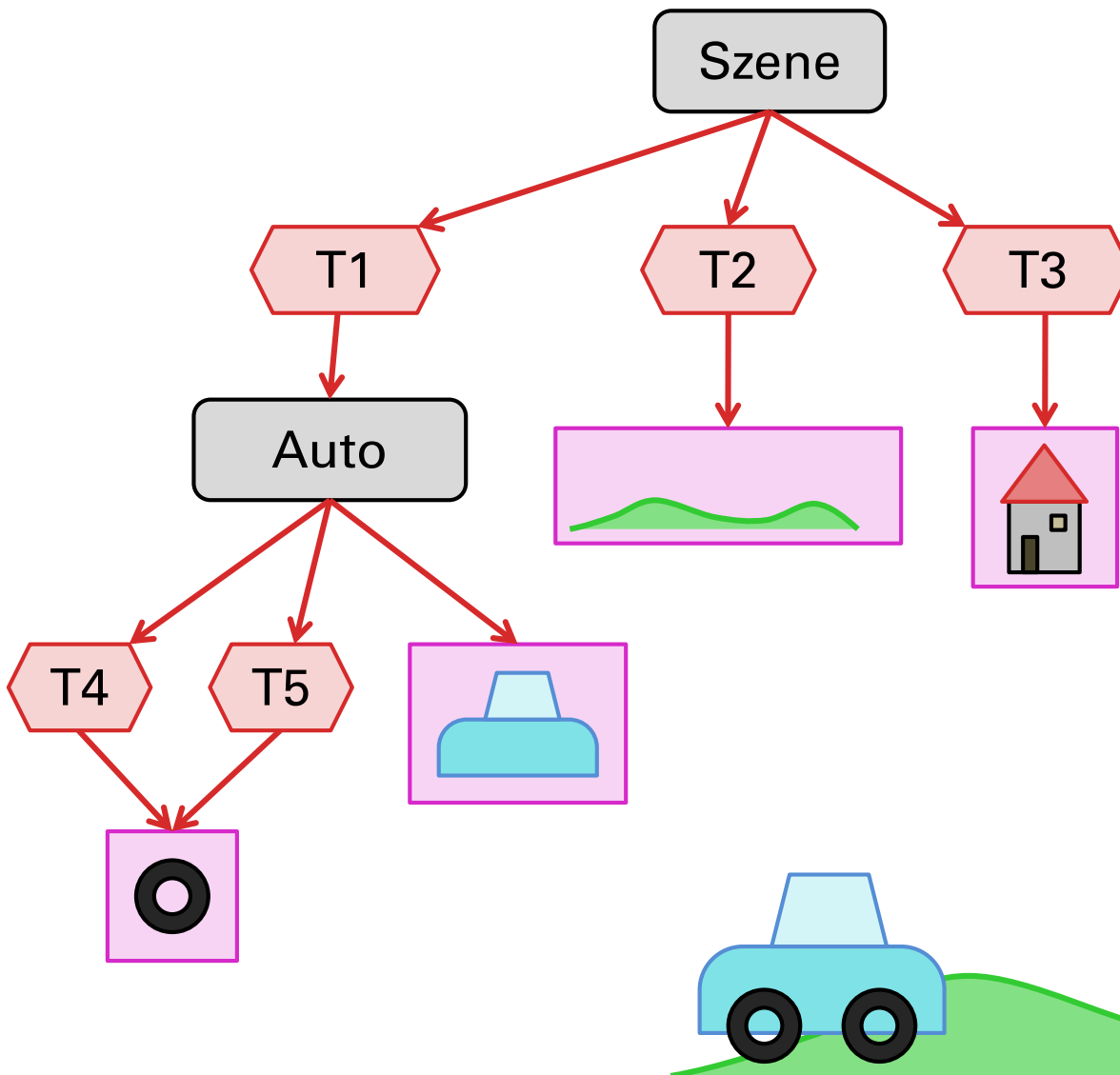
- Die meisten Dateiformate für polygonale Netze speichern die Geometrie in indizierter Darstellung
- Das einfachste und sehr oft benutzte Format ist Wavefront obj, das zeilenbasiert ist
- Jede Zeile beginnen mit einer Abkürzung, die angibt was der Rest der Zeile spezifiziert
 - v** ... Knotenposition mit x y z
 - vt** ... Texturposition mit u v
 - vn** ... Normale mit x y z
 - f** ... Facette mit einem Tripel von Positions-/Textur-/Normalenindex pro Keil. Die Attribute werden dazu beginnend mit 1 durchgezählt

- Würfel mit Normalen aber ohne Texturkoordinaten in obj:


```
v -1 -1 1
v 1 -1 1
v -1 1 1
v 1 1 1
v -1 -1 -1
v 1 -1 -1
v -1 1 -1
v 1 1 -1
vn -1 0 0
vn 1 0 0
vn 0 -1 0
vn 0 1 0
vn 0 0 -1
vn 0 0 1
f 1//1 3//1 7//1 5//1
f 2//2 6//2 8//2 4//2
f 1//3 5//3 6//3 2//3
f 3//4 4//4 8//4 7//4
f 5//5 7//5 8//5 6//5
f 1//6 2//6 4//6 3//6
```



Szenenverwaltung

Szenengraph – Beispiel



 ... Gruppenknoten

 ... Transformationsknoten

 ... Geometrie-knoten

Szenenverwaltung

Szenengraph – Übersicht



- Komplexe Szenen werden meist in einer Graphstruktur hierarchisch organisiert
- Ein Wurzelknoten definiert die gesamte Szene
- Die Szene wird aus verschiedenen Knoten zusammengesetzt
 - Gruppenknoten
 - Transformationsknoten
 - Geometrieknoten
 - Kameraknoten
 - Materialknoten
 - Lichtknoten
 - Animationsknoten
- Kanten definieren Eltern-Kind-Beziehungen
- Jeder Pfad von der Wurzel zu einem Blatt definiert eine Instanz eines Szenenelements
- Einfachste Implementierung eines Szenengraphen über Polymorphismus
- Es wird ein Interface mit den im Szenengraphen notwendigen Traversierungen definiert und für unterschiedliche Knotentypen in einer abgeleiteten Klasse spezialisiert.
- Erweiterung der Traversierungen aufwendig, weil alles neu zu übersetzen ist.

Gerichtete Graphen

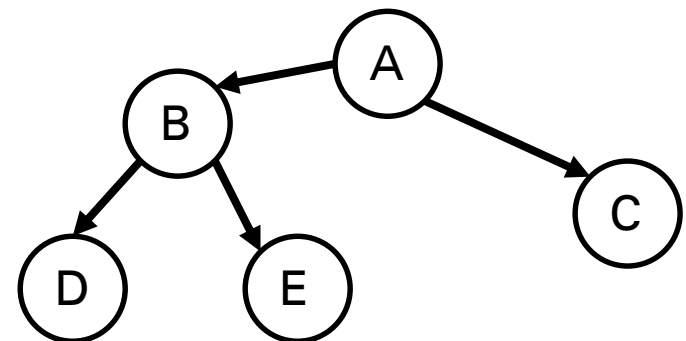
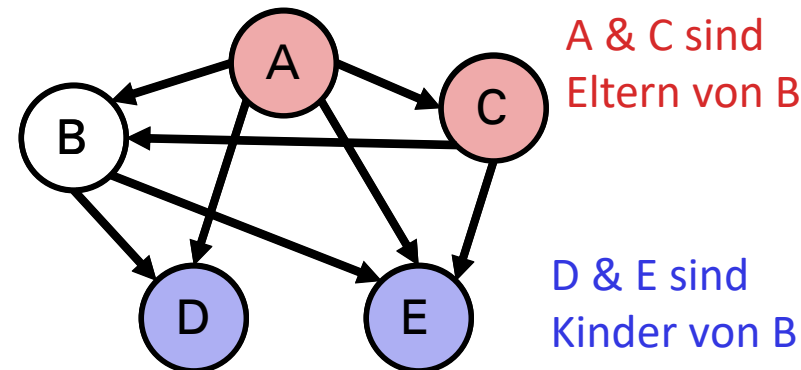
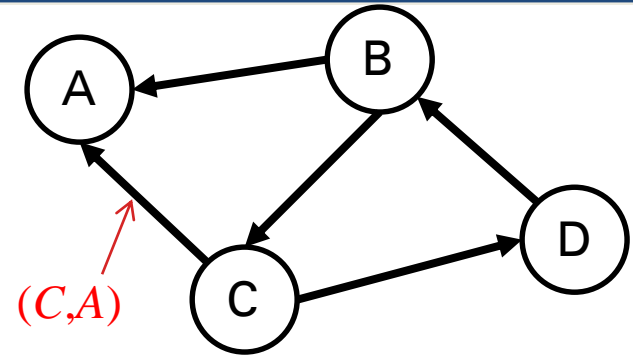
- ... sind definiert durch eine Menge von Knoten V und eine Menge von gerichteten Kanten $E \subset V \times V$, die aus jeweils einem Start- und einem Endknoten definiert sind

Azyklischer gerichteter Graph

- ... haben keine Zyklen, d.h. es existiert keine Kantenfolge der Art $(A,B), (B,C), \dots, (E,F), (F,A)$

Baum

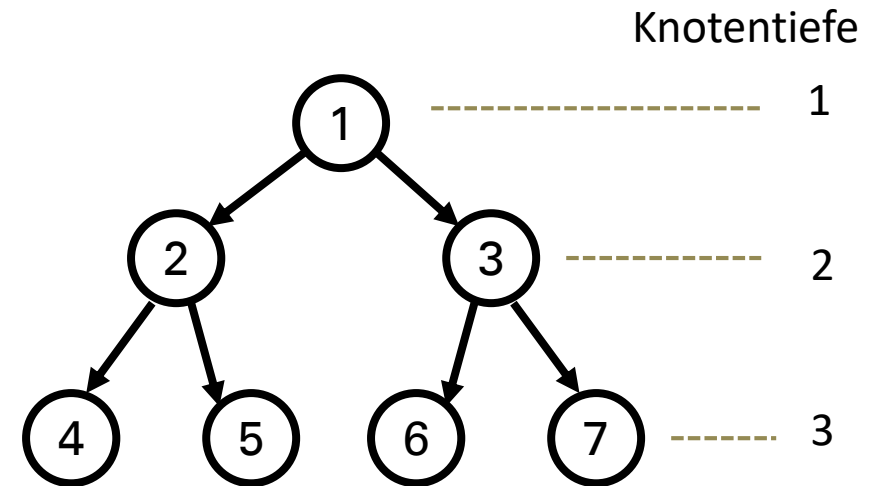
- ... verbundener gerichteter azyklischer Graph, wobei auf keinen Knoten mehr als eine Kanten zeigt



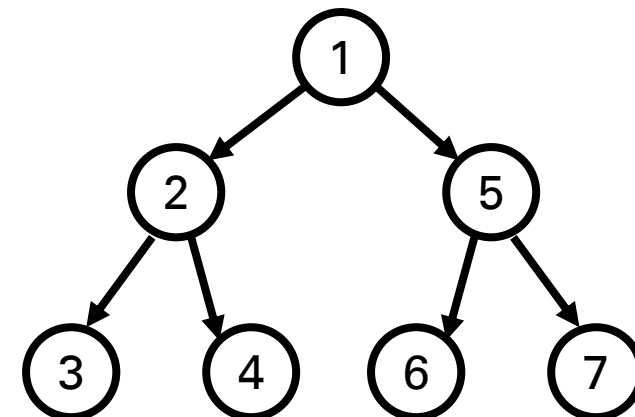


- Die Knoten von Bäumen und azyklischen Graphen können unterschiedlich traversiert (durchwandert) werden
- Knotentiefe ... für jeden Knoten die Länge des kürzesten Pfades zur Wurzel (Wurzel und Knoten mitzählen)
- Breadth First ... Knoten von links nach recht eine Knotentiefe nach der anderen
- Depth First ... Teilbäume der Kinder nacheinander besuchen. Bevor das nächste Kind besucht wird, den vorigen Teilbaum rekursiv abarbeiten

Breadth First

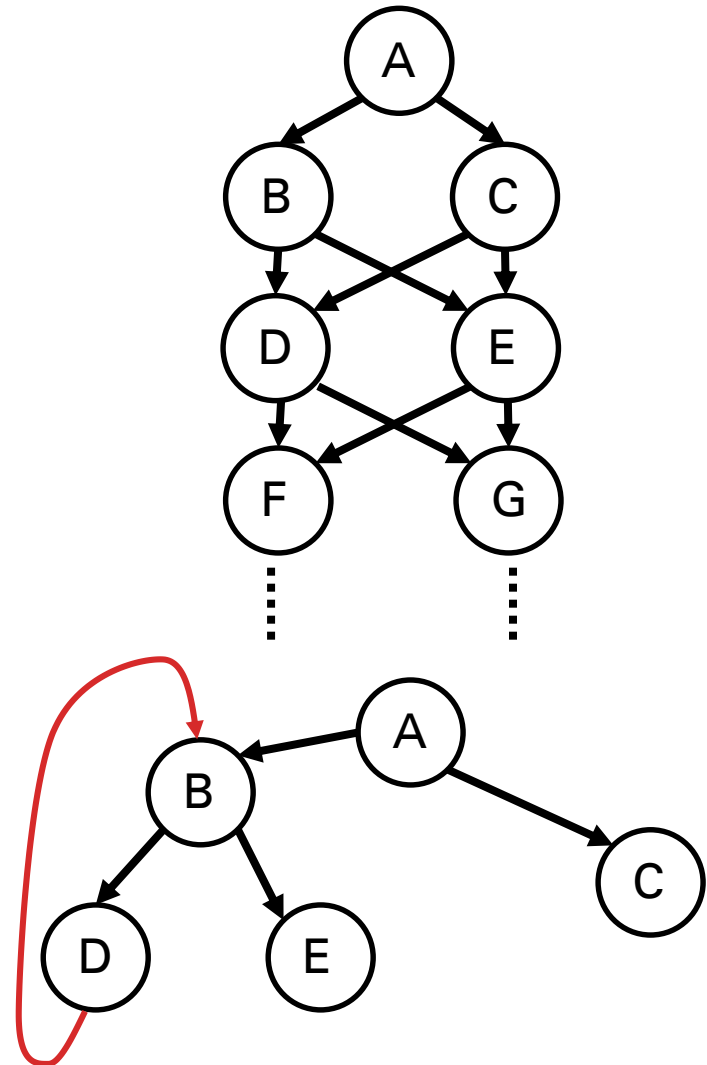


Depth first



Vorsicht

- Die Anzahl der durch einen azyklischen Graphen definierte Instanzen kann exponentiell in der Anzahl der Knoten wachsen.
- Wenn ein vermeintlicher Baum oder azyklischer Graph dennoch einen Zyklus hat, stürzen beide Traversierungsansätze ab. Zu welcher Ausnahmebedingung kommt es???



- Idee: repräsentiere Kurve / Fläche als Nullstellenmenge einer Funktion f über $\mathbf{R}^2 / \mathbf{R}^3$.

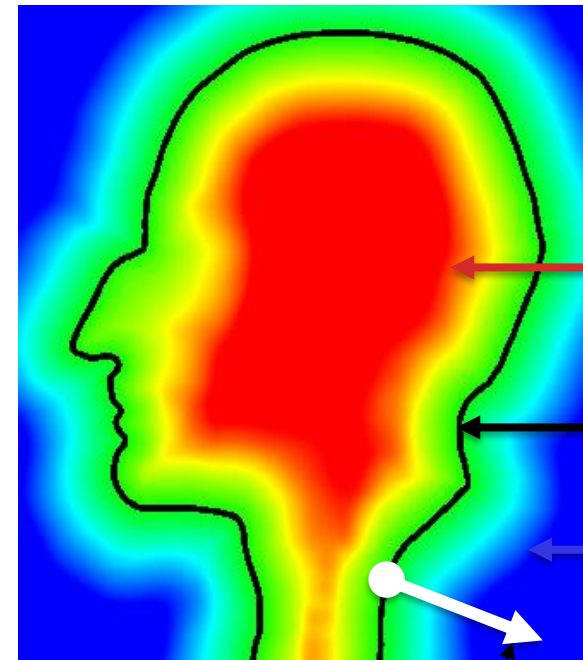
- Bsp.: Einheitskugel

$$f(x, y, z) := x^2 + y^2 + z^2 - 1 = 0$$

- Schreibt man die partiellen von f nach den Koordinaten x, y , und z in einen Vektor, so erhält man den Gradienten $\vec{\nabla} f$:

$$\vec{\nabla} f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{pmatrix} \stackrel{\text{Kugel}}{=} \begin{pmatrix} 2x \\ 2y \\ 2z \end{pmatrix}$$

- An Flächenpunkten zeigt der Gradient in Normalenrichtung



$f(x, y, z) < 0$
(innen)

$f(x, y, z) = 0$
(Fläche)

$f(x, y, z) > 0$
(außen)

$$f(x, y, z) = 0 \Rightarrow \hat{n} \propto \vec{\nabla} f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{pmatrix}$$

Implizite Modellierung

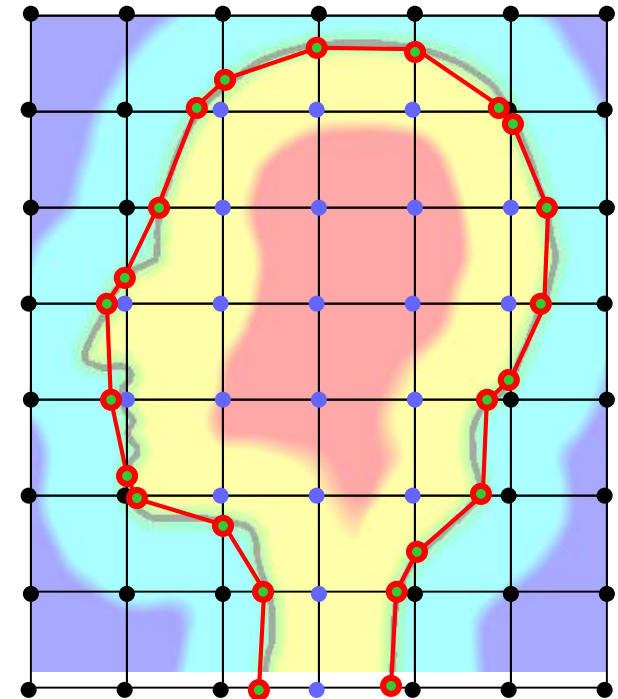
Approximation der impliziten Fläche



- Um eine implizit repräsentierte Kurve / Fläche darzustellen kann man diese mit dem Marching Squares bzw. Cubes Verfahren durch einen Polygonzug bzw. ein Dreiecksnetz annähern

Bsp: Marching Squares

- Definiere Gitter aus Ausdehnung und Auflösung
- Pro Knoten: werte Funktion aus und klassifiziere in innen / außen
- Pro Kante, die innen mit außen verbindet, erzeuge Kantenpunkt
- Pro Zelle verbinde Kantenpunkte mit Liniensegmenten

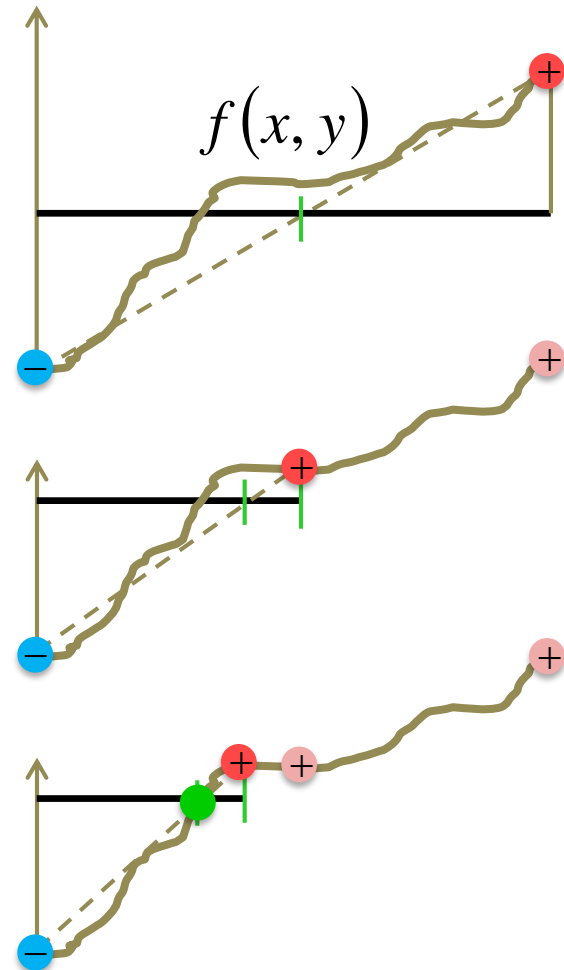


Implizite Modellierung

MS – Kantenpunkt erzeugen



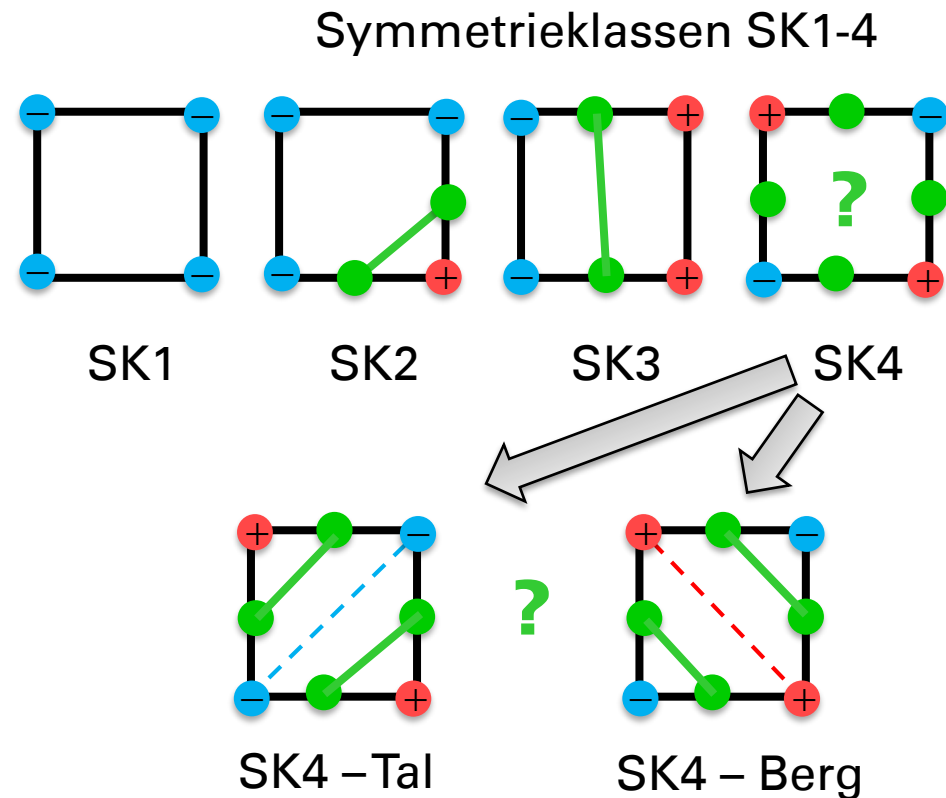
- Wenn eine Kante innen mit außen verbindet, muss es auf der Kante eine Nullstelle der Funktion geben.
- Nimmt man einen linearen Funktionsverlauf an, kann die Position der Nullstelle geschätzt werden
- Ist die lineare Näherung nicht genau genug, kann man die Kante solange an der geschätzten Nullstelle unterteilen und iterieren, bis die gewünschte Genauigkeit erreicht ist



Implizite Modellierung

MS – Kantenpunkte verbinden

- Pro Zelle gibt es $2^4 = 16$ mögliche Klassifikationen der anliegenden Knoten
- Diese kann man in vier Symmetrieklassen einteilen
- Bei der vierten Klasse gibt es zwei Möglichkeiten, die vier Kantenpunkte zu verbinden
- Man kann zum Beispiel den Funktionswert in der Zellmitte bestimmen und sich für die Talverbindung entscheiden, wenn der Wert kleiner Null ist.



Implizite Modellierung

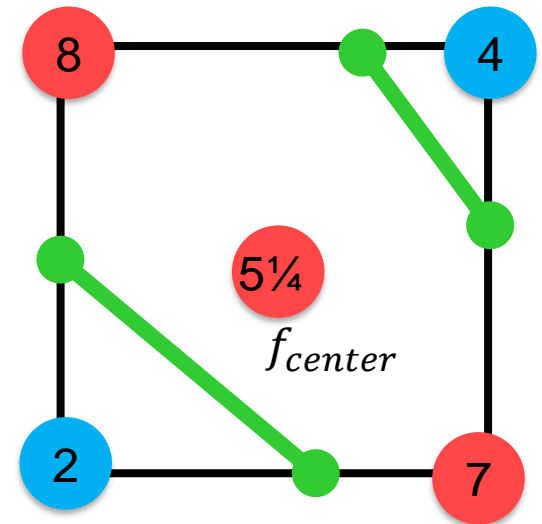
MS – Bsp Mehrdeutigkeiten auflösen



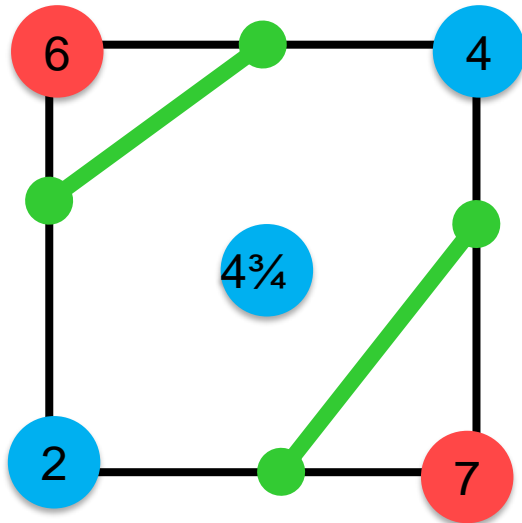
- ◆ Abweichend vom Standard
 $f(x, y) = 0$ soll die Kurve
 $f(x, y) = 5$ extrahiert werden.
- ◆ Die 5 nennt man auch Isolevel
- ◆ Wert in Zellmitte schätzt man
aus Mittelung der Knotenwerte:

$$f_{center} = \frac{8 + 4 + 2 + 7}{4} = 5\frac{1}{4}$$

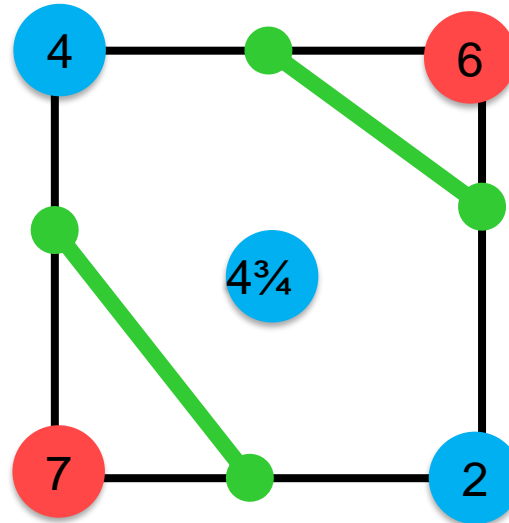
- ◆ Wert in Zellmitte ist größer 5.
Deshalb werden Kantenpunkte
so verbunden, dass ein
Bergkamm entsteht, der die
beiden roten Knoten mit
Werten >5 verbindet.



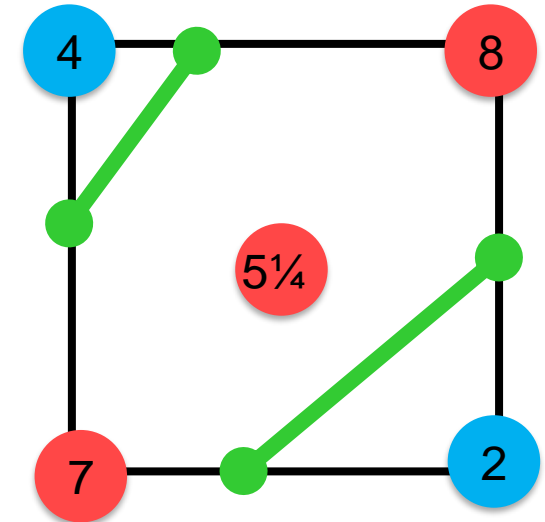
Bsp eines mehrdeutigen Falls.
Zahlen in den Knotenwerte
zeigen die Werte für f .



Bsp für ein Tal



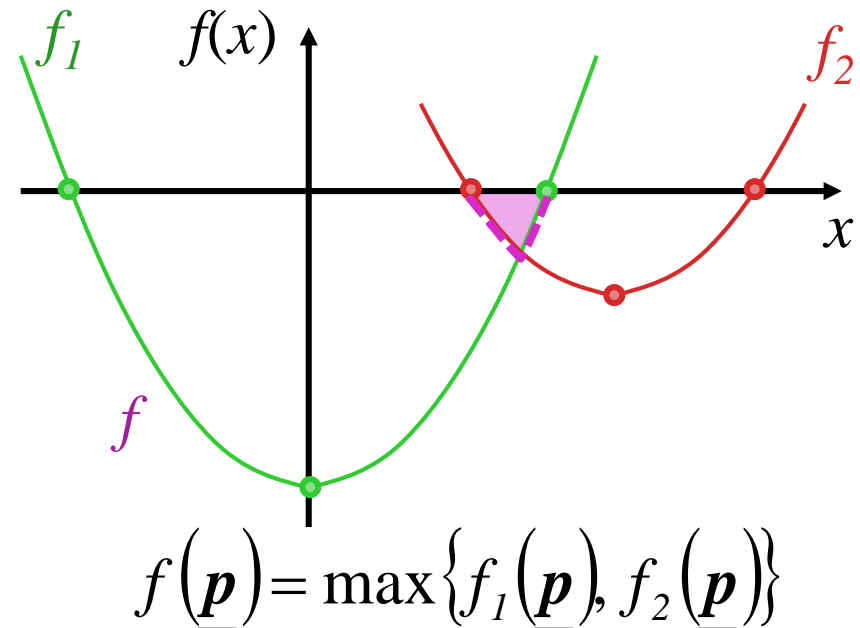
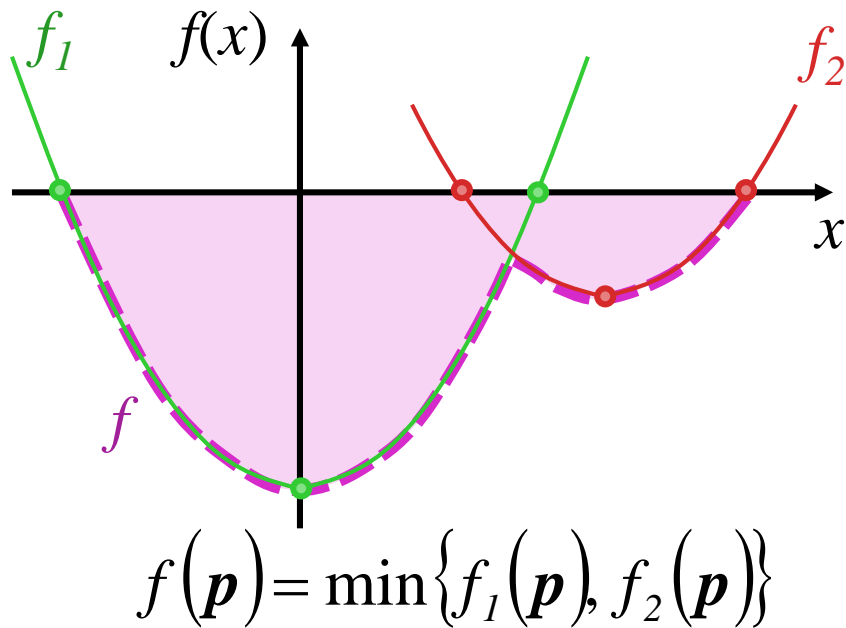
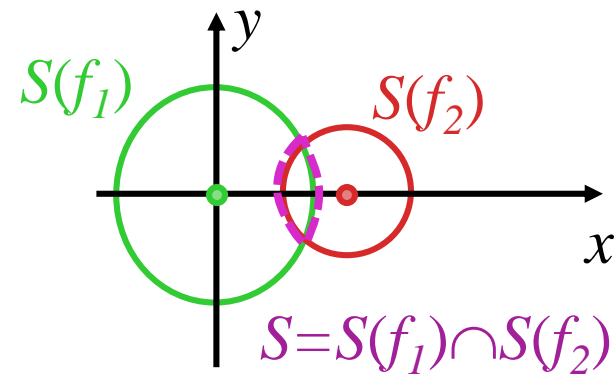
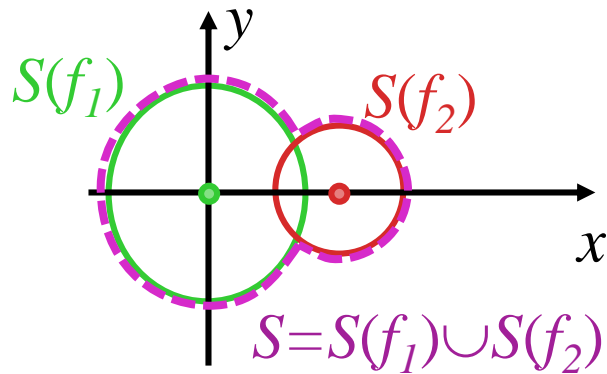
Bsp für Tal in
andere Richtung



Bsp für Bergkamm
in andere Richtung

Implizite Modellierung

Vereinigung und Schnittmenge





CSG – Operation

- Vereinigung

$$S(f) = S(f_1) \cup S(f_2) \quad \Leftrightarrow$$

- Schnittmenge

$$S(f) = S(f_1) \cap S(f_2) \quad \Leftrightarrow$$

- Komplement

$$\overline{S(f)} \quad \Leftrightarrow$$

- Differenz

$$\begin{aligned} S(f) &= S(f_1) - S(f_2) \quad \Leftrightarrow \\ &= S(f_1) \cap \overline{S(f_2)} \end{aligned}$$

Implizite Funktion

- Minimum der Funktion

$$\min\{f_1(\underline{p}), f_2(\underline{p})\}$$

- Maximum der Funktion

$$\max\{f_1(\underline{p}), f_2(\underline{p})\}$$

- negierte Funktion

$$-f(\underline{p})$$

- Maximum mit negierter

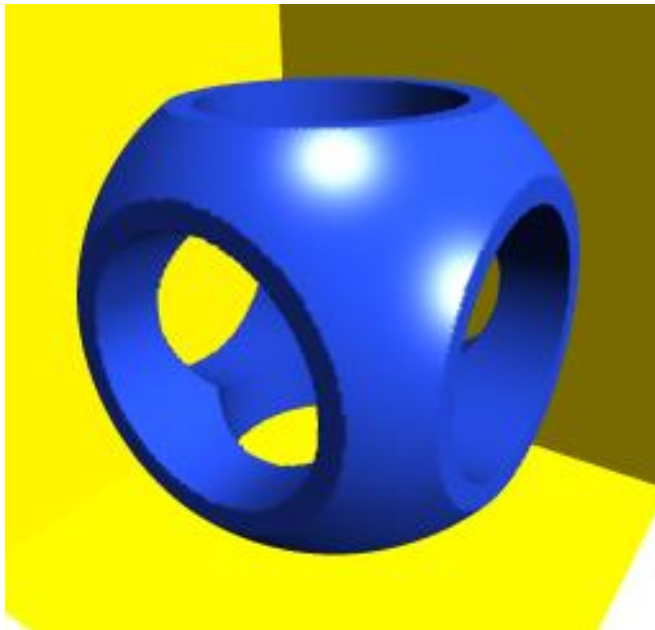
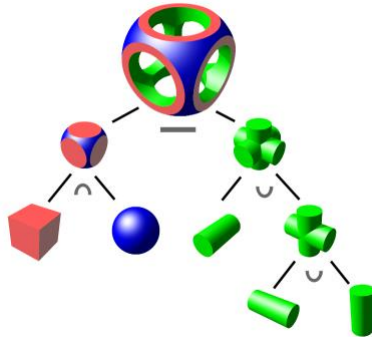
$$\max\{f_1(\underline{p}), -f_2(\underline{p})\}$$

Implizite Modellierung

Anwendung Mengenoperationen



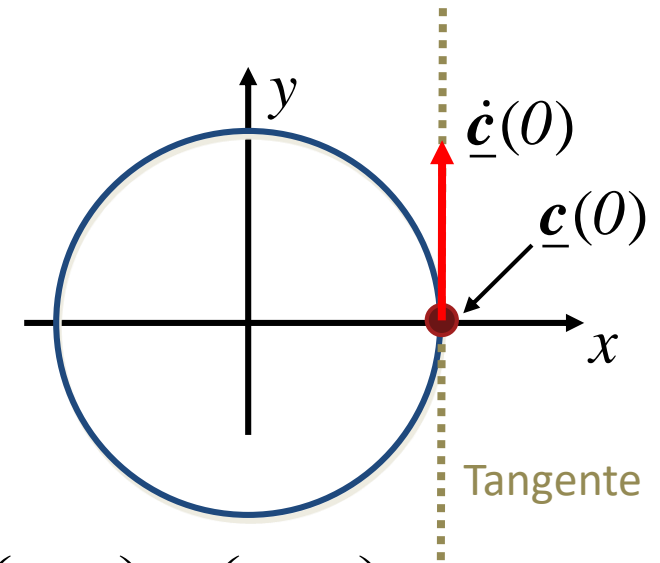
Constructive Solid Geometry (CSG) – Beispiel:



Kompakte Szenenbeschreibung
für den CGV-Viewer für implizite
Flächen:

```
-<root> (           ... difference
    * (             ... intersection
        S,           ... unit sphere
        u[s=0.8] (   ... scale unif.
            B         ... unit box
        )
    ),
    u[s=0.5] (
        + (           ... union
            Y,         ... cylinder
            r[] (Y),    ... rotation
            r[nx=0;ny=1] (Y)
        )
    )
)
```

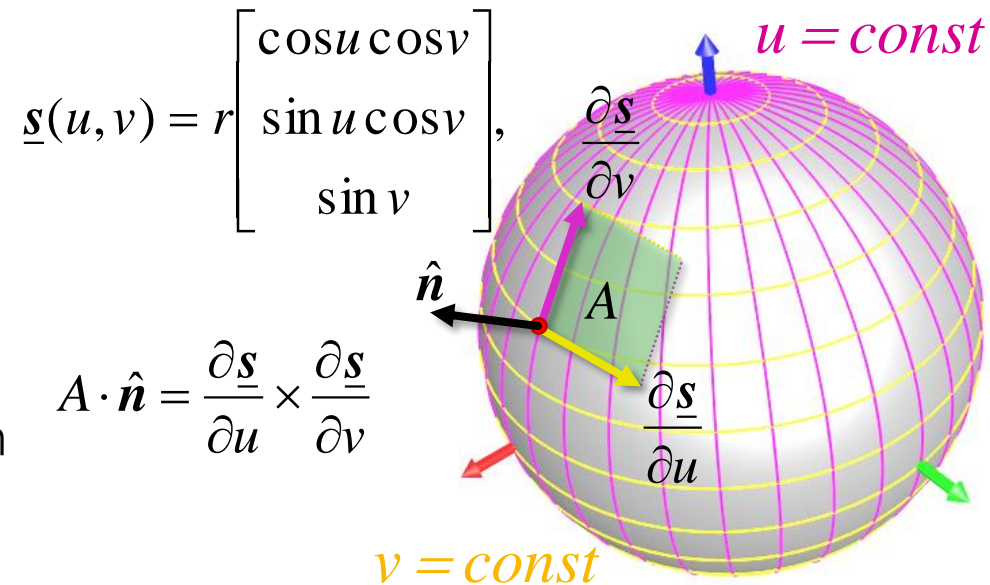
- Die Ableitung nach t wird meist mit einem Punkt auf der Funktion abgekürzt
- Beim Ableiten der Kurve werden beide Koordinatenfunktionen abgeleitet
- Es resultiert ein Vektor, der dem Geschwindigkeitsvektor eines Punktes entspricht, der sich entlang der Kurve mit t als Zeit interpretiert bewegt.
- Der Vektor heißt Tangentenvektor und spannt zusammen mit dem Kurvenpunkt die Tangente auf



$$\underline{c}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} \cos t \\ \sin t \end{pmatrix}, t \in [0, 2\pi]$$

$$\dot{\underline{c}}(t) = \begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \end{pmatrix} = \begin{pmatrix} -\sin t \\ \cos t \end{pmatrix}$$

- Parametrische Flächen werden über zwei Parameter u und v parametrisiert
- Der Parameterraum ist zweidimensional
- Zwei Tangentenvektoren ergeben sich aus den partiellen Ableitungen nach u und v .
- Sie spannen ein Parallelogramm der Fläche A sowie die Tangentialebene auf
- Das Kreuzprodukt der Tangentenvektoren ergibt die mit A skalierte Flächennormale



2D Parameterraum:

