Computergraphik II

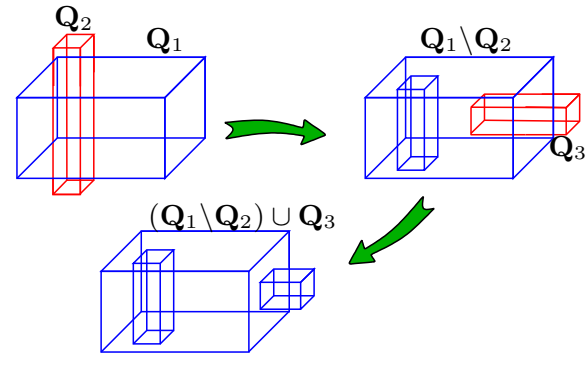
Winter 2011/2012

5 Modellierungstechniken

Versionsdatum: 3. November 2011

5.1 Solid Modeling

Motivation: Modellierung komplexer, *geschlossener* Körper (*Solids*) aus Primitiven unter Verwendung von Boolschen Operatoren.



Grundsätzliche Überlegungen zur Solid-Repräsentation

Modell-Repräsentation: Polygon, parametrisch, implizit, Subdivision

Eindeutigkeit: Zu jeder Geometrie sollte es nur eine Repräsentation geben

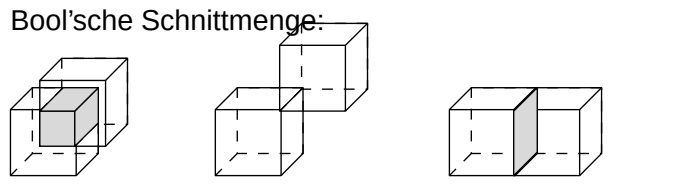
Repräsentationsformen: *Volumen-basiert* (alle Punkte **im** Objekt) oder *Oberflächen-basiert (boundary-representation, b-rep)* (Punkte **auf** Objektoberfläche)

Übliche Einschränkung: Polygonale b-reps

Boolsche Operatoren

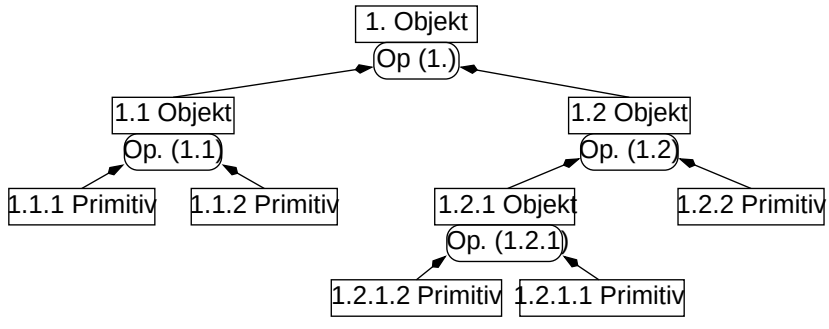
Ziel: Einsatz boolscher Operationen zur Erzeugung komplexen Geometrien

Problem: Standard Boolsche Mengen-Operatoren erzeugen aus Solids u.U. keine Solids



Regularisierte boolsche Operatoren: Entfernen von Nicht-Solid Ergebnissen

Binärer Konstruktionsbaum: Sukzessive Anwendung binärer boolscher Operationen



Ansatz: Umsetzung Boolscher Operationen

Ziel: Ermittlung der b-rep einer Boolschen Operationen

Zerlegung in Teilpolygone: Auftrennen aller Polgone an den Schnittkanten mit den Polgonen des anderen Primitives

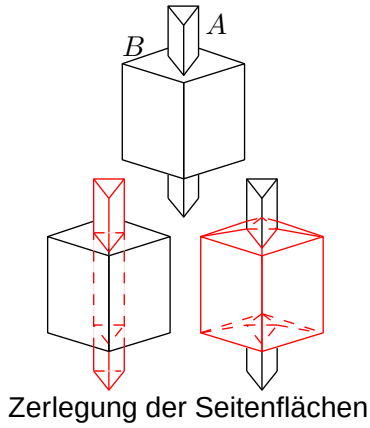
Innen-Außen-Klassifikation: Jedes entstandene Teilpolygone liegt bzgl. des anderen Primitives innen oder aussen

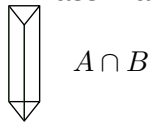
Zusammensatzung des Solids:

$A \bigcap B$: Alle Innen-Polygone von A und B

$A \bigcup B$: Alle Aussen-Polygone von A und B

$A \backslash B$: Alle Aussen-Polygone von A und alle Innen-Polygone von B





Erzeugung komplexer, selbstähnlicher Geometrien (Berge, Pflanzen)

Erzeugungsprozess: Zur Erzeugung fraktaler Geometrien werden benötigt:

1. Ausgangselement aus einer Menge von Elementen
2. Rekursion (bis zu Rekursionstiefe): Ersetze aktuelles Element durch anderes (komplexeres) Element (Ersetzungsregel)
3. Geometrische Interpretation

Beispiel: Koch'sche Schneeflocke

Elementmenge: Polygonzüge

Ausgangselement: Polygonzug mit vier Kanten

Regel: ersetze jede Kante des aktuellen Polygons durch Ausgangselement

Ausgangselement 1. Rekursion 4. Rekursion

Stochastische Fraktale

Stochastische Fraktale: Einsatz von Zufallsgrößen in Ersetzungsregel

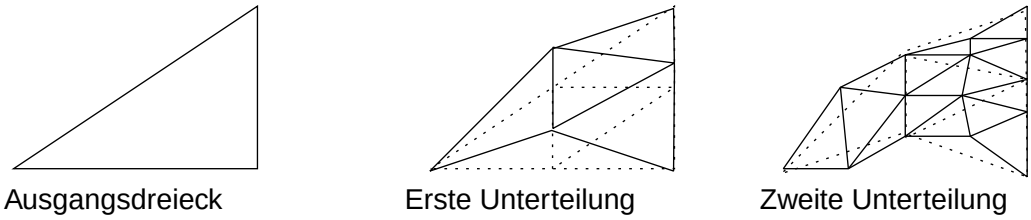
Fraktale Berge:

Elementmenge: Reguläre $3D$-Dreiecksmeshes (Skizze!)

Ausgangselement: Ein beliebiges Dreieck in der $x-y-$Ebene

Regel:

- Unterteile jedes Dreieck in vier ähnliche Dreiecke
- stochastische Störung der neuen Eckpunkte (z-Richtung)



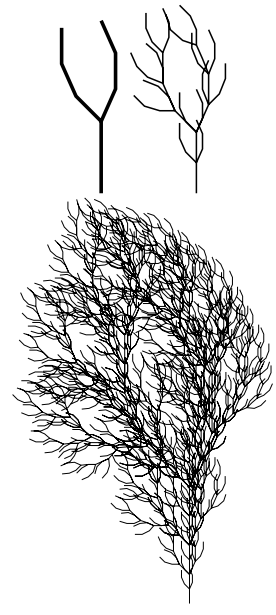
Ausgangsdreieck Erste Unterteilung Zweite Unterteilung

Modellierung verzweigter Geometrien (z.B. Pflanzen)

Grundidee: Rekursive Anwendung von Ersetzungsregeln für geom. Primitive

Grundlage: Formale Sprachen ($\rightarrow$ EI-I):

- **Alphabet V:** Beschreibt geom. Primitive, Transformationen & Stack-Befehle (Verzweigungen)
- **Wortmenge V^*:** Menge aller Kombinationen aus Elementen aus V
- **Regeln:** Für Elemente des Alphabets: $a \mapsto p(a) \in V^*$, $\forall a \in V$ bzw. für Worte: $\omega = a_1 a_2 a_3 \ldots \mapsto p(\omega) = p(a_1)p(a_2)p(a_3)...$
- **Axiom:** Ausgangselement $\omega_0 \in V^*$ der Rekursion



Beispiel: Alphabet bestehend aus

a	zeichne Linie (initiale Richtung: oben)
$+, -$	Richtungswechsel um $\pm 22.5°$
$[,]$	Stackingbefehl (Push- und Pop-Stack)

Axiom a und Regel $p(a) = aa+[+a-a-a]-[-a+a+a]$, sonst $p(x) = x$

Geometrische Interpretation: Elemente des Alphabetes stehen für geometrische Primitive, Transformationen oder Stack-Befehle

Algorithmus konkret zum vorherigen Beispiel:

```
void InterpreteLSystem ( char* word, Point P, Vector d) {
  moveto(P);        // Startposition
  while ( (*word) != EndOfWord ) {
    switch ( *word ) {
      case 'a':  P = P + d;     lineto(P);      break;
      case '+':  d = rotate(d,  phi);           break;
      case '-':  d = rotate(d, -phi);           break;
      case '[':  InterpreteLSystem(word , P, d);
                 moveto(P);                      // Zeichenpos. VOR '['
                 word = FindMatchingBrace(word); // ueberspringe [....]
                 break;
      case ']':  return;
    }
    word ++;          // naechster Buchstabe im Wort
  }
}
```

Erinnerung: TP-Flächen verallgemeinern ein Kurvenschema

Problem: Angabe von $n \cdot m$ Kontrollpunkten unhandlich/nicht problemorientiert

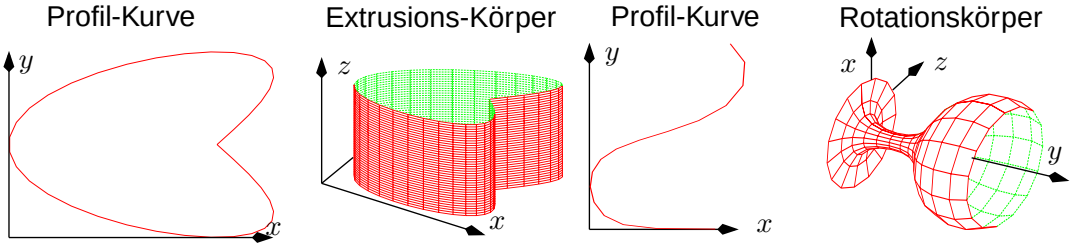
Sweep-Fläche: Alternativer Kurvenansatz:

Kurven: Bewege *Profil-Kurve* $\mathbf{S}(u)$ entlang eines *Pfades* $\mathbf{C}(v)$.
Variante: Veränderung der Profil-Kurve $\mathbf{S}$ in Abhängigkeit von v

Ausprägungen: Konkrete Kurven geben spezielle Klassen von Flächen:

Rotationskörper: Der Pfad $\mathbf{C}$ ist ein Kreis

Extrusionskörper: Geschlossenes 2D-Profil $\mathbf{S}$; Pfad $\mathbf{C}$ ist häufig eine Linie.



Beispiel: Einfacher Extrusionskörper

Ziel: Bestimme Polygon-Approximation des herzförmigen Extrusionskörpers

Profil-Kurve $\mathbf{S}(u),\ u \in [0,1]$ und Pfad-Kurve $\mathbf{C}(v),\ v \in [0,1]$ gegeben durch

$$\mathbf{S}(u) = \begin{pmatrix} s_x(u) \\ s_y(u) \end{pmatrix} = \begin{pmatrix} \cos(2\pi u) \cdot \sin(\pi u) \\ \cos(2\pi u) \end{pmatrix}, \quad \mathbf{C}(v) = \begin{pmatrix} 0 \\ 0 \\ v \end{pmatrix}$$

Parametrische Darstellung: $\mathbf{F}(u,v) = \begin{pmatrix} s_x(u) \\ s_y(u) \\ v \end{pmatrix}, \quad (u,v) \in [0,1]^2$

Polygonalisierung:

- Gitter im Parametergebiet $[0,1]^2$ mit Auflösung $(N_u+1) \times (N_v+1)$:

  $\mathbf{P}_{ij} = \begin{pmatrix} i\Delta_u \\ j\Delta_v \end{pmatrix},\ i \in \{0,\ldots,N_u\},\ j \in \{0,\ldots,N_v\}$ wobei $\Delta_u = \frac{1}{N_u},\ \Delta_v = \frac{1}{N_v}$

- Polygone ergeben sich zu dann zu:
  - $(\mathbf{F}(\mathbf{P}_{i,j}), \mathbf{F}(\mathbf{P}_{i+1,j}), \mathbf{F}(\mathbf{P}_{i+1,j+1}), \mathbf{F}(\mathbf{P}_{i,j+1}))$

Allgemeine Extrusionskörper

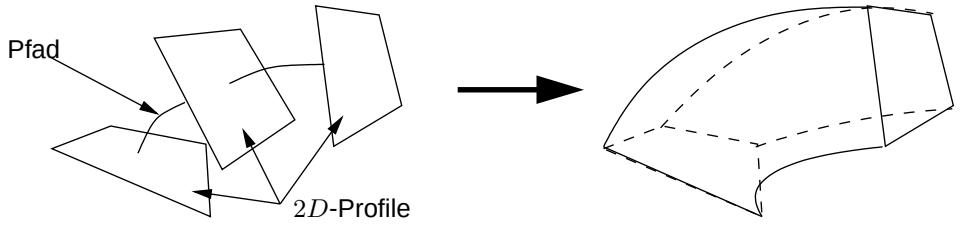
Ausgangslage: Allgemeine Extrusionsfläche sind gegeben durch

$2D$**-Profile:** Ebene, geschlossene Kurven
$\mathbf{S}_i : u \mapsto \mathbf{S}_i(u) \in \mathbb{R}^2,\ u \in [0,1],\ \ i = 1, \ldots, n$

Pfad: Kurve, entlang derer die Profilkurven $\mathbf{S}_i$ aufgereiht werden:
$\mathbf{C} : v \mapsto \mathbf{C}(v) \in \mathbb{R}^3,\ v \in [0,1]$

Zuordnung: Für jedes $\mathbf{S}_i$ werden zusätzlich benötigt:
- ein Ausrichtungspunkt $\mathbf{M}_i$ (z.B. Schwerpunkt)
- ein Pfad-Parameter $v_i \in [0,1]$ mit $\mathbf{M}_i \hat{=} \mathbf{C}(v_i)$ und $v_{i-1} < v_i < v_{i+1}$

Arbeitsschritte: 1. Ausrichtung der Profile am Pfad (*Frenetsches Dreibein*)
2. Interpolation zwischen den Profilen (z.B. B-Splines)



Algorithmus: Polygon-Approximation des Extrusionsobjektes

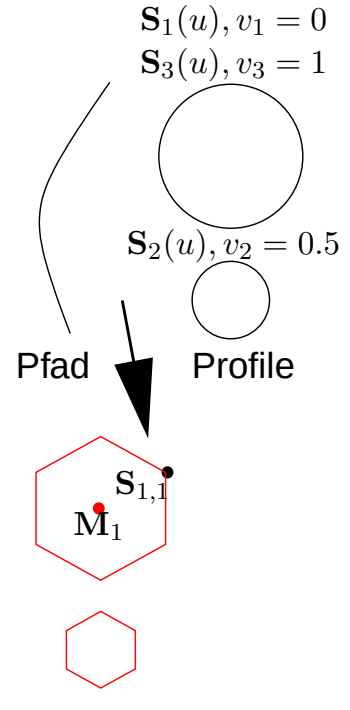
Gegeben: 1. Pfad-Kurve: $\mathbf{C} : v \mapsto \mathbf{C}(v) \in \mathbb{R}^3,\ v \in [0,1]$
2. Ebene $2D$-Profile-Kurven $\mathbf{S}_i : u \mapsto \mathbf{S}_i(u) \in \mathbb{R}^2,\ u \in [0,1],\ \ i = 1, \ldots, n$ mit
   2.1. Ausrichtungspunkt $\mathbf{M}_i$
   2.2. Pfad-Parameter $v_i \in [0,1]$ mit $\mathbf{M}_i \hat{=} \mathbf{C}(v_i)$
   2.3. Auflösung für den Pfad $r_\mathbf{C}$ und Profile $r_\mathbf{S}$

Linearisierung der Profile: Approximation der Profile mit $r_\mathbf{S}$ Punkten bzgl. Schwerpunkt $\mathbf{M}_i$:

$$\mathbf{S}_{i,j} = \mathbf{S}_i(u_j) - \mathbf{M}_i \in \mathbb{R}^2$$

$$\text{mit } u_j = \frac{j}{r_\mathbf{S}}$$

$\mathbf{M}_i$ liegt damit im Ursprung

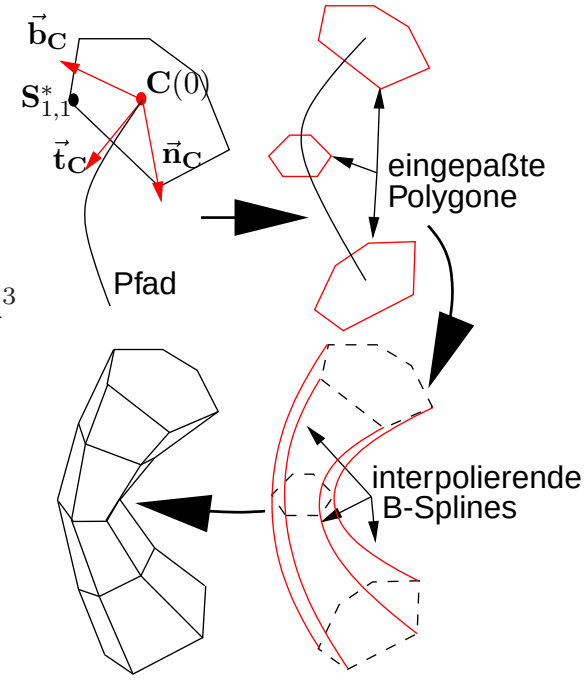


Algorithmus: Polygon-Approximation des Extrusionsobjektes (Forts.)

Einpassen der Polygone für Profil $\mathbf{S}_i$ am Frenet'schen Dreibein in $\mathbf{C}(v_i)$:

$$\mathbf{S}_{i,j} = \begin{pmatrix} x_{i,j} \\ y_{i,j} \end{pmatrix} \mapsto \mathbf{S}^*_{i,j} \text{ mit}$$

$$\mathbf{S}^*_{i,j} = \mathbf{C}(v_i) + x_{i,j}\vec{\mathbf{n}}_\mathbf{C}(v_i) + y_{i,j}\vec{\mathbf{b}}_\mathbf{C}(v_i) \in \mathbb{R}^3$$

B-Spline Interpolation der Punkte $\mathbf{S}^*_{i,j}, i = 1, \ldots, n$ durch Kurve $\mathbf{S}^*_j(v)$ (B-Spline, Catmull-Rom)

Linearisierung der B-Splines durch Polygonzüge bestehend aus $r_\mathbf{C}$ Strecken