# Seminar aus Computergraphik
# SS 03

## Bakkalaureatsarbeit

**Peter Rautek**
**Matrikelnummer 9925695**

# Implizites Modellieren

# Überblick

In dieser Arbeit wird ein Überblick über nicht-traditionelle Modellierungsmethoden gegeben und im speziellen auf implizite Modellierung eingegangen. Weiters wird der BlobTree erläutert, der eine Datenstruktur zur einheitlichen Organisation von impliziten Primitiven und Operationen, die auf diesen Primitiven ausgeführt werden können, darstellt.

# Inhaltsverzeichnis

**ÜBERBLICK ..... 2**

**1. EINLEITUNG ..... 3**

**1.1. TRADITIONELLES VS. NICHT-TRADITIONELLES MODELLIEREN ..... 3**
**1.2. PROZEDURALE–, IMPLIZITE– UND PUNKTMODELLIERUNG ..... 3**

**2. IMPLIZITE MODELLIERUNG ..... 4**

**2.1. PARAMETRISCHE UND IMPLIZITE FLÄCHEN ..... 4**
**2.2. FELDFUNKTIONEN UND ISOFLÄCHEN ..... 5**
**2.3. DER BLOBTREE ..... 6**
2.3.1. KNOTEN DES BLOBTREE ..... 6
2.3.2. VISUALISIERUNG ..... 10

**3. ZUSAMMENFASSUNG ..... 14**

**4. REFERENZEN ..... 15**

# 1. Einleitung

## 1.1. Traditionelles vs. nicht-traditionelles Modellieren

Mit traditionellem Modellieren respektive traditionellen Modellen sind Modelle der Computergraphik gemeint, die auf Polygonbeschreibungen beziehungsweise auf Beschreibungen von parametrischen Flächen (patches) basieren. Diese Methoden sind weit verbreitet und finden ihre Anwendung in allen Gebieten der Computergraphik. In Echtzeitanwendungen sind sie State-Of-The-Art, da sie sich auf kommerziell erhältlicher Grafikhardware schnell visualisieren lassen. Nicht traditionelle Modellierung bezeichnet hingegen, die Beschreibung von Objekten durch andere Methoden, die typischerweise nicht in Hardware umgesetzt sind und sich somit nur sehr rechenaufwendig (CPU-lastig) und daher relativ langsam darstellen lassen. Diese Modelle besitzen aber dafür einige andere Vorteile, wodurch sie sich mit steigender Rechenleistung herkömmlicher Personal-Computer wachsender Beliebtheit erfreuen. Die Grenze zwischen diesen zwei Bereichen der Modellierung in der Computergraphik ist keineswegs scharf und wird immer wieder durch neue Hardwareimplementierungen verschoben. In dieser Arbeit sind mit nicht-traditionellen Modellen, prozedurale, implizite, und Punkt-Modelle gemeint.

## 1.2. Prozedurale–, Implizite– und Punktmodellierung

Prozedurale Modelle wurden Mitte der Achziger-Jahre verwendet, um realistische Texturen für Holz, Stein, Sand und andere Naturmaterialien zu generieren. Diese Methoden wurden weiterentwickelt um ganze Modelle von Wasser, Feuer, Rauch, Nebel, etc. nachbilden zu können. Außerdem finden prozedurale Methoden ihre Anwendung in Shading Languages, die durch Pixar in RenderMan erstmals ihren Einsatz fanden.

**Abbildung 1:** Prozedural modelliertes Stadtmodell; Quelle: [Centralpictures2001]

Charakteristisch für prozedurale Modellierungsmethoden ist die Verwendung von Algorithmen oder Codesegmenten, um ein Modell zu beschreiben. Diese Algorithmen können sowohl deterministischer als auch stochastischer Natur sein, wobei vom Designer meist eine Menge von Parametern einstellbar ist. So können zum Beispiel bei einem prozeduralem Nebelmodell typischerweise nur sehr wenige Parameter wie zum Beispiel die Stärke und die Ausbreitung des Nebels gewählt werden, wobei bei einem prozeduralem Stadtmodell (siehe Abbildung 1) vergleichsweise viele Parameter eingestellt werden können.

Prozedurale Modellierung kann sowohl auf physikalischer Simulation basieren, als auch rein künstliche Effekte erzeugen. Bekannte prozedurale Modelle sind Fraktale, grammatikbasierte Modelle, etc. Diese Methoden werden ausführlich in [Ebert1998], [Peitgen1992], [Prusinkiewicz1990], [Smith1984] und [Hart1995] beschrieben.

Ein anderer Ansatz zur Modellierung komplexer geometrischer Modelle sind implizite Flächen. Implizite Flächen sind definiert als Isoflächen die Skelettprimitive umgeben. In dieser Arbeit wird ein Überblick über implizite Modelle gegeben und eine Datenstruktur zur einheitlichen Organisation der Primitive und der Operationen, die auf diesen Primitiven ausgeführt werden können, behandelt.

Der dritte nicht–traditionelle Ansatz zur Modellierung geometrischer Objekte ist die Punktmodellierung. Hierbei geht es nicht um die graphische Repräsentation geometrischer Flächen, sondern um die direkte Visualisierung einer Punktwolke. Solche Daten ohne explizit gegebene Nachbarschaftsbeziehungen entstehen häufig bei Messungen im wissenschaftlichen Bereich. Zum Beispiel liefern Computertomographen solche Punktwolken. Typischerweise fällt bei Messungen eine regelrechte Datenflut an, die sehr schwer zu verarbeiten ist. Der Vorteil der punktbasierten Darstellung liegt darin, dass nicht der Umweg über eine komplizierte Repräsentation der Daten (wie zum Beispiel Polygonnetze) gegangen werden muss, sondern das Modell direkt dargestellt werden kann. Weiters ist kein weiterer Speicheraufwand nötig, um Kanten und Flächen abzuspeichern.

Im Folgenden beschäftigt sich diese Arbeit nicht mehr mit prozeduralen- und punktbasierten- sondern ausschließlich mit impliziten Modellierungsmethode.

# 2. Implizite Modellierung

## 2.1. Parametrische und implizite Flächen

Viele Anwendungen verwenden parametrische Flachen zur Modellierung. Parametrische Flächen sind stets der Gestalt:

$$P = (x(u,v), y(u,v), z(u,v))$$

wobei u und v die Parameter sind, die in einem gegebenen Intervall variiert werden. Zum Beispiel ist

$$P = C + rSin(\theta)Cos(\phi),\ \ rSin(\theta)Sin(\phi),\ \ rCos(\theta)),\qquad \theta \in (0,\pi)\qquad \phi \in (0,2\pi)$$

die parametrische Darstellung einer Kugel wobei die Punkte auf der Oberfläche der Kugel durch Variieren der Parameter $\Phi$ und $\theta$ in den Intervallen $[0, \pi]$ respektive $[0, 2\pi]$, gewonnen werden können. Parametrische Formeln sind demnach ein konstruktiver Ansatz um Punkte auf einer Oberfläche zu erzeugen. Diese Methode eignet sich gut für die Erstellung von Polygonnetzen, die eine Approximation für die gegebene Oberfläche darstellen, und sehr schnell in Hardware visualisiert werden können. Außerdem ist es einfach, auf diesen Flächen eine Textur aufzutragen, da der Parameterraum und der Texturraum zweidimensional sind. Die Schwierigkeit bei den parametrischen Flächen liegt eher in ihrer Erstellung. Die Definition und Modellierung parametrischer Modelle zur adäquaten Repräsentation eines Objektes stellt sich als äußerst schwierig heraus. Trotzdem finden sie ein

breites Anwendungsfeld im Zuge von Fast-Prototyping Methoden im Bereich des Design.

Bei dem genannten Problem der Modellierung setzten die impliziten Flächen an. Ihre Gestalt ist meist viel einfacher. Viele Modelle, die sich nur schwierig in parametrischer Form darstellen lassen, haben eine vergleichsweise einfache implizite Darstellung. Implizite Flächen haben die Form:

$$F(x,y,z) = c$$

Die Punkte auf der Oberfläche sind implizit gegeben. Jeder Punkt der eine Lösung der Gleichung darstellt, ist ein Punkt der impliziten Fläche.

Zum Beispiel ist

$$F(x,y,z) = x^2 + y^2 + z^2 = r^2$$

die Gleichung für eine Kugel mit Radius r. Der Punkt (1,0,0) ist eine Lösung der Gleichung, bei gegebenem Radius r=1 und somit Element der Oberfläche. Im Allgemeinen ist das Auffinden der Lösungen ein nicht trivialer Vorgang, der die Visualisierung der impliziten Flächen sehr rechenaufwendig macht. Implizite Gleichungen können sowohl begrenzte Flächen definieren, wie zum Beispiel die Oberfläche einer Kugel, als auch unbegrenzte Flächen, wie zum Beispiel eine Ebene.

## 2.2. Feldfunktionen und Isoflächen

Die implizite Funktion F(x,y,z) ist eine Abbildung: R3 -> R. Sie kann an jedem beliebigen Punkt ausgewertet werden und ordnet diesem Punkt aus dem dreidimensionalen Raum einen skalaren Wert zu. Der Wertebereich dieser Funktion wird auch skalares Feld oder Potentialfeld genannt. Die Funktion F nennt man auch Feldfunktion oder Potentialfunktion. In dieser Arbeit werden im Weiteren nur noch die Namen Feldfunktion für die Funkton F(x,y,z) und skalares Feld für den Wertebereich der Feldfunktion verwendet. Flächen die durch die Funktion F und den Wert c definiert sind heißen Isoflächen, weil sie entlang eines bestimmten Wertes im skalaren Feld verlaufen.

Wie man leicht sieht, kann man die Gleichung F(x,y,z) = c auch als

$$G(x, y, z) = F(x, y, z) - c$$

schreiben. Ein Punkt der auf der Oberfläche liegt muss also die Bedingung G(x,y,z) = 0 erfüllen. Hier wird auch der in der Literatur teilweise verwendete Name der Nullflächen (engl.: zero surface) klar.

Die Funktion G(x,y,z) definiert oft auch ein Maß für den Abstand des Punktes zur Oberfläche des gegebenen Modells. Ist G(x,y,z) < 0 so ist der Punkt innerhalb des impliziten Volumens; ist G(x,y,z) > 0 so ist der Punkt außerhalb der gegebenen Oberfläche.

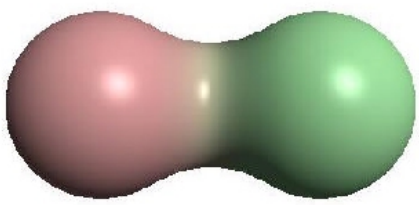

**Abbildung 2:** Isofläche, die zwei durch den Standard Blendoperator verbundene Punktprimitive umgibt; Quelle: [Wyvill2002]

Die vielfältigen Kombinationsmöglichkeiten von mehreren impliziten Flächen bilden einen großen Vorteil dieser Methode. So kann man sich zum Beispiel zwei Punkte vorstellen, die entsprechend ihrer Feldfunktionen einen Beitrag für das skalare Feld in ihrer nächsten Umgebung liefern. Addiert man nun die beiden Feldfunktionen erhält man ein neues Feld dessen Isoflächen von den einzelnen Feldfunktionen der Primitive abhängen. Sind die zwei Punkte nahe genug beisammen, so beeinflussen sich ihre skalaren Felder wechselseitig und die beiden Oberflächen verschmelzen miteinander, wie in Abbildung 2 dargestellt. Diese Art der Kombination der einzelnen Feldfunktionen wird Standard Blending genannt. Es gibt aber eine Reihe anderer Kombinationsmöglichkeiten, die noch genauer ausgeführt werden.

## 2.3. Der BlobTree

Der BlobTree ist eine Datenstruktur zur einheitlichen Organisation von Primitiven und Operationen die auf diesen Primitiven ausgeführt werden können. Der BlobTree wurde von Brian Wyvill und Andrew Guy von der University of Calgary vorgeschlagen. In ihrem Paper [Wyvill1999] beschreiben sie die grundlegende Datenstruktur und Möglichkeiten zur Visualisierung. Die Vorteile des Blobtrees sind die einheitliche Handhabung von bool'schen Operationen, Blending, Space Warp und affinen Transformationen. Alle Operationen sind Knoten des Blobtrees, die vielfältige Manipulationen und Kombinationen der in den Blättern gespeicherten Primitive zulassen.

### 2.3.1. Knoten des BlobTree

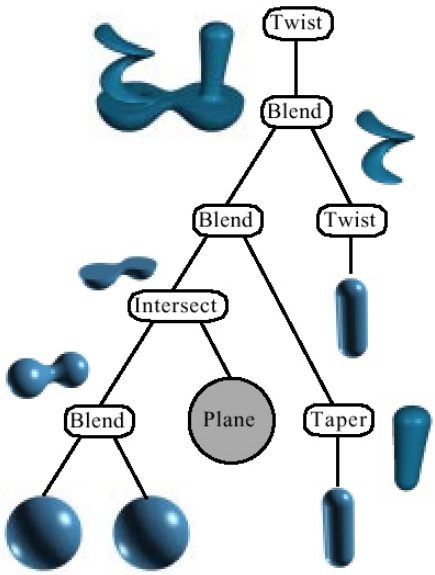


**Abbildung 3**: BlobTree des Objektes links oben; Quelle: [Wyvill2002]

Der BlobTree besitzt wie der Name schon sagt eine Baumstruktur. Seine Blattknoten enthalten geometrische Primitive die mittels unärer und binärer Operationen miteinander Verknüpft werden. Diese Operationen werden ebenfalls in Knotenform dargestellt. In Abbildung 3 wird ein Beispiel für einen BlobTree dargestellt. Er besteht aus vier geometrischen Primitiven, die durch die Operationen Blend, nicht-lineares Skalieren (Taper), Durchschnitt (Intersect) und Drehen (Twist) miteinander kombiniert werden.

Die verschiedenen Arten von Knoten und die Auswertung ihrer Feldfunktion werden im Folgenden erläutert.

#### 2.3.1.1. Blattknoten

Ein Blattknoten des BlobTree enthält ein geometrisches Primitiv und eine zugehörige Feldfunktion, die ein Maß für den Einflussbereich des Primitives ist. Die Primitive legen die Form des resultierenden Objektes wie ein Skelett fest. Man nennt sie daher auch skelettartige Elemente. Die Form der skelettartigen Elemente kann prinzipiell beliebig komplex sein. Die Auswertung der Feldfunktion F eines Blattknoten N im Punkt M, erfolgt in den meisten Fällen in zwei Schritten. Zuerst wird die kleinste Distanz r zwischen dem Punkt M und dem geometrischen Primitiv ermittelt.

Im einfachsten Fall ist das geometrische Primitiv ein Punkt und die Distanz zwischen dem Primitiv und dem Punkt M gleich der Länge des Vektors zwischen den beiden Punkten. Ist das skelettartige Element aber komplexer, kann die Auswertung dieser Distanz weitaus komplizierter sein als im einfachen Fall eines Punktprimitives. Wenn die Distanz ermittelt ist wird die Feldfunktion des Blattknoten ausgewertet.

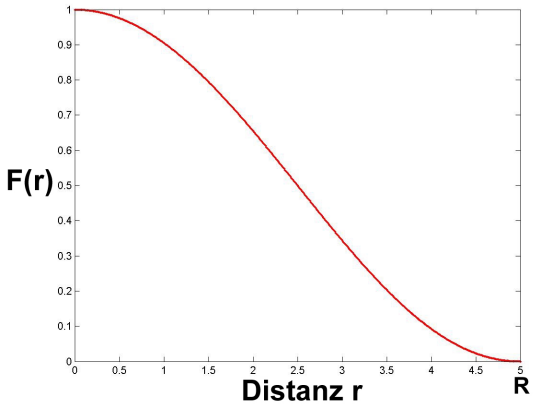


**Abbildung 4:** Kubische Feldfunktion

Die Feldfunktion hängt nur noch von der zuvor ausgewerteten Distanz r ab und hat meist die Form einer schnell abfallenden Kurve. In Abbildung 4 wird die kubische Feldfunktion dargestellt. Wobei R jene Entfernung ist, ab der die Feldfunktion den Wert Null annimmt.

#### 2.3.1.2. Boolsche Operationen

Im Folgenden bezeichne $F_A$ die Feldfunktion des Knoten A und $F_B$ die Feldfunktion des Knoten B.

Die Vereinigung und der Durchschnitt von Impliziten Flächen werden als

$$F_{A \cup B} = max\,(F_A, F_B)$$

respektive

$$F_{A \cap B} = min\,(F_A, F_B)$$

definiert.

Die Differenz zweier Flächen wird als Negation und Durchschnitt ausgedrückt:

$$F_{A-B} = F_{A \cap (-B)}$$

Man könnte allerdings obige Operationen genauso gut auf andere Weise definieren, wodurch die Unstetigkeiten, die aus der Anwendung der üblichen Formeln resultieren, vermieden werden können. Solche Formeln werden in [Pasko1995] vorgeschlagen.

### 2.3.1.3. Blending

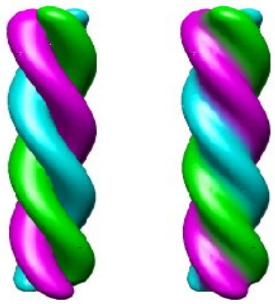

**Abbildung 5:** Links: Vereinigung der drei Primitive
Rechts: Blending; Quelle: [Wyvill1997]

Blending ist eine beliebte Methode implizit gegebene Oberflächen miteinander verschmelzen zu lassen. Die Blending Operation erzeugt im gegensatz zur Vereinigung weiche Übergänge zwischen zwei Oberflächen (siehe Abbildung 5). Das Ergebnis einer Standard Blendingoperation ist die Summe der Feldfunktionen der Operanden.

$$F_{A+B} = F_A + F_B$$

Die Generalisierung des Blending wird Super-Elliptic-Blending genannt und kann wie folgt berechnet werden:

$$F_{A \diamond B} = \left( {F_A}^n + {F_B}^n \right)^{\frac{1}{n}}$$

Super-Elliptic-Blending besitzt einige angenehme Eigenschaften:

Standard Blending ist ein Spezialfall des Super-Elliptic-Blending mit n=1.

Geht n gegen Unendlich nähert sich das Super-Elliptic-Blending der Vereinigung zweier Primitive; geht n gegen minus Unendlich nähert es sich dem Durchschnitt.

$$\lim_{n \to +\infty} \left( {F_A}^n + {F_B}^n \right)^{\frac{1}{n}} = max\left(F_A, F_B\right)$$

$$\lim_{n \to -\infty} \left( {F_A}^n + {F_B}^n \right)^{\frac{1}{n}} = min\left(F_A, F_B\right)$$

Weiters ist Super-Elliptic-Blending assoziativ. Durch die Assoziativität kann Super-Elliptic-Blending auf einfache Weise auf eine n-äre Operation generalisiert werden.

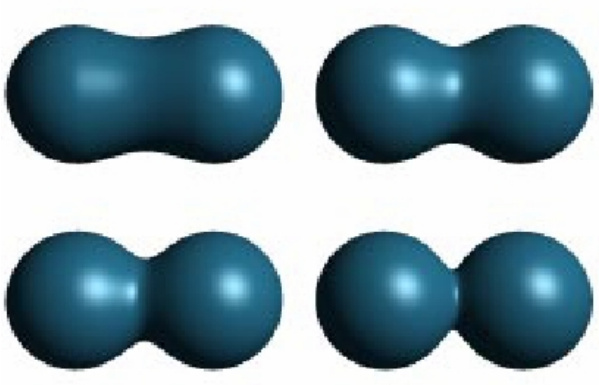

**Abbildung 6**: Interpolation zwischen Blending (links oben) und Vereinigung (rechts unten) mittels Super-Elliptic-Blending; Quelle: [Wyvill2002]

Außerdem gilt: Lässt man den Paramerter n von 1 bis Unendlich gehen, erhält man eine Interpolation der Standard Blending Operation und der Vereinigung (siehe Abbildung 6).

### 2.3.1.4. Space Warp

Der Space Warp ist der einzige unäre Operator im BlobTree. Durch Space Warps können Deformationen und affine Transformationen des Modells erreicht werden. Prinzipiell ist jede R3 -> R3 – Abbildung ein Space Warp. Da aber nicht jeder Space Warp eine sinnvolle Operation darstellt und es im Allgemeinen nicht möglich ist die Oberflächennormalen der deformierten Fläche zu berechnen, werden einige Space Warps speziell behandelt.

### 2.3.1.5. Affine Transformationen

Affine Transformationen werden im BlobTree als Space Warp Knoten behandelt. Es macht mathematisch gesehen keinen Unterschied, ob man eine Transformation oder einen Space Warp auf die Primitive des modellierten Objekts anwendet. In jedem Fall wird das geometrische Primitiv in einer kanonischen Form modelliert und anschließend in Weltkoordinaten transformiert. Der Vorteil des Space Warp ist jedoch das er auf beliebige und vor allem konsistente Weise mit anderen Warps kombiniert werden kann.

### 2.3.1.6. Barr Deformationen

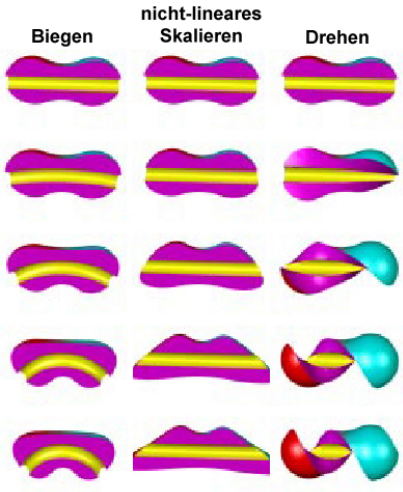


**Abbildung 7:** Barr Deformationen; Quelle: [Wyvill1997]

Die Barr Deformationen wurden 1984 von Alan Barr [Barr1984] eingeführt. Es handelt sich dabei um die Operationen Biegen (Bend), nicht-lineares Skalieren (Taper) und Drehen (Twist) (siehe Abbildung 7). Auch diese Operationen sind Spezialfälle des Space Warp.

Das nicht-lineare Skalieren wird meist linear entlang einer Achse angewandt. Im Allgemeinen können aber beliebige Skalierungen vorgenommen werden. Die Skalierung, linear entlang der z Achse, bedeutet, dass sowohl x als auch y Koordinaten verändert werden. In der Formel

$$w(x,y,z) = \left\{ \begin{array}{c} x * (1 + z * warpFactor\ ) \\ y \\ z \end{array} \right\}$$

wird nur x-Koordinate verändert. Sie kann aber analog für die y-Koordinate angewendet werden. Der warpFactor gibt an wie stark in Abhängigkeit von der z-Koordinate skaliert werden soll.

Beim Drehen werden die 2 Koordinaten in Abhängigkeit von der dritten gedreht. Beim Drehen um die z-Achse werden also die x und y Koordinaten wie folgt verändert:

$$w(x,y,z) = \left\{ \begin{array}{c} x * cos(\theta) - y * sin(\theta) \\ x * sin(\theta) + y * cos(\theta) \\ z \end{array} \right\}$$

Wobei der Winkel θ das Produkt von z und dem warpFactor ist. Wobei der warpFactor wieder angibt, wie stark die x- und y-Koordinate in Abhängigkeit von der z-Koordinate gedreht werden sollen.

Das Biegen erfolgt ebenfalls entlang einer Hauptachse. Wobei k angibt wie viele Grad das Objekt pro Einheitslänge gebogen wird. Beim Biegen entlang der z-Achse ist x0 jener x-Wert, um den gebogen wird. Der Winkel θ ist definiert als (x-x0)*k.

$$w(x,y,z) = \left\{ \begin{array}{c} -sin(\theta) * (y - 1/k) + x0 \\ cos(\theta) * (y - 1/k) + 1/k \\ z \end{array} \right\}$$

### 2.3.2. Visualisierung

Der BlobTree kann auf zwei Arten visualisiert werden. Durch vorhergehende Polygonisation und spätere Ausgabe auf einer 3D-Grafikkarte, oder durch Raytracing–Algorithmen.

Es gibt einige Funktionen, die in jedem Fall auf dem BlobTree ausgeführt werden müssen.

#### 2.3.2.1. Berechnung der Feldfunktion im BlobTree

Die Auswertung der Feldfunktion muss an sehr vielen Punkten durchgeführt werden. Die Werte, die die Feldfunktion liefert, sind von der Art der Knoten abhängig. Sei im Folgenden F(N,M) der Wert der Feldfunktion des Knotens N im Punkt M. Weiters seien L(N) und R(N) der linke respektive rechte Kindknoten des Knoten N.

Der BlobTree wird dabei, beginnend bei der Wurzel, wie folgt traversiert:

| N | F(N,M) |
|---|---|
| Primitiv (Blattknoten) | F(M) |
| Warp: | F(L(N), w(M))<br>wobei w(M) die Anwendung der Warp-Operation auf den Punkt M bezeichnet. |
| Blend: | F(L(N),M) + F(R(N),M) |
| Vereinigung: | max(F(L(N),M), F(R(N),M)) |
| Durchschnitt: | min(F(L(N),M), F(R(N),M)) |
| Differenz: | min(F(L(N),M), -F(R(N),M)) |

Aber nicht nur die Auswertung der Feldfunktion spielt bei der Visualisierung eine zentrale Rolle. Es müssen auch die Oberflächeneigenschaften und Normalvektoren in geeigneter Form ermittelt werden.

### 2.3.2.2. Berechnung der Normalen und Oberflächeneigenschaften im BlobTree

Die Oberflächennormalen können aus den gegebenen Normalen der Primitive errechnet werden. Zum Beispiel sind die Oberflächennormalen der Einheitskugel gegeben durch N(P) = (x, y, z).

Der Punkt P (1,0,0) hat also den Normalvektor (1,0,0). Dreht man das Vorzeichen bei der Berechnung der Normalen um, vertauscht man im Falle geschlossener Volumen die Bezeichnungen innerhalb und außerhalb des Volumens.

Bei einem Blendknoten können die Normalen als der gewichtete Durchschnitt aus den einzelnen Normalen der Kindknoten errechnet werden. Die Gewichtung erfolgt in Bezugnahme auf die impliziten Funktionen der Kindknoten. Bei boolschen Operationen werden die Normalen eines jeweiligen Kindknotens verwendet. Bei Warpoperationen erfolgt die Berechnung über den Kindknoten und die Jacobi Matrix des Warps. Die Ermittlung der Jacobi Matrix ist jedoch teilweise schwierig oder sogar unmöglich. Im Falle von Barr Deformationen sind die Matrizen in [Barr1984] beschrieben. Kann die Jacobi Matrix nicht ermittelt werden, können die Oberflächennormale numerisch durch den Gradient

$$\nabla F(x,y,z) = \frac{1}{2\delta}\begin{pmatrix} F(x+\delta,y,z) - F(x-\delta,y,z) \\ F(x,y+\delta,z) - F(x,y-\delta,z) \\ F(x,y,z+\delta) - F(x,y,z-\delta) \end{pmatrix}$$

approximiert werden. Wobei δ ein relativ kleiner Wert ist, der angibt, wie weit neben der eigentlichen Oberfläche die Feldfunktion ausgewertet werden soll. Die Berechnung des Gradienten in einem Punkt ist sehr aufwendig, da man die Feldfunktion sechs mal auswerten muss.

Die Kombination von Materialeigenschaften wie zum Beispiel Farbe, Reflektions- oder Brechungskoeffizienten, erfolgt bei Blendknoten und boolschen Operationen nach dem gleichen Schema wie die Berechnung der

Oberflächennormalen. Space Warps beeinflussen hingegen die Materialeigenschaften nicht.

### 2.3.2.3. Polygonisation

Die Polygonisation kann vor allem für das Darstellen des Modells in Echtzeit verwendet werden. Ist das Modell einmal in Polygone zerlegt, kann es schnell und effizient auf der Grafikhardware dargestellt werden. Dieses Verfahren eignet sich vor allem um dem Designer ein einfaches und interaktives Werkzeug zu bieten, mit dem er schnell und intuitiv Modelle entwerfen kann. Die Polygonisation findet heute aber auch vermehrt in der Computerspiele Branche Einsatz. Dort werden Modelle, die auf andere Weise schwierig zu modellieren sind, durch implizite Modelle beschrieben und anschließend in Polygonnetze verschiedener Komplexität zerlegt.

Die Polygonisation ist für eine differzierbare implizite Funktion immer möglich [Whitney1957]. Methoden dazu werden in [Kalvin1992] und [Schroeder1996] behandelt und in [Ning1993] analysiert.

Schwierigkeiten treten vor allem bei boolschen-Operationen auf, da der Durchschnitt, die Vereinigung und die Differenz von Primitiven zu Unstetigkeitsstellen in der Oberfläche führen können. Hier wird ein Nachbearbeitungsschritt notwendig. Bei diesem Nachbearbeitungsschritt müssen Kanten aufgefunden werden, die Unstetigkeiten erzeugen und die Unstetigkeitsstellen angenähert werden. Dieses Verfahren wird in [Wyvill1996] näher beschrieben.

### 2.3.2.4. Raytracing

Der Nachteil der Poligonisierung liegt in der Ungenauigkeit dieser Methode. Komplizierte Oberflächen werden durch eine relativ kleine Anzahl von Dreiecken beziehungsweise Polygonen repräsentiert. Dabei müssen entweder Details unberücksichtigt bleiben, oder die Anzahl der Polygone drastisch erhöht werden, sodass der Vorgang der Poligonisierung sehr komplex wird und das Polygonnetz eventuell nicht mehr in Echtzeit dargestellt werden kann.

Polygonisierung eignet sich also hauptsächlich für rasches Prototyping. Zur genauen Darstellung impliziter Flächen eignen sich Raytracing-Algorithmen besser. Um eine Szene mittels Raytracing darstellen zu können, müssen die Schnittpunkte der Strahlen mit den Modellen berechnet werden. Sei F(x,y,z)=0 die implizite Funktion der darzustellenden Geometrie und

$$r(t) = r_0 + \Delta t$$

die parametrisierte Darstellung eines Strahles. Ein Schnittpunkt ist dann genau dort zu finden, wo F( r(t) ) = 0 gilt.

Um Lösungen der Gleichung zu finden, können Minimierungsmethoden oder Intervalanalysemethoden, wie in [Gascuel1995] und [Kalra1989] beschrieben, verwendet werden.

Alle Ansätze beruhen auf einem iterativen Ansatz. Die implizite Funktion wird an verschiedenen Stellen ausgewertet um entscheiden zu können, wo nach einer Nullstelle zu suchen ist. Die einzelnen Verfahren unterscheiden sich nur durch die Vorschrift, die angibt an welcher Stelle die Funktion als nächstes ausgewertet wird. Dieser iterative Prozess wird solange ausgeführt, bis eine Lösung gefunden wird, die im Bezug auf eine vorgegebene

Genauigkeitsschranke hinreichend gut ist. Manche Methoden verwenden zusätzlich zur Auswertung der impliziten Funktion auch deren erste Ableitung um schneller zu einer Lösung zu gelangen. Dabei werden sowohl die Funktion selbst, als auch deren erste Ableitung an einem bestimmten Punkt wie oben beschrieben ausgewertet.

Um eine Effizienzsteigerung zu erreichen, ist es hilfreich die implizite Funktion der Primitive beziehungsweise Subgraphen des BlobTrees nur dann auszuwerten, wenn sie überhaupt einen Beitrag zur Feldfunktion im Punkt der Funktionsauswertung liefern. Hier finden alle gängigen Methoden ihre Anwendung, die zur strukturierten Unterteilung des Raumes in Unterräume dienen. Man speichert in jedem Unterraum, welche Primitive respektive deren Feldfunktionen Einfluss auf diesen Bereich haben. Will man die implizite Funktion an einem gegebenen Punkt auswerten, muss man nur noch jene Primitive berücksichtigen, die sich in der näheren Umgebung dieses Punktes befinden und einen Beitrag zur Feldfunktion liefern. Eine Unterteilung des R3 kann durch einen so genannten Octree erfolgen, der den Raum in hierarchisch organisierte Voxels unterteilt.

Alternativ dazu können die Primitive auch systematisch in Bounding Boxes zusammengefasst werden. Bounding Boxes von Primitiven werden definiert als einschließendes Volumen in dem das Primitiv Einfluss hat. Die Bounding Box von einem Vereinigungsknoten aber auch von einem Blendingknoten, ist die Vereinigung der Bounding Boxes. Die Bounding Box des Durchschnittes von zwei Knoten ist der Durchschnitt der Bounding Boxes. Die Bounding Box der Differenz wird nach obigen Regeln als der Durchschnitt der einen Bounding Box mit der Negation der anderen Bounding Box der Knoten berechnet. Für Space Warp Knoten gilt, hat man eine Bounding Box B für einen Knoten N so ist die gewarpte Bounding Box w(B) ein einschließendes Volumen des gewarpten Knoten w(N).

$$\mathcal{N} \subset \mathcal{B} \Rightarrow w(\mathcal{N}) \subset w(\mathcal{B})$$

Da w(B) im Allgemeinen keine Box ist, ist die Bounding Box des Knotens w(N) gleich der Bounding Box von w(B). Die Berechnung einer konkreten Box hängt von der Art des Warps ab.

Für das Raytracing bei diesem Ansatz, schneidet man den Strahl zuerst mit den Bounding Boxes und merkt sich das Intervall vom Eintritt des Strahls bis zum Austritt des Strahls. Innerhalb dieses Intervalls schneidet man den Strahl mit Bounding Boxes die in der Ersten Bounding Box zusammengefasst sind, und so fort, bis man schließlich den Strahl mit den einzelnen Primitiven schneidet.

**Abbildung 8**: Links: BlobTree eines Kerzenständers; Rechts: Mittels Raytracing visualisierte Szene die mit einem BlobTree modelliert wurde. Quelle: [Wyvill2002]

In Abbildung 8 ist ein Beispiel für die Verwendung des BlobTrees zu sehen. Es wurden drei, mit Halbkugeln abgerundeten Zylindern vereinigt. Dieser Subgraph wird dann gedreht und mit zwei anderen Primitiven mittels Super-Elliptic-Blending kombiniert.

# 3. Zusammenfassung

In dieser Arbeit wurden einige nicht-traditionelle Modellierungstechniken angeführt und kurz beschrieben. Danach wurde näher auf die Grundlagen von impliziten Modellen eingegangen und mit dem BlobTree schließlich eine Methode angeführt, welche es erlaubt, die Operationen Blending, Vereinigung, Durchschnitt, Differenz, Barr Deformationen, affine Transformationen und generelle Space Warps, auf implizite Primitive anzuwenden und diese somit zu komplexen Modellen zu kombinieren.

# 4. REFERENZEN


- [Ning 1993] P.Ning, J.Bloomenthal, An Evaluation of Implicit Surface Tilers, IEEE Computer Graphics and Applications, 13:6, Nov. 1993, pp. 33-41.
- [Centralpictures2001] www.centralpictures.com, 2001.
- [Barr1984] A. Barr. Global and Local Deformations of Solid Primitives. Computer Graphics (Proc. SIGGRAPH 84), pages 21–30, July 1984
- [Ebert1998] Ebert, D., Musgrave, F., Peachey, D., Perlin, K., Worley, S., Texturing and Modeling: A Procedural Approach, Second Edition, AP Professional, July 1998.
- [Gascuel1995] J.-D. Gascuel. Implicit patches: An optimised and powerful ray intersection algorithm. In Implicit Surfaces '95,Apr. 1995.
- [Hart1995] Hart, J., "Procedural Models of Geometric Detail," SIGGRAPH '95 Course 33 Notes, ACM SIGGRAPH 1995.
- [Kalra1989] D. Kalra and A. Barr. Guaranteed Ray Intersections with Implicit Functions. Computer Graphics (Proc. SIGGRAPH 89), 23(3):297–306, July 1989.
- [Kalvin1992] A.Kalvin, A Survey of Algorithms for Constructing Surfaces from 3D Volume Data, IBM Research Report RC 17600, Jan. 1992.
- [Pasko1995] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling: concepts, implemen-tation and applications. The Visual Computer, 2(8):429–446, 1995.
- [Peitgen1992] Peitgen, H., Hartmut, J., and Saupe, D., Chaos and Fractals: New Frontiers of Science, Springer-Verlag 1992.
- [Prusinkiewicz1990] Prusinkiewicz, P., Lindenmayer, A., The Algorithmic Beauty of Plants, Springer- Verlag, 1990.
- [Schroeder1996] W.Schroeder, K.Martin, W.Lorensen, The Visualization Toolkit, An Object-Oriented Approach To 3D Graphics, Prentice Hall, 1996.
- [Smith1984] Smith, A., "Plants, Fractals and Formal Languages," Computer Graphics (SIGGRAPH '84 Proceedings), ACM SIGGRAPH, 1984.
- [Whitney1957] H.Whitney, Elementary Structure of Real Algebraic Varieties, Annals of Mathematics 66:3, pp. 545-556, 1957.
- [Wyvill1996] B. Wyvill and K. van Overveld. Polygonization of Implicit Surfaces with Constructive Solid Geometry. Journal of Shape Modelling, 2(4):257–273, 1996.
- [Wyvill1997] B. Wyvill and K. van Overveld, Warping as a Modelling Tool for CSG/Implicit Models, Shape Modelling Conference, University of Aizu, Japan, IEEE Society Computer Press ISBN 0-8186-7867-4, pages 205-214,1997
- [Wyvill1999] B.Wyvill, E.Galin, and A.Guy, Extending The CSG Tree. Warping, Blending and Boolean Operations in an Implicit Surface Modeling System. Computer Graphics Forum 18, 2 (June), 149–158, 1999.
- [Wyvill2002] B. Wyvill and D. Ebert and M. Gross. Non-Traditional Modeling. SIGGRAPH '2002, Course \#41, 2002.