Dynamische NURBS

Fussen Alvaro

June 10, 1997

Diese Arbeit gibt den Inhalt des Vortrages über Dynamische NURBS wieder, wie sie im Artikel von *Hong Qin* und *Demetri Terzopulous, 'D-NURBS: A Physics-Based Framework for Geometric Design', IEEE Transactions on visualization and computer graphics, Vol. 2, No. 1, März 1996* vorgeschlagen werden. Da Kenntnisse über Splines und NURBS nicht vorausgesetzt werden, geben die ersten beiden Kapitel einen kleinen Einblick in diese Themen.

1 B-Spline Kurven

Ziel ist es, eine möglichst effiziente aber denoch flexible Beschreibung von Kurven anzugeben. Einerseits soll die Kurve einfach zu modellieren sein, andererseits soll der Anwender die Kriterien über die Glattheit der Kurve angeben können.

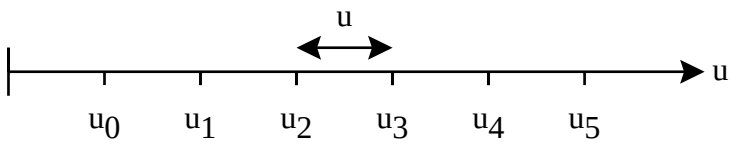
B-Spline Kurven erfüllen diese Anforderungen weitgehend. Sie sind folgendermassen definiert:

$$c(u) = \sum_{i=0}^{n} d_i B_i^k(u)$$

Dabei sind die d_i Kontrollpunkte, die $B_i^n(u)$ Basen des Grades k.

Die Kurve $c(u)$ ist unendlich glatt, ausgenommen an den Stellen $c(u_k)$ für $i = 0, \cdots, n$ auf der Kurve, wo sie nur C^{k-1} stetig ist.

$c(u)$ wird über alle $u_i, i = 1, \cdots, n$ ausgewertet:



Die Knoten u_k im Parameterraum sind äquidistant.

Sollen die Punkte auf $c(u)$ zwischen den Kontrollpunkten $[d_2, d_3]$ ausgewertet, läuft der Parameter u von u_2 bis u_3.

Die Idee der Basen besteht darin, dass nur eine begrenzte Anzahl der Kontrollpunkte einen Einfluss auf die Kurve in der Nähe dieses Kontrollpunktes haben (local support).

Also

$$B_i^k(u) = 0, \forall u \notin [u_i, u_{i+k+1}]$$

Dass heisst, dass ein Punkt auf der B-Spline Kurve c(u) einen lokalen Suport von $k+1$ Kontrollpunkten hat.

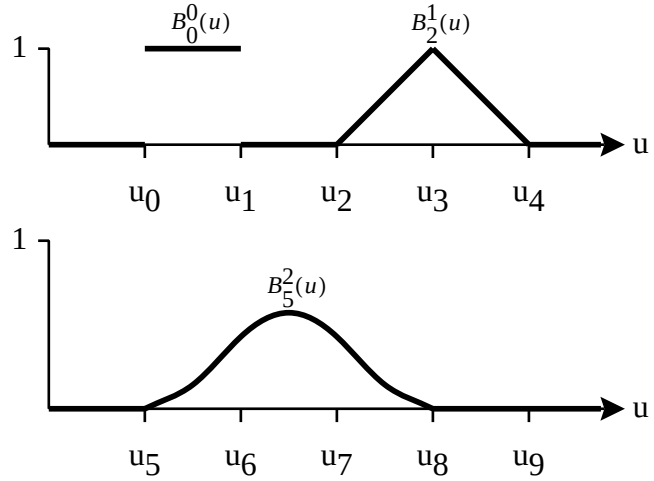
Die Basisfunktionen sind Rekursiv definiert:

$$B_i^k(u) = ((u - u_i)\frac{B_i^{k-1}(u)}{u_{i+k} - u_i} + (u_{i+k+1} - u)\frac{B_{i+1}^{k-1}(u)}{u_{i+k+1} - u_{i+1}}$$

wobei

$$B_i^0(u) = \begin{cases} 1, & u \in [u_i, u_{i+1}] \\ 0, & sonst \end{cases}$$

Die Basen für Grad 0, 1, 2 sehen dann wie folgt aus.



Eine B-Spline Kurve mit Basen des Grades 1 besteht aus dem Streckenzug der Kontrollpunkte.

Die B-Spline Basen besitzen folgende Eigenschaften:

- Partition of unity:

$$\sum_i B_i^k(u) = 1$$

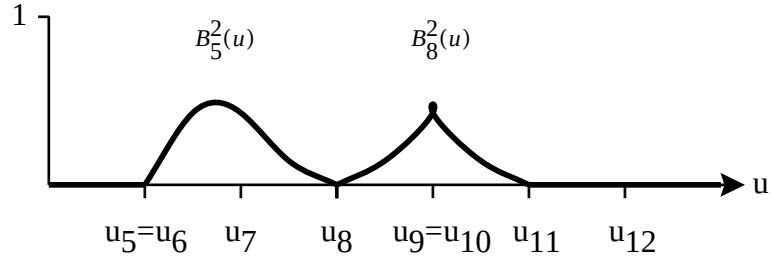
- Positivität:
$$B_i^k(u) \geq 0$$
- Lokaler Support:
$$B_i^k(u) = 0, \forall u \notin [u_i, u_{i+k+1}]$$
- Stetigkeit:
$$B_i^k \text{ ist } (k-1) \text{ mal stetig differenzierbar}$$

Im folgenden Diagramm sind die drei Basen 2. Grades eingezeichnet, die für die Punkte auf der Kurve im Intervall $[u_7, u_8]$ Support besitzen.

ncludegraphics[]Basen2.eps

Wie bereits erwähnt sind die Knoten u_n im Allgemeinen äquidistant. Allerdings können mehrere Kontrollpunkte zusammenfallen. Dieser Sonderfall ist speziell für die Endpunktinterpolation wichtig. Dabei muss aber beachtet werden, dass weniger als $k+1$ Punkte zusammenfallen, um Singularitäten zu vermeiden. Fallen p Knoten zusammen, wird die B-Spline-Kurve an dieser Stelle nur C^{k-1-p} stetig sein.

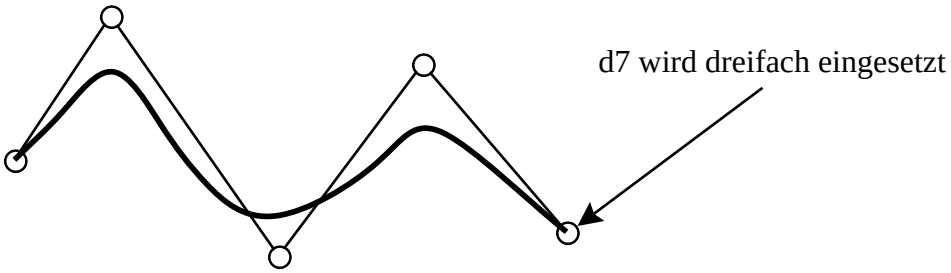
Beispiel: Quadratische B-Spline Basen mit doppelten Knoten:



Zum Abschluss nun eine B-Spline Kurve mit Endpunktinterpolation.

Grad der Kurve: 3

$d_0 = d_1 = d_3, d_4, d_5, d_6, d_7 = d_8 = d_9$



Um die Kurve effizient auszuwerten benützt man eine Generalisierung des De-Casteljau Verfahrens. Der Algorithmus ist bekannt unter dem Namen *De-Boor Algorithmus* und kann beispielsweise in *Numerical Recipes in C, The Art of Scientific Computing, von Press, Teukolsky, Vetterling und Flannery* nachgelesen werden.

2 Non uniform rational B-Spline Kurven

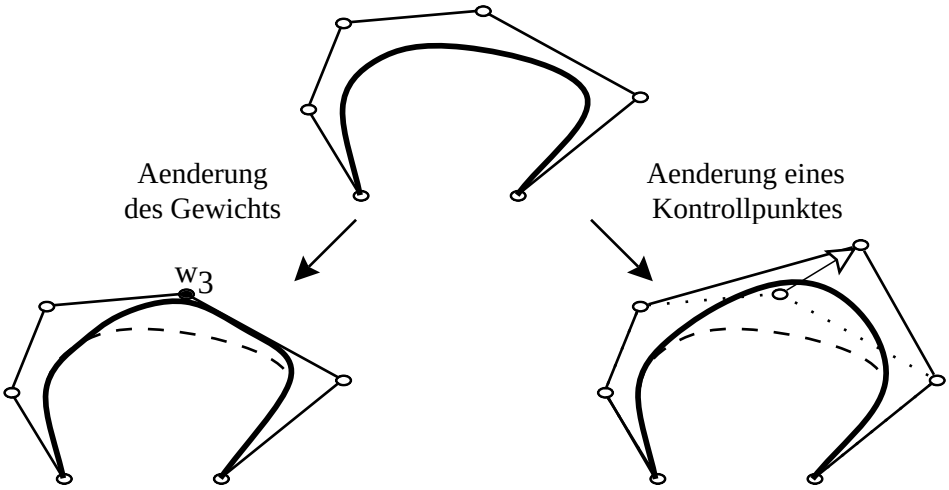
Non uniform rational B-Splines oder kurz *NURBS* sind eine Verallgemeinerung der B-Splines. Die Kontrollpunkte müssen nun nicht mehr den gleichen Abstand voneinander haben, und zudem wird für jeden Kontrollpunkt ein Gewicht $w_i \geq 0$ eingeführt.

$$c(u) = \frac{\sum_{i=0}^{n} w_i p_i C_i^k(u)}{\sum_{i=0}^{n} w_i C_i^k(u)}$$

Als Basen verwendet man die aus dem Kapitel 1 bekannten B-Spline Basen.

Die Normierung durch $\sum_{i=0}^{n} w_i B_i^k(t)$ führt dazu, dass wir es hier mit rationalen Kurven zu tun haben. Die rationale Form der Kurvengleichung ist zur Zeit die allgemeinste Art der Repräsentation von Kurven.

Obgleich NURBS sehr flexibel sind, haben sie den Nachteil, dass die Modellierung der Kurven nicht sehr intuitiv ist. Der Designer muss sich entscheiden, ob er die Kurve durch Verschieben der Kontrollpunkte, durch Variation der Gewichte oder durch Manipulation der Knotenabstände verändern will.



3 Dynamische NURBS

3.1 D-NURBS Geometrie

Ein von der Mechanik her motivierter Ansatz zur Modellierung von Kurven wurde von Terzopoulos 1987 eingeführt. Die Idee ist es, dass die Kurve durch eine Lagrangesche Dynamik beschrieben wird, um daraus die Gewichte und Kontrollpunkte zu erhalten. Die Modellierung der Kurve vereinfacht sich dabei wesentlich: Es werden einfach Kräfte an das D-NURBS gelegt, wobei die Kurve den Zustand der kleinsten Energie annimmt. Aufgrund der zugrundeliegenden Dynamik kann das Verhalten der Kurve durch eine Massen-, Steifheits- und Dämpfungsmatrix vorgegeben werden.

Da sich die Kurve in der Zeit verändert, schreiben wir die NURBS-Gleichung mit dem Parameter u und der Zeit t:

$$c(u,t) = \frac{\sum_{i=0}^{n} w_i(t) p_i(t) B_i^k(u)}{\sum_{i=0}^{n} w_i(t) B_i^k(u)}$$

Die Kontrollpunkte $p_i(t)$ und die Gewichte $w_i(t)$ schreiben wir zusammengefasst als

$$p(t) := [p_0^T \ w_0 \ \cdots \ p_n^T \ w_n]^T$$

Damit kann die Kurve in Abhängigkeit von u und $p(t)$ ausgedrückt werden. Nun wird unser NURB über folgende Gleichung an eine Dynamic gekoppelt. Die

Geschwindigkeit der Dynamischen Kurve ist also

$$\dot{c}(u,p) = J\dot{p}$$

wobei die Punkte eine Ableitung nach der Zeit bezeichnen, und $J(u,p)$ die $d \times (d+1)(n+1)$-Jakobi Matrix ist (d ist die Dimension der Kurve). Ausgeschrieben sieht J im 2-Dimensionalen Fall dann so aus:

$$J = \left[\begin{bmatrix} \frac{\partial c_x}{\partial p_{0,x}} & 0 \\ 0 & \frac{\partial c_y}{\partial p_{0,y}} \end{bmatrix} \frac{\partial c}{\partial w_0} \quad \cdots \quad \begin{bmatrix} \frac{\partial c_x}{\partial p_{n,x}} & 0 \\ 0 & \frac{\partial c_y}{\partial p_{n,y}} \end{bmatrix} \frac{\partial c}{\partial w_n} \right]$$

Sei

$$B_i(u,p) := \begin{bmatrix} N_i(u,p) & 0 \\ 0 & N_i(u,p) \end{bmatrix}$$

mit

$$N_i(u,p) := \frac{\partial c_x}{\partial p_{i,x}} = \frac{\partial c_y}{\partial p_{i,y}} = \frac{w_i B_i^k}{\sum_{j=0}^{n} w_j B_j^k}$$

Ausserdem setzen wir

$$w_i(u,p) := \frac{\partial c}{\partial w_i} = \frac{\sum_{j=0}^{n} (p_i - p_j) w_j B_i^k B_j^k}{(\sum_{j=0}^{n} w_j B_j^k)^2}$$

und fassen wir B_i und w_i zusammen

$$B(u,p) = [B_0 \cdots B_n]$$

$$W(u,p) = [w_0 \cdots w_n]$$

Damit kann die Jakobimatrix geschrieben werden als

$$J(u,p) = [B_0 w_0 \cdots B_n w_n]$$

Offensichtlich gilt

$$Jp(t) = Bp_b + Wp_w$$

wobei $p_b = [p_0^T(t), \cdots, p_n^T(t)]^T$ und $p_w(t) = [w_0(t), \cdots, w_n(t)]^T$.
Die NURBS Gleichung kann jetzt umgeschrieben werden als

$$c(u,p) = Bp_b$$

Es lässt sich beweisen, dass $Wp_w = 0$ (durch Einsetzen der Definitionen und nächtelangem Umformen) und damit kommen wir zu dem wichtigen Ergebnis

$$c(u,p) = Jp$$

3.2 D-NURBS Bewegungsgleichung

Und nun können wir dir die Bewegungsgleichung der D-NURBS angeben. Die Dynamik ist aus der Lagrangeschen Mechanik entnommen und sieht in linearisierter Form so aus

$$M\ddot{p} + D\dot{p} + Kp = f_p - I\dot{p}$$

mit

$$f_p = \int\int J^T f(u,v,t) du dv$$

und

$$I(p) = \int\int \mu J^T \dot{J} du dv$$

Ueber die Funktion f(u, v, t) kann angegeben werden, welche Kraft an der Stelle (u, v) des D-NURBS wirkt. Das nächste Kapitel geht darauf näher ein. $M(p)$ die Massenmatrix, $D(p)$ die Dämpfungsmatrix und Steifheitsmatrix. Diese können explizit angegeben werden:

$$M(p) = \int\int \mu J^T J du dv$$

$$D(p) = \int\int \gamma J^T J du dv$$

$$K(p) = \int\int (\alpha_{1,1} J_u^T J_u + \alpha_{2,2} J_v^T J_v + \beta_{1,1} J_{uu}^T J_{uu} + \beta_{1,2} J_{uv}^T J_{uv} + \beta_{2,2} J_{vv}^T J_{vv}) du dv$$

Wobei $\mu(u,v)$ eine Dichtefunktion über einen parametrisierten Bereich ist, und $\gamma(u,v)$ eine Dämpfungsfunktion darstellt. Die $\alpha_{i,j}(u,v)$ und $\beta_{i,j}(u,v)$ sind Elastizitätsfunktionen.

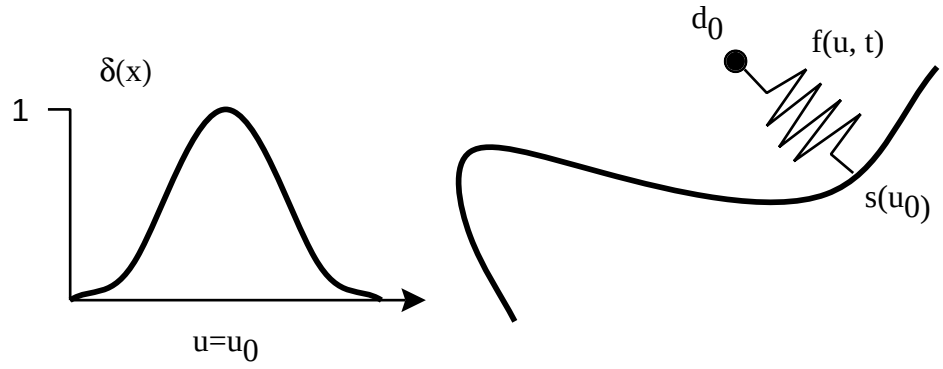
3.3 Modellierung von D-NURBS

Nun kennen wir die Bewegungsgleichung der D-NURBS und fragen uns, wie die Kurven denn nun modelliert werden. Es stehen dafür mehrere Möglichkeiten zur Verfügung. Der Designer kann Kräfte an die Kurve legen und er kann die Beschaffenheit der Kurve über die Dichte-, Dämpfungs- und Elastizitätsfunktion angeben.

Um Kräfte an das D-NURBS zu legen, müssen wir eine Kräftefunktion $f(u,t)$ definieren. Dies kann eine Gravitationskraft, eine Federkraft oder eine elektromagnetische Kraft sein. Der Phantasie wird freier Lauf gelassen. Betrachten wir die Federkraft

$$f(u,t) = \int k(d_0 - c(u,t)\delta(u-u_0))du$$

wobei δ die Einheitsfunktion Delta ist. Diese kann zum Beispiel wie eine Gaussfunktion aussehen, und hat dann eine lokal beschränkte Auswirkung auf die Kurve.



4 Anwendungen

4.1 Flächen

4.1.1 Tensor-Produkt D-NURBS Flächen

Da wir nun die berechtigte Hoffnung haben, auf angenehme Weise mit Kurven umgehen zu können, erweitern wir die D-NURBS um eine Dimension, um damit Flächen zu modellieren. Man kann sich vorstellen, dass wir nun nicht mehr ein einfaches Kontrollpolygon haben, sondern ein Netz von Kontrollpunkten. Die D-NURBS Kurve sieht dann wie folgt aus

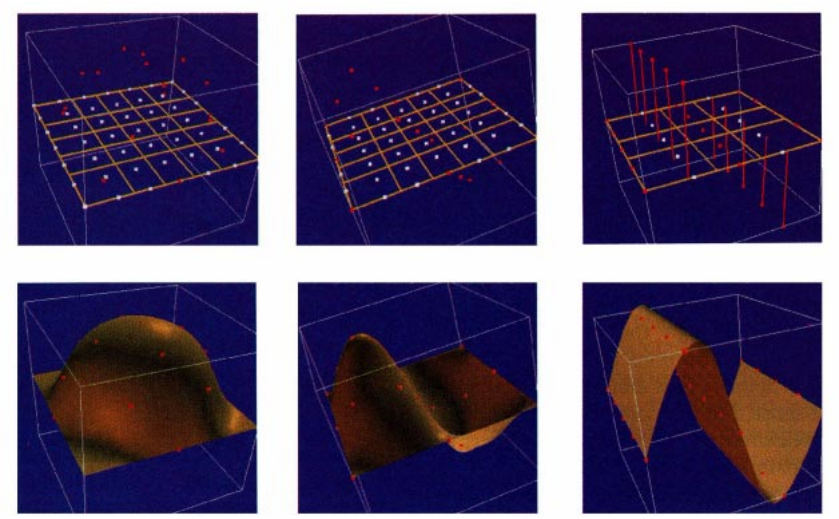
$$c(u,v,t) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{n} w_{i,j}(t)p_{i,j}(t)B_i^k(u)B_j^l(v)}{\sum_{i=0}^{n}\sum_{j=0}^{n} w_{i,j}(t)B_i^k(u)B_j^l(v)}$$

Aehnlich wie in Kapitel 3 gesehen, können wir diese Gleichung umformen und erhalten

$$c(u,v,t) = Jp$$

4.1.2 Optimales Anpassen

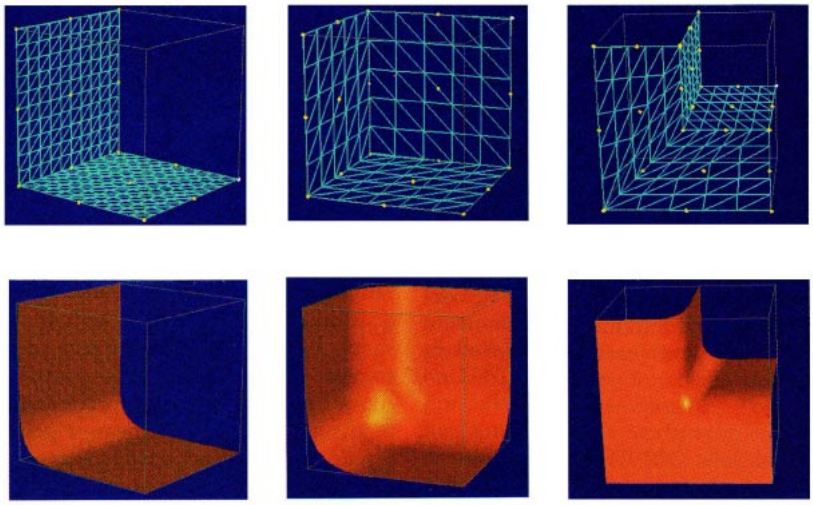
D-NURBS eignen sich auch dazu, eine Fläche durch eine Anzahl von Punkte zu legen. Mechanisch gesehen haben wir es hier mit einer dünnen Platte zu tun, die sich unter einer Deformationsenergie verformt, um diese zu minimieren. Man kann sich also vorstellen, dass bei jedem Punkt, durch die die Kurve gehen soll, Federn an die Fläche gelegt wird. Die Federkonstante gibt dann an, wie stark die Fläche an den jeweiligen Punkt herangezogen werden soll.



Auf dieser Abbildung sind einige angepasste Flächen zu sehen. Sie befinden sich nach der Simulation in einem Energiegleichgewicht. Dichte-, Dämpfungs- und Elastizitätsfunktionen für die ersten beiden Flächen sind konstant und wie folgt definiert:

$$\alpha_{1,1} = 10.0,\ \alpha_{2,2} = 10.0,\ \beta_{1,1} = 5.0,\ \beta_{1,2} = 5.0,\ \beta_{2,2} = 5.0$$

4.1.3 Rundungen



Durch die Möglichkeit, lineare (und auch nicht lineare) Zwänge an die Kurve vorgeben zu können, lassen sich mehrere Flächen-Patches recht einfach zusammenfügen. Durch die Vorgaben der Gewichte und des Kontrollnetzes, der Dynamik und der Constraints, werden Stetigkeitsbedingungen automatisch erfüllt. Das Bild zeigt wunderschöne Rundungen, wie sie jederman gerne sieht.

5 Anmerkungen

Die von der Mechanik her motivierten D-NURBS Kurven sind eine interessante Alternative zu den herkömmlichen NURBS. Die Modellierung mag tatsächlich intuitiver sein als dies bei den NURBS der Fall ist. Allerdings ist zu bezweifeln, ob es wirklich möglich ist, die Kurve exakt in die gewünschte Form zu bringen, ausser man betreibt enormen Aufwand.

Man kann sich vorstellen, dass die Formgebung von Kurven über die Dichte-, Dämpfungs- und Elastizitätsfunktion und Kräfte an die Kurve (wie zum Beispiel Federkräfte) sicher sehr mächtig ist, aber die ursprüngliche Motivation, nämlich eine einfache, intuitive Modellierung von Kurven zu ermöglichen, etwas in den Hintergrund gerät. (Diese Funktionen wirken sich ja auf die ganze Kurve aus, sofern sie nicht über Lokalitätsbedingungen verfügen, was die Sache noch einmal komplexer macht.)

Man kann sich auch überlegen, dass es ganz nützlich sein könnte, an wenigen Stellen der Kurve mit den normalen NURBS-Parameter zu arbeiten. Dies ist zwar problemlos möglich, da sämtliche Parameter für NURBS in der D-NURBS Repräsentation vorhanden sein müssen. Allerdings ist es kaum möglich, aus den Kontrollpunkten und Gewichten eine Kräfteverteilung für das D-NURBS zu berechnen. Das heisst, dass wir ein NURBS kaum auf einfache Weise wieder in ein D-NURBS transformieren können.

5.1 Verwendetes Material

- *Hong Qin* und *Demetri Terzopulous, 'D-NURBS: A Physics-Based Framework for Geometric Design'*, IEEE Transactions on visualization and computer graphics, Vol. 2, No. 1, März 1996
- *Hong Qin* und *Demetri Terzopulous, 'Dynamic NURBS with Geometric Constraints for Interactive Sculpting'* aus ACM Transactions on Graphics, Vol. 13. No. 2, April 1994
- Skript *Graphische Datenverarbeitung II* von Prof. M. Gross
- Das Buch *Numerical Recipes in C, The Art of Scientific Computing*, von Press, Teukolsky, Vetterling und Flannery
- Der Text wurde mit LaTeX unter NeXTTeX gesetzt, die Grafiken sind mit *Diagram!* erstellt worden.