Kapitel 7

Kurven

Die bisher besprochenen 2D-Objekte haben – bis auf den Kreis – den Nachteil, daß sie im weitesten Sinne ”eckig” sind. Wenn ein Objekt mit ”runder” Form verlangt wird, z.B. ein Herz, ein Schiffsrumpf, ein Kotflügel oder ein Flugzeugflügel, kann entweder ein Polygon mit einer sehr großen Anzahl von Eckpunkten verwendet werden oder eine *Kurve*. Die Kurve hat dabei zwei Vorteile:

- Sie braucht weniger Speicherplatz als das Polygon und
- sie bleibt ”glatt” wenn man sie aus der Nähe betrachtet.

Kurven entstammen dem *Computer Aided Geometric Design (CAGD)*, einer Disziplin, die sich mit der computerinternen Darstellung von Objekten beschäftigt, die aus der Geometrie stammen.

Problem: Spezifiziere und zeichne eine beliebige “glatte” Kurve.

7.1 Algebraischer Ansatz

Gegeben $n+1$ Stützpunkte.
Bestimme Polynom n-ten Grades

$$y = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_1 \cdot x + a_0$$

welches durch alle Stützpunkte läuft.
Problem: Bei der Auswertung des Polynoms treten wegen der hohen Potenzen hohe Rechenzeiten, große Rundungsfehler und Oszillationen auf.

7.2 Kubische Splines

Gegeben $n+1$ Stützpunkte. Wähle n Kurven (Polynome) kleiner Ordnung für den Verlauf innerhalb der n Intervalle.

Kurven erster Ordnung (Geraden) ergeben nur den Linienzug. Ein solcher Linienzug ist C^0-stetig, da die einzelnen Linien an den Stützpunkten dieselben x-und y-Werte haben.

Kurven zweiter Ordnung (Parabeln) kann man so wählen, daß sie an den Intervallgrenzen einmal stetig differenzierbar (C^1-stetig) sind. Geometrisch bedeutet das, daß dort nicht nur die Orte übereinstimmen, sondern auch die Steigungen. Mit Parabeln lassen sich nur Kegelschnitte darstellen; die Einsatzmöglichkeiten sind also begrenzt.

Kurven dritter Ordnung mit stetigen ersten und zweiten Ableitungen (C^2-stetig) an den Intervallgrenzen (kubische Splines) können zu einer Gesamtkurve mit weichen Übergängen an den Nahtstellen zusammengesetzt werden. "weich" heißt, daß das Auge der gezeichneten Kurve nicht mehr ansehen kann, wo die Stützpunkte liegen. Neben der Steigung ist auch die Krümmung identisch. Dies sind die einfachsten Kurven, mit denen sich bereits komplexe Formen erzeugen lassen.

Gesucht: Approximation einer Kurve $f(t)$, die durch $n+1$ vorgegebene Punkte laufen soll mithilfe von n Kurvenabschnitten.

Bemerkung: f wird in parametrisierter Form ermittelt (d.h. $x = g(t), y = h(t)$), da die explizite Form $y = f(x)$ pro x-Wert nur einen y-Wert erlaubt.

Die $n+1$ Stützpunkte seien $f(t_i)$, die resultierenden Intervalle seien $[t_i, t_{i+1}]$, die für das Intervall i zuständige Funktion sei $f_i(t)$, $1 \leq i \leq n$.

Jedes f_i hat die Form

$$\begin{aligned} f_i(t) &= f(t_i) + a_i \cdot (t - t_i) + b_i \cdot (t - t_i)^2 + c_i \cdot (t - t_i)^3 \\ & \quad t_i \leq t \leq t_{i+1},\ i = 1, \ldots, n \end{aligned}$$

Es muß gelten

$f_i(t_i)$	$=$	$f_{i+1}(t_i)$	für	$i = 1, \ldots, n-1$	gleicher Wert
$f_i'(t_i)$	$=$	$f_{i+1}'(t_i)$	für	$i = 1, \ldots, n-1$	gleiche Steigung
$f_i''(t_i)$	$=$	$f_{i+1}''(t_i)$	für	$i = 1, \ldots, n-1$	gleiche Steigungsänderung

Durch Festlegung von $f''(t_0) = f''(t_n) = 0$ wird das Gleichungssystem abgeschlossen.

Für die Folge t_i reicht jede monoton wachsende Folge. Geeignet ist z.B. die Folge der euklidischen Abstände zwischen den Stützpunkten. Statt den Abstand $d = \sqrt{dx^2 + dy^2}$ zu berechnen, verwendet man die Näherungsformel $\frac{1}{3} \cdot (|dx| + |dy| + 2 \cdot \max(|dx|, |dy|))$.

7.3 Bézier-Kurven

Spezifiziere die Kurve durch $n+1$ Stützpunkte $P_0, P_1, \ldots, P_n$. Die Kurve läuft nicht durch alle Stützpunkte, sondern wird von ihnen beeinflußt. Zeitgleich in den 1960'ern entwickelt von P. Bézier bei Renault und von P. de Casteljau bei Citroen. Damals sollten Karosserien entworfen werden, die den ästhetischen Ansprüchen der Designer und den technischen Ansprüchen der Ingenieure genügten.

$$P(t) = \sum_{i=0}^{n} B_{i,n}(t) \cdot P_i, \quad 0 \leq t \leq 1$$

Der Punkt P_i wird gewichtet mit Hilfe eines Bernstein-Polynoms.

$$B_{i,n}(t) = \frac{n!}{i! \cdot (n-i)!} \cdot t^i \cdot (1-t)^{n-i},\ i = 0, \ldots, n$$

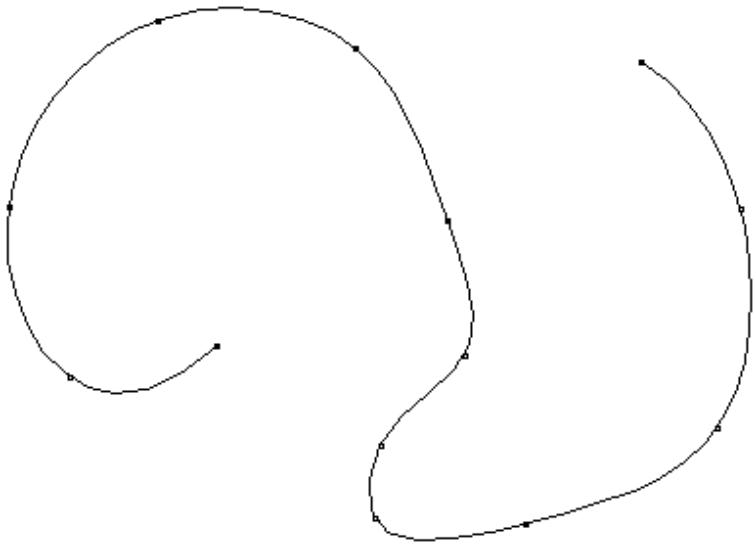

Abbildung 7.1: Vom Spline-Algorithmus erzeugte Kurveninterpolation

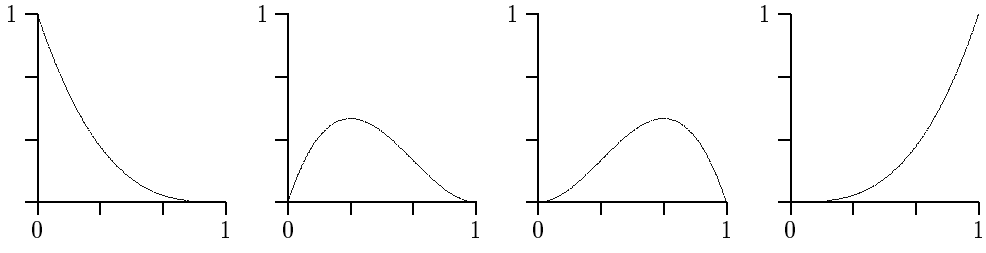

Abbildung 7.2: Verlauf der Bernsteinpolynome für n=3

Die wichtigsten Eigenschaften der Bernsteinpolynome:

- alle Bernsteinpolynome sind positiv auf $[0,1]$,
- für jedes feste t gilt $\sum_{i=0}^{n} B_{i,n}(t) = 1$,
- $B_{i,n}(t) = B_{n-i,n}(1-t)$,
- $\frac{dB_{i,n}(t)}{dt} = n \cdot (B_{i-1,n-1}(t) - B_{i,n-1}(t)), i > 0$,
- Maxima von $B_{i,n}(t)$ in $[0,1]$ an den Stellen $t = \frac{i}{n}, i = 0, \ldots, n$.

Da alle Bernsteinpolynome für $0 < t < 1$ ungleich Null sind, beeinflussen alle Stützpunkte in diesem Intervall den Kurvenverlauf.

Die Kurve beginnt im Stützpunkt P_0 tangential zur Geraden $\overline{P_0P_1}$, endet im Stützpunkt P_n tangential zur Geraden $\overline{P_{n-1}P_n}$ und verläuft innerhalb der konvexen Hülle der Stützpunkte. Somit können

mehrere Bézier-Kurven aneinandergesetzt werden durch Identifikation von Endpunkt und Anfangspunkt aufeinanderfolgender Bézierkurven. Einen stetig differenzierbaren Übergang erreicht man bei Kollinearität der Punkte $P_{n-1}, P_n = Q_0, Q_1$.

Berechnung der Bézier-Kurve nach de Casteljau

Um die aufwendige Auswertung der Bernsteinpolynome zu vermeiden, wurde von de Casteljau ein Iterationsverfahren vorgeschlagen, welches durch fortgesetztes Zerteilen von Linien den gewünschten Kurvenpunkt approximieren kann.

Es gilt:

$$P(t)_{0,n} = (1-t) \cdot P(t)_{0,n-1} + t \cdot P(t)_{1,n}$$

wobei die tiefgestellten Indizes angeben, welche Stützpunkte in die Berechnung einfließen. D.h. eine Bézier-Kurve vom Grad n läßt sich durch zwei Bézier-Kurven vom Grad $n-1$ definieren, indem für ein festes t die Punkte beider Kurven berechnet werden, und die Verbindungsstrecke im Verhältnis t geteilt wird.

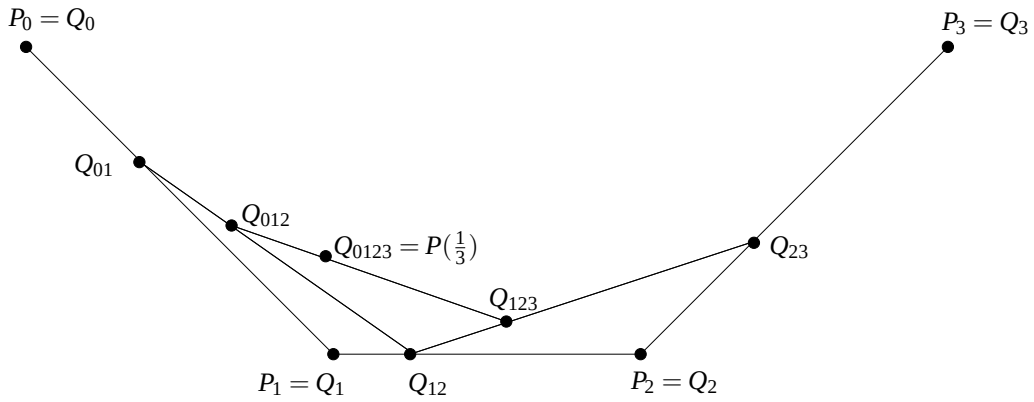

Abbildung 7.3: Approximation einer Bezier-Kurve nach de Casteljau

Abbildung 7.3 zeigt die Berechnung eines Kurvenpunktes für $t = \frac{1}{3}$. Die Indizes geben an, welche Stützpunkte beteiligt sind. Abbildung 7.4 zeigt das Ergebnis dieser Iteration für zwei aneinanderliegende kubische Bézierkurven (Rekursionstiefe 3).

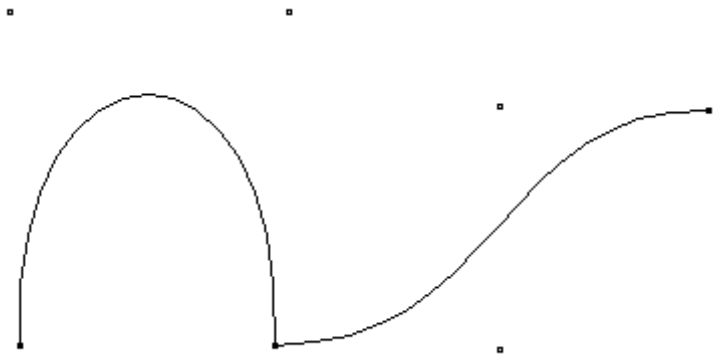

Abbildung 7.4: Vom De Casteljau-Algorithmus mit Rekursionstiefe 3 gezeichnete Bezierkurven

```
/*************************************************************************/
/*                                                                       */
/*        Zeichnen einer Bezierkurve 3. Grades nach De Casteljau         */
/*                                                                       */
/*************************************************************************/
private void bezier(
  double px0, double py0,                       // mit Stuetzpunkt p0
  double px1, double py1,                       // mit Stuetzpunkt p1
  double px2, double py2,                       // mit Stuetzpunkt p2
  double px3, double py3,                       // mit Stuetzpunkt p3
  int depth)                                    // aktuelle Rekursionstiefe
{
  double qx01,qy01,qx12,qy12,qx23,qy23,         // Hilfspunkte
         qx012,qy012,qx123,qy123,qx0123,qy0123;

  if (depth == 0)                               // Iterationstiefe erreicht
    drawLine(new Point( (int)px0, (int)py0),    // Linie zeichnen
             new Point( (int)px3, (int)py3));
    else {
      depth--;

      qx01   = (px0+px1)/2;     qy01   = (py0+py1)/2;
      qx12   = (px1+px2)/2;     qy12   = (py1+py2)/2;
      qx23   = (px2+px3)/2;     qy23   = (py2+py3)/2;
      qx012  = (qx01+qx12)/2;   qy012  = (qy01+qy12)/2;
      qx123  = (qx12+qx23)/2;   qy123  = (qy12+qy23)/2;
      qx0123 = (qx012+qx123)/2; qy0123 = (qy012+qy123)/2;
}
      bezier(px0,py0,qx01,qy01,qx012,qy012,qx0123,qy0123,depth);
      bezier(qx0123,qy0123,qx123,qy123,qx23,qy23,px3,py3,depth);
  }
}
```

7.4 B-Splines

Es sollen nun nicht alle Stützpunkte Einfluß auf den gesamten Kurvenverlauf haben und der Grad der Polynome soll unabhängig von der Zahl der Stützpunkte sein.

Spezifiziere die Kurve durch $n+1$ Stützpunkte $P_0,\ldots,P_n$, und einen Knotenvektor $T = (t_0,t_1,\ldots,t_{n+k}), t_j \leq t_{j+1}$ und Polynome $N_{i,k}$

$$P(t) = \sum_{i=0}^{n} N_{i,k}(t) \cdot P_i$$

Die Punkte P_i wirken sich nur auf maximal k Kurvenabschnitte aus und werden gewichtet durch die Basis des B-Splines, Polynome $N_{i,k}(t)$, abschnittsweise vom Grad $k-1$ ($0 \leq i \leq n$) :

$$N_{i,1}(t) = \begin{cases} 1 & \text{falls } t_i \leq t < t_{i+1} \\ 0 & \text{sonst} \end{cases} \tag{7.1}$$

$$N_{i,k}(t) = \frac{t-t_i}{t_{i+k-1}-t_i} \cdot N_{i,k-1}(t) + \frac{t_{i+k}-t}{t_{i+k}-t_{i+1}} \cdot N_{i+1,k-1}(t),\ k>1 \tag{7.2}$$

Bei Division durch Null wird der Quotient gleich 0 gesetzt.
Durch die Wahl von k und T geht jeder Stützpunkt auf einzigartige Weise in die Kurve ein. T kann entweder *uniform* (die t_j sind äquidistant) oder *nicht uniform* gewählt werden. Jede der beiden Arten kann *offen* (Anfang und Ende von T bestehen jeweils aus k-mal dem kleinsten bzw. größten t_j) oder *periodisch* (es ergeben sich periodische Gewichtungspolynome, die durch einfaches Verschieben auseinander hervorgehen) sein.

Der Knotenvektor wird häufig wie folgt gewählt für $0 \leq j \leq n+k$:

$$t_j = \begin{cases} 0 & \text{falls } j < k \\ j-k+1 & \text{falls } k \leq j \leq n \\ n-k+2 & \text{falls } j > n \end{cases}$$

Hierdurch wird t im Intervall $[0, n-k+2]$ definiert.

Beispiel: $k=3$, $n=4$ ergibt einen offenen uniformen quadratischen B-Spline mit dem Knotenvektor $T=(0,0,0,1,2,3,3,3)$. Die Stützpunkte P_0,P_1,P_2,P_3,P_4 haben nur lokal Einfluß auf den Kurvenverlauf, und zwar in den Intervallen $t \in [0,1],[0,2],[0,3],[1,3],[2,3]$.

Sonderfall

Für $k=n+1$ ergibt sich für den Knotenvektor der Sonderfall

$$T = (\underbrace{0,\ldots,0}_{k \text{ mal}},\underbrace{1,\ldots,1}_{k \text{ mal}}).$$

Die zugehörigen B-Splinefunktionen haben die schon bekannte Form

$$N_{i,k}(t) = \frac{(k-1)!}{i! \cdot (k-1-i)!} \cdot t^i \cdot (1-t)^{k-1-i},\ i=0,\ldots,k-1$$

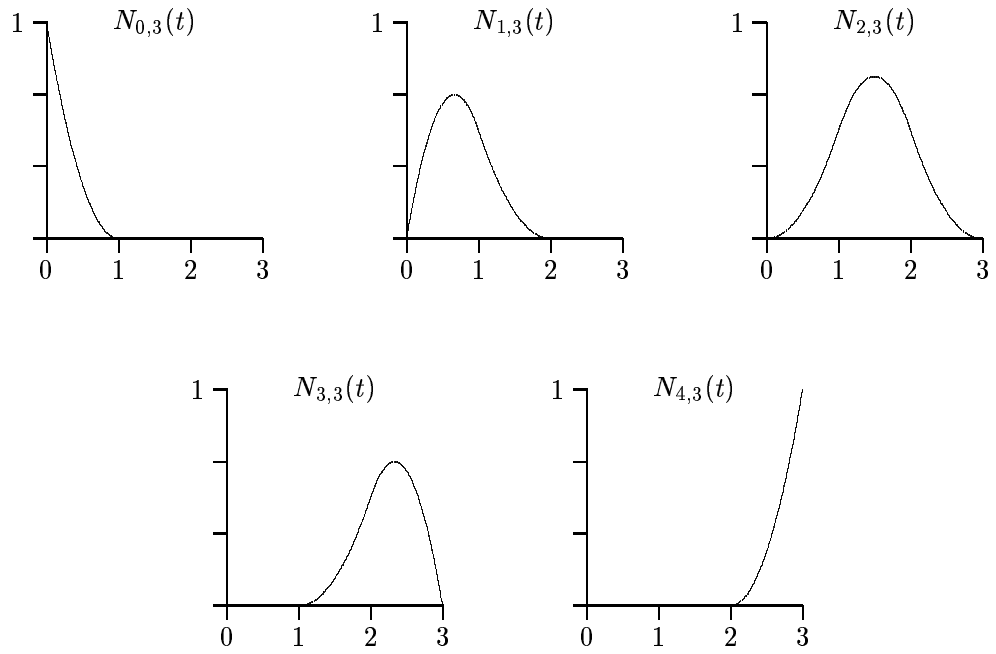

Abbildung 7.5: Verlauf der Gewichtungspolynome $N_{i,k}$ für $k = 3$

Somit lassen sich also die B-Splinefunktionen als Verallgemeinerung der Bernsteinpolynome auffassen.

Wie bei den Bernsteinpolynomen gilt auch für die B-Splinefunktionen $N_{i,k}$, daß sie positiv sind, und $\sum_{i=0}^{n} N_{i,k} = 1$.

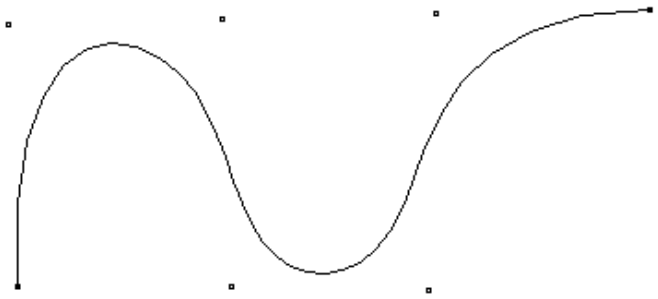

Abbildung 7.6: B-Spline mit $n = 6$, $k = 4$ und 35 Interpolationspunkten

Eigenschaften

B-Splines haben zwei weitere Eigenschaften, die für den praktischen Einsatz von großem Interesse sind:

- Sie verlaufen innerhalb des aus den Stützpunkten gebildeten konvexen Polygons (*konvexe Hülle*.

- Sie sind invariant unter *affinen Abbildungen*.

Konvexe Hülle

Die Kurvenpunkte eines B-Splines ergeben sich durch eine baryzentrische Kombination der Stützpunkte. Da die Gewichtungspolynome für alle Parameterwerte t größer oder gleich Null sind, handelt es sich sogar um eine *konvexe Kombination*. Deren Werte liegen innerhalb der konvexen Hülle der Stützpunkte.

7.5 Affine Abbildungen und Invarianz

Affine Abbildungen bestehen aus einer linearen Abbildung und einem translativen Anteil:

$$B\vec{x} = A\vec{x} + \vec{t}$$

Es handelt sich um eineindeutig umkehrbare Abbildungen, bei denen

- Geradlinigkeit,
- Zusammengehörigkeit (Objekt wird nicht zerissen),
- Parallelität und
- Teilverhältnisse auf jeder Geraden

erhalten bleiben. Wogegen

- Positionen,
- Längen,
- Winkel,
- Flächeninhalte und
- Orientierungen (Umlaufsinn)

sich ändern können.

Alle bisher in der Vorlesung behandelten Transformationen sind affine Abbildungen.

Invarianz unter linearen Abbildungen bedeutet in diesem Zusammenhang, daß es keinen Unterschied macht, ob erst die Kurve gezeichnet und dann jeder der gezeichneten Kurvenpunkte abgebildet wird oder ob die Stützpunkte abgebildet werden und die Kurve dann auf Basis diese neuen Stützpunkte gezeichnet wird.

7.6 NURBS

Über die Eigenschaften der B-Splines hinaus, wäre für den 3D-Fall auch eine Invarianz unter *projektiven Abbildungen* wünschenswert, da dort die 3D-Kurven auf den 2D-Bildschirm projeziert werden müssen.

Um möglichst große Freiheit bei den darzustellenden Formen zu haben, sollten die Kurven beliebige Kegelschnitte (also Kreise, Parabeln und Hyperbeln) annähern können, wie sie in Abbildung 7.7 gezeigt werden.

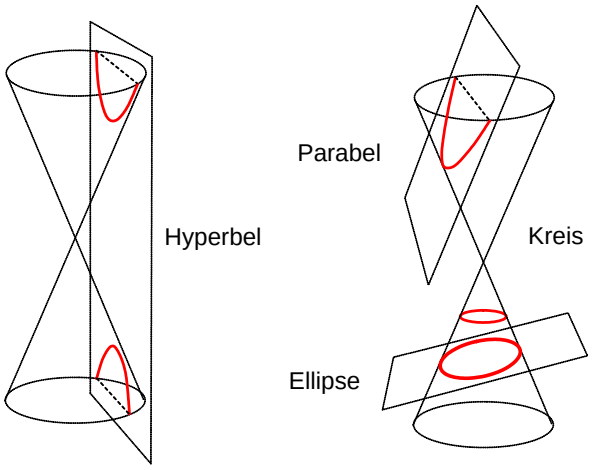

Abbildung 7.7: Kreis, Ellipse, Parabel und Hyperbel erzeugt durch Kegelschnitte

Beide Wünsche werden von der allgemeinsten Form zum Zeichnen von Kurven erfüllt. Es handelt sich dabei um *NURBS (nonuniform rational basis splines)*.

$$P(t) = \sum_{i=0}^{n} R_{i,k}(t) \cdot P_i \quad \text{mit}$$

$$R_{i,k}(t) = \frac{h_i \cdot N_{i,k}(t)}{\sum_{j=0}^{n} h_j \cdot N_{j,k}(t)}$$

Nonuniform bedeutet, dass die Knoten auf dem Knotenvektor nicht äquidistant sein müssen; *rational* bedeutet, dass die Gewichtung eines Kontrollpunktes durch den Quotienten zweier Polynome definiert wird.

Abbildung 7.8 zeigt den Einfluss der Gewichte auf den Verlauf der Kurve. Ein Gewicht $h_2 > 1$ verschiebt die Kurve in Richtung Kontrollpunkt P_2, ein Gewicht $h_2 < 1$ verringert den Einfluss von Kontrollpunkt P_2 (und erhöht dadurch den Einfluss von Kontrollpunkt P_1.

Zur Berechnung von NURBS-Kurven bedient man sich der homogenen Koordinaten: Zunächst wird ein Punkt $P_i(x_i, y_i)^T$ in seine homogenen Koordinaten $P(x_i, y_i, 1)^T$ überführt und dann mit seinem Gewicht h_i multipliziert. Die so entstandenen Punkte $P_i'(w_i \cdot x_i, w_i \cdot y_i, h_i)^T$ werden nun mit den Gewichtspolynomen $N_{i,k}$ verarbeitet: $P'(t) = \sum_{i=0}^{n} N_{i,k}(t) \cdot P_i'$

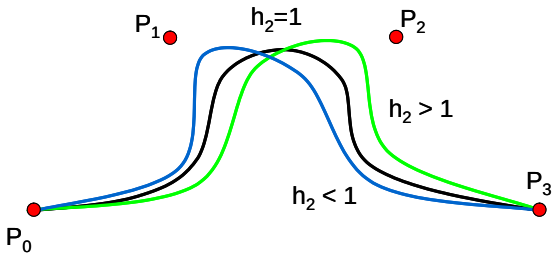

Abbildung 7.8: Einfluss des Gewichtes auf den Verlauf der Kurve

Zur Anzeige der errechneten Kurve werden die homogenen Koordinaten durch Division der dritten Komponente in Punktkoordinaten überführt:

$$(x,y,z)^T \rightarrow (\frac{x}{z},\frac{y}{z})^T$$

NURBS bieten gegenüber B-Splines einige Vorteile:

- NURBS sind eine Verallgemeinerung der B-Splines. Für $h_i = 1 \ \ \forall i$ reduziert sich die NURBS-Kurve zur entsprechenden B-Spline-Kurve.

- NURBS sind invariant bzgl. perspektivischer Projektion. Dies bedeutet, daß zur Projektion einer Kurve nicht alle Kurvenpunkte projeziert werden müssen, sondern es reicht, die Stützpunkte zu projezieren und dann zu einer Kurve zu verbinden.

- NURBS sind (im Gegensatz zu nicht-rationalen *B-Splines*) in der Lage, Kreise zu beschreiben. Abbildung 7.9 zeigt die 8 Kontrollpunkte zusammen mit ihren Gewichten für den Einheitskreis mit Radius 1. Der zugehörige Knotenvektor lautet (0,0,1,1,2,2,3,3,4,4), wobei der erste und der letzte Kontrollpunkt jeweils doppelt benutzt werden.

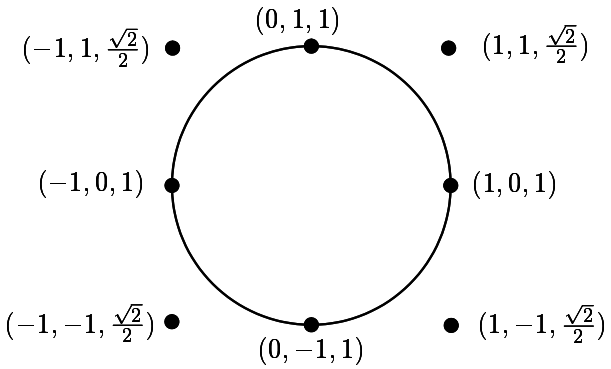

Abbildung 7.9: Kreis erzeugt durch NURBS

7.7 Java-Applet zu Splines

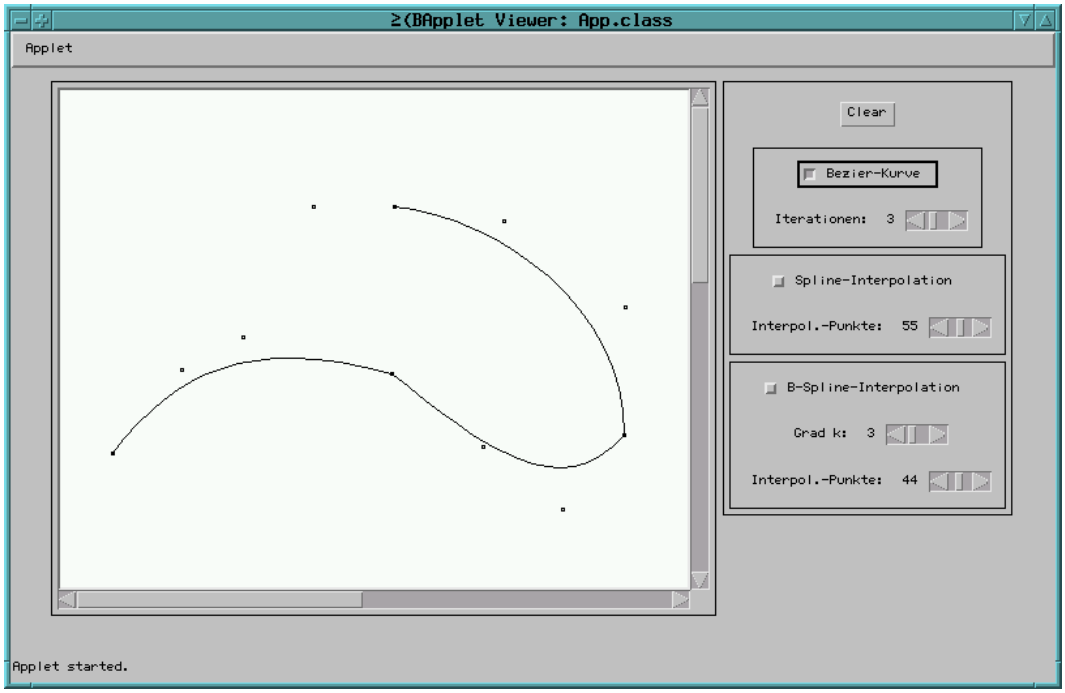

Abbildung 7.10: Screenshot vom Splines-Applet