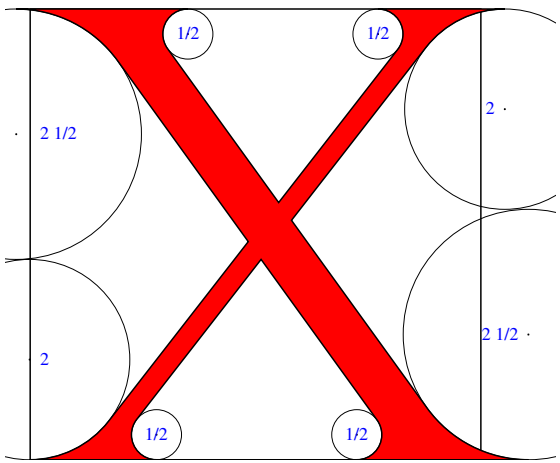
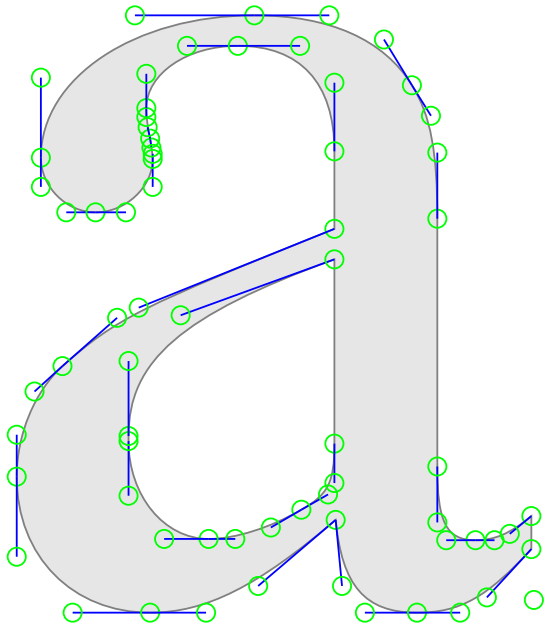


Wie können oder sollten Kurven repräsentiert werden?

- Torniello kam mit Geraden und Kreisen aus.
- Einige versuchten es nur mit Punkten und Geraden (etwa Hershey im Jahr 1972).
- 1976 wurden Splines bei Xerox eingesetzt.
- Ronald McIntosh und Peter Purdy schlugen in einer 1978 veröffentlichten Arbeit Spiralen vor.

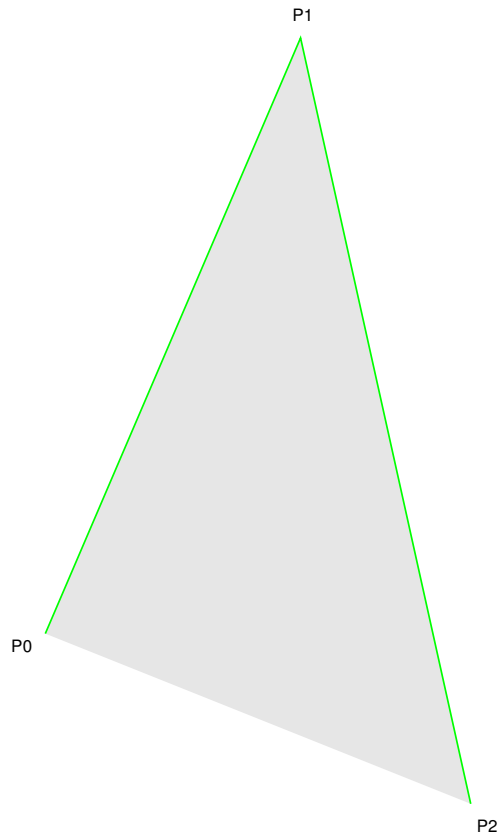
Ziele einer Repräsentierung für Kurven



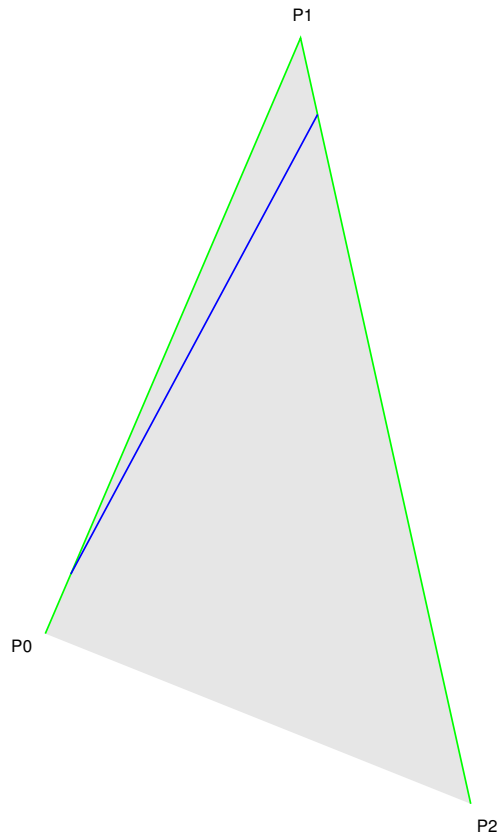
Die Darstellung einer Kurve sollte

- das Original bis zu einer gegebenen Auflösung möglichst gut approximieren, so dass dem menschlichen Auge der Unterschied nicht auffällt,
- ausreichend komprimierend sein,
- Veränderungen einer Kurve erleichtern,
- effizient und numerisch stabil berechenbar sein und
- affine Abbildungen unterstützen.

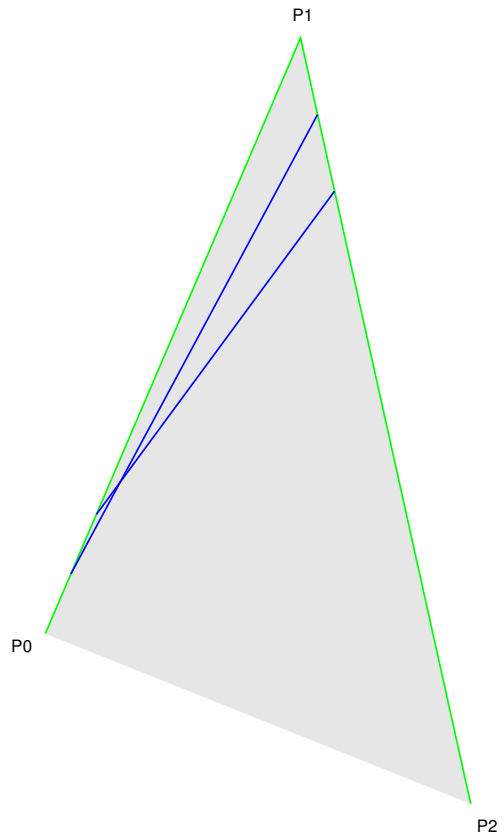
- Pierre Etienne Bézier (1910–1999) war bei Renault Leiter der Abteilung für die Entwicklung von Werkzeugmaschinen für die Automobilproduktion.
- 1960 gab es erste Computer-kontrollierte Maschinen (*Computer Aided Manufacturing*), die die Herstellung von 3-dimensionalen Oberflächen aus Holz oder Stahl erlaubten. Das Problem war die Programmierung einer passenden 3-dimensionalen Fläche.
- Bézier löste das Problem mit den nach ihm benannten 2-dimensionalen Kurven, die mit zwei Endpunkten und zwei Kontrollpunkten in der Mitte vollständig spezifiziert waren. Die 2-dimensional definierten Kurven lassen sich kreuzweise verflechten, um eine 3-dimensionale Fläche zu definieren.
- Unabhängig davon stießen Paul de Casteljau (bei Citroen) und der Flug-Ingenieur James Ferguson (bei Boeing) zuvor auf die gleichen Kurven. Das stellte sich jedoch erst später heraus, weil damals diese Techniken weitgehend als Firmengeheimnisse unter Verschluss blieben.



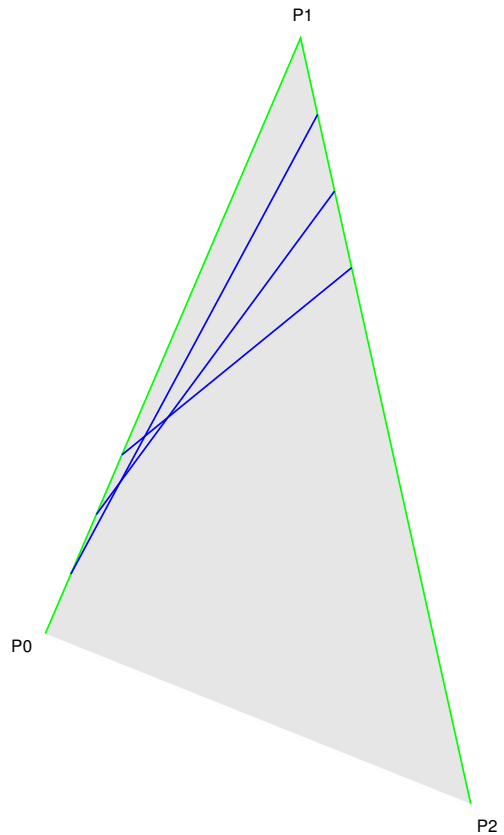
- Gegeben seien drei Punkte P_0, P_1 und P_2 in $\mathbb{R}^2$.



- Gegeben seien drei Punkte P_0, P_1 und P_2 in $\mathbb{R}^2$.
- Notation für die Spezifikation eines Punktes auf einer Linie zwischen zwei Punkten für $t \in [0, 1]$: $t[(x_1, y_1), (x_2, y_2)] = (x_1, y_1) + t(x_2 - x_1, y_2 - y_1)$
- Wir verbinden $0.1[P_0, P_1]$ mit $0.1[P_1, P_2]$.

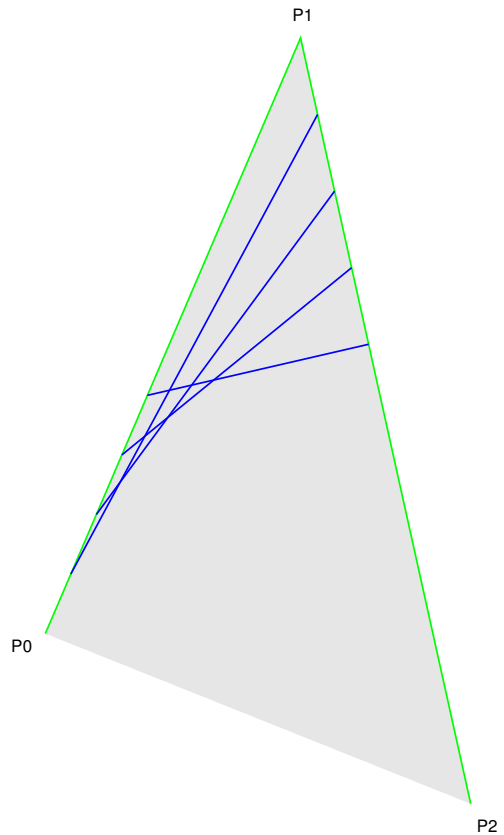


- Gegeben seien drei Punkte P_0, P_1 und P_2 in $\mathbb{R}^2$.
- Notation für die Spezifikation eines Punktes auf einer Linie zwischen zwei Punkten für $t \in [0, 1]$: $t[(x_1, y_1), (x_2, y_2)] = (x_1, y_1) + t(x_2 - x_1, y_2 - y_1)$
- Wir verbinden $0.2[P_0, P_1]$ mit $0.2[P_1, P_2]$.

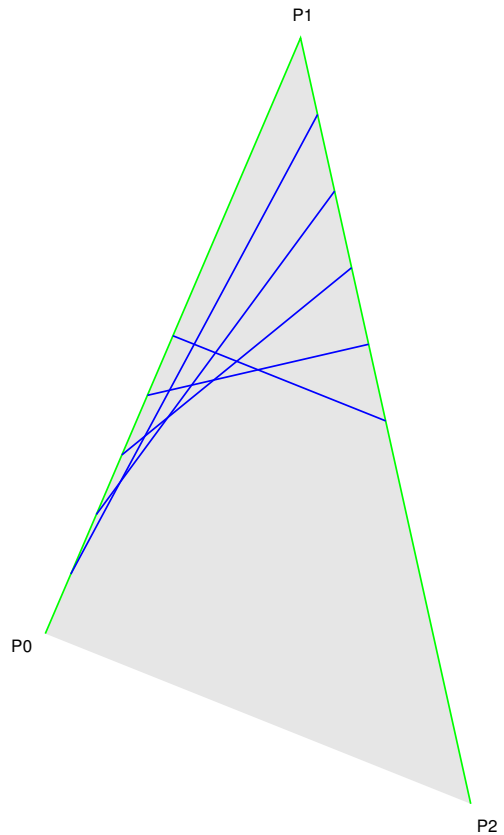


- Gegeben seien drei Punkte P_0, P_1 und P_2 in $\mathbb{R}^2$.
- Notation für die Spezifikation eines Punktes auf einer Linie zwischen zwei Punkten für $t \in [0, 1]$: $t[(x_1, y_1), (x_2, y_2)] = (x_1, y_1) + t(x_2 - x_1, y_2 - y_1)$
- Und so weiter...

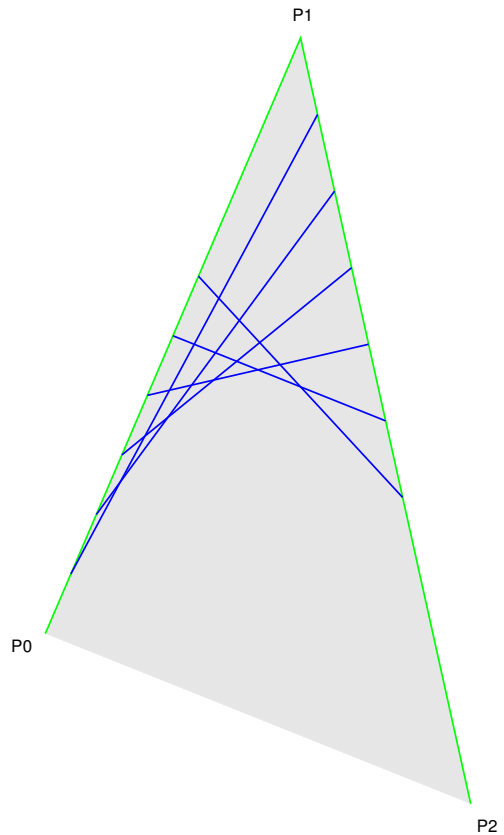
Eine erste Annäherung an Bézier-Kurven



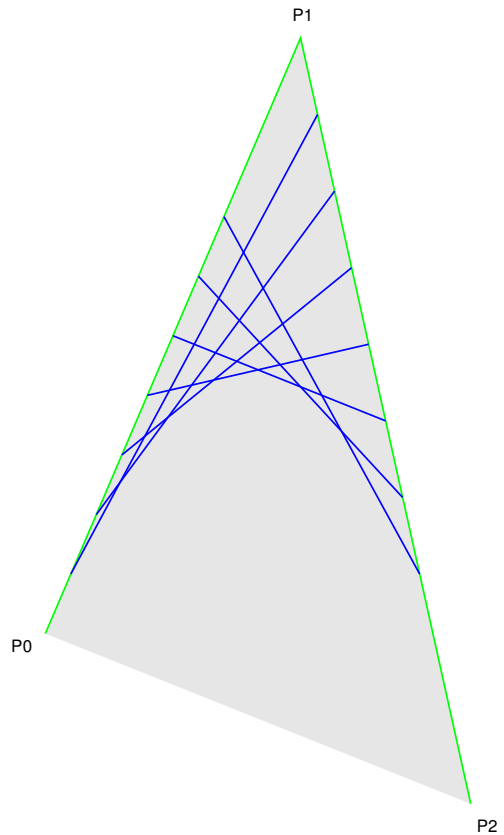
- Gegeben seien drei Punkte P_0, P_1 und P_2 in $\mathbb{R}^2$.
- Notation für die Spezifikation eines Punktes auf einer Linie zwischen zwei Punkten für $t \in [0, 1]$: $t[(x_1, y_1), (x_2, y_2)] = (x_1, y_1) + t(x_2 - x_1, y_2 - y_1)$
- Und so weiter...



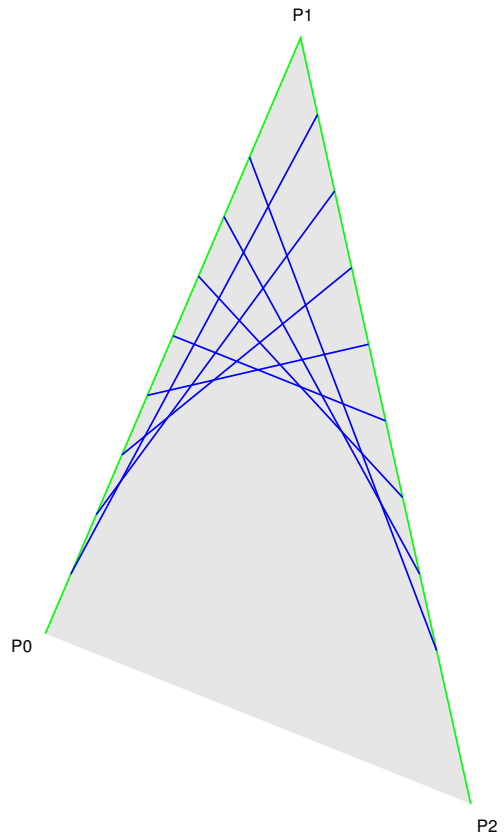
- Gegeben seien drei Punkte P_0, P_1 und P_2 in $\mathbb{R}^2$.
- Notation für die Spezifikation eines Punktes auf einer Linie zwischen zwei Punkten für $t \in [0, 1]$: $t[(x_1, y_1), (x_2, y_2)] = (x_1, y_1) + t(x_2 - x_1, y_2 - y_1)$
- Und so weiter...



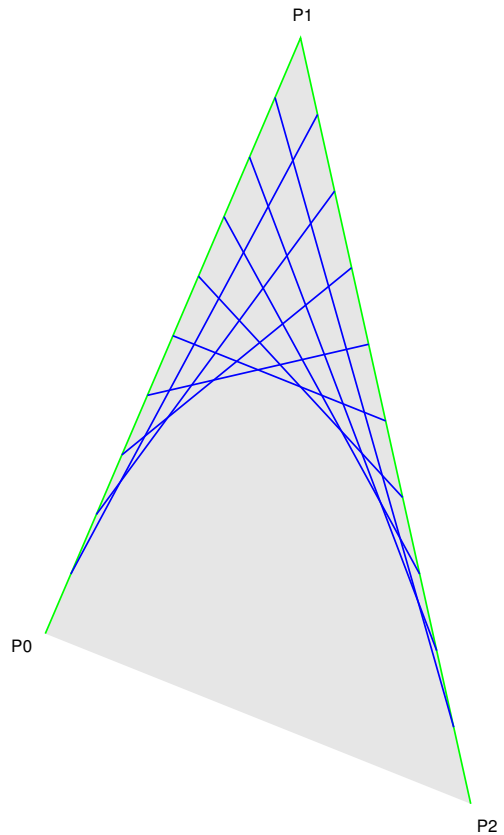
- Gegeben seien drei Punkte P_0, P_1 und P_2 in $\mathbb{R}^2$.
- Notation für die Spezifikation eines Punktes auf einer Linie zwischen zwei Punkten für $t \in [0, 1]$: $t[(x_1, y_1), (x_2, y_2)] = (x_1, y_1) + t(x_2 - x_1, y_2 - y_1)$
- Und so weiter...



- Gegeben seien drei Punkte P_0, P_1 und P_2 in $\mathbb{R}^2$.
- Notation für die Spezifikation eines Punktes auf einer Linie zwischen zwei Punkten für $t \in [0, 1]$: $t[(x_1, y_1), (x_2, y_2)] = (x_1, y_1) + t(x_2 - x_1, y_2 - y_1)$
- Und so weiter...

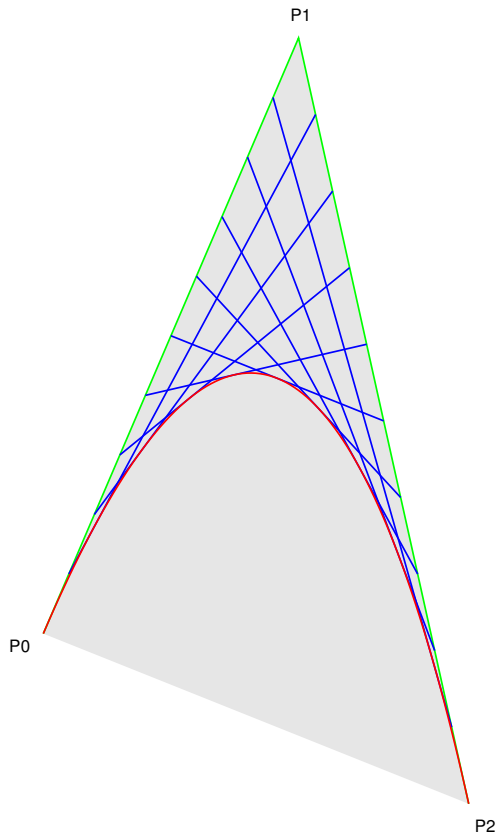


- Gegeben seien drei Punkte P_0, P_1 und P_2 in $\mathbb{R}^2$.
- Notation für die Spezifikation eines Punktes auf einer Linie zwischen zwei Punkten für $t \in [0, 1]$: $t[(x_1, y_1), (x_2, y_2)] = (x_1, y_1) + t(x_2 - x_1, y_2 - y_1)$
- Und so weiter...

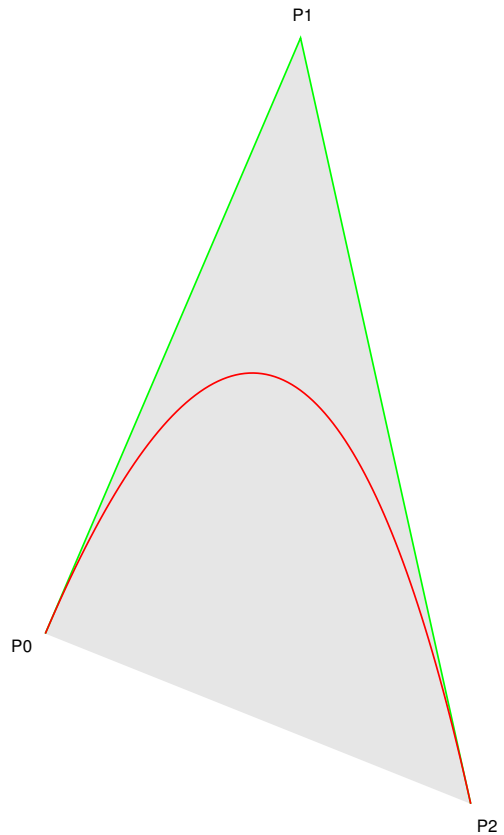


- Gegeben seien drei Punkte P_0, P_1 und P_2 in $\mathbb{R}^2$.
- Notation für die Spezifikation eines Punktes auf einer Linie zwischen zwei Punkten für $t \in [0, 1]$: $t[(x_1, y_1), (x_2, y_2)] = (x_1, y_1) + t(x_2 - x_1, y_2 - y_1)$
- Und so weiter...

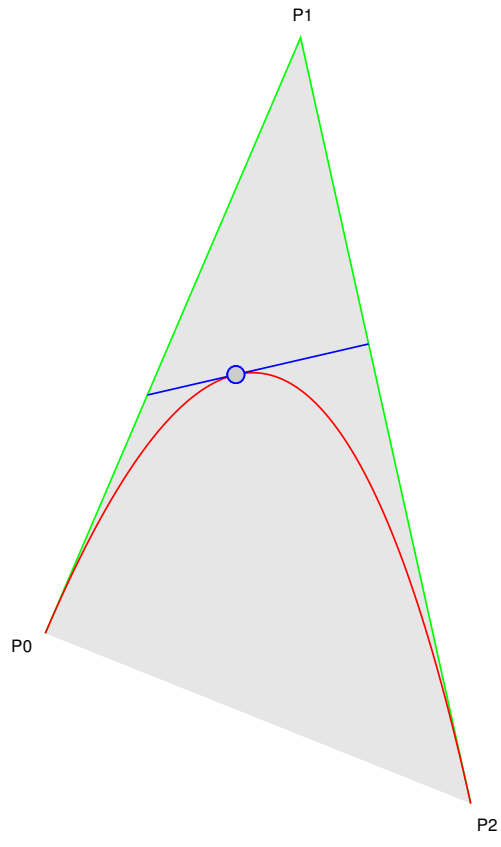
Eine erste Annäherung an Bézier-Kurven



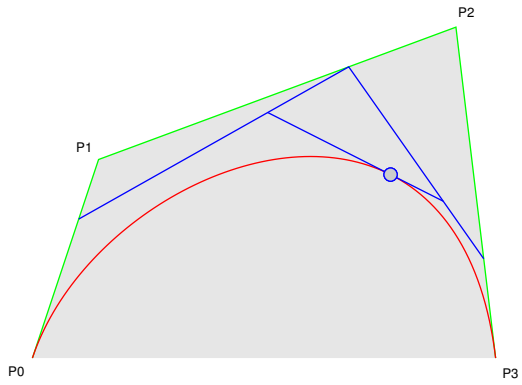
- Wenn das Spiel weiter getrieben wird, nähern wir uns der roten Kurve.



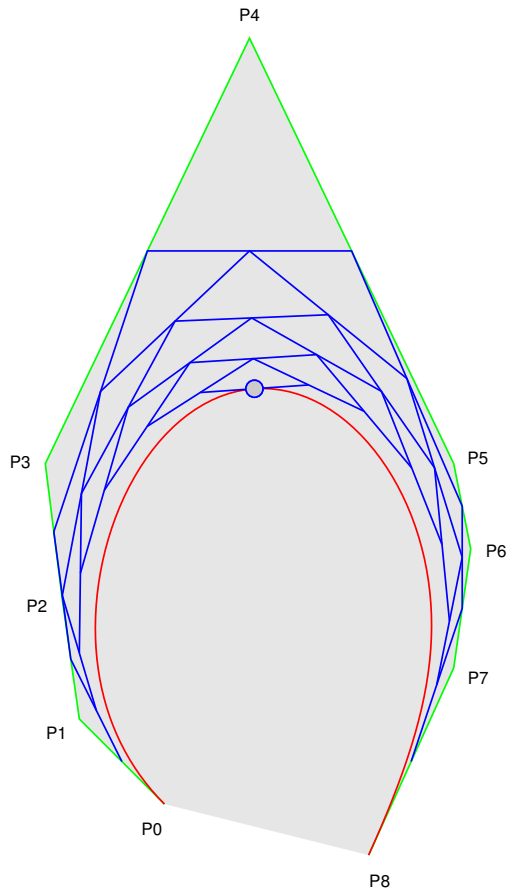
- Wenn das Spiel weiter getrieben wird, nähern wir uns der roten Kurve.



- Die rote Kurve kann auch definiert werden über $C(t) = t[t[P_0, P_1], t[P_1, P_2]]$ für $t \in [0, 1]$.
- Das lässt sich umrechnen zu: $C(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$ für $t \in [0, 1]$.
- Das Diagramm demonstriert dies für $t = 0,4$.



- Wenn ein Punkt hinzukommt, dann vergrößert sich das System der Zwischenpunktberechnungen um eine Ebene.
- Entsprechend erhalten wir folgende Kurve:
  $t[t[t[P_0, P_1], t[P_1, P_2]], [t[P_1, P_2], t[P_2, P_3]]]$
  für $t \in [0, 1]$.
- Das lässt sich umrechnen zu:
  $C(t) =$
  $(1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$
- Das Diagramm demonstriert dies für $t = 0,7$.

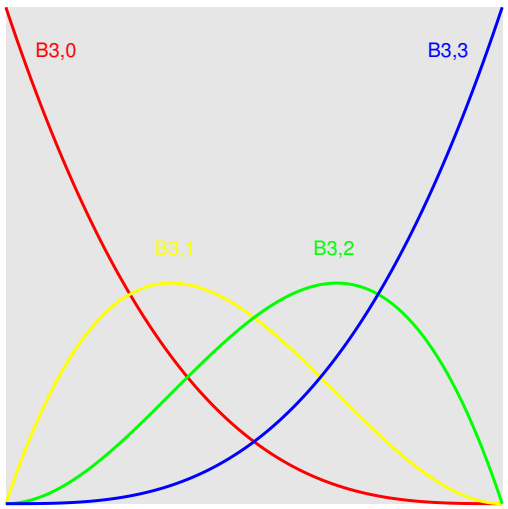


- Eine Bézier-Kurve n-ten Grades wird definiert über $n+1$ Punkte $P_0 \ldots P_n$:

$$C(t) = \sum_{i=0}^{n} B_{n,i}(t) P_i$$

- Die Bernstein-Polynome $B_{n,i}(t)$ werden wie folgt definiert:

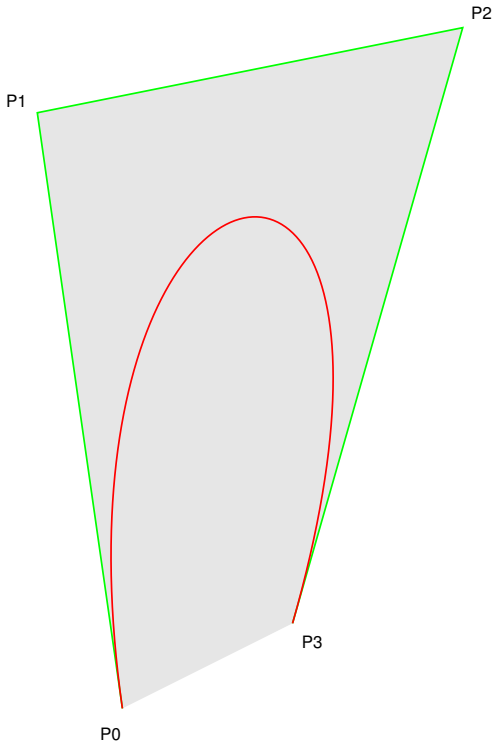
$$B_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$



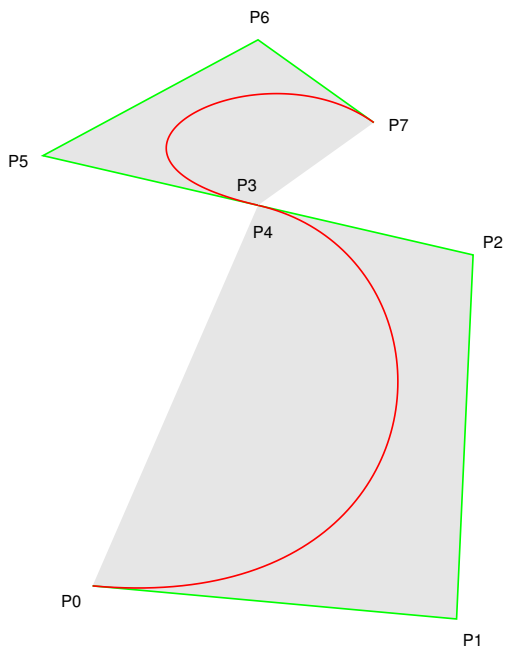
- Die Bernstein-Polynome sind benannt nach Sergei Natanovich Bernstein, der sie 1911 zuerst in einem Beweis verwendete. Sie haben folgende Eigenschaften auf dem Intervall $[0, 1]$:
- Positivität: $B_{n,i}(t) > 0$ für $t \in (0, 1)$
- Zerlegung der Eins:

$$\sum_{i=0}^{n} B_{n,i}(t) = 1 \quad \forall t \in [0, 1]$$

Anfang und Ende der Bézier-Kurve

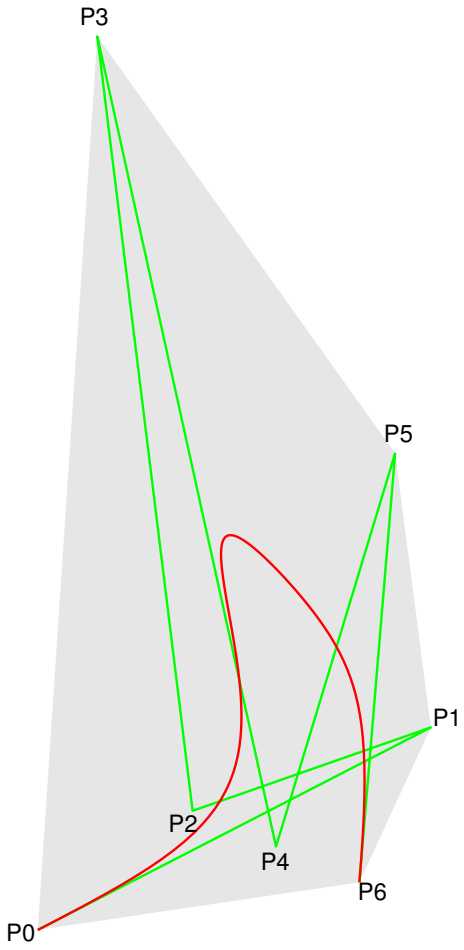


- $\overline{P_0P_1}$ und $\overline{P_{n-1}P_n}$ sind Tangenten der Bézier-Kurve, die diese am Anfangs- bzw. Endpunkt berühren.
- Somit kontrollieren die Kontrollpunkte P_1 und P_{n-1} direkt die tangentiale Ausrichtung der Kurvenenden.
- Das erleichtert das knick-freie Zusammenfügen mehrerer Bézier-Kurven.

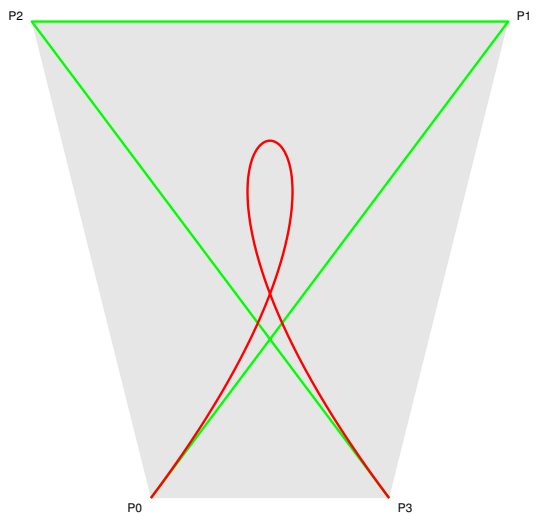


- In diesem Diagramm sind zwei Bézier-Kurven knickfrei zusammengelegt worden (G^1-Stetigkeit).
- Allerdings ist damit noch keine C^1-Stetigkeit gewonnen, da die Tangenten zwar in der Richtung übereinstimmen, jedoch noch nicht notwendigerweise in ihrem Betrag.
- Dies wird gelegentlich als G^1-Stetigkeit bezeichnet.

Konvexe Hülle der Kontrollpunkte

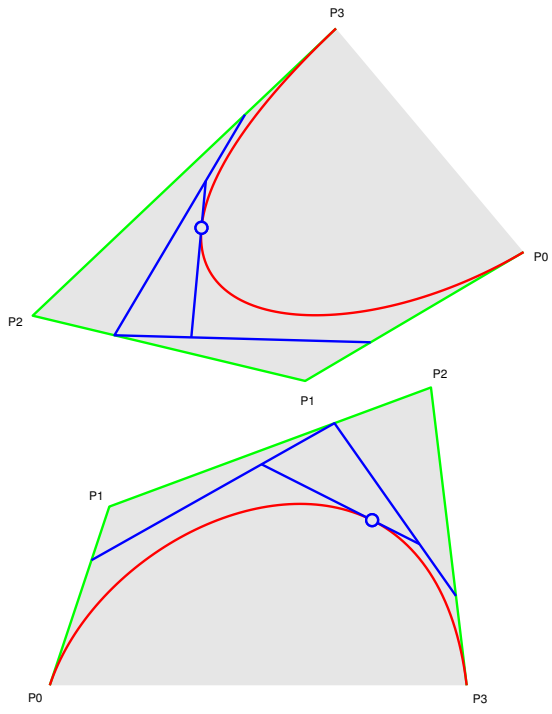


- Bézier-Kurven verlaufen immer innerhalb der konvexen Hülle ihrer Kontrollpunkte.
- Im Diagramm ist die konvexe Hülle grau hinterlegt.

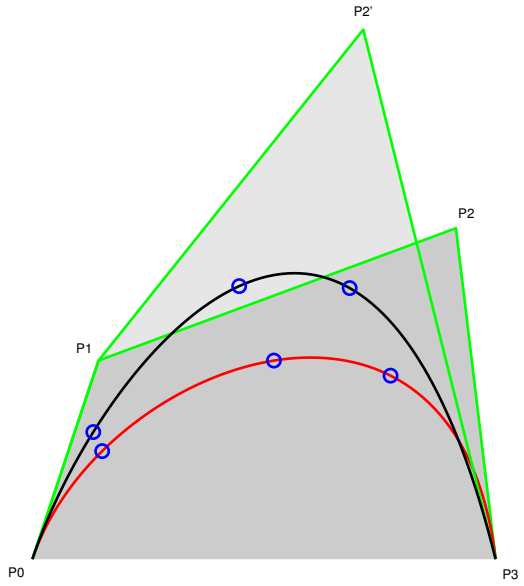


- Grundsätzlich können sich Bézier-Kurven sich selbst überschneiden.
- Allerdings ist die Zahl der Überschneidungen einer Bézier-Kurve nach oben beschränkt durch die Zahl der Überschneidungen des Polygon-Zuges, der durch die Kontrollpunkte geht.

Affine Invarianz



- Bei affinen Transformationen von Bézier-Kurven genügt es, die Kontrollpunkte $P_0, \ldots, P_n$ entsprechend abzubilden und dann ausgehend von den abgebildeten Kontrollpunkten die Kurve neu zu zeichnen.
- Das vereinfacht dramatisch die Implementierung der Operatoren `translate`, `scale` und `rotate` in PostScript.
- Diese Vereinfachung betrifft sämtliche Kurven in PostScript, da jede Kurve in PostScript nur aus einer Folge von Bézier-Kurven besteht.

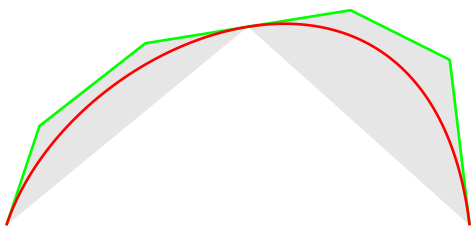


- Wenn genau ein Kontrollpunkt P_i durch P'_i ersetzt wird, dann bewegen sich alle Punkte der Bézier-Kurve in der Richtung von $\vec{P_i P'_i}$.
- Diese Bewegung wird in genau dem Maße gedämpft, wie es dem Gewicht von P_i entsprechend des betreffenden Gliedes des Bernstein-Polynoms an der jeweiligen Kurvenstelle entspricht:

$$C'(t) = C(t) + B_{n,i}(t)(P'_i - P_i)$$

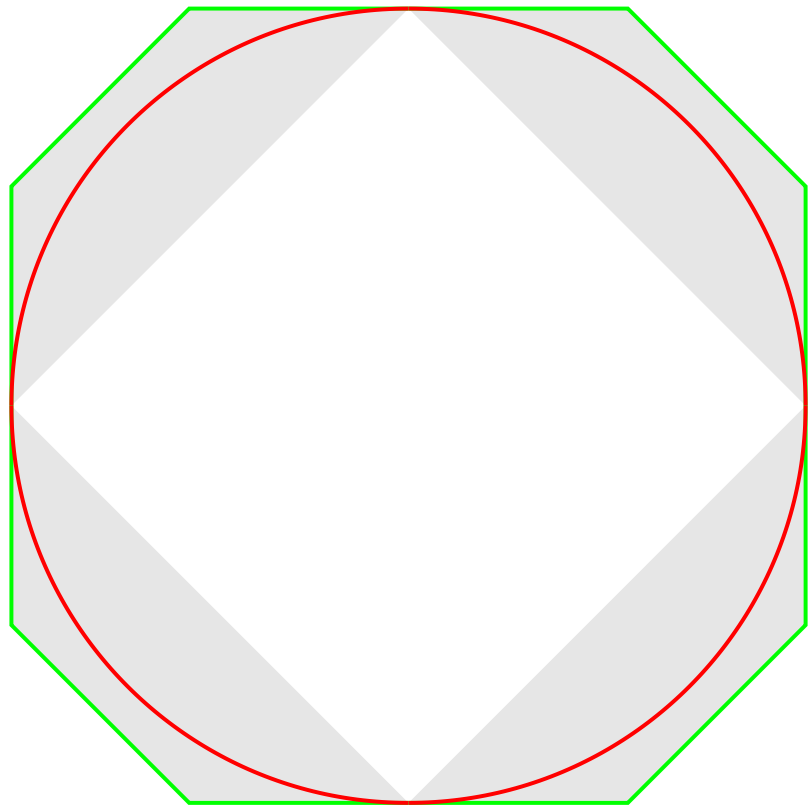
- Jede Änderung eines Kontrollpunktes macht sich **global** bemerkbar.

- Zu berechnen ist $C(t)$ für $t \in [0,1]$.
- Rekursiv werden Punkte $P_{i,j}$ definiert:
  - $P_{0,i} = P_i$ für $i = 0 \ldots n$
  - $P_{k,i} = t[P_{k-1,i}, P_{k-1,i+1}]$ für $k = 1 \ldots n$ und $i = 0 \ldots n-k$
- Dann ist $C(t) = P_{n,0}$.
- Um doppelte Berechnungen zu vermeiden, empfiehlt sich ein Pyramidenschema.
- Der Berechnungsaufwand ist $O(n^2)$.
- Bei kubischen Bézier-Kurven sind 6 Linear-Kombinationen zu berechnen pro Zwischenpunkt.
- Dieses Berechnungsverfahren ist effizient und numerisch stabil.



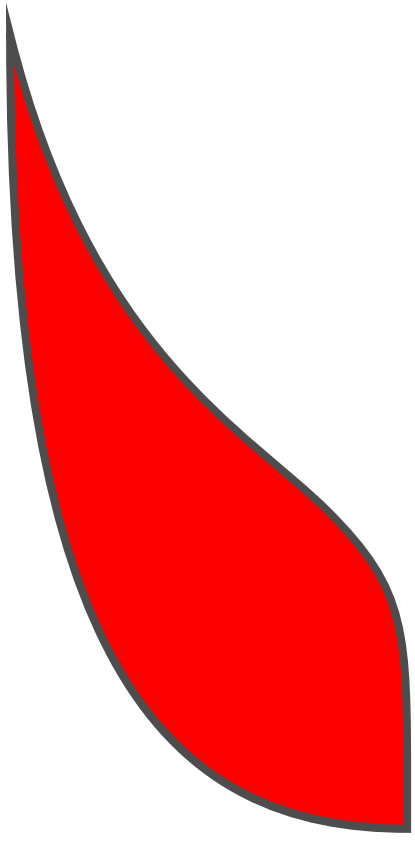
- Jede Bézier-Kurve kann für jedes $t \in (0, 1)$ in zwei Bézier-Kurven gleichen Grades zerlegt werden.
- Die Kontrollpunkte der beiden neuen Kurven ergeben sich aus den Zwischenpunkten, die nach dem Algorithmus von de Casteljau berechnet worden sind:
  - 1. Kurve: $P_{0,0}, P_{1,0}, \ldots, P_{n,0}$
  - 2. Kurve: $P_{0,n}, P_{1,n-1}, \ldots, P_{n,0}$

- Ziel ist ein Kreis um den Punkt $(0,0)$ mit Radius 1.
- Der Kreis wird in 4 Segmente entsprechend den Quadranten aufgeteilt.
- Für das Kreissegment von $(1,0)$ nach $(0,1)$ wird eine kubische Bézier-Kurve gesucht, die folgende Forderungen erfüllt:
  - Die Bézier-Kurve hat die gleiche Tangente wie der Kreis in $(1,0)$ und $(0,1)$.
  - Die Bézier-Kurve verläuft durch $M = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$.
- Wegen der ersten Forderung müssen die Kontrollpunkte von der Form $(1,a)$ bzw. $(b,1)$ sein. Wegen der Symmetrie ergibt sich $a = b$. Somit ist nur noch a so zu wählen, dass die Bézier-Kurve durch M läuft.
- Da $C(0,5) = \frac{1}{8}(4+3a, 4+3a)$, ergibt sich $a = 4\frac{\sqrt{2}-1}{3}$.
- In PostScript generiert der Operator `arc` eine Folge von Bézier-Kurven nach diesem Schema.



- In PostScript lassen sich kubische Bézier-Kurven mit dem Operator `curveto` zeichnen.
- Dabei gilt die aktuelle Position als P_0. Die Punkte P_1, P_2 und P_3 sind explizit zu spezifizieren.
- Beispiel: `newpath 100 100 moveto 150 250 420 350 450 100 curveto stroke`
- Die Kontrollpunkte müssen also explizit angegeben werden.
- John D. Hobby veröffentlichte 1986 Formeln zur Bestimmung der Kontrollpunkte, so dass nach von ihm angegebenen formalen Kriterien die Kurven möglichst "gut" aussehen und passen.
- Dieses Verfahren ist Bestandteil von METAFONT/METAPOST, die eine Reihe alternativer Parameter erlauben, die alternativ zu den Kontrollpunkten angegeben werden können.

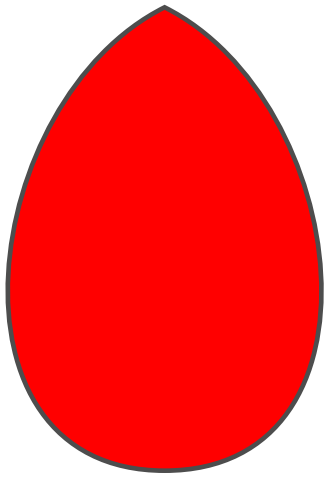
Tulpen mit Bézier-Kurven: Linkes Blatt



tulips.eps

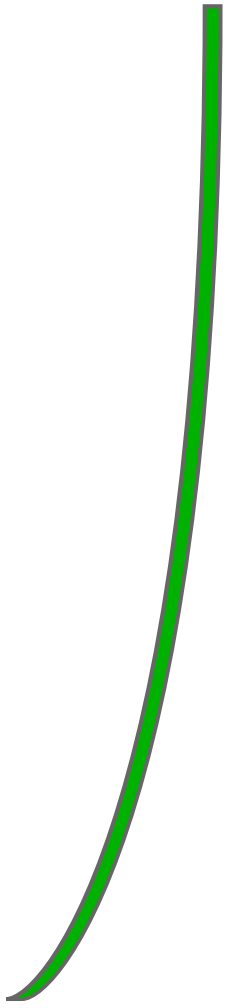
```
/LeftTulipLeaf {
   newpath
   0 0 moveto
   -30 0 -50 20 -50 100 curveto
   -40 60 -20 50 -10 40 curveto
   0 30 0 25 0 5 curveto
   closepath
} def
```

Tulpen mit Bézier-Kurven: Mittleres Blatt



tulips.eps

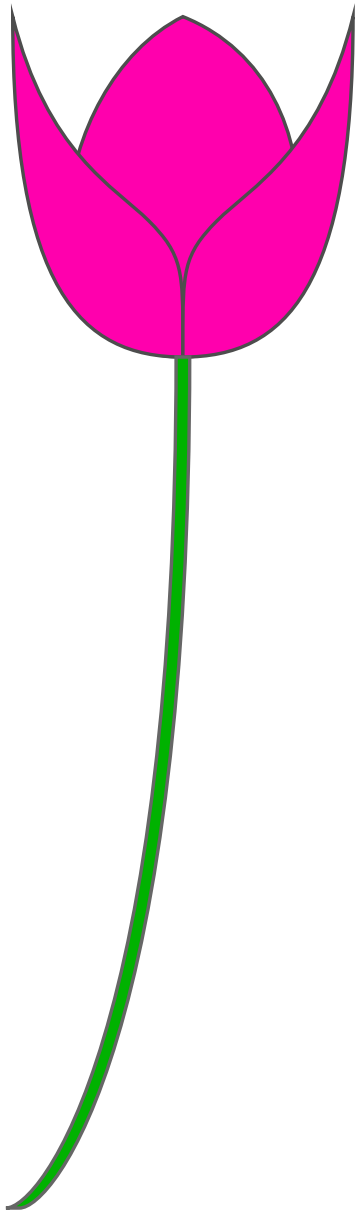
```
/MiddleTulipLeaf {
   newpath
   0 0 moveto -50 0 -40 80 0 100 curveto
   40 80 50 0 0 0 curveto
   closepath
} def

% color FillTulip -
/FillTulip {
   1 dict begin
   /color exch def
   gsave
      color aload pop sethsbcolor
      fill
   grestore
   0.3 setgray stroke
   end
} def
```



tulips.eps

```
/Stem {
   gsave
      -2 0 moveto
      -2 -200  -42 -250  -52 -250 curveto
      -48 -250 lineto
      -38 -250  2 -200  2 0 curveto
      closepath
      gsave
         0 0.7 0 setrgbcolor
         fill
      grestore
      0.4 0.4 0.4 setrgbcolor stroke
      stroke
   grestore
} def
```

Tulpen mit Bézier-Kurven: Zusammenbau



tulips.eps

```
/RandomColor {
   [
      rand 32 mod 220 add 256 div
      1 1
   ]
} def

/PlotTulip {
   1 dict begin
   /color RandomColor def
   gsave
      MiddleTulipLeaf color FillTulip
      LeftTulipLeaf color FillTulip
      RightTulipLeaf color FillTulip
      Stem
   grestore
   end
} def

PlotTulip
```