BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

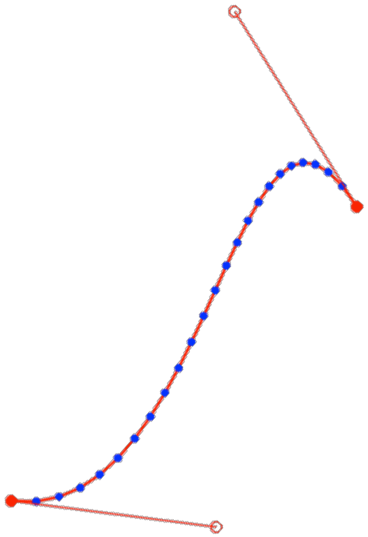
Computergrafik II
- Kurven und JavaScript II -

Bachelor Medieninformatik
Wintersemester 2012/2013

Prof. Dr.-Ing. Hartmut Schirmacher
http://schirmacher.beuth-hochschule.de
hschirmacher@beuth-hoschule.de

Lernziele heute

- Geometrische Modellierung: Kurven
  - Was ist Interpolation
  - Lineare vs. kubische Interpolation
  - Kubische Bezier-Kurven
  - Auswerten und Darstellen von Kurven



- JavaScript II
  - Wiederholung von JavaScript Teil I
  - Prototypen und Vererbung
  - Sonstige Typen und “falsy values“

```
this.x = x || 0;
```

Abb.: Mario Botsch, Uni Bielefeld

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Wiederholung: JavaScript I und Canvas

Objekte, Arrays, `undefined`

- Objekte

```
var o = { x:1, y:true, z:"super simpel!" };
var x = o.x;
var y = o["y"];
```

- Arrays

```
var a = [];
a[2] = "Hello Array!";
a.length
```

- `undefined`
  - Zuweisung `var x = undefined` OK
  - Verwendung eines `undefined.x()` → Exception
  - Funktionsparameter weglassen → `undefined`

Funktionen, Closures

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

- Funktionen

```
function f(x,y) { return x+y; }
var g = function(x,y) { return x+y; }
```

- Closure

```
var outer = function(outerParam) {
    var inClosure = ...;
    var inner = function(x) {
        return x + inClosure + outerParam;
    };
    return inner;
};
```

Konstruktoren und Methoden

- Konstruktorfunktion, `new()` und `this`

```
var Scene = function(bgColor) {
    this.bgColor = bgColor;
};
```

```
Scene.prototype.draw = function(context) {
    context.fillStyle = this.bgColor;
    fillRect(...);
    ...
};
```

```
var theScene = new Scene("#FF0000");
...
theScene.draw(...);
```

Kapselung und Module

- Kapselung mittels anonymer Funktion

```
(function() { /* gekapselt */ }());
```

- Modul-Pattern

```
var MODNAME = (function($, util) {
    ...
    return <interface>;
}($, util));
```

- RequireJS, *Asynchronous Module Definition* AMD

```
define([”jquery”, "util”], (function($, util) {
    ...
}));
```

HTML5 Canvas und der 2D-Rendering-Kontext

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

- Canvas-Element in HTML 5

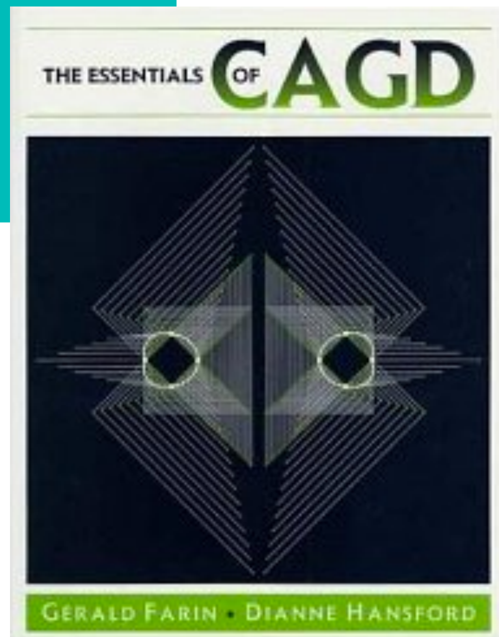
```
<canvas id="myCanvas" width="400px" height="200px">
  Kann weiteren <i>HTML-Inhalt</i> enthalten!
</canvas>
```

- 2D-Rendering-Kontext anfordern und verwenden

```
var canvas = $("#myCanvas").get(0);
var context = canvas.getContext("2d");

context.beginPath();
context.moveTo(...);
context.lineTo(...);
context.stroke();
context.arc(...);
```

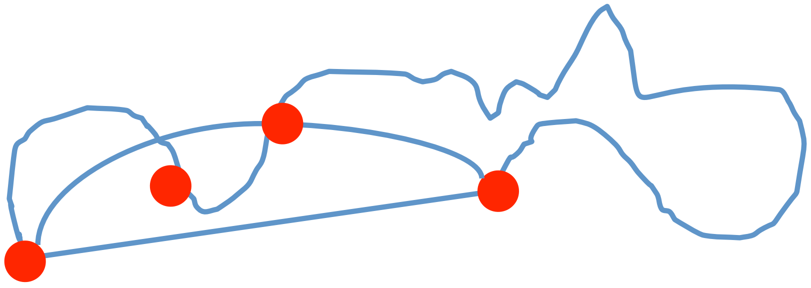
Kurvenmodellierung

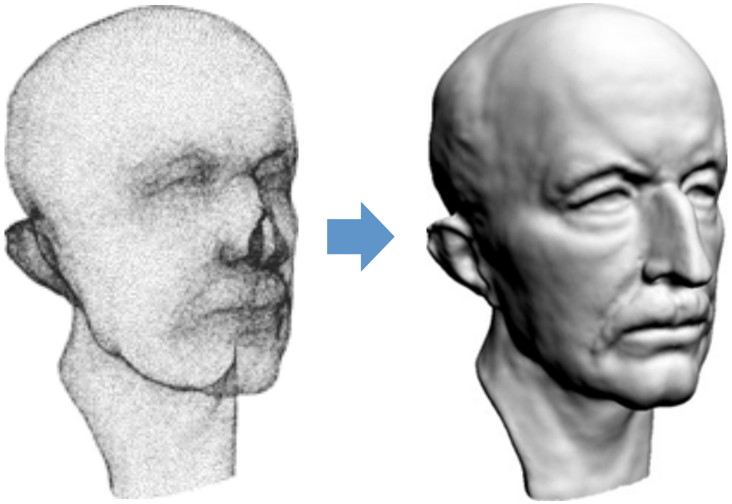


Farin/Hansford, *The Essentials of CAGD*,
Peters, Wellesley 2000, 248 Seiten

Interpolation

- Was ist Interpolation?
  - Gegeben eine Menge von Punkten
  - Interpolante = stetige Funktion, die die Punktemenge beschreibt



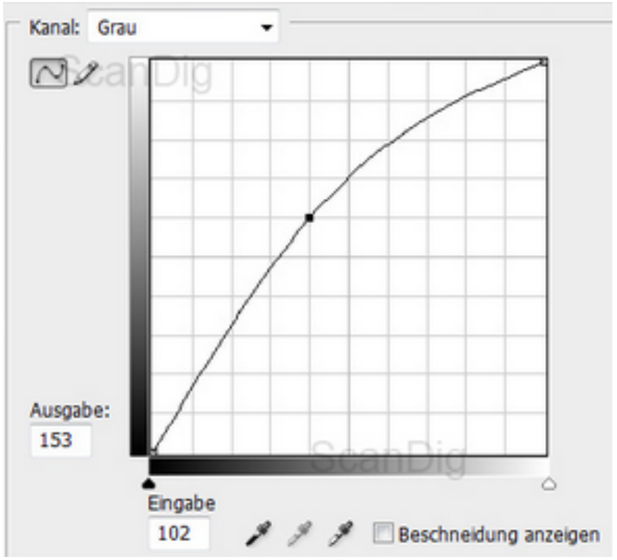


- Wozu?
  - Modellierung komplexer Formen mittels weniger Parameter
  - Kontrolle über Glattheit und ähnliche Eigenschaften der Form durch entsprechende Wahl der Interpolationsfunktion

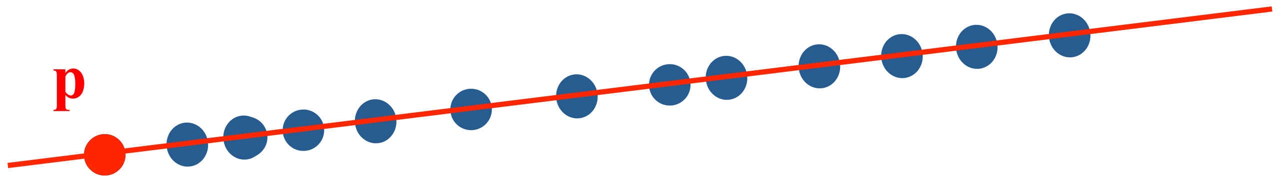
Interpolation – nicht nur auf Geometrie anwendbar!

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

- Interpolation von Pfaden
  - Glatte Kamerapfade
  - Automatische Wegfindung mit „natürlichen“ Pfaden
- Interpolation von Audio / Video
  - Compositing / Cross Fading
  - Manipulation von Farbkurven
  - ...



Lineare Interpolation



- Interpolation mittels eines Liniensegments
  - Gesucht: Beschreibung des Liniensegments, auf dem die Punkte liegen
  - Explizite Form:

$$y = mx + b$$

← „für ein gegebenes *x* auf der Linie, rechne *y* aus“

  - Parametrische Form:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} + t \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

← Beschreibung durch **Anfangspunkt** und **Richtungsvektor**

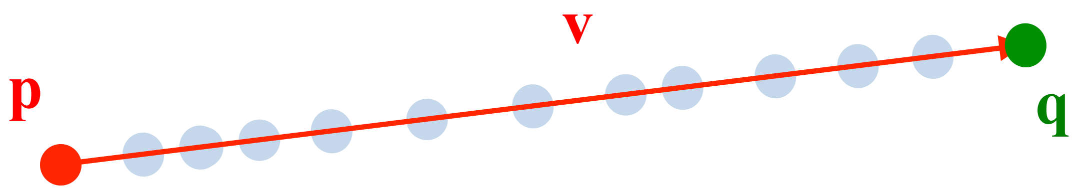
  - Vektorschreibweise:

$$\mathbf{x} = \mathbf{p} + t\mathbf{v}$$

... so kommen wir weiter: als **Anfangspunkt *p*** verwenden wir einen bekannten Punkt ● auf der Linie...

Lineare Interpolation durch zwei Punkte

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences



- Parametrische Form (Vektorschreibweise): $\mathbf{x} = \mathbf{p} + t\mathbf{v}$
- Wie ermittele ich den Richtungsvektor $\mathbf{v}$?
  - Wähle den Endpunkt $\mathbf{q}$ des zu beschreibenden Segments
  - Bilde Richtungsvektor vom Anfangs- zum Endpunkt: $\mathbf{v} = \mathbf{q} - \mathbf{p}$
  - Einsetzen in die parametrische Form:

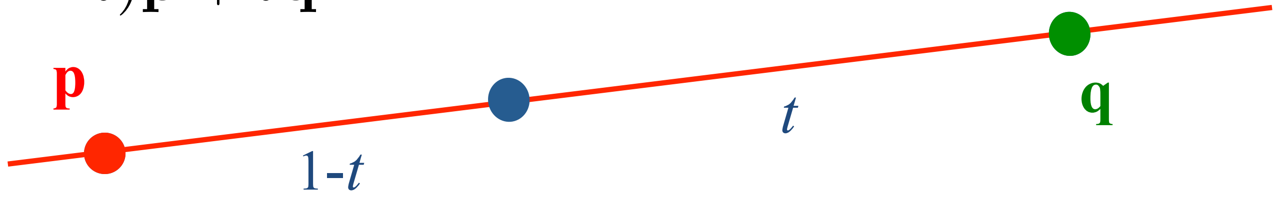
$$\begin{aligned} \mathbf{x}(t) &= \mathbf{p} + t(\mathbf{q} - \mathbf{p}) \\ &= \mathbf{p} + t\mathbf{q} - t\mathbf{p} \\ &= (1-t)\mathbf{p} + t\mathbf{q} \end{aligned}$$

Die Schreibweise $\mathbf{x}(t)$ soll verdeutlichen, dass x eine Funktion von t ist.

beschreibt eine **Gerade** durch die beiden Punkte $\mathbf{p}$ und $\mathbf{q}$

Lineare Interpolation: die Rolle des Parameters t

$$\mathbf{x}(t) = (1-t)\mathbf{p} + t\mathbf{q}$$



- Werte die Gleichung für einige interessante t aus:

$$\mathbf{x}(0) = \mathbf{p} \qquad \mathbf{x}(1) = \mathbf{q} \qquad \mathbf{x}\left(\frac{1}{2}\right) = \frac{1}{2}\mathbf{p} + \frac{1}{2}\mathbf{q}$$

  - Im Intervall [0:1] beschreibt t alle Punkte auf der Strecke von $\mathbf{p}$ nach $\mathbf{q}$
  - t auf [0:1] beschränken → **Liniensegment** von $\mathbf{p}$ nach $\mathbf{q}$
  - Das **Verhältnis** $\dfrac{1-t}{t}$ beschreibt, wie nahe der Punkt an $\mathbf{p}$ bzw. $\mathbf{q}$ ist

Animation: de.wikipedia.org

Parametrische Kurven

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

- Das Liniensegment wird beschrieben durch

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} (1-t)p_x + tq_x \\ (1-t)p_y + tq_y \end{bmatrix}$$

- Eine parametrische Kurve hat i.A. die Form

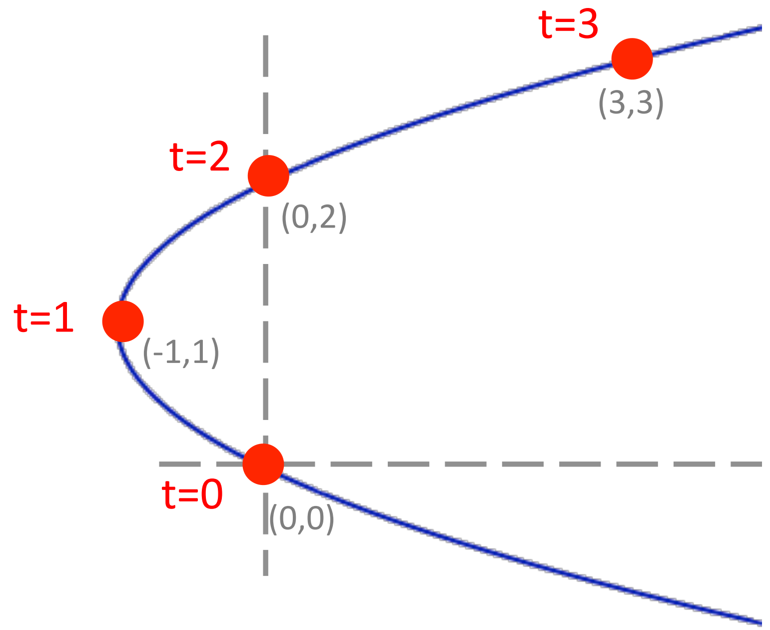
$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f(t) \\ g(t) \end{bmatrix}$$

Wertebereich:
2D Koordinaten (x,y)

Definitionsbereich:
1D Parameter t

- Beispiel: quadratische Kurve

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -2t + t^2 \\ t \end{bmatrix}$$



Auswerten durch Einsetzen von t

Parametrische Kurven im 3D

- Das gleiche Prinzip lässt sich im 3D anwenden

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f(t) \\ g(t) \\ h(t) \end{bmatrix}$$

- Beispiel: Helix

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} sin(t) \\ cos(t) \\ t \end{bmatrix}$$

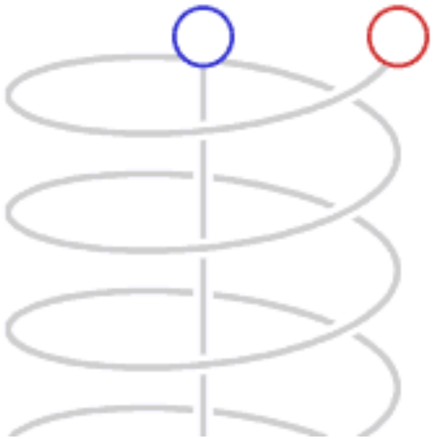


Illustration*: [cleonis / wikipedia]
Lizenz: [CC-by-SA-2.5]

*) die Uhren illustrieren einen Raum-Zeit-Effekt und haben nichts mit der geometrischen Modellierung einer Helix zu tun

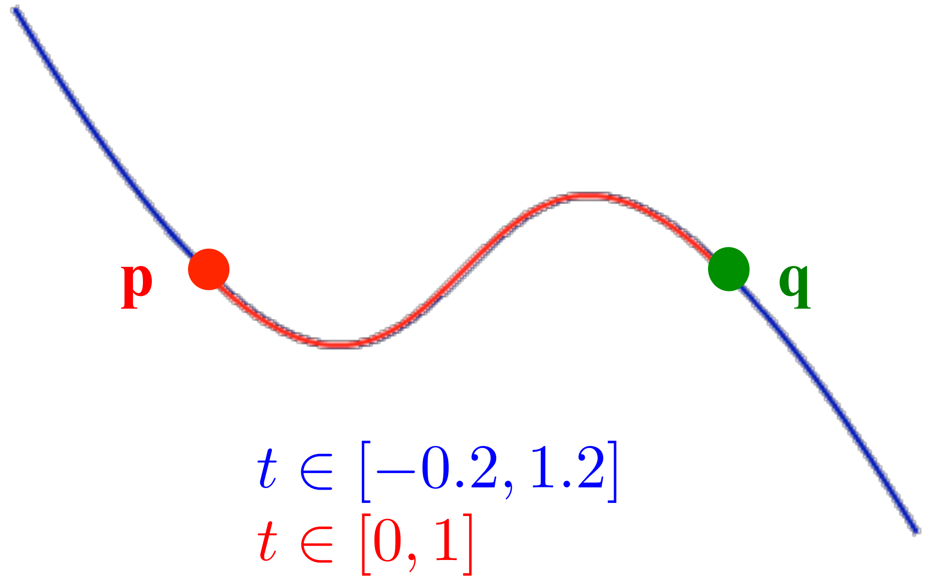
Kubische Kurven

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

- Gegeben sei folgende Kurve:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -(1-t)^3 + t^3 \\ 3(1-t)^2 t - 3(1-t)t^2 \end{bmatrix}$$

- Wie sieht sie aus?
  - nicht offensichtlich...

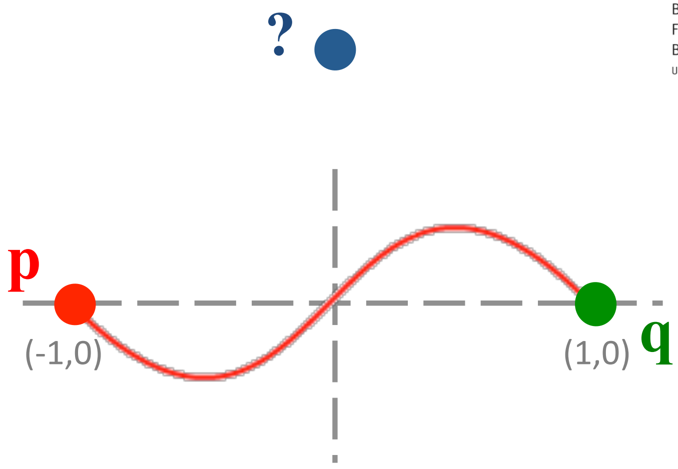


- Gesucht: *intuitivere* Beschreibung
  - Kurve konstruieren, die
    - in Punkt $\mathbf{p}$ anfängt,
    - in Punkt $\mathbf{q}$ aufhört,
    - einen bestimmten Verlauf zeigt?

Eine kubische Bézier-Kurve

- Gegeben sei folgende Kurve:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -(1-t)^3 + t^3 \\ 3(1-t)^2 t - 3(1-t)t^2 \end{bmatrix}$$

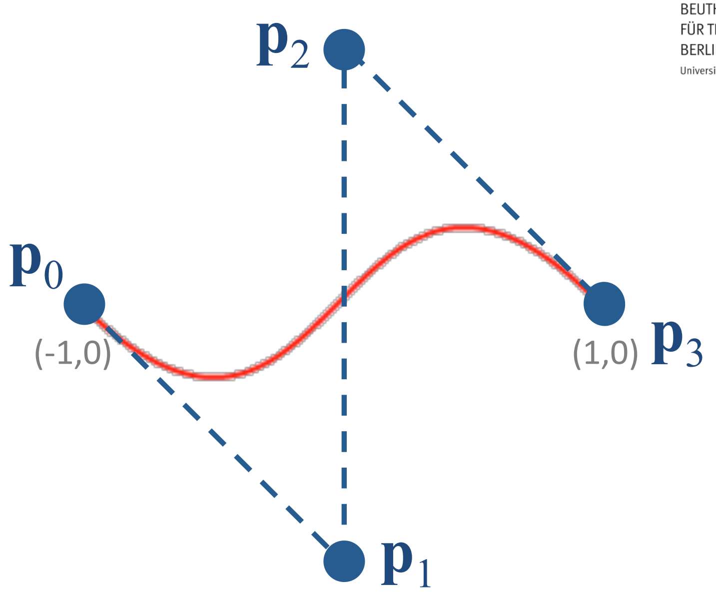


- Umformen in Bézier-Darstellung:

Kubische Bernstein-Polynome B_i^3

$$\begin{bmatrix} x \\ y \end{bmatrix} = \overbrace{(1-t)^3}^{B_0^3} \underbrace{\begin{bmatrix} -1 \\ 0 \end{bmatrix}}_{\mathbf{p}} + \overbrace{3(1-t)^2 t}^{B_1^3} \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{?} + \overbrace{3(1-t)t^2}^{B_2^3} \underbrace{\begin{bmatrix} 0 \\ -1 \end{bmatrix}}_{?} + \overbrace{t^3}^{B_3^3} \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\mathbf{q}}$$

- Kontrollpolygon $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$
  - Verlauf des Polygons beschreibt „grob“ den Verlauf der Kurve!
  - Startet in Richtung $(\mathbf{p}_1\text{-}\mathbf{p}_0)$
  - Besitzt Wendepunkt
  - Endet in Richtung $(\mathbf{p}_3\text{-}\mathbf{p}_2)$

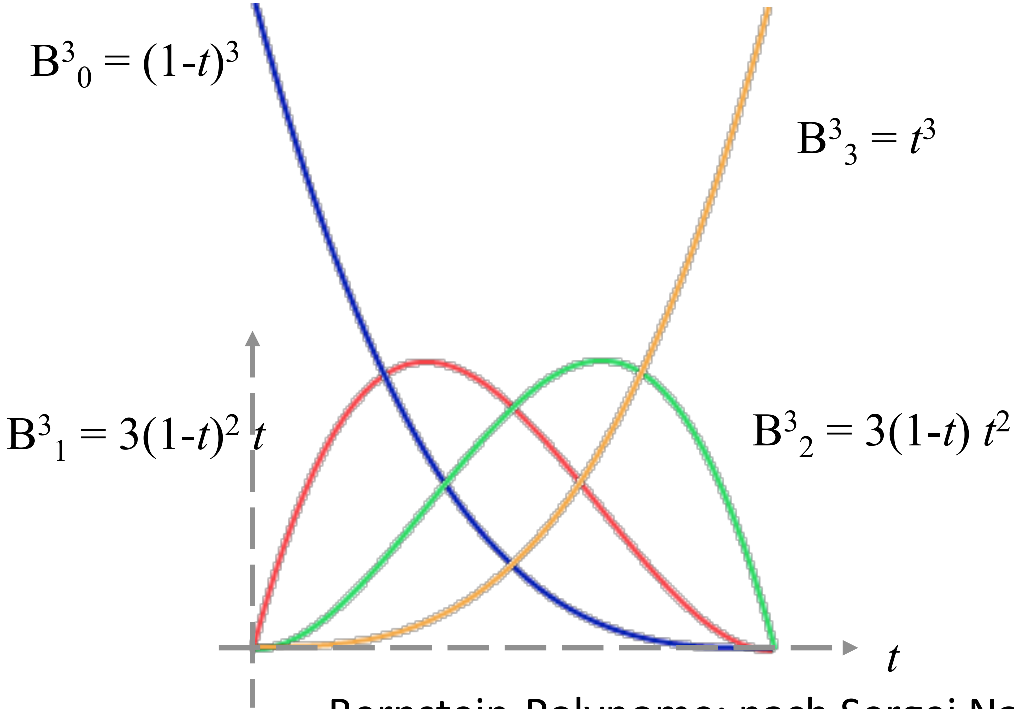


$$\begin{bmatrix} x \\ y \end{bmatrix} = \overbrace{(1-t)^3}^{B_0^3} \underbrace{\begin{bmatrix} -1 \\ 0 \end{bmatrix}}_{\mathbf{p}_0} + \overbrace{3(1-t)^2 t}^{B_1^3} \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{\mathbf{p}_1} + \overbrace{3(1-t)t^2}^{B_2^3} \underbrace{\begin{bmatrix} 0 \\ -1 \end{bmatrix}}_{\mathbf{p}_2} + \overbrace{t^3}^{B_3^3} \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\mathbf{p}_3}$$

Kubische Bézier-Kurven und Bernstein-Polynome

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

$$\mathbf{x}(t) = \begin{bmatrix} x \\ y \end{bmatrix} = B_0^3\mathbf{p_0} + B_1^3\mathbf{p_1} + B_2^3\mathbf{p_2} + B_3^3\mathbf{p_3}$$



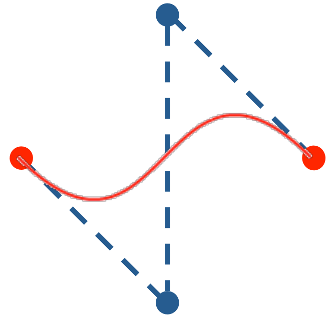
- Anfang der 1960 er Jahre
- Erfinder:
  Pierre Bézier (Renault)
  Paul de Casteljau (Citroën)
  (unabhängig voneinander)
- Anwendung: CAD

Bernstein-Polynome: nach Sergei Natanowitsch Bernstein, 1911

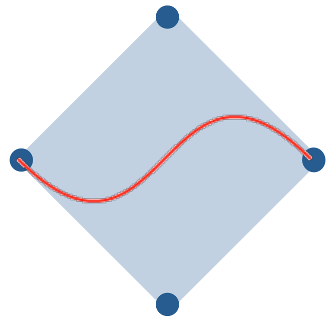
Eigenschaften kubischer Bézier-Kurven (1)

$$\mathbf{x}(t) = \begin{bmatrix} x \\ y \end{bmatrix} = B_0^3\mathbf{p_0} + B_1^3\mathbf{p_1} + B_2^3\mathbf{p_2} + B_3^3\mathbf{p_3}$$

- Kombination von vier Punkten $\mathbf{p}_0$ ... $\mathbf{p}_3$
  - Kontrollpunkte beschreiben Kurve vollständig
- Interpolation
  - $\mathbf{p}_0$ und $\mathbf{p}_3$ werden von der Kurve interpoliert
  - $\mathbf{x}(0) = \mathbf{p}_0$, $\mathbf{x}(1) = \mathbf{p}_3$
- Konvexe Hülle
  - Kurve liegt vollständig in der konvexen Hülle der Kontrollpunkte



Interpolation von $\mathbf{p}_0$ und $\mathbf{p}_3$



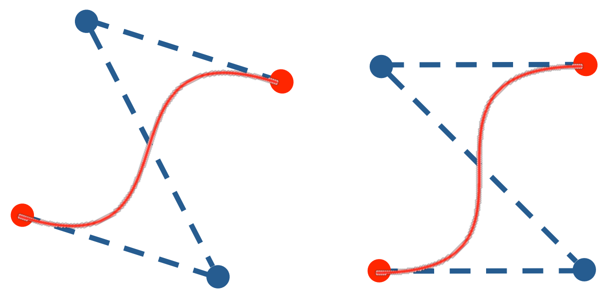
Konvexe Hülle der Kontrollpunkte

Eigenschaften kubischer Bézier-Kurven (2)

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

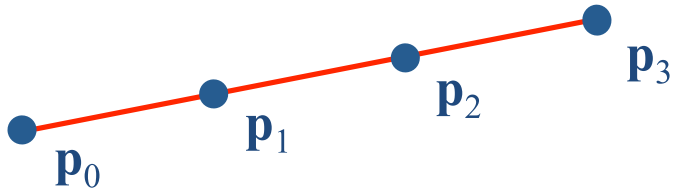
$$\mathbf{x}(t) = \begin{bmatrix} x \\ y \end{bmatrix} = B_0^3 \mathbf{p_0} + B_1^3 \mathbf{p_1} + B_2^3 \mathbf{p_2} + B_3^3 \mathbf{p_3}$$

- Symmetrie
  - Kurve durch $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ = Kurve durch $(\mathbf{p}_3, \mathbf{p}_2, \mathbf{p}_1, \mathbf{p}_0)$
  - Lediglich die „Durchlaufrichtung“ von t ändert sich



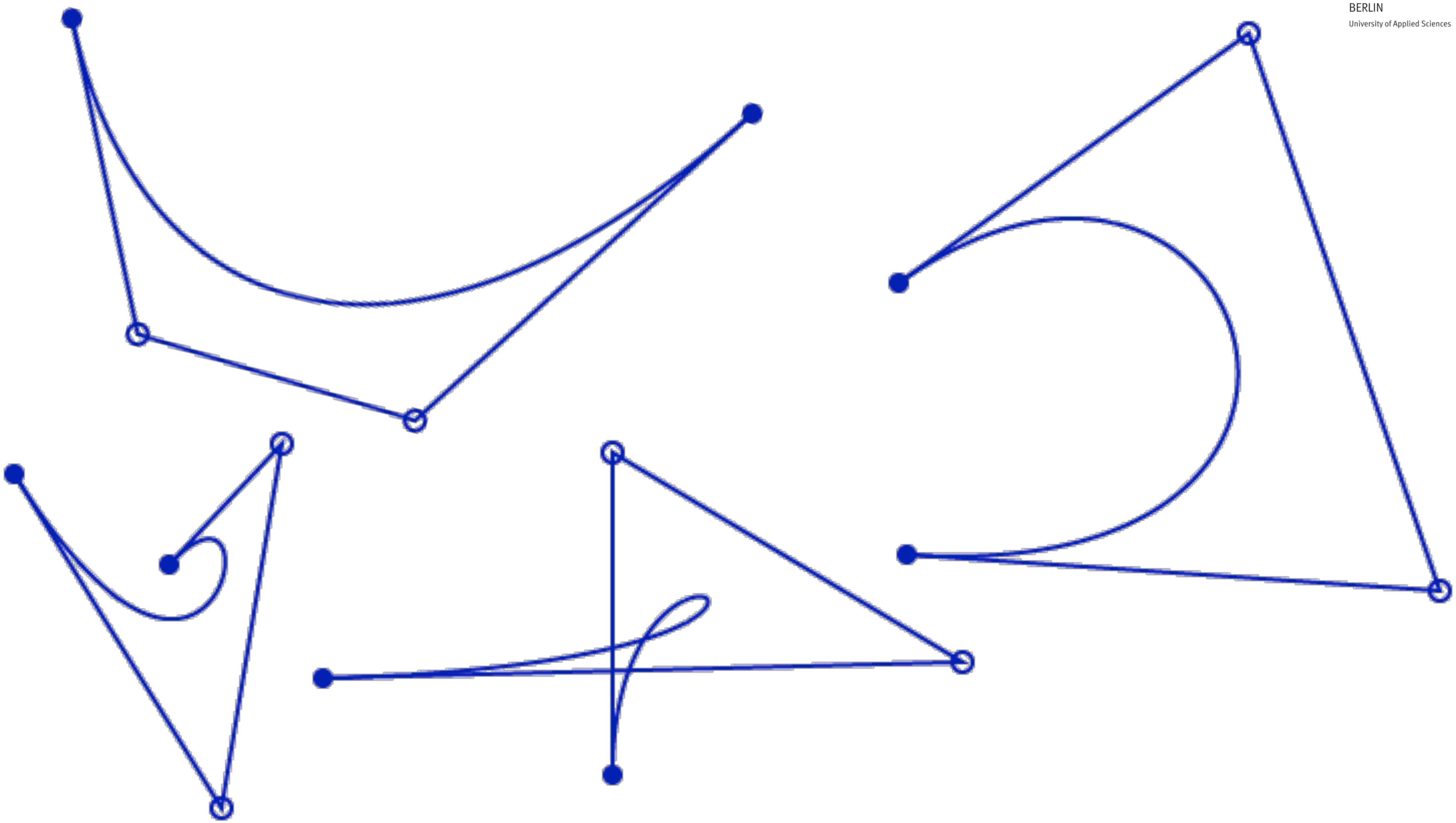
Rotationen des Kontrollpolygons

- Invarianz unter affinen Transformationen
  - Transformation des Kontrollpolygons überträgt sich auf die Kurve
- Spezialfall lineare Interpolation
  - Wenn Kontrollpunkte gleichverteilt auf Strecke zwischen $\mathbf{p}_0$ und $\mathbf{p}_3$, dann Kurve = lineare Interpolante



Lineare Interpolation mittels Bézier-Kurve

Beispiele kubischer Bézier-Kurven



Fragen: Interpolation und kubische Bézier-Kurven

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

- Wie interpoliert man linear zwischen zwei Datenpunkten?
- Was können diese Datenpunkte z.B. darstellen?
- Was beschreiben (t-1) und t? Was ist bei t=0 und t=1?
- Was ist eine parametrische Kurve?
- Definitionsbereich? Wertebereich? 2D vs 3D?
- Welche Struktur hat eine kubische Bézierkurve?
  Wodurch wird sie vollständig beschrieben?
- Welche Zusammenhänge gibt es zwischen den Kontrollpunkten und der Kurve?

Berechnung der Ableitung

- Ableiten der Kurve $\mathbf{x}(t)$ nach t

$$\mathbf{x}(t) = \underbrace{(1-t)^3}_{-3(1-t)^2}\mathbf{p}_0 + \underbrace{3(1-t)^2 t}_{-6(1-t)t + 3(1-t)^2}\mathbf{p}_1 + \underbrace{3(1-t)t^2}_{-3t^2 + 6(1-t)t}\mathbf{p}_2 + \underbrace{t^3}_{3t^2}\mathbf{p}_3$$

- ergibt:

$$\dot{\mathbf{x}}(t) = -3(1-t)^2\mathbf{p}_0 + \left(3(1-t)^2 - 6(1-t)t\right)\mathbf{p}_1 + \left(+6(1-t)t - 3t^2\right)\mathbf{p}_2 + 3t^2\mathbf{p}_3$$

- umgruppiert:

$$\dot{\mathbf{x}}(t) = 3(1-t)^2\left[\mathbf{p}_1 - \mathbf{p}_0\right] + 6(1-t)t\left[\mathbf{p}_2 - \mathbf{p}_1\right] + 3t^2\left[\mathbf{p}_3 - \mathbf{p}_2\right]$$

Vorwärtsdifferenzen, Tangenten

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

- Umgeformte Ableitung der Kurve

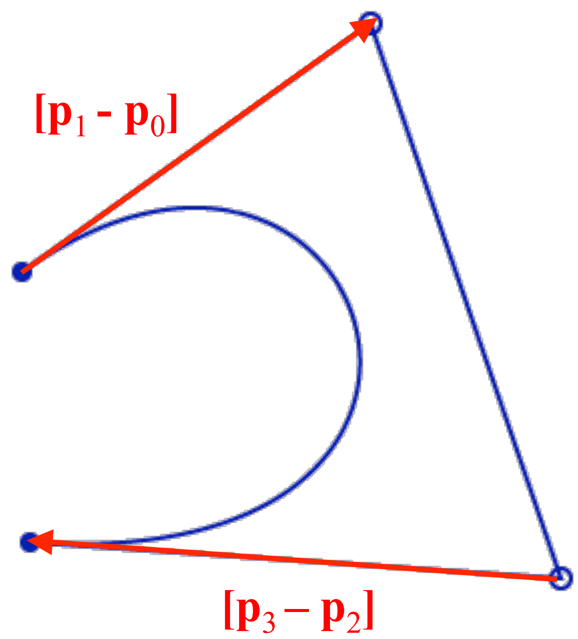
$$\dot{\mathbf{x}}(t) = 3(1-t)^2 \underbrace{[\mathbf{p}_1 - \mathbf{p}_0]}_{\text{Vorwärtsdifferenz in } \mathbf{p}_0} + 6(1-t)t \underbrace{[\mathbf{p}_2 - \mathbf{p}_1]}_{\text{Vorwärtsdifferenz in } \mathbf{p}_1} + 3t^2 \underbrace{[\mathbf{p}_3 - \mathbf{p}_2]}_{\text{Vorwärtsdifferenz in } \mathbf{p}_2}$$

- Setze $t = 0$ ein:

$$\dot{\mathbf{x}}(0) = 3\,[\mathbf{p}_1 - \mathbf{p}_0]$$

- Setze $t = 1$ ein:

$$\dot{\mathbf{x}}(1) = 3\,[\mathbf{p}_3 - \mathbf{p}_2]$$



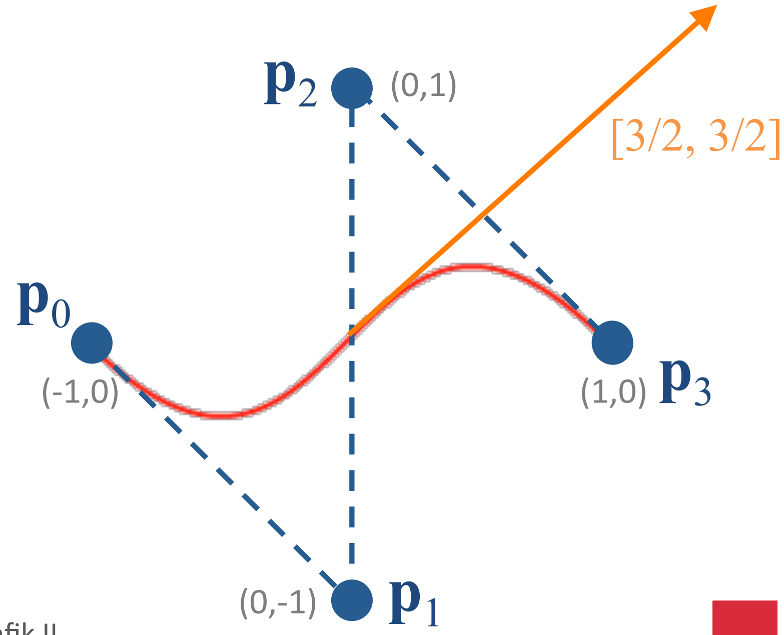
Ableitung an Stelle t
= Tangente in $\mathbf{x}(t)$

Tangenten an den Endpunkten:
sehr wichtig für das
Zusammensetzen von Kurven

Beispiel: Ableitung einer Bézier-Kurve bei $t = 0.5$

$$\dot{\mathbf{x}}(t) = \underbrace{\underbrace{3(1-t)^2}_{3/4}\,\underbrace{[\mathbf{p}_1 - \mathbf{p}_0]}_{\begin{matrix}1\\-1\end{matrix}}}_{\begin{matrix}3/4\\-3/4\end{matrix}} + \underbrace{\underbrace{6(1-t)t}_{6/4}\,\underbrace{[\mathbf{p}_2 - \mathbf{p}_1]}_{\begin{matrix}0\\2\end{matrix}}}_{\begin{matrix}0\\3\end{matrix}} + \underbrace{\underbrace{3t^2}_{3/4}\,\underbrace{[\mathbf{p}_3 - \mathbf{p}_2]}_{\begin{matrix}1\\-1\end{matrix}}}_{\begin{matrix}3/4\\-3/4\end{matrix}}$$

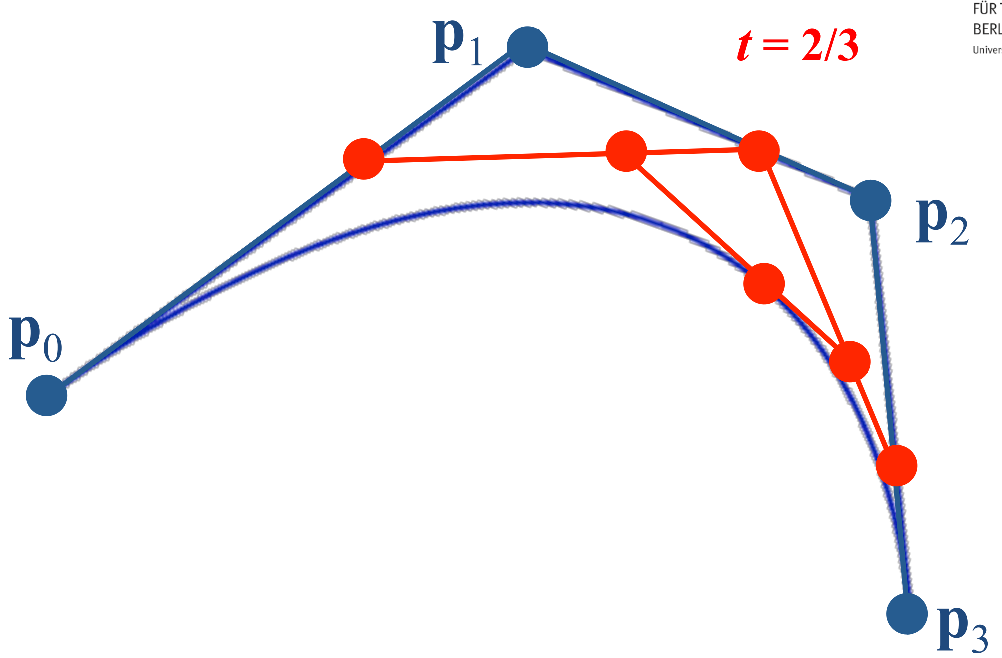
$$= \begin{matrix}3/4 + 0 + 3/4\\ -3/4 + 3 - 3/4\end{matrix} = \begin{matrix}6/4\\ -6/4 + 12/4\end{matrix} = \begin{matrix}3/2\\ 3/2\end{matrix}$$

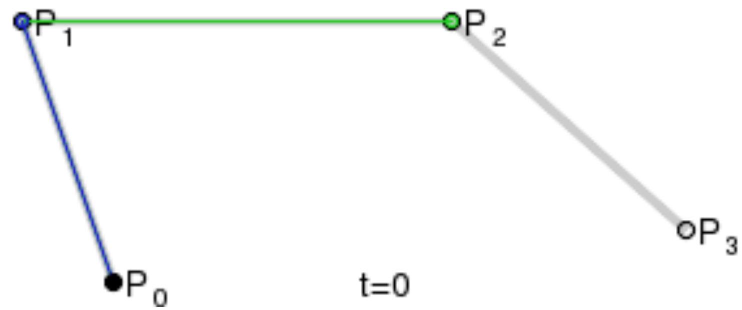


Der Algorithmus von *de Casteljau*

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

- Aufgabe
  - berechne $\mathbf{x}(t)$ auf einer kubischen Bézierkurve
- Gegeben
  - $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$
  - t
- Algorithmus
  - Drei einfache Schritte
  - Ausschließlich lineare Interpolation !!!
  - Liefert exakten Punkt auf Kurve
  - Liefert Tangente / Ableitung in dem Punkt





Animation: de.wikipedia.org

Der Algorithmus von *de Casteljau*

- Schritt 1

$$\mathbf{a}_0 = (1-t)\mathbf{p}_0 + t\mathbf{p}_1$$

$$\mathbf{a}_1 = (1-t)\mathbf{p}_1 + t\mathbf{p}_2$$

$$\mathbf{a}_2 = (1-t)\mathbf{p}_2 + t\mathbf{p}_3$$

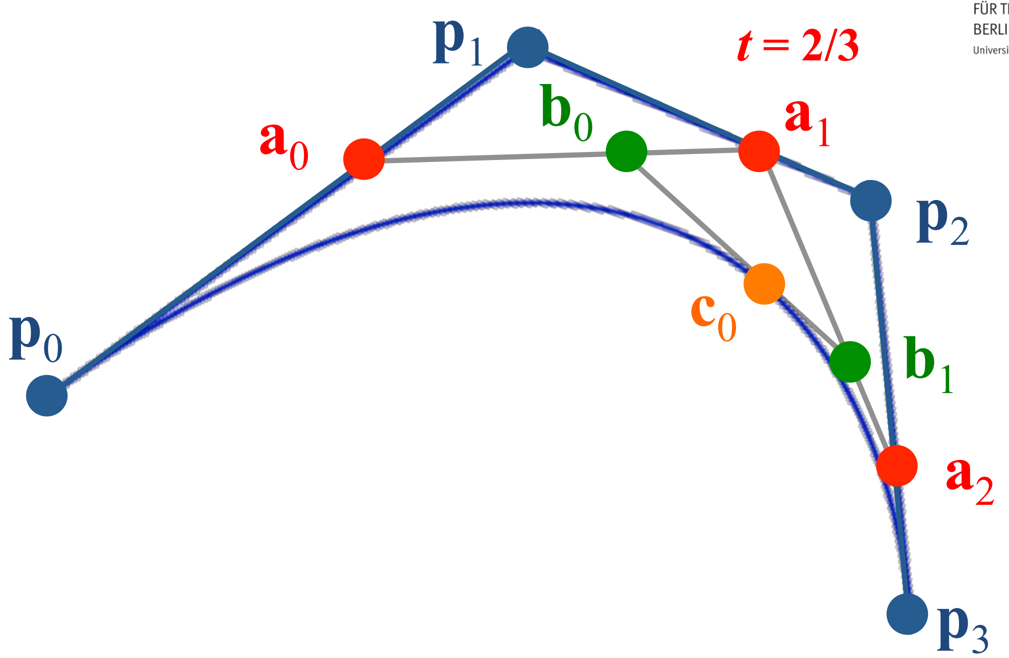
- Schritt 2

$$\mathbf{b}_0 = (1-t)\mathbf{a}_0 + t\mathbf{a}_1$$

$$\mathbf{b}_1 = (1-t)\mathbf{a}_1 + t\mathbf{a}_2$$

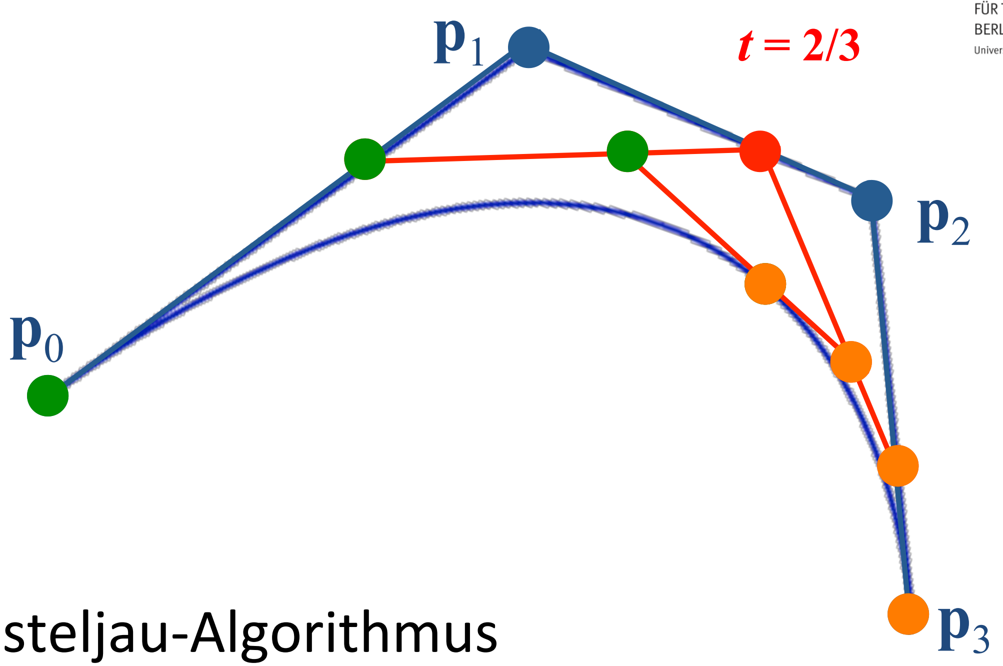
- Schritt 3

$$\mathbf{c}_0 = (1-t)\mathbf{b}_0 + t\mathbf{b}_1$$



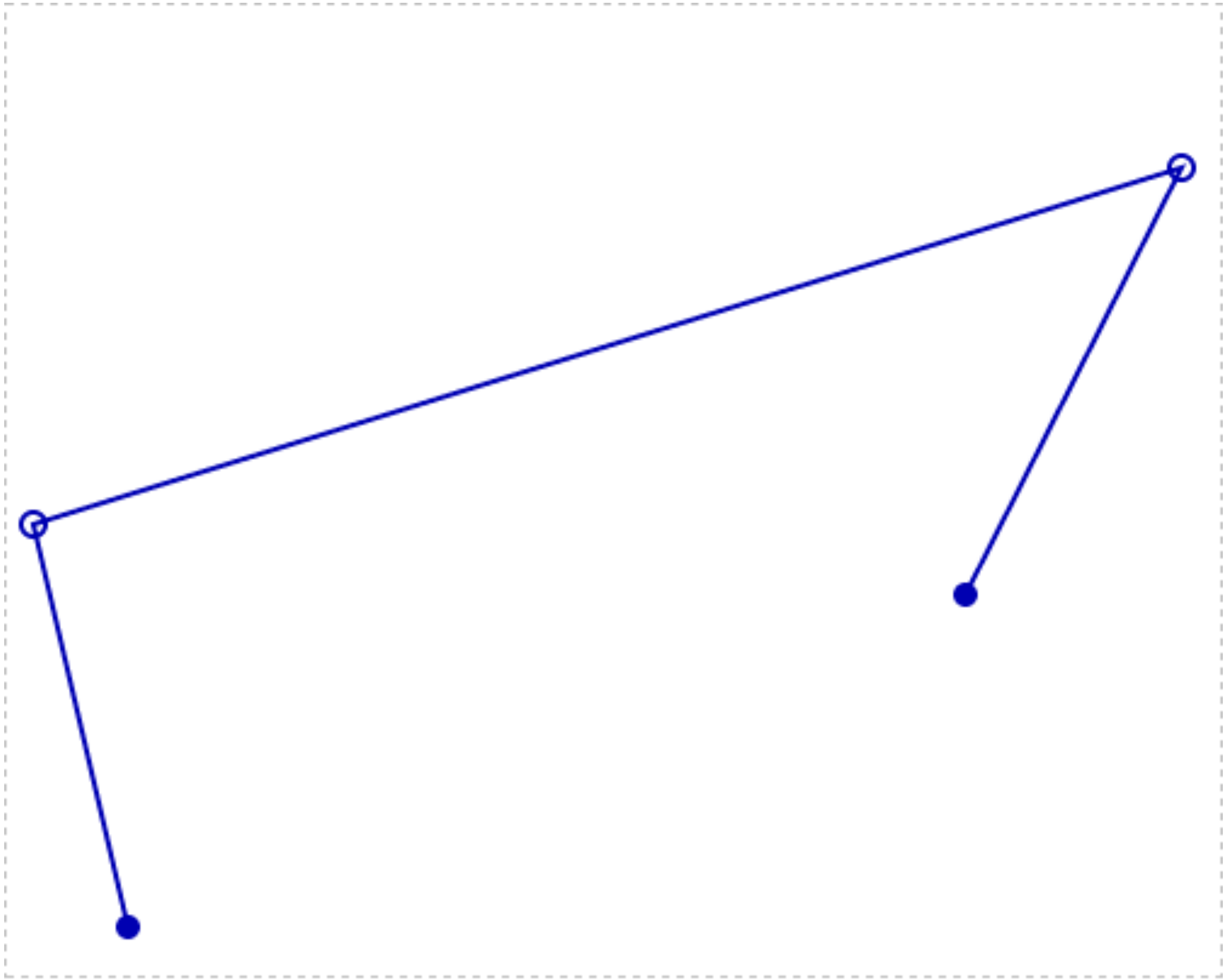
Subdivision mittels de Casteljau

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

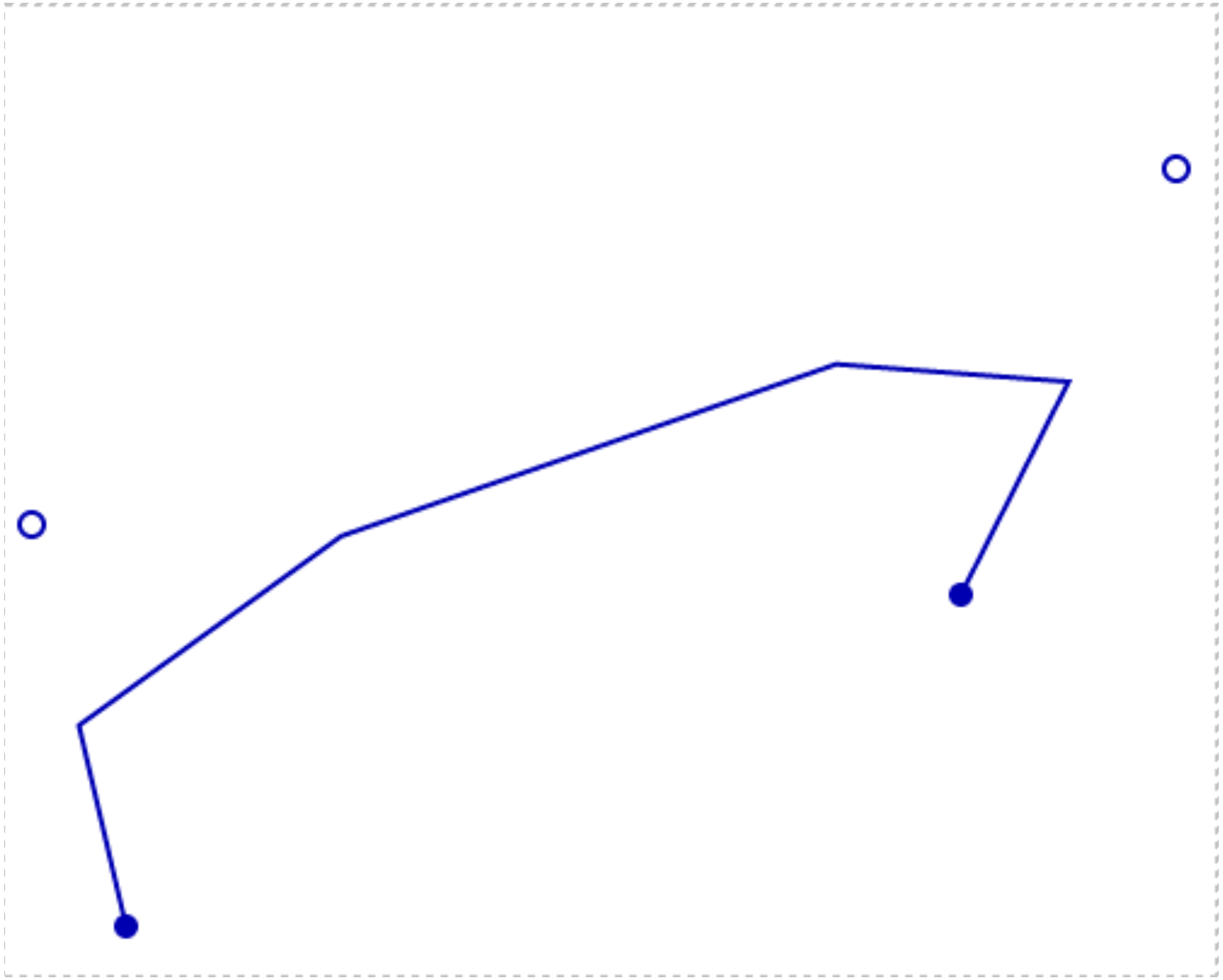


- Durch Anwendung des de-Casteljau-Algorithmus erhält man zwei neue Kontrollpolygone („links“ + „rechts“)
- Split der Kurve in zwei Segmente
- Kontrollpolygone konvergieren gegen Kurve!

Gleichmäßige Unterteilung bei $t = 0.5$: Stufe 0

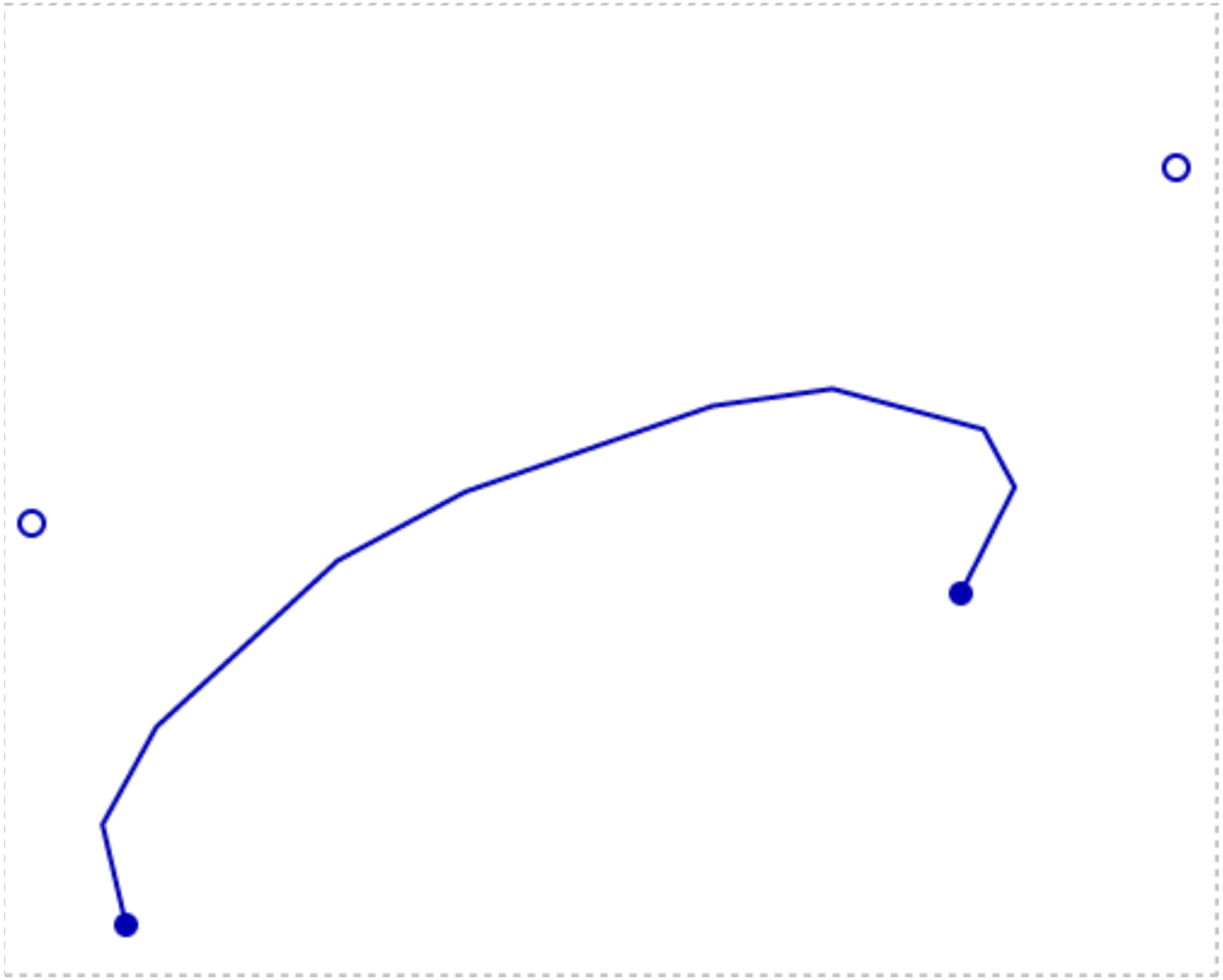


Gleichmäßige Unterteilung bei $t = 0.5$: Stufe 1

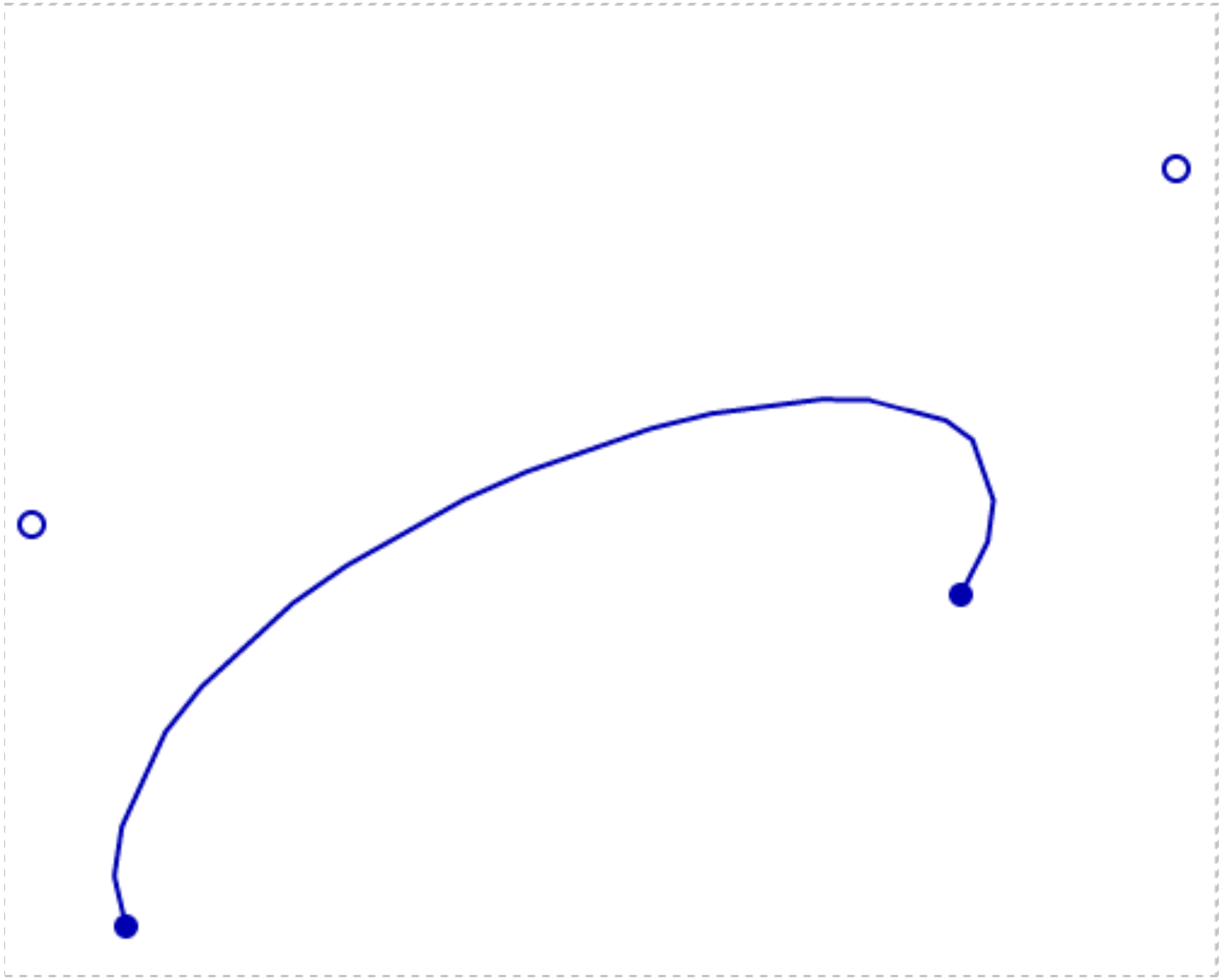


BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

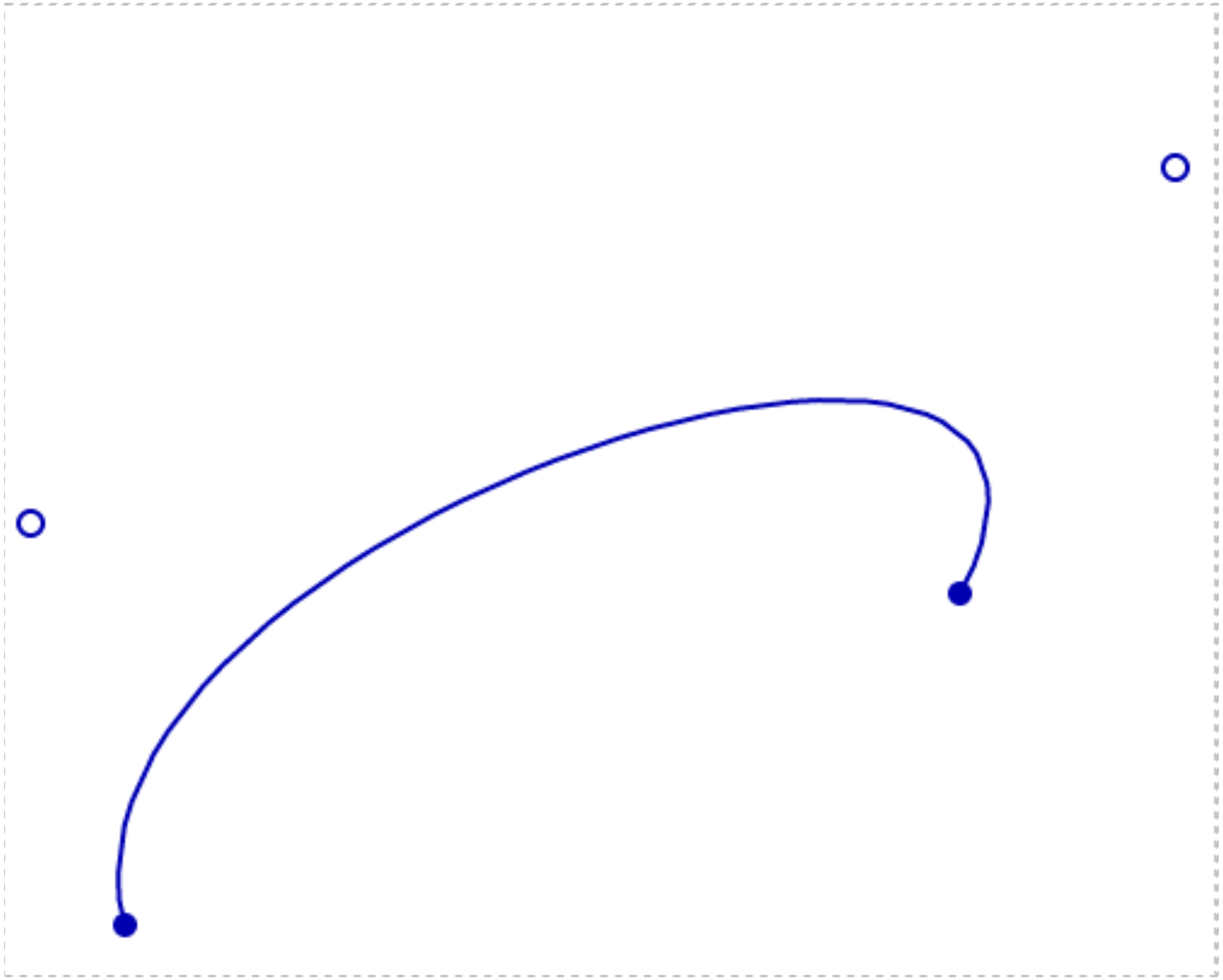
Gleichmäßige Unterteilung bei $t = 0.5$: Stufe 2



Gleichmäßige Unterteilung bei $t = 0.5$: Stufe 3



Gleichmäßige Unterteilung bei $t = 0.5$: Stufe 4



Adaptive Unterteilung

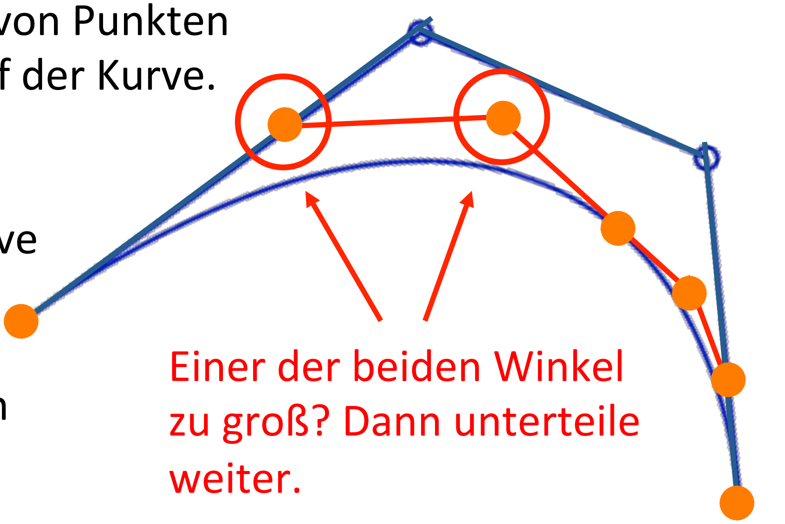
- Adaptive Unterteilung:
  - Unterteile das Kontrollpolygon nur dort weiter, wo es die echte Kurve schlecht approximiert
  - Typischerweise dort, wo die Krümmung stärker ist

- Mögliche Maße für die Qualität der Approximation
  - „Abstand“ Polygon-Kurve

    Least Squares: Summe der Quadrate des Abstands von Punkten auf dem Polygon zu dem jeweils nächsten Punkt auf der Kurve.

  - Starke Krümmung?

    Krümmung = Betrag der zweiten Ableitung der Kurve
    = Änderung der Tangentenrichtung
    → „Billige“ Abschätzung z.B. durch die Winkel zwischen den drei Segmenten des Kontrollpolygons...



Fragen: Ableitung, Tangenten, de Casteljau

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

- Welche Bedeutung hat die Ableitung einer Kurve?
- Welchen Zusammenhang gibt es zwischen den Kontrollpunkten und der Ableitung einer kubischen Bezier-Kurve?
- Wofür ist die Ableitung in den Endpunkten wichtig?
- Wozu dient der de-Casteljau-Algorithmus?
- Wie läuft der de-Casteljau-Algorithmus ab?
- Wieso kann de Casteljau für Subdivision verwendet werden?

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

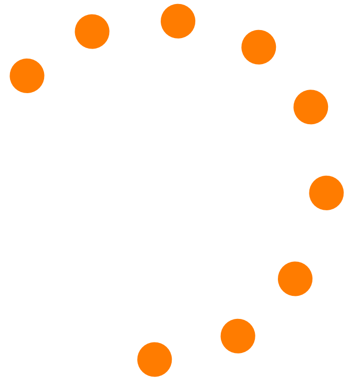
Kurven: Auswertung, Darstellung und Interaktion

Darstellung einer parametrischen Kurve

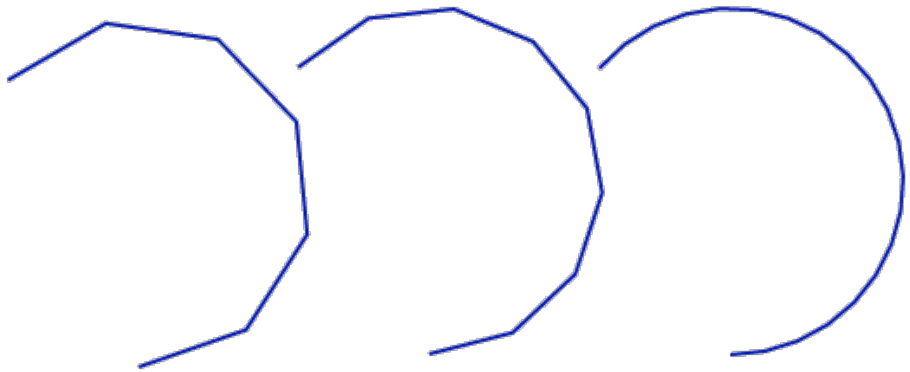
- Kurve gegeben durch $$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f(t) \\ g(t) \end{bmatrix}$$

- Z.B. in JavaScript einfach so: ...

```
var f = function(t) { return Math.sin(t); }
var g = function(t) { return Math.cos(t); }
var c = new ParametricCurve(f,g);
```



- Zwei Möglichkeiten der Darstellung:
  - Punkte malen, und hoffen, dass Sie dicht genug beieinander liegen
  - Punkte mittels Liniensegmenten verbinden



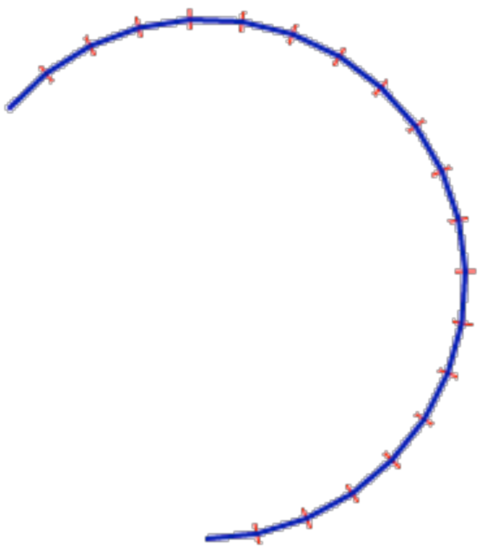
- Wie viele Punkte / Segmente malen?
  - Möglichst wenige → schneller
  - Je mehr, desto genauere Approximation

Darstellung einer beliebigen parametrischen Kurve

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

- Auswerten der Kurve an N+1 Stellen
  - Bestimmte $N+1$ Stellen t_i , an denen die Kurve ausgewertet wird
  - Für jedes t_i werte die Kurve aus: $(x_i , y_i) = (f(t_i) , g(t_i))$

- Darstellen der Kurve durch N Liniensegmente
  - Schleife von $i = 1 \dots N$
  - Zeichne Linie von (x_{i-1} , y_{i-1}) zu (x_i , y_i)



- Demo / Aufgabe 1.2

Adaptive Darstellung von Bézier-Kurven mittels Subdivision

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

- Bézier-Kurven kann man genauso mittels N Liniensegmenten darstellen wie jede beliebige parametrische Kurve
  - Verwende für $f(t)$ und $g(t)$ einfach die Bézier-Formeln

- Noch besser: rekursiv-adaptiv:
  - Bezier_rekursiv(p_0, p_1, p_2, p_3, t)
    - Wenn Krümmung in p1 und p2 nicht mehr zu groß
      - Zeichne Polygon (p0,p1,p2,p3)
      - Return
    - Unterteile in zwei Kontrollpolygone
    - Zeichne linke Hälfte rekursiv
    - Zeichne rechte Hälfte rekursiv
  - Vorteil: keine Annahmen über die Anzahl von Segmenten

Demo?

Benutzer-Interaktion mit Bezier-Kurven

- Verschieben von Anfangs- und Endpunkt
- Verschieben der beiden restlichen Kontrollpunkte
- Darstellung der Kontrollpolygon-Linie, um die Tangentenrichtungen zu visualisieren
- Beim Zusammensetzen mehrerer Kurven:
  - Richtung und Länge der Tangenten gleich (C1-Stetigkeit)
  - Richtung der Tangenten gleich (G1-Stetigkeit)

Demo?

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

JavaScript II
Prototypen, Vererbung und Co.

Prototypen

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

- Jedes Objekt besitzt ein Attribut namens `prototype`
  - `prototype` ist ein Objekt und hat ggf. seinerseits einen Prototyp...

```
var a = { x:3 };
var b = { y:4 };
b.prototype = a;
```

  - Beim *lesenden* Zugriff auf ein Attribut durchsucht JavaScript potentiell die gesamte Prototyp-Kette.

```
b.x   → 3
b.y   → 4
a.y   → undefined
```

  - Beim *schreibenden* Zugriff auf ein Attribut wird der Prototyp nicht verändert; das Attribut wird ggf. neu angelegt.

```
b.x = 7;   ← neues Attribut von b
b.x        → 7
a.x        → 3
b.z = 4;   ← neues Attribut von b
```

Eigene Attribute vs. Prototypen-Attribute

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

- Wie unterscheide ich, ob ein Objekt selbst eine bestimmte Eigenschaft besitzt, oder nur einer seiner Prototypen?

```
var a = { x:3 };
var b = { y:4 };
b.prototype = a;
if(b.x != undefined) ...
```

→ wenn **a** oder **b** ein Attribut **x** besitzen...

- Dafür gibt es `hasOwnProperty()`

```
a.hasOwnProperty("x")
b.hasOwnProperty("x")
```

→ **false**

→ **true**

Verdeckung und Löschung

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

- Attribute gleichen Namens verdecken sich in der Prototyp-Kette

```
var base = { x:0 };
var obj = { x:1, y:2, z:3 };
obj.prototype = base;
obj.x                   → 1
obj.prototype.x         → 0
```

- `delete` erlaubt das Löschen von Variablen oder Attributen. Aber es löscht Attribute nicht „richtig" – die entsprechenden Attribute des Prorotyps bleiben verdeckt*

```
delete obj.x;
obj.x                   → undefined  (nicht etwa 0)
delete obj;
obj                     → exception: obj is undefined
```

*) laut Crockford sollte das eigentlich nicht so sein...

Selektive Vererbung über den Prototyp

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

```
var Base = function(...) {...};
Base.prototype.f = ...;
Base.prototype.g = ...;

var Derived = function(...) {..};
Derived.prototype.f = Base.prototype.f;
Derived.prototype.g_super = Base.prototype.g;
Derived.prototype.g = function(...) {
    this.g_super(...);
    // do some more...
};
```

→ Erben einer Methode

→ Erben unter anderem Namen

→ Verwendet intern eine geerbte Funktion

→ Selektive Vererbung sehr mächtig, aber schnell verwirrend!
Oftmals ist Komposition der sauberere Weg!

Vergleich mit klassischer Vererbung: http://www.crockford.com/javascript/inheritance.html

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

JavaScript II
Weitere Typen und Konstrukte

Strings

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

```
var str = 'Say "hello" to my new BELLO!';
var str = "Say 'hello' to my new BELLO!";
```

```
str.length                → 28
str.slice(5,5)            → "'hello'"
str.slice(-6)             → "BELLO!"
str.split(" ")            → [ "Say", "'hello'", "to", "my",
                              "new", "BELLO!" ]

str.match(/.ello/gi)      → [ "hello", "BELLO" ]
str.replace("BELLO","dog")
```

Regular Expression, dazu später mehr.

http://www.w3schools.com/jsref/jsref_obj_string.asp

undefined, null und Co.: “falsy values“

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

- Alle diese Werte gelten als *falsy*
  - `false`
  - `0`
  - `""`
  - `NaN`
  - `null`
  - `undefined`

```
x = NaN;
if(!x) {
  // this could be undefined, NaN, 0, ...
};
```

- Was wann verwenden?
  - `0` = reguläres Ergebnis numerischer Berechnungen
  - `NaN` = irreguläres Ergebnis, z.B. Division durch 0
  - `null` eher für explizites „verweist auf kein Objekt“ (seltener)
  - `undefined` steht für „existiert nicht“

Objekt-Attribute und die Operatoren || und &&

- „Auffüllen" eines Objekts mit Standard-Werten

```
var middle = person.middle_name || "(none)";
```

entspricht:

```
var middle = person.middle_name;
if(middle == undefined) {
  middle = "(none)";
};
```

- Exception wegen eines nicht definierten Attributs vermeiden:

```
// obj.model is undefined
var id = obj.model.id;
var id = obj.model && obj.model.id;
```

→ *Exception: obj.model is undefined*

→ **`id === undefined`**

Exceptions – String vs. Error

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

```
try {
  // ...
  throw "just throw a string";
} catch(err) {
  console.log("caught it: " + err)
};
```

```
try {
  // ...
  throw new Error("message");
} catch(err) {
  console.log("caught it: " + err.message)
};
```

Code einfügen mittels `eval()`

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

- `eval(str)` evaluiert einen beliebigen String so, als würde er in der aktuellen Closure im Code stehen.

```
var Scene = function(bgColor) (
    this.bgColor = bgColor;
    eval("this.draw = function(context) {...};");
    ...
);
```

- Natürlich ist `eval()` in echten Webanwendungen **böse**!
- Aber wirklich nützlich beim Skripting. Beispiel: Kurvenplotter

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Hinweise zur Übungsaufgabe

Übungsaufgaben 1.2 und 1.3

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

- Beliebige parametrische Kurve darstellen
  - Benutzer gibt Formel frei ein, Programm verwendet `eval()`
  - „Tick Marks" darstellen, senkrecht zur Kurve
- Bezier-Kurve im Canvas darstellen
  - nicht mittels `bezierCurveTo()`, sondern selbstgemacht
  - für sehr gute Note: adaptiv mittels de Casteljau - Unterteilung
- Kontrollpolygon interaktiv darstellen
  - Point Draggers
  - Polygon / Tangenten zeichnen (z.B. mittels neuem Dragger-Typ)
- Mehrere Bézier-Kurven zusammenfügen
  - Neues Objekt „zusammengesetzte Kurve"
  - Übergänge immer C0-stetig, wahlweise C1-stetig

1.3 / für nächste Woche

Anhang: Gültigkeitsbereiche und Hoisting

Gültigkeitsbereich und *Hoisting*

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

wird interpretiert wie

```
var foo = 1;
function bar() {

   window.console.log(foo);
   {
     var foo = 10;
   }
   window.console.log(foo);
};
bar();
```

→ undefined
10

```
var foo = 1;
function bar() {
   var foo;
   window.console.log(foo);
   {
     foo = 10;
   }
   window.console.log(foo);
};
bar();
```

→ Hoisting: alle Deklarationen werden an den Anfang der Funktion "gehoben"

- Ratschlag: immer alle „var" Definitionen an den Anfang der Funktion schreiben. Alles andere verwirrt nur!

http://www.adequatelygood.com/2010/2/JavaScript-Scoping-and-Hoisting