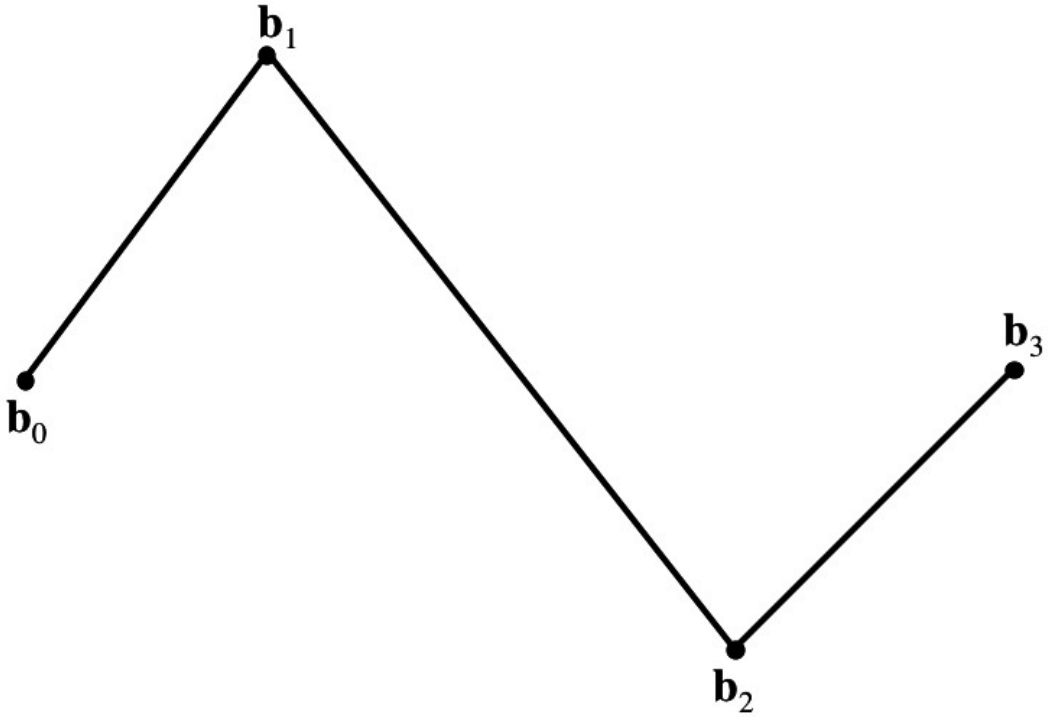
Übungsblatt 9

Willkommen zur neunten Übung der Veranstaltung Generative Computergrafik. Ziel dieses Übungsblatts ist, dass Sie sich mit parametrischen Kurven, insbesondere mit Bézier und B-Spline-Kurven vertraut machen.

Dieses Blatt enthält die dritte verpflichtende Abgabe. Die **Aufgaben 3** ist bis Montag, den 03. Juli 2023 um 23:55 Uhr über `https://read.mi.hs-rm.de` abzugeben. Spätere Abgaben oder Abgaben auf anderem Wege, wie z.B. per Email können leider nicht berücksichtigt werden und müssen mit 0 Punkten bewertet werden. Sie sollten daher **frühzeitig mit der Arbeit an diesem Blatt beginnen** und evtl. auch schon früh erste lauffähige Versionen ihrer Lösung hochladen.

> Um das Praktikum zu bestehen müssen Sie *insgesamt (über alle drei Abgaben) mindestens 50% aller erreichbaren Punkte* erreichen, wobei *bei jeder der drei Abgaben mindestens 25% der erreichbaren Punkte* erreicht werden müssen. **Wichtig:** *Alle von Ihnen abgegebenen Programme müssen auf den Poolrechnern unter Linux laufen!*

Aufgabe 1. Das unten abgebildete Polygon sei das Kontrollpolygon einer, über dem Intervall $[0, 1]$ definierten, Bézier-Kurve.



1. Welchen Polynomgrad hat die zugehörige Bézier-Kurve?
2. Kann die zugehörige Bézier-Kurve einen Wendepunkt haben?
3. Muss die zugehörige Bézier-Kurve einen Wendepunkt haben?

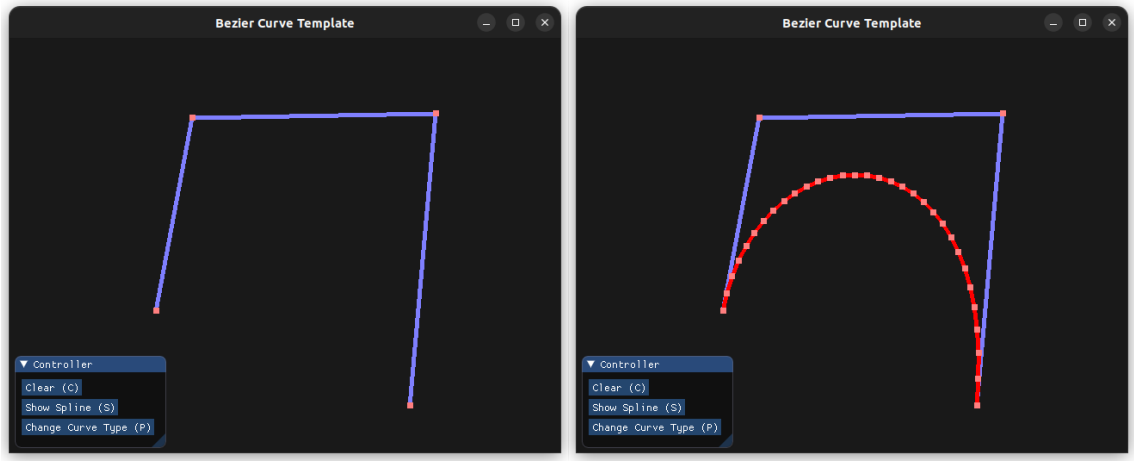
4. Zeichnen Sie die Konvexe-Hülle des Kontrollpolygons ein.

5. Zeichnen Sie die Kurven-Punkte und Tangenten zu den Parameterwerten 0, $\frac{1}{2}$ und 1 ein.

6. Zeichnen Sie eine Näherung der Bézier-Kurve ein, indem Sie die Kurve mit Hilfe des de Casteljau-Algorithmus an den Parameterwerten $\frac{1}{4}$, $\frac{1}{2}$ und $\frac{3}{4}$ unterteilen. Eine gute Näherung an die ursprüngliche Bézier-Kurve besteht dann aus den Kontrollpolygonen zu den Bézier-Kurven über den Intervallen $[0, \frac{1}{4}]$, $[\frac{1}{4}, \frac{1}{2}]$, $[\frac{1}{2}, \frac{3}{4}]$ und $[\frac{3}{4}, 1]$.

Aufgabe 2. Unter `www.mi.hs-rm.de/~schwan/teaching/computergrafik/` finden Sie das Programm `bezierTemplate.py`. Es erlaubt Ihnen, mit der Maus Punkte in einem Fenster auszuwählen. Sind mindestens zwei Punkte ausgewählt worden, wird ein Polygon durch die Punkte gezeichnet (siehe Abbildung unten links).

Erweitern Sie das Programm so, dass es zusätzlich die Bézier-Kurve einzeichnet, welche durch das (Kontroll-)Polygon definiert wird (siehe Abbildung unten rechts).



Implementieren Sie die Darstellung der Bézier-Kurve auf die beiden folgenden, unterschiedlichen Arten.

1. Gegen Sie davon aus, dass die Bézier-Kurve über dem Parameterintervall $[a, b]$ definiert ist und berechnen Sie mit Hilfe des de Casteljau-Algorithmus $N + 1$ Kurven-Punkte $\mathbf{x}(t_i)$ zu den Parameterwerten $t_i = a + \frac{i}{N}(b - a), i = 0, \ldots, N$. Zur Vereinfachung können Sie $a = 0$ und $b = 1$ setzen. Als Näherung für die Bézier-Kurve zeichnen Sie dann das Polygon durch die Punkte $\mathbf{x}(t_i)$, $i = 0, \ldots, N$.

2. Unterteilen Sie die Bézier-Kurve mit Hilfe des de Casteljau-Algorithmus K-mal in der Mitte des Parameterintervalls. Setzt man $a = 0$ und $b = 1$, so entstehen 2^K Bézier-Kurven über den Parameterintervallen $\left[\frac{i}{2^K}, \frac{i+1}{2^K}\right]$, $i = 0, \ldots, 2^K - 1$ welche alle zusammen die ursprüngliche Bézier-Kurve beschreiben. Zeichnen Sie die Kontroll-Polygone dieser Bézier-Kurven als Näherung für die ursprüngliche Bézier-Kurve.

   Hinweis: Dieses Verfahren *konvergiert sehr schnell* gegen die Bézier-Kurve. Es reichen daher im Allgemeinen 3 oder 4 Unterteilungsschritte, um eine sehr gute Näherung der Bézier-Kurve zu erreichen.

Aufgabe 3. Schreiben Sie ein `glfw`-Programm, welches die Eingabe von Punkten $\mathbf{p}_0, \ldots, \mathbf{p}_n$ mit Hilfe der Maus erlaubt und das Polygon durch die Punkte sowie eine Kurve zeichnet. Die dargestellte Kurve soll eine **B-Spline-Kurve**

$$\mathbf{x}(t) = \sum_{i=0}^{n} N_i^k(t)\mathbf{p}_i$$

der Ordnung k mit uniformen Knotenvektor (der Länge $n + k + 1$)

$$K = \{\underbrace{0, \ldots, 0}_{k}, 1, 2, \ldots, n - (k-1), \underbrace{n - (k-2), \ldots, n - (k-2)}_{k}\}$$

und (Kontroll-)Punkten $\mathbf{p}_0, \ldots, \mathbf{p}_n$ sein.

1. (5 Punkte) Schreiben Sie dazu eine Funktion

   ```
   deboor(degree, controlpoints, knotvector, t)
   ```

   die mit Hilfe des *de-Boor-Algorithmus* einen Punkt auf einer B-Spline-Kurve berechnet. Die Parameter der Funktion haben dabei folgende Bedeutungen:

   - `degree`: **Polynomgrad** der B-Spline-Kurve
   - `controllpoints`: **Kontrollpunkte** der B-Spline-Kurve
   - `knotvector`: **Knotenvektor** der B-Spline-Kurve
   - `t`: **Parameterwert**, zu dem der Punkt auf der B-Spline-Kurve berechnet werden soll

   Verwenden Sie die Funktion `deboor()` dann, um eine feste Anzahl m von Punkten auf der Kurve zu berechnen und anschließend als Polygon darzustellen. Beachten Sie, dass eine B-Spline-Kurve der Ordnung k erst dann vollständig definiert ist, wenn wenigstens k Punkte angegeben wurden. Bis zur

Eingabe des k-ten Punktes soll lediglich das (Kontroll-)Polygon durch die Punkte dargestellt werden.

(5 Punkte) Ihr Programm soll es weiterhin ermöglichen die beiden Parameter k (*Ordnung* der Kurve) und m (*Anzahl* zu berechnender *Kurvenpunkte)* zu beeinflussen. Die *Ordnung* soll dabei mit den Tasten `k` bzw. `K`, die *Anzahl Kurvenpunkte* mit den Tasten `m` bzw. `M` *verringert* bzw. *erhöht* werden können.

2. (10 Punkte) Erweitern Sie Ihr Programm aus Teil 1. so dass es eine **nicht uniforme rationale B-Spline-Kurve (NURBS)**

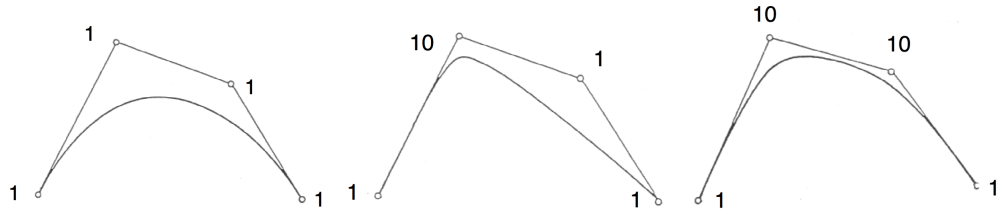
$$\mathbf{x}(t) = \sum_{i=0}^{n} N_i^k(t)\tilde{\mathbf{p}}_i$$

der Ordnung k mit Knotenvektor (der Länge $n+k+1$)

$$K = \{\underbrace{0,\ldots,0}_{k}, 1, 2, \ldots, n-(k-1), \underbrace{n-(k-2),\ldots,n-(k-2)}_{k}\}$$

und *homogenen (Kontroll-)Punkten* $\tilde{\mathbf{p}}_0 = \omega_0(x_0, y_0, 1)^\top, \ldots, \tilde{\mathbf{p}}_n = \omega_n(x_n, y_n, 1)^\top$ darstellen kann. Die Gewichte ω_i können als zusätzliche Designparameter eingesetzt werden.

Ihr Programm soll zusätzlich zur Funktionalität aus Teil 1. die Möglichkeit bieten, das Gewicht ω_i jedes Kontrollpunkts $\mathbf{p}_i$ im Intervall $[1, 10]$ mit der Maus zu ändern (z.B. indem man bei gedrückter SHIFT-Taste auf den entsprechenden Kontrollpunkt klickt und die Maus bewegt). Die nachfolgenden Abbildungen zeigen, wie sich die Kurven bei Veränderung der entsprechenden Gewichte verhält.



Haben alle Gewichte den gleichen Wert (z.B. $\omega_i = 1 \forall i$) so handelt es sich bei der rationalen Kurve um eine einfache polynomielle Kurve (siehe Abbildung links). Wird das Gewicht eines Kontrollpunktes (relativ zu den Gewichten der benachbarten Kontrollpunkte) erhöht, so wird die Kurve zu dem entsprechenden Kontrollpunkt hingezogen (siehe mittlere und rechte Abbildung).