Media Informatics /
Graphics & Geometric Computing

Prof. Dr. Marcel Campen
Steffen Hinderink

UNIVERSITÄT OSNABRÜCK

Computergrafik (SoSe 2023)

Blatt 9

fällig Sonntag **25. Juni**, 24:00

Verspätet eingereichte Abgaben können nicht berücksichtigt werden.

Aufgabe 1 (Theorie: Bézierkurven und -flächen)

Eine kubische Bézierkurve $C : [0,1] \to \mathbb{R}^2$ sei definiert durch die Kontrollpunkte $b_0 = (1,0)^\mathsf{T}$, $b_1 = (4,1)^\mathsf{T}$, $b_2 = (0,1)^\mathsf{T}$ und $b_3 = (3,0)^\mathsf{T}$.

(a) (1P) Zeichnen Sie die Kontrollpunkte und das Kontrollpolygon von C.

(b) (1P) Werten Sie C mittels de Casteljau-Algorithmus *grafisch* (d. h. ohne Rechnung) an der Stelle $t = \frac{1}{2}$ aus.

(c) (1P) Werten Sie C mittels de Casteljau-Algorithmus *grafisch* (d. h. ohne Rechnung) an der Stelle $t = \frac{1}{3}$ aus.

(d) (2P) Werten Sie C mittels de Casteljau-Algorithmus *rechnerisch* an der Stelle $t = \frac{1}{4}$ aus.

(e) (1P) Geben Sie einen Richtungsvektor (Vorzeichen egal) der Tangente an C an der Stelle $t = 1$ an. Begründen Sie Ihre Angabe geometrisch, ohne Rechnung.

(f) (1P) Skizzieren Sie C.

(g) (1P) Eine lineare Bézierkurve $C : [0,1] \to \mathbb{R}^4$ sei gegeben durch $b_0 = (1,2,3,4)^\mathsf{T}$ und $b_1 = (1,3,3,7)^\mathsf{T}$. Werten Sie diese Kurve mittels de Casteljau-Algorithmus rechnerisch an der Stelle $\frac{2}{3}$ aus.

(h) (2P) Eine Bézierfläche $S : [0,1] \times [0,1] \to \mathbb{R}^3$, welche in beide Parameterrichtungen linear ist, sei gegeben durch $b_{00} = (0,-4,0)^\mathsf{T}$, $b_{01} = (-4,0,0)^\mathsf{T}$, $b_{10} = (4,0,0)^\mathsf{T}$ und $b_{11} = (0,2,2)^\mathsf{T}$. Werten Sie diese Fläche rechnerisch an der Stelle $\left(\frac{1}{2}, \frac{3}{4}\right)$ aus.

Aufgabe 2 (Praxis: Bézierkurven)

(a) (5P) Die Klasse `BezierCurve` implementiert eine Bézierkurve vom Grad n. Sie besitzt dazu $n+1$ Punkte im Array `points`, die die Kontrollpunkte der Kurve angeben. Implementieren Sie in dieser Klasse die Methode `drawFrame(t)`.

Diese Methode soll die Kontrollpunkte und sämtliche Zwischenpunkte b_i^k, k von 0 bis n, mit dem de Casteljau-Algorithmus für den Wert t berechnen und zeichnen. Die üblichen Verbindungslinien dieser Punkte sollen ebenfalls gezeichnet werden. Sie können beliebig viele weitere Hilfsmethoden hinzufügen.

Zum Verwalten von Punkten können Sie die Klasse `Point` verwenden. Punkte können mit der Funktion `drawPoint` gezeichnet werden. Zum Verwalten von Linien können Sie die Klasse `Line` verwenden. Linien können mit der Funktion `drawLine` gezeichnet werden.

Bei korrekter Implementierung können die Kontrollpunkte mit der Maus verschoben und der Grad sowie der t-Wert mit den Slidern eingestellt werden.

(b) (5P) Implementieren Sie in der Klasse `BezierCurve` die Methode `drawCurve`, die die eigentliche Bézierkurve zeichnen soll.

Die Kurve soll durch viele Punkte gezeichnet werden, die so nah aneinander liegen, dass sie wie eine Kurve aussehen. Dafür muss also die Kurve für viele t-Werte ausgewertet werden und an jeder Stelle ein Punkt gezeichnet werden. Iterieren Sie dazu über t; eine geeignete Schrittweite ist z. B. 0.001. Damit die Kurve die gleiche Breite wie die Linien hat, sollte der Radius dieser Punkte die halbe Linienbreite (also 1) betragen. Sie können wieder beliebig viele weitere Hilfsmethoden hinzufügen.

Aufgabe 3 (Bonus: Bézierflächen)

Ergänzen Sie den Code so, dass eine Bézierfläche angezeigt werden kann.

Vervollständigen Sie dazu die Funktion `drawSurface` der Klasse `BezierSurface`. Diese enthält im Array `curves` $n+1$ Bézierkurven. Nähern Sie die Fläche durch mehrere Dreiecke an. In die Variablen `vertices` und `faces` können Sie das Netz als Shared Vertex Datenstruktur schreiben um es zu rendern. Die Auflösung, die durch den Slider verändert werden kann, befindet sich in der Variable `numPoints`.

Mit den Schaltflächen kann zwischen Aufgabe 2 und Bonusaufgabe gewechselt werden und das Netz durch den virtuellen Trackball rotiert werden.