


Praktikum Computergrafik, Blatt 5

*** Aufgabe 1 (B-Splines)

In Moodle  finden Sie die Datei `bspline.zip` mit den Rahmenprogrammen:

BSplinePanel.java: JPanel-Klasse mit main-Routine zur Darstellung, muss *zunächst* nicht editiert werden. Sie können gegen Ende der Bearbeitung von periodischen zu nicht-periodischen uniformen B-Splines wechseln.

BSpline.java: Das ist die Klasse, in der Sie den Algorithmus zum Zeichnen von B-Spline-Kurven implementieren müssen.

BSplineTest.java: JUnit4-Test, muss nicht editiert werden. Identisch mit dem vom APA-Server ausgeführten Test.

Point.java: Einfache Klasse, die einen 2D-Punkt repräsentiert.

Bedienung des grafischen Panels: BSplinePanel erlaubt es, durch Mausklicks eine beliebige Zahl von Kontrollpunkten P_0, \dots, P_{n-k} zu wählen. Dabei werden die Punkte als kleine Quadrate gezeichnet sowie der verbindende Linienzug (etwas fälschlich als Kontrollpolygon bezeichnet).

Durch erneutes Klicken auf einen bereits gewählten Punkt wird die Auswahl von Kontrollpunkten beendet und die Berechnung der B-Spline-Kurve mit $k = 2$ gestartet. Dies ist allerdings noch zu implementieren (s. Teilaufgaben unten).

Nach dem Zeichnen der B-Spline-Kurve können Sie Kontrollpunkte mit der Maus über *drag and drop* verschieben. Die Kurve folgt dem geänderten Kontrollpolygon.

Ein weiterer Klick ins Panel abseits von Kontrollpunkten inkrementiert k , d.h. man kann den Grad dynamisch hochschalten.

a) Implementieren Sie in der Klasse BSpline den Konstruktor

```
BSpline(List<Point> points, int k, double h)
```

Die Bedeutung der Parameter ist in den JavaDoc-Kommentaren angegeben. Speichern Sie die Parameter in Attribute. Sie können, falls Sie die Notwendigkeit dafür sehen, im weiteren Verlauf noch weitere Attribute hier initialisieren.

b) Implementieren Sie dann in der Klasse BSpline die Methode

```
double nik(int i, int k, double t)
```

die mit einem Casteljau-artigen Algorithmus für einen Kurvenparameter t den Wert der B-Spline-Basisfunktion $N_{i,k}(t)$ berechnet. Vermeiden Sie Rekursion, da diese extrem ineffizient ist.

Verwenden Sie stattdessen eine zweidimensionale Hilfsstruktur, z.B. ein Attribut¹

```
private double[][] nik;
```

¹Zur Schonung des Garbage Collectors.

Beachten Sie, dass bei ungleichen Knotenwerten nur ein $N_{i,1}(t)$ den Wert 1 hat, die anderen sind 0. Darauf aufbauend erzeugen Sie nach der Formel

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

die $N_{i,k}(t)$ mit $k > 1$.

Beachten Sie auch die $\frac{x}{0} = 0$ -Regel!

Da die ganze Index-Fummelei extrem fehleranfällig ist, rate ich Ihnen, die JUnit-Tests zur Orientierung zu nutzen. Die Methode `nik` sollte die beiden Tests `testNikPeriodicBSpline` und `testNikNonPeriodicBSpline` fehlerfrei ausführbar machen.

- c) Leiten Sie von `BSpline` die Klasse `PeriodicUniformBSpline` ab, die den Knotenvektor in Abhängigkeit von der Zahl der Punkte in `points` und von k passend initialisiert.

Damit müsste der Test `testKnotVectorPeriodicUniformBSpline` erfüllt werden.

- d) Leiten Sie von `BSpline` die Klasse `NonPeriodicUniformBSpline` ab, die den Knotenvektor in Abhängigkeit von der Zahl der Punkte in `points` und von k passend initialisiert.

Damit müsste der Test `testKnotVectorNonPeriodicUniformBSpline` erfüllt werden.

- e) Implementieren Sie jetzt in der Klasse `BSpline` die Methode

```
Point bSpline(double t)
```

die für einen Kurvenparameter t den zugehörigen Punkt

$$\vec{x}(t) = \sum_{i=0}^{n-k} N_{i,k}(t) P_i$$

auf der B-Spline liefert.

Hier sollten Sie eine Optimierung einbauen: Falls $t < t_i$ oder $t > t_{i+k}$ ist, ist $N_{i,k}(t) = 0$ und man spart sich die Berechnung.

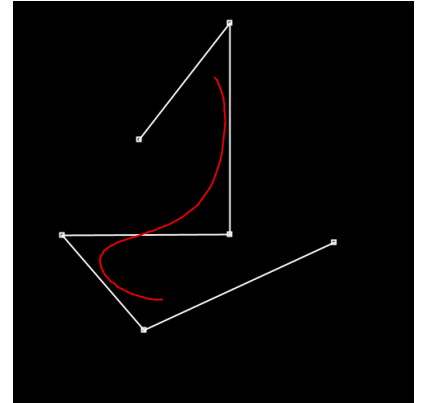
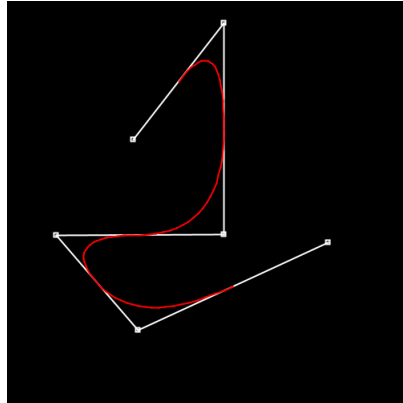
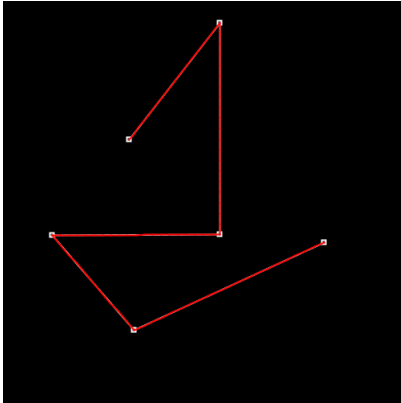
Damit müsste der Test `testPeriodicUniformBSpline` erfüllt werden.

- f) Implementieren Sie dann in der Klasse `BSpline` die Methode

```
void render(Graphics graphics)
```

die die B-Spline-Kurve zeichnet. Dazu lässt man den Kurvenparameter t mit der Schrittweite h in einer Schleife das Parameterintervall $[t_{k-1}; t_{n-k+1}]$ durchlaufen. Mit der Methode `bSpline` werden die Punkte $\vec{x}(t)$ der B-Spline-Kurve berechnet und mit der `drawLine`-Methode des `graphics`-Objekts verbunden.

Der letzte gültige Parameterwert t_{n-k+1} wird durch die Iteration normalerweise nicht exakt erreicht. Stellen Sie sicher, dass Sie $\vec{x}(t_{n-k+1})$ in jedem Fall berechnen und dass es mit dem Vorgängerpunkt korrekt verbunden ist.



Periodische uniforme B-Spline orientiert an dem Kontrollpolygon aus dem letzten Testfall für $k = 2, 3, 4$.

Fragen zur Übung (prüfungsrelevant)

- 1.) Die folgende Grafik zeigt das Kontrollpolygon und eine darauf basierende uniforme periodische B-Spline vom Grad 3.

Kennzeichnen Sie durch Verbindungspfeile, welche Kontrollpunkte übereinandergelegt werden müssen, damit die B-Spline einen geschlossenen Kurvenzug ergibt.

