Übungen - Bildgenierung

Übung 09.

Jose Jimenez

Angewandte Informatik
Bergische Universität Wuppertal

January 10, 2023

Table of Contents

Hermite-Kurven

Was wir in der letzten mal gelernt haben ...
Und Rahmen Program.

Hermite-Kurven

In der Vorlesung habt ihr die Hermite-Basis-Polynome kennengelernt

Hermite-Kurven

In der Vorlesung habt ihr die Hermite-Basis-Polynome kennengelernt

$$Q(t) = (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4$$

Hermite-Kurven

In der Vorlesung habt ihr die Hermite-Basis-Polynome kennengelernt

$$Q(t) = (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4$$

Der Parameter ist t. Schreiben wir $Q(t)$ als Polynom.

Hermite-Kurven

In der Vorlesung habt ihr die Hermite-Basis-Polynome kennengelernt.
Der Parameter ist t, dannn

$$Q(t) = (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4$$
$$Q(t) = 2t^3P_1 - 3t^2P_1 + P_1 - 2t^3P_4 + 3t^2P_4 + t^3R_1 - 2t^2R_1 + tR_1 + t^3R_4 - t^2R_4$$

Hermite-Kurven

In der Vorlesung habt ihr die Hermite-Basis-Polynome kennengelernt.
Der Parameter ist t, dannn

$$Q(t) = (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4$$
$$Q(t) = 2t^3 P_1 - 3t^2 P_1 + P_1 - 2t^3 P_4 + 3t^2 P_4 + t^3 R_1 - 2t^2 R_1 + tR_1 + t^3 R_4 - t^2 R_4$$

und...

Hermite-Kurven

In der Vorlesung habt ihr die Hermite-Basis-Polynome kennengelernt.
Der Parameter ist t, dannn

$$Q(t) = (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4$$
$$Q(t) = 2t^3P_1 - 3t^2P_1 + P_1 - 2t^3P_4 + 3t^2P_4 + t^3R_1 - 2t^2R_1 + tR_1 + t^3R_4 - t^2R_4$$
$$Q(t) = t^3(2P_1 - 2P_4 + R_1 + R_4) + t^2(-3P_1 + 3P_4 - 2R_1 - R_4) + t(R_1) + P_1$$

Hermite-Kurven

In der Vorlesung habt ihr die Hermite-Basis-Polynome kennengelernt.
Der Parameter ist t, dannn

$$Q(t) = (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4$$
$$Q(t) = 2t^3P_1 - 3t^2P_1 + P_1 - 2t^3P_4 + 3t^2P_4 + t^3R_1 - 2t^2R_1 + tR_1 + t^3R_4 - t^2R_4$$
$$Q(t) = t^3\underbrace{(2P_1 - 2P_4 + R_1 + R_4)}_{c_0} + t^2\underbrace{(-3P_1 + 3P_4 - 2R_1 - R_4)}_{c_1} + t\underbrace{(R_1)}_{c_2} + \underbrace{P_1}_{c_3}$$

In der Vorlesung habt ihr die Hermite-Basis-Polynome kennengelernt.
Der Parameter ist t, dannn

$$Q(t) = (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4$$
$$Q(t) = 2t^3P_1 - 3t^2P_1 + P_1 - 2t^3P_4 + 3t^2P_4 + t^3R_1 - 2t^2R_1 + tR_1 + t^3R_4 - t^2R_4$$
$$Q(t) = t^3\underbrace{(2P_1 - 2P_4 + R_1 + R_4)}_{c_0} + t^2\underbrace{(-3P_1 + 3P_4 - 2R_1 - R_4)}_{c_1} + t\underbrace{(R_1)}_{c_2} + \underbrace{P_1}_{c_3}$$
$$Q(t) = c_0t^3 + c_1t^2 + c_2t + c_3$$

Neun Multiplikationen...

Hermite-Kurven

In der Vorlesung habt ihr die Hermite-Basis-Polynome kennengelernt.
Der Parameter ist t, dannn

$$Q(t) = (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4$$

$$Q(t) = 2t^3P_1 - 3t^2P_1 + P_1 - 2t^3P_4 + 3t^2P_4 + t^3R_1 - 2t^2R_1 + tR_1 + t^3R_4 - t^2R_4$$

$$Q(t) = t^3\underbrace{(2P_1 - 2P_4 + R_1 + R_4)}_{c_0} + t^2\underbrace{(-3P_1 + 3P_4 - 2R_1 - R_4)}_{c_1} + t\underbrace{(R_1)}_{c_2} + \underbrace{P_1}_{c_3}$$

$$Q(t) = c_0t^3 + c_1t^2 + c_2t + c_3$$

$$Q(t) = (c_0t^2 + c_1t + c_2)t + c_3$$

Hermite-Kurven

In der Vorlesung habt ihr die Hermite-Basis-Polynome kennengelernt.
Der Parameter ist t, dannn

$$Q(t) = (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4$$
$$Q(t) = 2t^3P_1 - 3t^2P_1 + P_1 - 2t^3P_4 + 3t^2P_4 + t^3R_1 - 2t^2R_1 + tR_1 + t^3R_4 - t^2R_4$$
$$Q(t) = t^3\underbrace{(2P_1 - 2P_4 + R_1 + R_4)}_{c_0} + t^2\underbrace{(-3P_1 + 3P_4 - 2R_1 - R_4)}_{c_1} + t\underbrace{(R_1)}_{c_2} + \underbrace{P_1}_{c_3}$$
$$Q(t) = c_0t^3 + c_1t^2 + c_2t + c_3$$
$$Q(t) = (c_0t^2 + c_1t + c_2)t + c_3$$
$$Q(t) = ((c_0t + c_1)t + c_2)t + c_3$$

Drei Multiplikationen!

Hermite-Kurven

In der Vorlesung habt ihr die Hermite-Basis-Polynome kennengelernt.

$$Q(t) = t^3 \underbrace{(2P_1 - 2P_4 + R_1 + R_4)}_{c_0} + t^2 \underbrace{(-3P_1 + 3P_4 - 2R_1 - R_4)}_{c_1} + t \underbrace{(R_1)}_{c_2} + \underbrace{P_1}_{c_3}$$

$$Q(t) = ((c_0 t + c_1)t + c_2)t + c_3$$

c_i sind Vektoren, d.h., $c_i(x, y)$. Und wir zeichnen Segmente. Wie viele Segmenten brauchen wir?

Hermite-Kurven

In der Vorlesung habt ihr die Hermite-Basis-Polynome kennengelernt.

$$Q(t) = t^3 \underbrace{(2P_1 - 2P_4 + R_1 + R_4)}_{c_0} + t^2 \underbrace{(-3P_1 + 3P_4 - 2R_1 - R_4)}_{c_1} + t \underbrace{(R_1)}_{c_2} + \underbrace{P_1}_{c_3}$$

$$Q(t) = ((c_0 t + c_1)t + c_2)t + c_3$$

c_i sind Vektoren, d.h., $c_i(x, y)$. Und wir zeichnen Segmente. Wie viele Segmenten brauchen wir?

```
double cx[4], cy[4];
  DPoint2D anf, end;
  double t;
  double ninv = 1.0 / n;
```

Hermite-Kurven

In der Vorlesung habt ihr die Hermite-Basis-Polynome kennengelernt.

$$Q(t) = t^3 \underbrace{(2P_1 - 2P_4 + R_1 + R_4)}_{c_0} + t^2 \underbrace{(-3P_1 + 3P_4 - 2R_1 - R_4)}_{c_1} + t \underbrace{(R_1)}_{c_2} + \underbrace{P_1}_{c_3}$$

$$Q(t) = ((c_0 t + c_1)t + c_2)t + c_3$$

c_i sind Vektoren, d.h., $c_i(x, y)$. Und wir zeichnen Segmente. Wie viele Segmenten brauchen wir?

```
double cx[4], cy[4];
  DPoint2D anf, end;
  double t;
  double ninv = 1.0 / n;
```

Schauen wir uns den Code an! + Aufgabe 29

Bézier-Kurven und B-Splines

Für die **Hermite-Kurven**:

$$Q(t) = t^3 \underbrace{(2P_1 - 2P_4 + R_1 + R_4)}_{c_0} + t^2 \underbrace{(-3P_1 + 3P_4 - 2R_1 - R_4)}_{c_1} + t \underbrace{(R_1)}_{c_2} + \underbrace{P_1}_{c_3}$$

$$Q(t) = c_0 t^3 + c_1 t^2 + c_2 t + c_3 t$$

$$Q(t) = ((c_0 t + c_1)t + c_2)t + c_3$$

c_i sind Vektoren, d.h., $c_i(x, y)$.

Für die **Hermite-Kurven**:

$$Q(t) = c_0 t^3 + c_1 t^2 + c_2 t + c_3 t$$
$$Q(t) = ((c_0 t + c_1)t + c_2)t + c_3$$

c_i sind Vektoren, d.h., $c_i(x, y)$.

Bézier-Kurven und B-Splines sind auch Polynome, d.h. man kann sie so beschreiben

$$Q(t) = ((c_0 t + c_1)t + c_2)t + c_3$$

Dan, das code sieth gleich aus

$$Q(t) = ((c_0 t + c_1)t + c_2)t + c_3$$

```
for (i = 1; i <= n; ++i){
  t = ninv * i;
  anf = end;
  end.x = ((cx[0] * t + cx[1]) * t + cx[2]) * t + cx[3];
  end.y = ((cy[0] * t + cy[1]) * t + cy[2]) * t + cy[3];
  pic.drawLine(round(anf), round(end));
  }
```

Die Frage ist denn natürlich: was sind die Koeffizienten c_i?

$$Q(t) = ((c_0 t + c_1)t + c_2)t + c_3$$

Die Frage ist denn natürlich: was sind die Koeffizienten c_i?

$$Q(t) = ((c_0 t + c_1)t + c_2)t + c_3$$

Die antwort ist: Hermite-Form

- **Bézier-Kurven**

$$Q(t) = G_B M_B T$$

Bézier-Kurven und B-Splines

Die Frage ist denn natürlich: was sind die Koeffizienten c_i?

$$Q(t) = ((c_0 t + c_1)t + c_2)t + c_3$$

Die antwort ist: Hermite-Form

- **Bézier-Kurven**

$$Q(t) = G_B M_B T$$

- **B-Splines**

$$Q_i(t) = G_{BS_i} M_{BS} T_i$$

Bézier-Kurven und B-Splines

Die Frage ist denn natürlich: was sind die Koeffizienten c_i?

$$Q(t) = ((c_0 t + c_1)t + c_2)t + c_3$$

Die antwort ist: Hermite-Form

- **Bézier-Kurven**

$$Q(t) = G_B M_B T = C_{Be} T$$

- **B-Splines**

$$Q_i(t) = G_{BS_i} M_{BS} T_i = C_B T_i$$

Bézier-Kurven und B-Splines

Die Frage ist denn natürlich: was sind die Koeffizienten c_i?

$$Q(t) = ((c_0 t + c_1)t + c_2)t + c_3$$

Die antwort ist: Hermite-Form

- **Bézier-Kurven**
$$Q(t) = G_B M_B T = C_{Be} T$$
- **B-Splines**
$$Q_i(t) = G_{BS_i} M_{BS} T_i = C_B T_i$$

Hier:

$$G = (P_{i-3}, P_{i-2}, P_{i-1}, P_i) \qquad \text{und} \qquad T = (t^3, t^2, t, 1)^T \tag{1}$$

und M sind die Basismatrizen.

$$Q(t) = c_0 t^3 + c_1 t^2 + c_2 t + c_3 t$$

- **Bézier-Kurven**

$$Q(t) = G_B M_B T = C_{Be} T$$

- **B-Splines**

$$Q_i(t) = G_{BS_i} M_{BS} T_i = C_B T_i$$

Hier:

$$G = (P_{i-3}, P_{i-2}, P_{i-1}, P_i) \qquad \text{und} \qquad T = (t^3, t^2, t, 1)^T \tag{2}$$

und M sind die Basismatrizen.

$$M_B = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 4 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad M_{BS} = \frac{1}{6} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 0 & 4 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Ok, fertig, jetzt nur noch den Code implementieren. (Werte von C rechnen)

Bézier-Kurven und B-Splines

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T$
- **B-Splines** $Q_i(t) = G_{BS_i} M_{BS} T_i = C_B T_i$

Hier:

$$G = (P_{i-3}, P_{i-2}, P_{i-1}, P_i) \qquad \text{und} \qquad T = (t^3, t^2, t, 1)^T \tag{3}$$

und M sind die Basismatrizen.

$$M_B = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 4 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad M_{BS} = \frac{1}{6} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 0 & 4 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

```
for (k = 3; k <= m; k += 3) {
    // Die Kurve besteht dann aus m/3 einzelnen Kurvenstücken.
    }
for (k = 3; k <= m; ++k){
    /* "die Funktion die den zu p0, ..., pm gehörenden
        B-Spline malt."*/
    }
```

Bézier-Kurven und B-Splines

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T$
- **B-Splines** $Q_i(t) = G_{BS_i} M_{BS} T_i = C_B T_i$

Hier:

$$G = (P_{i-3}, P_{i-2}, P_{i-1}, P_i) \qquad \text{und} \qquad T = (t^3, t^2, t, 1)^T \tag{4}$$

und M sind die Basismatrizen.

$$M_B = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 4 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad M_{BS} = \frac{1}{6} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 0 & 4 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

```
for (k = 3; k <= m; k += 3) { //Bézier
    cx[0] = -p[k - 3].x + 3 * p[k - 2].x - 3 * p[k - 1].x + p[k].x;
    cx[1] = 3 * p[k - 3].x - 6 * p[k - 2].x + 3 * p[k - 1].x;
    cx[2] = -3 * p[k - 3].x + 3 * p[k - 2].x;
    cx[3] = p[k - 3].x;
    }
```

Und gleich für $cy[i]$

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T$
- **B-Splines** $Q_i(t) = G_{BS_i} M_{BS} T_i = C_B T_i$

Hier:

$$G = (P_{i-3}, P_{i-2}, P_{i-1}, P_i) \qquad \text{und} \qquad T = (t^3, t^2, t, 1)^T \tag{5}$$

und M sind die Basismatrizen.

$$M_B = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 4 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad M_{BS} = \frac{1}{6} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 0 & 4 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

```
for (k = 3; k <= m; ++k) {
  cx[0] = (-p[k - 3].x + 3 * p[k - 2].x - 3 * p[k - 1].x + p[k].x) / 6.0;
  cx[1] = (3 * p[k - 3].x - 6 * p[k - 2].x + 3 * p[k - 1].x) / 6.0;
  cx[2] = (-3 * p[k - 3].x + 3 * p[k - 1].x) / 6.0;
  cx[3] = (p[k - 3].x + 4 * p[k - 2].x + p[k - 1].x) / 6.0;
    }
```

Und gleich für $cy[i]$

Catmull-Rom-Splines (ganz ähnlich)

- **Catmull-Rom-Splines** $Q(t) = G_B M_{CR} T = C_{CR} T$

Hier:

$$G = (P_{i-3}, P_{i-2}, P_{i-1}, P_i) \qquad \text{und} \qquad T = (t^3, t^2, t, 1)^T \tag{6}$$

und M sind die Basismatrizen.

$$\mathrm{M}_{CR} = \frac{1}{2}\begin{pmatrix} -1 & 2 & -1 & 0 \\ 3 & -5 & 0 & 2 \\ -3 & 4 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{pmatrix}$$

```
for (k = 3; k <= m; ++k){
  cx[0] = (-p[k - 3].x + 3 * p[k - 2].x - 3 * p[k - 1].x + p[k].x) / 2.0;
  cx[1] = (2 * p[k - 3].x - 5 * p[k - 2].x + 4 * p[k - 1].x - p[k].x) / 2.
  cx[2] = (-p[k - 3].x + p[k - 1].x) / 2.0;
  cx[3] = p[k - 2].x;
  }
```

Und gleich für $cy[i]$.