Übungen - Bildgenierung

Übung 09.

Jose Jimenez

Angewandte Informatik
Bergische Universität Wuppertal

January 17, 2024

Table of Contents

Rotationskörper

Beim letzten mal und im Vorlesung haben wir gelernt:

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T$

Hier:

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

und M sind die Basismatrizen.

Rotationskörper

Beim letzten mal und im Vorlesung haben wir gelernt:

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T$

Hier:

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

und M sind die Basismatrizen.

$$M_B = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T \qquad T = (t^3, t^2, t, 1)^T$

Idee:

```
void berechneRotationsKoerper( vector<Kante>& vk, const
↪  vector<Vec3D> &p, int anzkurv = 5, int anzlinku = 20,
int anzkreis = 5, int anzlinkr = 20 ){
  int m = p.size() - 1;
  for (k = 3; k <= m; k += 3) {
    /*-------------- berechne vorweg die Matrizen C[d] ------*/


    /*------------------- male Kurvenstücke ----------------*/


    /*-------------------- male Kreise  --------------------*/

   }
```

Wie?: letzten mal haben wir es gemacht

Beim letzten mal und im Vorlesung haben wir gelernt:

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T$

Hier:

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

und M sind die Basismatrizen.

$$M_B = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

```
int m = p.size() - 1;
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;
    }
```

Es ist gleich für y, dann...

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T$

Hier:

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

und M sind die Basismatrizen.

$$M_B = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

```
int m = p.size() - 1;
for (k = 3; k <= m; k += 3) {
    for (d = 0; d < 2; d++){ //für x und y
      c[0][d] = -p[k - 3].el[d] + 3 * p[k - 2].el[d] - 3 * p[k -
↪   1].el[d]
        + p[k].el[d];
      c[1][d] = 3 * p[k - 3].el[d] - 6 * p[k - 2].el[d]
        + 3 * p[k - 1].el[d];
      c[2][d] = -3 * p[k - 3].el[d] + 3 * p[k - 2].el[d];
      c[3][d] = p[k - 3].el[d];
    }
```

Rotationskörper

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T \qquad T = (t^3, t^2, t, 1)^T$

Idee:

```
void berechneRotationsKoerper( vector<Kante>& vk, const
↪  vector<Vec3D> &p, int anzkurv = 5, int anzlinku = 20,
int anzkreis = 5, int anzlinkr = 20 ){
  int m = p.size() - 1;
  for (k = 3; k <= m; k += 3) {
    /*-------------- berechne vorweg die Matrizen C[d] ------*/
                            /*DONE*/


    /*-------------------- male Kurvenstücke ----------------*/


    /*--------------------- male Kreise  --------------------*/

   }
```

Kurvenstücke malen

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T \qquad T = (t^3, t^2, t, 1)^T$

Die erste stück der Kurve ist:

```
...
  for (k = 3; k <= m; k += 3){
 /*-------------- berechne vorweg die Matrizen C[d] ------*/
                               /*DONE*/

    /*-------------------- male Kurvenstücke ----------------*/
    anf = //Q(0) = C * T(0)
    end = //Q(Δt) = CΔT
    /*--------------------- male Kreise  --------------------*/

   }
```

Wie kann man $Q(t)$ beschreiben?

Kurvenstücke malen

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T \qquad T = (t^3, t^2, t, 1)^T$

Die erste stück der Kurve ist:

```
...

 for (k = 3; k <= m; k += 3){
/*------------- berechne vorweg die Matrizen C[d] ------*/
                               /*DONE*/

   /*-------------------- male Kurvenstücke ----------------*/
   anf = //Q(0) = C * T(0)
   end = //Q(Δt) = CΔT
   /*--------------------- male Kreise  --------------------*/

  }
```

Beim letzten Mal haben wir erfahren, dass das Produkt wie folgt hergestellt wird:

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

dann...

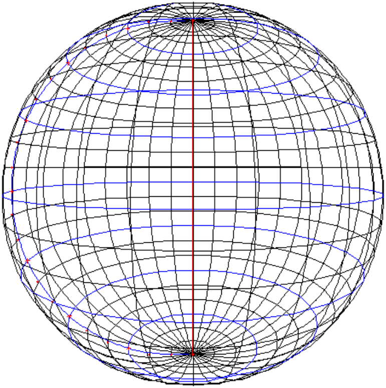
Kurvenstücke malen

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T \qquad T = (t^3, t^2, t, 1)^T$

```
...
  for (k = 3; k <= m; k += 3){
 /*-------------- berechne vorweg die Matrizen C[d] ------*/
                                  /*DONE*/


    /*-------------------- male Kurvenstücke ----------------*/
  anf = Vec4D(c[3][0], c[3][1], 0, 1);

  end = Vec4D(((c[0][0] * t + c[1][0]) * t + c[2][0]) * t + c[3][0],
↪  ((c[0][1] * t + c[1][1]) * t + c[2][1]) * t + c[3][1], 0, 1);

  vk.push_back(Kante(anf, end, BLUE));


   }
```

wir drehen unsere Kante von 0 auf 2π.

Kurvenstücke malen



Kurvenstücke malen

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T \qquad T = (t^3, t^2, t, 1)^T$

```
...
  double deltakurv = 2.0 * M_PI / anzkurv;
  for (k = 3; k <= m; k += 3){
 /*------------- berechne vorweg die Matrizen C[d] ------*/
                                   /*DONE*/

    /*------------------- male Kurvenstücke ----------------*/
    anf = Vec4D(c[3][0], c[3][1], 0, 1);
    end = Vec4D(((c[0][0] * t + c[1][0]) * t + c[2][0]) * t +
↪   c[3][0],
                  ((c[0][1] * t + c[1][1]) * t + c[2][1]) * t +
↪   c[3][1],
                  0, 1);
    vk.push_back(Kante(anf, end, BLUE));


   }
```

wir drehen unsere Kante von 0 auf 2π.

Kurvenstücke malen

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T \qquad T = (t^3, t^2, t, 1)^T$

```
...
  double deltakurv = 2.0 * M_PI / anzkurv;
  for (k = 3; k <= m; k += 3){
 /*-------------- berechne vorweg die Matrizen C[d] ------*/
                                /*DONE*/

    /*-------------------- male Kurvenstücke ----------------*/
    anf = //Q(0) = C * T(0)
    end = //Q(Δt) = CΔT
    vk.push_back(Kante(anf, end, BLUE));

    for (phi = deltakurv; phi <= 2*M_PI, phi += deltakurv){
        anf2 = ?
        end2 = ?
      }
   }
```

Kurvenstücke malen

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T \qquad T = (t^3, t^2, t, 1)^T$

```
...
  double deltakurv = 2.0 * M_PI / anzkurv;
  for (k = 3; k <= m; k += 3){
 /*-------------- berechne vorweg die Matrizen C[d] ------*/
                                   /*DONE*/
    /*-------------------- male Kurvenstücke ----------------*/
    anf = //Q(0) = C * T(0)
    end = //Q(Δt) = CΔT
    vk.push_back(Kante(anf, end, BLUE));

    for (phi = deltakurv; phi <= 2*M_PI, phi += deltakurv){
        anf2 = Vec4D(anf.el[0], cos(phi) * anf.el[1],
                      sin(phi) * anf.el[1], 1);
        end2 = Vec4D(end.el[0], cos(phi) * end.el[1],
                      sin(phi) * end.el[1], 1);
        vk.push_back(Kante(anf2, end2, BLACK));
      }
   }
```

Das ist für die erste geradenstück, wir müssen dasselbe für die anderen **anzlinku** geradenstücke tun.

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T \qquad T = (t^3, t^2, t, 1)^T$

Für die andere geradenstücke:

```
...
  for (k = 3; k <= m; k += 3){
 /*-------------- berechne vorweg die Matrizen C[d] ------*/
                                   /*DONE*/
    /*-------------------- male Kurvenstücke ----------------*/
    //-------------- Alle Kante erzeugen-----------------------
    end = C*T(0);
    for (i = 1, t = deltalinku; i <= anzlinku; i++, t+= deltalinku){
      anf = end;
      end = C*T(t);
      vk.push_back(Kante(anf, end, BLUE));
    //--------------for(0 ... 2π) {Kante drehen}
```

male Kurvenstücke: Completed!

```
...
  double deltakurv = 2.0 * M_PI / anzkurv;
  for (k = 3; k <= m; k += 3){
 /*-------------- berechne vorweg die Matrizen C[d] ------*/
                                   /*DONE*/
   // male Kurvenstuecke
    end = Vec4D(c[3][0], c[3][1], 0, 1);
    for (i = 1, t = deltalinku; i <= anzlinku; i++, t+= deltalinku){
      anf = end;
      end = //C*T(t);
      vk.push_back(Kante(anf, end, BLUE));
      for (j = 1, phi = deltakurv; j < anzkurv; j++, phi +=
↪   deltakurv) //drehen
      {
        anf2 = Vec4D(anf.el[0], cos(phi) * anf.el[1],
                     sin(phi) * anf.el[1], 1);
        end2 = Vec4D(end.el[0], cos(phi) * end.el[1],
                     sin(phi) * end.el[1], 1);
        vk.push_back(Kante(anf2, end2, BLACK));
      }
    }
   }
```

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T \qquad T = (t^3, t^2, t, 1)^T$

Was wir bisher gemacht haben:

```
...
  for (k = 3; k <= m; k += 3){
 /*-------------- berechne vorweg die Matrizen C[d] ------*/
                                /*DONE*/

    /*-------------------- male Kurvenstücke -------------------
        -------------- 1. Alle Kante erzeugen ------------------
        -------------- 2. for(0 ... 2π) --------------------*/
                             /*DONE*/



    /*-------------------- male Kreise  --------------------*/
  }
```

Für die Kreise, ist das Konzept dasselbe. d.h:

Rotationskörper

- **Bézier-Kurven** $Q(t) = G_B M_B T = C_{Be} T \qquad T = (t^3, t^2, t, 1)^T$

```
...
  for (k = 3; k <= m; k += 3){
 /*-------------- berechne vorweg die Matrizen C[d] ------*/
                                   /*DONE*/


    /*-------------------- male Kurvenstücke --------------------
        -------------- 1. Alle Kante erzeugen ------------------
        -------------- 2. for(0 ... 2π) --------------------*/
                                /*DONE*/

    /*--------------------- male Kreise  ----------------------
       -------------- 1. Alle Kante erzeugen ------------------
       -------------- 2. for(0 ... 2π) --------------------*/
```

$$Q(t) = G_B M_B T = C_{Be} T \qquad Q(t) = ((c_0 t + c_1)t + c_2)t + c_3$$

```
...
  double deltakreis = 1.0 / (anzkreis - 1);
  for (k = 3; k <= m; k += 3){
   /*-------------- berechne vorweg die Matrizen C[d] ------*/
                              /*DONE*/

    /*-------------------- male Kurvenstücke -------------------*/
                           /*DONE*/

    /*-------------------- male Kreise  ----------------------
       -------------- 1. Alle Kante erzeugen ------------------*/
    end = C*T(0);  //Q(t) = ((c0 t + c1)t + c2)t + c3
    for (i = 0, t = 0; i < anzkreis; i++, t+= deltakreis)
    {
      anf = C*T(t)

     /* ------------- 2. for(0 ... 2π) --------------------*/
```

$$Q(t) = G_B M_B T = C_{Be} T \qquad Q(t) = ((c_0 t + c_1)t + c_2)t + c_3$$

```
...
  double deltakreis = 1.0 / (anzkreis - 1);
  for (k = 3; k <= m; k += 3){
   ...
      -------------- 1. Alle Kante erzeugen ------------------*/
    end = C*T(0);   //Q(t) = ((c0 t + c1)t + c2)t + c3
    for (i = 0, t = 0; i < anzkreis; i++, t+= deltakreis)
    {
      anf = C*T(t)

     /* -------------- 2. for(0 ... 2π) ---------------------*/
       end2 = anf;
       for (phi = deltalinkr; phi <= 2*M_PI; phi += deltalinkr){
         anf2 = end2;
         end2 = Vec4D(anf.el[0], cos(phi) * anf.el[1],
                      sin(phi) * anf.el[1], 1);
         vk.push_back(Kante(anf2, end2, BLACK));
        }
    }
```

Rotationskörper

- $Q(t) = G_B M_B T = C_{Be} T \qquad Q(t) = ((c_0 t + c_1)t + c_2)t + c_3$

Am Ende sieht unser Code so aus::

```
...
  int m = p.size() - 1;
  for (k = 3; k <= m; k += 3){
    /*-------------berechne vorweg die Matrizen C[i]------------*/

    /*-------------------- male Kurvenstücke -------------------
        -------------- 1. Alle Kante erzeugen ------------------
        -------------- 2. for(0 ... 2π) + pushBack --------*/

    /*--------------------- male Kreise  ----------------------
       -------------- 1. Alle Kante erzeugen ------------------
      -------------- 2. for(0 ... 2π) + pushBack  --------*/
}
```