Übungen - Bildgenierung

Übung 09.

Jose Jimenez

Angewandte Informatik
Bergische Universität Wuppertal

January 17, 2024

Table of Contents

Bézier-Flächen

Ich gebe das Material aus der Vorlesung ab. Lesen wir es.

Bézier-Flächen

Aufgabe: Zu jeweils 16 Punkten $p[i][j], ..., p[i+3][j+3]$ gehört ein Flächenstück bestehend aus **anzkurv** Kurvenstücken für jede der beiden Richtungen, wobei jedes Kurvenstück durch **anzlin** Linien approximiert wird.

Ihr habt in der Vorlesung gelernt:

Parametrisierte bikubische Fläche

$$Q(s,t) = T^T \cdot M^T \cdot \tilde{G} \cdot M \cdot S \qquad T = \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}, \qquad S = \begin{bmatrix} s^3 \\ s^2 \\ s \\ 1 \end{bmatrix}$$

oder **koordinatenweise**:

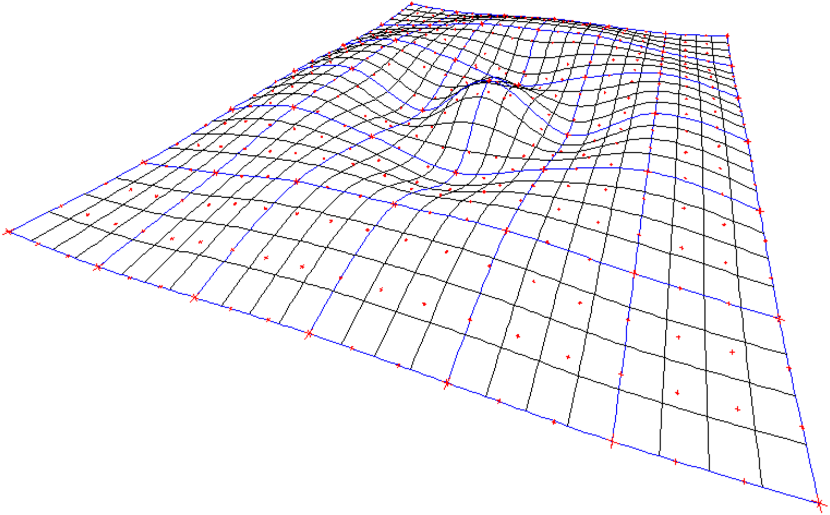
$$x(s,t) = T^T \cdot \underbrace{M^T \cdot \tilde{G}_x \cdot M}_{C_1} \cdot S$$

$$y(s,t) = T^T \cdot \underbrace{M^T \cdot \tilde{G}_y \cdot M}_{C_2} \cdot S$$

$$z(s,t) = T^T \cdot \underbrace{M^T \cdot \tilde{G}_z \cdot M}_{C_3} \cdot S$$

Bézier-Flächen

$$z(s,t) = T^T \cdot C_3 \cdot S \qquad s \in [?,?] \quad \textbf{und} \;\; t \in [?,?]$$



Bézier-Flächen

Zu jeweils 16 Punkten $p[i][j], \ldots, p[i+3][j+3]$ gehört ein Flächenstück bestehend aus **anzkurv** Kurvenstücken für jede der beiden Richtungen, wobei jedes Kurvenstück durch **anzlin** Linien approximiert wird.

Wir implementieren:

```
void berechneBezierFlaeche( vector<Kante>&  vk,
                            vector<vector<Vec3D> > &p,
                            int anzkurv = 5, int anzlin = 20 ){
  /* berechnet ein Netz aus Bezier-Flaechen, alle
  Kantenstuecke werden  dem Vektor vk hinzugefügt*/
}
```

Die Idee:

```
void berechneBezierFlaeche( vector<Kante>&  vk,
                            vector<vector<Vec3D> > &p,
                            int anzkurv = 5, int anzlin = 20 ){
  ...
  // Schleifen über die Einzel-Flaechen
  for (k = 3; k <= m; k += 3)  // m=?
    for (l = 3; l <= n; l += 3)// n=?
      /*-------------- berechne vorweg die Matrizen C[d] ------*/

      /*--------------  Linien für jeweils festes s -------------*/
      /*--------------  Linien für jeweils festes t -------------*/
```

Bézier-Flächen

Zu jeweils 16 Punkten $p[i][j], ..., p[i+3][j+3]$ gehört ein Flächenstück bestehend aus **anzkurv** Kurvenstücken für jede der beiden Richtungen, wobei jedes Kurvenstück durch **anzlin** Linien approximiert wird.

Wir brauchen...

- einen Zähler für jede Richtung (von 0 bis ... ??? Denkt über P nach)

Bézier-Flächen

Zu jeweils 16 Punkten $p[i][j], \ldots, p[i+3][j+3]$ gehört ein Flächenstück bestehend aus **anzkurv** Kurvenstücken für jede der beiden Richtungen, wobei jedes Kurvenstück durch **anzlin** Linien approximiert wird.

Wir brauchen...

- einen Zähler für jede Richtung (von 0 bis ... ??? Denkt über P nach)
- zwei Deltas für die Kurven und Linien

Bézier-Flächen

Zu jeweils 16 Punkten $p[i][j], \ldots, p[i+3][j+3]$ gehört ein Flächenstück bestehend aus **anzkurv** Kurvenstücken für jede der beiden Richtungen, wobei jedes Kurvenstück durch **anzlin** Linien approximiert wird.

Wir brauchen...

- einen Zähler für jede Richtung (von 0 bis ... ??? Denkt über P nach)
- zwei Deltas für die Kurven und Linien
- –Bezier-Basismatrix–

Bézier-Flächen

Zu jeweils 16 Punkten $p[i][j], ..., p[i+3][j+3]$ gehört ein Flächenstück bestehend aus **anzkurv** Kurvenstücken für jede der beiden Richtungen, wobei jedes Kurvenstück durch **anzlin** Linien approximiert wird.

Wir brauchen...

- einen Zähler für jede Richtung (von 0 bis ... ??? Denkt über P nach)
- zwei Deltas für die Kurven und Linien
- –Bezier-Basismatrix–
- Geometriematrix

Bézier-Flächen

Zu jeweils 16 Punkten $p[i][j], ..., p[i+3][j+3]$ gehört ein Flächenstück bestehend aus **anzkurv** Kurvenstücken für jede der beiden Richtungen, wobei jedes Kurvenstück durch **anzlin** Linien approximiert wird.

Wir brauchen...

- einen Zähler für jede Richtung (von 0 bis ... ??? Denkt über P nach)
- zwei Deltas für die Kurven und Linien
- –Bezier-Basismatrix–
- Geometriematrix
- C_1, C_2, C_3. (Was sind die nochmal?)

Bézier-Flächen

Zu jeweils 16 Punkten $p[i][j], ..., p[i+3][j+3]$ gehört ein Flächenstück bestehend aus **anzkurv** Kurvenstücken für jede der beiden Richtungen, wobei jedes Kurvenstück durch **anzlin** Linien approximiert wird.

Wir brauchen...

- einen Zähler für jede Richtung (von 0 bis ... ??? Denkt über P nach)
- zwei Deltas für die Kurven und Linien
- –Bezier-Basismatrix–
- Geometriematrix
- C_1, C_2, C_3. (Was sind die nochmal?)

$$x(s,t) = T^T \cdot \underbrace{M^T \cdot \tilde{G}_x \cdot M}_{C_1} \cdot S \qquad y(s,t) = T^T \cdot \underbrace{M^T \cdot \tilde{G}_y \cdot M}_{C_2} \cdot S$$

$$z(s,t) = T^T \cdot \underbrace{M^T \cdot \tilde{G}_z \cdot M}_{C_3} \cdot S$$

Zu jeweils 16 Punkten $p[i][j], \ldots, p[i+3][j+3]$ gehört ein Flächenstück bestehend aus **anzkurv** Kurvenstücken für jede der beiden Richtungen, wobei jedes Kurvenstück durch **anzlin** Linien approximiert wird.

Wir brauchen...

- einen Zähler für jede Richtung
- zwei Deltas für die Kurven und Linien
- –Bezier-Basismatrix–
- Geometriematrix
- C_1, C_2, C_3.

```
void berechneBezierFlaeche( vector<Kante>&  vk,
                            vector<vector<Vec3D> > &p,
                            int anzkurv = 5, int anzlin = 20 ){
   /* berechnet ein Netz aus Bezier-Flaechen, alle
  Kantenstuecke werden  dem Vektor vk hinzugefügt*/
  int m = p.size() - 1;
  int n = p[0].size() - 1;
  double deltakurv = 1.0 / (anzkurv - 1);
  double deltalin = 1.0 / anzlin;
  Matrix4x4 C[3];
```

Bézier-Flächen

Wir brauchen...

- ...
- ~~Bezier-Basismatrix~~
- Geometriematrix
- C_1, C_2, C_3.
- Schleifen über die Einzel-Flächen

```
void berechneBezierFlaeche( vector<Kante>&  vk,
                            vector<vector<Vec3D> > &p,
                            int anzkurv = 5, int anzlin = 20 ){
   /* berechnet ein Netz aus Bezier-Flaechen, alle
  Kantenstuecke werden  dem Vektor vk hinzugefügt*/
  int m = p.size() - 1;
  int n = p[0].size() - 1;
  double deltakurv = 1.0 / (anzkurv - 1);
  double deltalin = 1.0 / anzlin;

  double mbel[4][4] = { { -1,  3, -3, 1 },
                        {  3, -6,  3, 0 },
                        { -3,  3,  0, 0 },
                        {  1,  0,  0, 0 } };
  Matrix4x4 MB(mbel);
```

Die Idee:

```
void berechneBezierFlaeche( vector<Kante>&  vk,
                            vector<vector<Vec3D> > &p,
                            int anzkurv = 5, int anzlin = 20 ){
  int m = p.size() - 1,          n = p[0].size() - 1;

  double deltakurv = 1.0 / (anzkurv - 1),   deltalin = 1.0 / anzlin;

  double mbel[4][4] = {...};   Matrix4x4 MB(mbel);  Matrix4x4 C[3];

  // Schleifen über die Einzel-Flaechen
  for (k = 3; k <= m; k += 3)
    for (l = 3; l <= n; l += 3)
      /*-------------- berechne vorweg die Matrizen C[d] ------*/

      /*--------------  Linien für jeweils festes s -------------*/
      /*--------------  Linien für jeweils festes t -------------*/
```

Fragen?

Die Idee:

```
void berechneBezierFlaeche( vector<Kante>&  vk,
                            vector<vector<Vec3D> > &p,
                            int anzkurv = 5, int anzlin = 20 ){
  int m = p.size() - 1,         n = p[0].size() - 1;

  double deltakurv = 1.0 / (anzkurv - 1),   deltalin = 1.0 / anzlin;

  double mbel[4][4] = {...};   Matrix4x4 MB(mbel);  Matrix4x4 C[3];

  // Schleifen über die Einzel-Flaechen
  for (k = 3; k <= m; k += 3)
    for (l = 3; l <= n; l += 3)
      /*-------------- berechne vorweg die Matrizen C[d] ------*/

      /*--------------  Linien für jeweils festes s -------------*/
      /*--------------  Linien für jeweils festes t -------------*/
```

Fragen? Wir implementieren den Fehlenden Code.

Bézier-Flächen

$$z(s,t) = T^T \cdot C_3 \cdot S \qquad C_3 = M^T \cdot \tilde{G}_z \cdot M$$

Berechne vorweg die Matrizen C[d].

```
...

  for (k = 3; k <= m; k += 3) // Schleifen über die Einzel-Flaechen
    for (l = 3; l <= n; l += 3){

  //-------------- berechne vorweg die Matrizen C[d]--------------
      for (d = 0; d < 3; d++){    //für jede Matrix C haben wir...
        for (i = 0; i < 4; i++)    // 16 KontrolPunkte im G
          for (j = 0; j < 4; j++)
            G.el[i][j] = p[k - 3 + i][l - 3 + j].el[d];
        C[d] = MB * G * MB; // MB^T = MB
      }
  ...
```

Warum haveb wir 3 Matrizen C?

Bézier-Flächen

$$z(s,t) = T^T \cdot C_3 \cdot S \qquad C_3 = M^T \cdot \tilde{G}_z \cdot M$$

```
void berechneBezierFlaeche( vector<Kante>&  vk,
                            vector<vector<Vec3D> > &p,
                            int anzkurv = 5, int anzlin = 20 ){

  int m = p.size() - 1,          n = p[0].size() - 1;

  double deltakurv = 1.0 / (anzkurv - 1),   deltalin = 1.0 / anzlin;

  double mbel[4][4] = {...};   Matrix4x4 MB(mbel);  Matrix4x4 C[3];

  for (k = 3; k <= m; k += 3) // Schleifen über die Einzel-Flaechen
    for (l = 3; l <= n; l += 3){
      /*-------------- berechne vorweg die Matrizen C[d] ------*/
                             /*DONE*/


      /*--------------  Linien für jeweils festes s -------------*/
      /*--------------  Linien für jeweils festes t -------------*/
```

Bézier-Flächen

$$z(s,t) = T^T \cdot C_3 \cdot S \qquad s \in [0,1) \quad \textbf{und} \ t \in [0,1).$$

s geht von 0 bis 1 mit schritten von *deltakruv*.

Bézier-Flächen

$$z(s,t) = T^T \cdot C_3 \cdot S \qquad s \in [0,1) \quad \textbf{und} \;\; t \in [0,1).$$

s geht von 0 bis 1 mit schritten von *deltakruv*.

```
...
  for (k = 3; k <= m; k += 3) // Schleifen über die Einzel-Flaechen
    for (l = 3; l <= n; l += 3){
      /*--------------  Linien für jeweils festes s -------------*/
      for (s = 0; s < 1; s += deltakurv)
      {
```

Bézier-Flächen

$$z(s,t) = T^T \cdot C_3 \cdot S \qquad s \in [0,1) \quad \textbf{und} \;\; t \in [0,1).$$

Für jede feste Kurve s haben wir eine Schleife über die Linien:

```
...
  for (k = 3; k <= m; k += 3) // Schleifen über die Einzel-Flaechen
    for (l = 3; l <= n; l += 3){
      /*--------------  Linien für jeweils festes s -------------*/
      for (s = 0; s < 1; s += deltakurv)
      {
        for (t = deltalin; t<=1; t += deltalin)
        {
...
```

"Mit dem Befehl **vk.push_back(Kante(anf, end, BLACK));** können Sie dem Kanten-Vektor die einzelnen Kantenstücke hinzufügen, wobei anf und end vom Typ Vec4D sind."

Bézier-Flächen

$$z(s,t) = T^T \cdot C_3 \cdot S \qquad s \in [0,1) \quad \textbf{und} \;\; t \in [0,1).$$

Für jede feste Kurve s haben wir eine Schleife über die Linien:

```
...
  for (k = 3; k <= m; k += 3) // Schleifen über die Einzel-Flaechen
    for (l = 3; l <= n; l += 3){
      /*--------------  Linien für jeweils festes s -------------*/
      for (s = 0; s < 1; s += deltakurv)
      {
        for (t = deltalin; t<=1; t += deltalin)
        {
...
```

"Mit dem Befehl **vk.push_back(Kante(anf, end, BLACK));** können Sie dem Kanten-Vektor die einzelnen Kantenstücke hinzufügen, wobei anf und end vom Typ Vec4D sind." Die erste Kante für festes s ist etwas wie...

Bézier-Flächen

$$z(s,t) = T^T \cdot C_3 \cdot S \qquad s \in [0,1) \quad \textbf{und} \;\; t \in [0,1).$$

Für jede feste Kurve s haben wir eine Schleife über die Linien:

```
...
  for (k = 3; k <= m; k += 3) // Schleifen über die Einzel-Flaechen
    for (l = 3; l <= n; l += 3){
      /*--------------  Linien für jeweils festes s -------------*/
      for (s = 0; s < 1; s += deltakurv)
      {
        for (t = deltalin; t<=1; t += deltalin)
        {
...
```

"Mit dem Befehl **vk.push_back(Kante(anf, end, BLACK));** können Sie dem Kanten-Vektor die einzelnen Kantenstücke hinzufügen, wobei anf und end vom Typ Vec4D sind." Die erste Kante für festes s ist etwas wie...

```
anf = mult(s, C, 0);
   end = mult(s, C, deltalin);  //deltalin = t
   vk.push_back(Kante(anf, end, BLACK));
```

Bézier-Flächen

$$z(s,t) = T^T \cdot C_3 \cdot S \qquad s \in [0,1) und t \in [0,1)$$

Und so weiter...

```
//------------------1--------------------
   anf = mult(s, C, 0);
   end = mult(s, C, deltalin);
   vk.push_back(Kante(anf, end, BLACK));
//------------------2--------------------
   anf = end;
   end = mult(s, C, 2*deltalin);
   vk.push_back(Kante(anf, end, BLACK));
//------------------3--------------------
   anf = end;
   end = mult(s, C, 3*deltalin);
   vk.push_back(Kante(anf, end, BLACK));
```

dann in unserer for-Schleife...

Bézier-Flächen

$$z(s,t) = T^T \cdot C_3 \cdot S \qquad s \in [0,1) \quad \textbf{und} \;\; t \in [0,1).$$

```
/*--------------  Linien für jeweils festes s -------------*/
       for (s = 0; s < 1; s += deltakurv)
      {
        end = mult(s, C, 0);
        for (t = deltalin; t<=1; t += deltalin)
        {
          anf = end;
          end = mult(s, C, t);
          vk.push_back(Kante(anf, end, BLACK));
        }
      }
```

Bis hier: $z(s,t) = T^T \cdot C_3 \cdot S \qquad s \in [0,1)$ **und** $t \in [0,1)$.

```
...
  for (k = 3; k <= m; k += 3) // Schleifen über die Einzel-Flaechen
    for (l = 3; l <= n; l += 3){
     /*-------------- berechne vorweg die Matrizen C[d] ------*/
                                /*DONE*/
      /*--------------  Linien für jeweils festes s -------------*/
       for (s = 0; s < 1; s += deltakurv)
      {
        end = mult(s, C, 0);    //???
        for (t = deltalin; t<=1; t += deltalin)
        {
          anf = end;
          end = mult(s, C, t); //???
          vk.push_back(Kante(anf, end, BLACK));
        }
      }
```

Bis hier:

$$z(s,t) = T^T \cdot C_3 \cdot S \qquad s \in [0,1) \quad \textbf{und} \ t \in [0,1).$$

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

      /*--------------  Linien für jeweils festes s -------------*/
                                /*DONE*/

      /*--------------  Linien für jeweils festes t -------------*/
                            /*Analog zu s*/
    }
```

Und... die Function mult?

Bézier-Flächen

$$z(s,t) = T^T \cdot C_3 \cdot S$$

Und... die Function mult?

```
Vec4D mult(double s, Matrix4x4 C[3], double t)
{
  // berechnet den Punkt fuer die Parameter s und t

  Vec4D ss(s * s * s, s * s, s , 1);
  Vec4D tt(t * t * t, t * t, t , 1);

  return Vec4D( skalarprod(ss, (C[0] * tt)),
                skalarprod(ss, (C[1] * tt)),
                skalarprod(ss, (C[2] * tt)),
                1 );
}
```

$$z(s,t) = T^T \cdot C_3 \cdot S \qquad s \in [0,1) \quad \textbf{und} \ t \in [0,1).$$

```
// Schleifen ueber die Einzel-Flaechen
  for (k = 3; k <= m; k += 3)
    for (l = 3; l <= n; l += 3){
      // berechne vorweg die Matrizen C[i]
      for (d = 0; d < 3; d++){
      for (i = 0; i < 4; i++)
          for (j = 0; j < 4; j++)
            G.el[i][j] = p[k - 3 + i][l - 3 + j].el[d];
        C[d] = MB * G * MB; // MB^T = MB
      }
      // Linien fuer jeweils festes s  (Link nach Rechts)
      for (s = 0; s < 1; s += deltakurv){
        end = mult(s, C, 0);
        for (t = deltalin; t<=1; t += deltalin){
          anf = end;
          end = mult(s, C, t);

          vk.push_back(Kante(anf, end, BLACK));
        }
      }
      // Linien fuer jeweils festes t
```