Übungen - Bildgenierung

Übung 09.

Jose Jimenez

Angewandte Informatik
Bergische Universität Wuppertal

January 17, 2024

Table of Contents

Bézier-Flächen mit OpenGL

1. Legen Sie mittels **glMap2f** und des Target-Parameters **GL_MAP2_VERTEX_3**, welcher dabei anzugeben ist, die Kontrollpunkte des aktuellen Flächenstücks fest.

Bézier-Flächen mit OpenGL

1. Legen Sie mittels **glMap2f** und des Target-Parameters **GL_MAP2_VERTEX_3**, welcher dabei anzugeben ist, die Kontrollpunkte des aktuellen Flächenstücks fest.

"The **glMap2f** function defines a two-dimensional evaluator. It generates some values depending of the **target**."

Bézier-Flächen mit OpenGL

1. Legen Sie mittels **glMap2f** und des Target-Parameters **GL_MAP2_VERTEX_3**, welcher dabei anzugeben ist, die Kontrollpunkte des aktuellen Flächenstücks fest.

"The **glMap2f** function defines a two-dimensional evaluator. It generates some values depending of the **target**."

Das Target das wir brauchen ist:

- **GL_MAP2_VERTEX_3**: Each control point is three floating-point values representing x, y, and z.

Bézier-Flächen mit OpenGL

Paremeter für glMAP2f

```
glMap2f( target,   // GL_MAP2_VERTEX_3
         t1, t2,   // 0, 1
         tstride,  //floats/doubles
         torder,   // 4
         s1, s2,   // 0, 1
         sstride,  // 3*4
         sorder,   // 4
         points )  // unsere Punkte
```

1. **tstride** The number of floats or doubles between the beginning of control point $R_{i,j}$ and the beginning of control point $R_{i+1,j}$.
2. **torder:** The dimension of the control point array in the t-axis. Must be positive.

Bézier-Flächen mit OpenGL

Paremeter für glMAP2f

```
glMap2f( target,   // GL_MAP2_VERTEX_3
         t1, t2,   // 0, 1
         tstride,  //floats/doubles
         torder,   // 4
         s1, s2,   // 0, 1
         sstride,  // 3*4
         sorder,   // 4
         points )  // unsere Punkte
```

1. **tstride** The number of floats or doubles between the beginning of control point $R_{i,j}$ and the beginning of control point $R_{i+1,j}$.
2. **torder:** The dimension of the control point array in the t-axis. Must be positive.

sstride The number of floats or doubles between the beginning of control point $R_{i,j}$ and the beginning of control point $R_{i,j+1}$.

Bézier-Flächen mit OpenGL

1. Legen Sie mittels **glMap2f** und des Target-Parameters **GL_MAP2_VERTEX_3**, welcher dabei anzugeben ist, die Kontrollpunkte des aktuellen Flächenstücks fest.
2. Aktivieren Sie die Kontrollpunkte mittels **glEnable**.

Bézier-Flächen mit OpenGL

1. Legen Sie mittels **glMap2f** und des Target-Parameters **GL_MAP2_VERTEX_3**, welcher dabei anzugeben ist, die Kontrollpunkte des aktuellen Flächenstücks fest.
2. Aktivieren Sie die Kontrollpunkte mittels **glEnable**.

```
glMap2f( target,   // GL_MAP2_VERTEX_3
         t1, t2,   // 0, 1
         tstride,  // floats/doubles = 3
         torder,   // 4
         s1, s2,   // 0, 1
         sstride,  // 3*4
         sorder,   // 4
         points )  // unsere Punkte

  glEnable( GL_MAP2_VERTEX_3 );
```

Bézier-Flächen mit OpenGL

1. Legen Sie mittels **glMap2f** und des Target-Parameters **GL_MAP2_VERTEX_3**, welcher dabei anzugeben ist, die Kontrollpunkte des aktuellen Flächenstücks fest.
2. Aktivieren Sie die Kontrollpunkte mittels **glEnable**.
3. Erzeugen Sie unter Verwendung des Befehls **glMapGrid2f** ein Mesh, das aus nMeshSize Partitionen in jeder Richtung besteht.

Bézier-Flächen mit OpenGL

"Defines a one-dimensional mesh."

- Paremeter für glMapGrid2f

```
void zeichneBezierFlaeche( const vector<vector<Vec3D> >& p,
                           int nMeshSize = 10 )
{
...
    glMapGrid2f( nMeshSize,
                 t1, t2,      // 0, 1
                 nMeshSize,
                 s1, s2 );    // 0, 1
}
```

Bézier-Flächen mit OpenGL

1. Legen Sie mittels **glMap2f** und des Target-Parameters **GL_MAP2_VERTEX_3**, welcher dabei anzugeben ist, die Kontrollpunkte des aktuellen Flächenstücks fest.
2. Aktivieren Sie die Kontrollpunkte mittels **glEnable**.
3. Erzeugen Sie unter Verwendung des Befehls **glMapGrid2f** ein Mesh, das aus nMeshSize Partitionen in jeder Richtung besteht.
4. Zeichnen Sie die Bézier-Fläche mit **glEvalMesh2**.

Bézier-Flächen mit OpenGL

"Computes a two-dimensional grid of points or lines."

- Paremeter für glEvalMesh2

```
glEvalMesh2( mode,      // GL_POINT oder GL_LINE
             i1, i2,    /* "The first integer value for grid domain
                           variable i."*/
             j1, j2;)   /* "The first integer value for grid domain
                           variable j."*/
```

d.h:

```
glEvalMesh2( GL_LINE
                  0, nMeshSize
                  0, nMeshSize);
```

Bézier-Flächen mit OpenGL

Bisher:

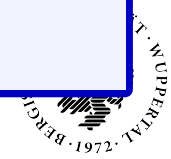
1. Legen Sie die Kontrollpunkte des aktuellen **Flächenstücks** fest.
2. Aktivieren Sie die Kontrollpunkte mittels **glEnable**.
3. Erzeugen Sie unter Verwendung des Befehls **glMapGrid2f** ein Mesh, das aus nMeshSize Partitionen in jeder Richtung besteht.
4. Zeichnen Sie die Bézier-Fläche mit **glEvalMesh2**.

Bézier-Flächen mit OpenGL

Bisher:

```
void zeichneBezierFlaeche( const vector<vector<Vec3D> >& p,
                           int nMeshSize = 10 ){

/*glMap2f( target, t1, t2, tstride, torder,
            s1, s2, sstride, sorder, points )*/
  glMap2f( GL_MAP2_VERTEX_3, 0.0, 1.0, 3, 4,
           0.0, 1.0, 3*4, 4, /*points*/ );

  glEnable( GL_MAP2_VERTEX_3 );

//glMapGrid2f( tn, t1, t2, sn, s1, s2);
  glMapGrid2f( nMeshSize, 0.0, 1.0, nMeshSize, 0.0, 1.0 );

//glEvalMesh2( mode, i1, i2, j1, j2 )
  glEvalMesh2( GL_LINE, 0, nMeshSize, 0, nMeshSize );
}
```

Ok, wie initialisieren wir die Punkte?. **Wie Fruher**



Bézier-Flächen mit OpenGL

Die Vorbereitung war wie folgt:

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

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

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

Nehmen wir denn alles! ctrl+ C

Bézier-Flächen mit OpenGL

```
void zeichneBezierFlaeche( const vector<vector<Vec3D> >& p,
                           int nMeshSize = 10 ){

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

  for ( int k = 3; k <= m; k += 3 )
    for ( int l = 3; l <= n; l += 3 ){

        glMap2f( GL_MAP2_VERTEX_3, 0.0, 1.0, 3, 4,
              0.0, 1.0, 3*4, 4, /*points*/ );

        glEnable( GL_MAP2_VERTEX_3 );

        glMapGrid2f( nMeshSize, 0.0, 1.0, nMeshSize, 0.0, 1.0 );

        glEvalMesh2( GL_LINE, 0, nMeshSize, 0, nMeshSize );
    }
```

Was denn mit die Punkte? Punkte an OpenGL übergeben.

Bézier-Flächen mit OpenGL

```
void zeichneBezierFlaeche( const vector<vector<Vec3D> >& p,
                           int nMeshSize = 10 ){

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

  // Array der Kontrollpunkte fuer OpenGL vorbereiten
  GLfloat *apPoints = new GLfloat[3*4*4];

  for ( int k = 3; k <= m; k += 3 )
    for ( int l = 3; l <= n; l += 3 ){

      //Übergeben 16 Kontrollpunkte der aktuellen Fläche an OpenGL.
        for( int i = 0; i < 4; i++ )
            for( int j = 0; j < 4; j++ ){
                apPoints[3*(4*i+j)+0] = p[k-3+i][l-3+j].el[0];  //x
                apPoints[3*(4*i+j)+1] = p[k-3+i][l-3+j].el[1];  //y
                apPoints[3*(4*i+j)+2] = p[k-3+i][l-3+j].el[2];  //z
            }
    }
```

```
void zeichneBezierFlaeche( const vector<vector<Vec3D> >& p,
                           int nMeshSize = 10 ){

  GLfloat *apPoints = new GLfloat[3*4*4];

  for ( int k = 3; k <= m; k += 3 )
    for ( int l = 3; l <= n; l += 3 ){

      //Übergeben 16 Kontrollpunkte der aktuellen Fläche an OpenGL.

      ...

      glMap2f( GL_MAP2_VERTEX_3, 0.0, 1.0, 3, 4, 0.0, 1.0, 3*4, 4,
↪   apPoints );
      glEnable( GL_MAP2_VERTEX_3 );
      glMapGrid2f( nMeshSize, 0.0, 1.0, nMeshSize, 0.0, 1.0 );
      glEvalMesh2( GL_LINE, 0, nMeshSize, 0, nMeshSize );
    }
  delete[] apPoints;
}
```

Rotationskörper

Rotationskörper Versucht es zu Hause.