Tutorial 4： B-spline Curve and Surface

Chenqi Luo

Assignment 2 is released.

- Please be noted that assignment2 is released on the course homepage. http://faculty.sist.shanghaitech.edu.cn/faculty/liuxp/course/cs171.01/assignment/2/assignment2.html
- You should carefully read the assignment requirements and the submission deadline. We suggest you start your assignment as early as possible.
  If you have any question about the assignment, don't be shy and please post them on Piazza.
- There are some details you need to pay attention to.

1. Put your source code in Coding/src or Coding/include, do not put your source code into build directory.
2. You should ensure that your total file size is smaller than **15MB**! Please clean your project intermediate files before submission.
3. Keep a nice and clean code style, e.g., do not use Pinyin for your variable name.
4. Write the report more formally and elaborately. We give you a sample_report.pdf for reference.

If you violate any rule stated above, we may impose a score deduction upon your assignments.

- Furthermore, we provide you a skeleton code just to help you understand what you need to write, but you can add or modify any part of it to implement your assignment requirements.

Assignment requirements

- **[must]** You are required to use de Boor algorithm to construct a B-Spline surface with given control points.
- **[must]** You must render your B-Spline surface with VAO, VBO in OpenGL.
- **[optional]** Based on the B-spline surface evaluation, construct a Non-Uniform Rational B-Spline (NURBS) surface.
- **[optional]** You can texture the B-Spline surface which is in the "resources" directory on the surface mesh your generated. You can try to think about how to reduce the texture distortion over your mesh.
- **[optional]** Implement the UI function that enables you to add control points and move control points to change the surface mesh.

B-spline curve

- Bezier Curve: the basis function is global.

- B-spline curve : the basis function of each control point can be local.

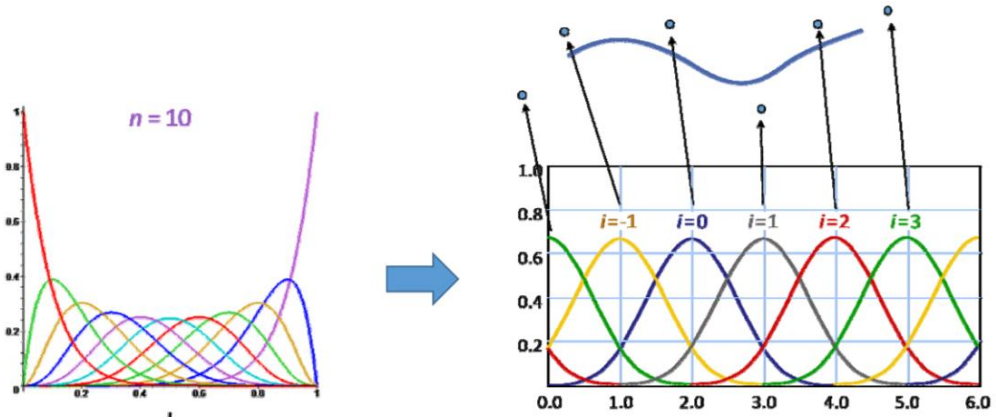
Recall the concepts

- Control points

  The control points determine the shape of the curve.

- Knots
  - The knot vector is a sequence of parameter values that determines where and how the control points affect the curve.

Comparison between Bezier and Bspline



Bezier

$$\mathbf{p}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} B_i^m(u)B_j^n(v)\mathbf{k}_{i,j}$$

Bspline

$$\mathbf{p}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)\mathbf{k}_{i,j}$$

De Boor Recursion: uniform case

- The uniform B-spline basis of order $\boldsymbol{k}$ (degree $\boldsymbol{k-1}$) is given as

$$N_i^1(t) = \begin{cases} 1, \text{if } i \le t < i+1 \\ 0, \quad \text{otherwise} \end{cases}$$

$$N_i^k(t) = \frac{t-i}{(i+k-1)-i} N_i^{k-1}(t) + \frac{(i+k)-t}{(i+k)-(i+1)} N_{i+1}^{k-1}(t)$$

$$= \frac{t-i}{k-1} N_i^{k-1}(t) + \frac{i+k-t}{k-1} N_{i+1}^{k-1}(t)$$

B-spline curves: general case

- Given: knot sequence $t_0 < t_1 < \cdots < t_n < \cdots < t_{n+k}$
  $((t_0, t_1, \ldots, t_{n+k})$ is called knot vector)
- Normalized B-spline functions $N_{i,k}$ of the order k (degree $k-1$) are defined as:

$$N_{i,1}(t) = \begin{cases} 1, & t_i \le t < t_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

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

for $k > 1$ and $i = 0, \ldots, n$

B-spline curves

- B-spline curves
  - Given: $n+1$ control points $\boldsymbol{d}_0, \dots, \boldsymbol{d}_n \in \mathbb{R}^3$

    knot vector $T = (t_0, \dots, t_n, \dots t_{n+k})$

  - Then, the B-spline curve $\boldsymbol{x}(t)$ of the order k is defined as

$$\boldsymbol{x}(t) = \sum_{i=0}^{n} N_{i,k}(t) \cdot \boldsymbol{d}_i$$

  - The points $\boldsymbol{d}_i$ are called *de Boor points*

Carl R. de Boor
German-American mathematician
University of Wisconsin-Madison

B-spline curves

- Interesting property:
  - B-spline functions $N_{i,k}$ $(i = 0, \dots, k-1)$ of the order k over the knot vector $T = (t_0, t_1, \dots, t_{2k-1}) =$ $(\underbrace{0, \dots, 0}_{k \text{ times}}, \underbrace{1, \dots, 1}_{k \text{ times}})$

    are Bernstein polynomials B_i^{k-1} of degree $k-1$

Basis properties

- For the so defined basis functions, the following properties can be shown:
  - $N_{i,k}(t) > 0$ for $t_i < t < t_{i+k}$
  - $N_{i,k}(t) = 0$ for $t_0 < t < t_i$ or $t_{i+k} < t < t_{n+k}$
  - $\sum_{i=0}^{n} N_{i,k}(t) = 1$ for $t_{k-1} \leq t \leq t_{n+1}$
- For $t_i \leq t_j \leq t_{i+k}$, the basis functions $N_{i,k}(t)$ are C^{k-2} at the knots t_j
- The interval $[t_i, t_{i+k}]$ is called support of $N_{i,k}$

The de Boor algorithm

- Given:

  $\boldsymbol{d}_0, \dots, \boldsymbol{d}_n$: de Boor points

  $(t_0, \dots, t_{k-1} = t_0, t_k, t_{k+1}, \dots, t_n, t_{n+1}, \dots, t_{n+k} = t_{n+1})$:

  Knot vector

- wanted:

  Curve point $\boldsymbol{x}(t)$ of the B-spline curve of the order k

The de Boor algorithm

1. Search index r with $t_r \leq t < t_{r+1}$
2. for $i = r - k + 1, \dots, r$

$$d_i^0 = d_i$$

- for $j = 1, \dots, k - 1$

for $i = r - k + 1 + j, \dots, r$

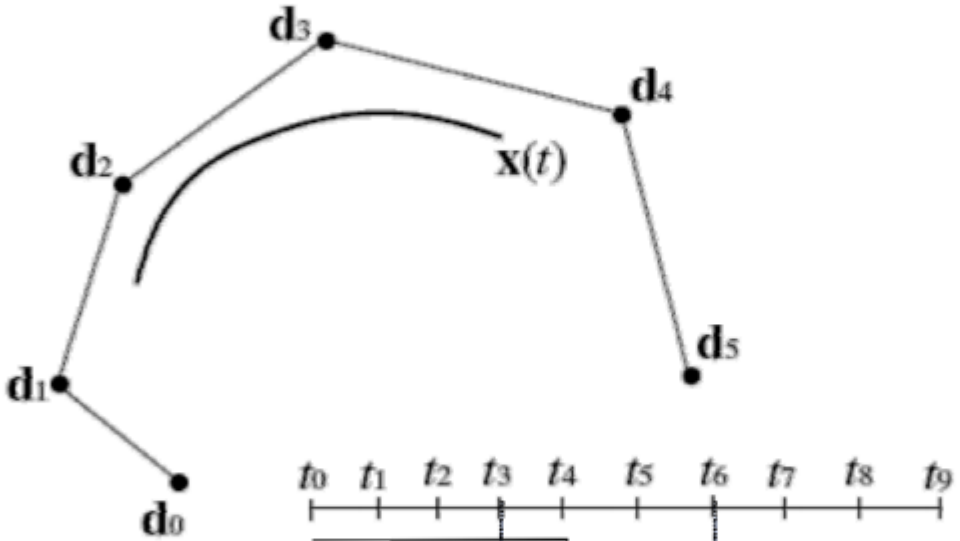
$$d_i^j = \left(1 - \alpha_i^j\right) \cdot d_{i-1}^{j-1} + \alpha_i^j \cdot d_i^{j-1}$$

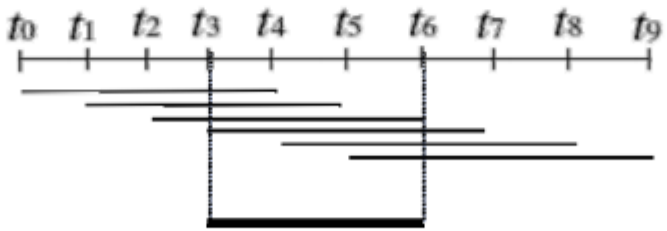
with $\alpha_i^j = \frac{t - t_i}{t_{i+k-j} - t_i}$

Then: $d_r^{k-1} = x(t)$

Example

- $k = 4, n = 5$





Support intervals of $N_{i,k}$

Curve defined in interval $t_3 \leq t \leq t_6$

B-spline curves

- Effect of multiple knots:
  - set: $t_0 = t_1 = \cdots = t_{k-1}$
  - and $t_{n+1} = t_{n+2} = \cdots = t_{n+k}$

$\boldsymbol{d}_0$ and $\boldsymbol{d}_n$ are interpolated

Example

- Degree = 3
- Insert u=0.4;

u0 = u1 = u2 = u3	u4	u5	u6	u7 = u8 = u9 = u10
0	0.25	0.5	0.75	1

Let us compute the second column. The involved a coefficients are

$$a_{4,1} = (u - u_4) / (u_{4+3} - u_4) = 0.2$$
$$a_{3,1} = (u - u_3) / (u_{3+3} - u_3) = 8/15 = 0.53$$
$$a_{2,1} = (u - u_2) / (u_{2+3} - u_2) = 0.8$$

Therefore, the first column is computed as follows:

$$p_{4,1} = (1 - a_{4,1})p_{3,0} + a_{4,1}p_{4,0} = 0.8p_{3,0} + 0.2p_{4,0}$$
$$p_{3,1} = (1 - a_{3,1})p_{2,0} + a_{3,1}p_{3,0} = 0.47p_{2,0} + 0.53p_{3,0}$$
$$p_{2,1} = (1 - a_{2,1})p_{1,0} + a_{2,1}p_{2,0} = 0.2p_{1,0} + 0.8p_{2,0}$$

To compute the second column, we need the following coefficients:

$$a_{4,2} = (u - u_4) / (u_{4+3-1} - u_4) = 0.3$$
$$a_{3,2} = (u - u_3) / (u_{3+3-1} - u_3) = 0.8$$

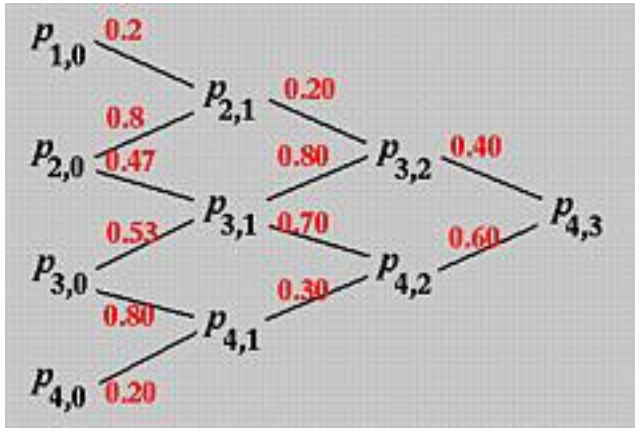
The points are

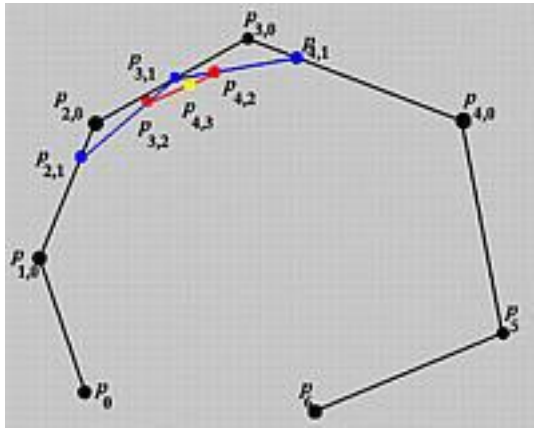
$$p_{4,2} = (1 - a_{4,2})\, p_{3,1} + a_{4,2}p_{4,1} = 0.7p_{3,1} + 0.3p_{4,1}$$
$$p_{3,2} = (1 - a_{3,2})\, p_{2,1} + a_{3,2}p_{3,1} = 0.2p_{2,1} + 0.8p_{3,1}$$

Finally, since $a_{4,3} = (u - u_4)/ (u_{4+3-2} - u_4) = 0.6$, the final point is

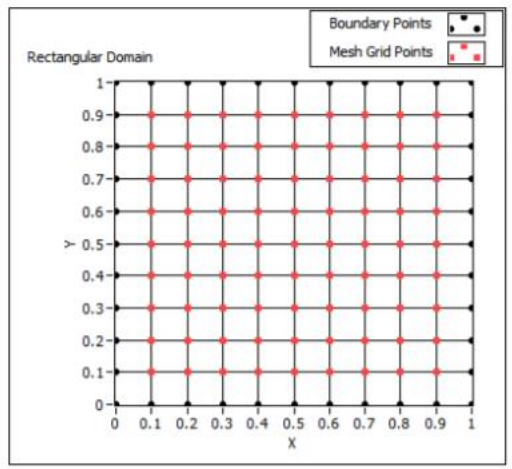
$$p_{4,3} = (1 - a_{4,3})p_{3,2} + a_{4,3}p_{4,2} = 0.4p_{3,2} + 0.6p_{4,2}$$

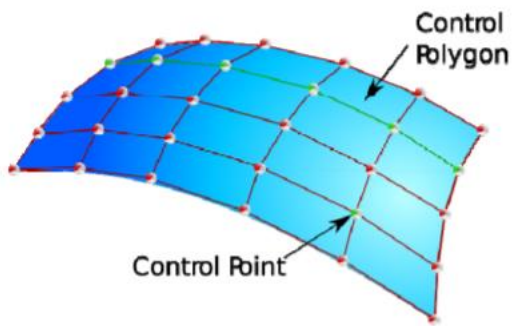
This is the point on the curve corresponding to $u = 0.4$.

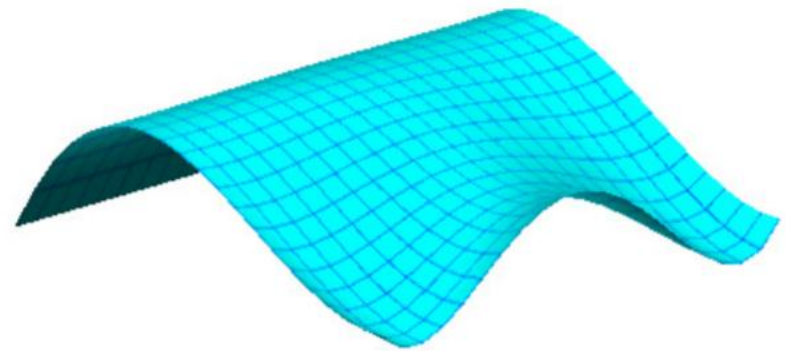




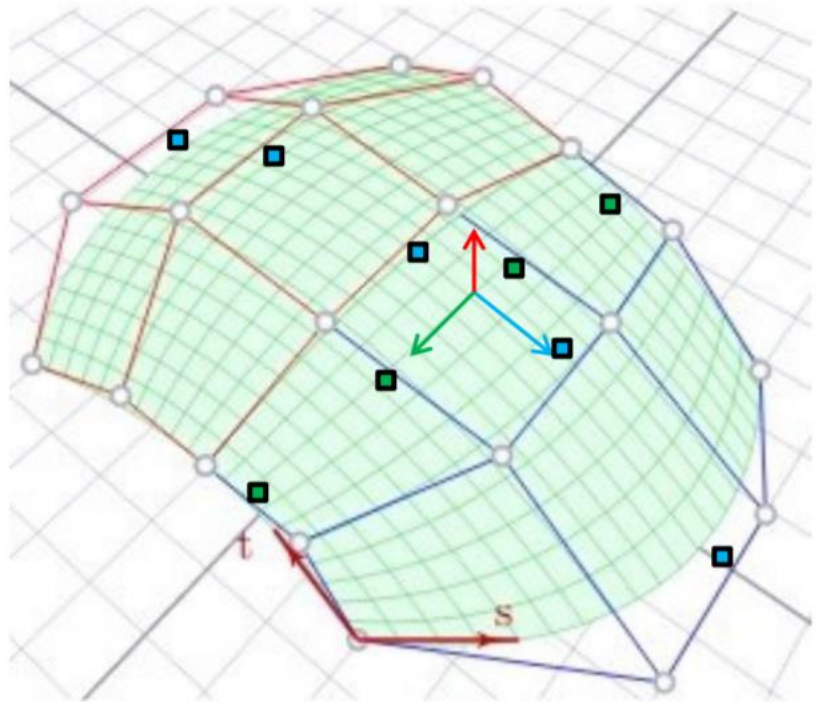
- **How to create meshes for free-form surfaces?**
  - create mesh in u-v parameter space







- **Computing the tangents and normal**
  - compute the tangent of two crossing Bézier curves
  - then take the cross product of these two tangents to form the normal



Coding

- What you need to implement in the code.

- Code demo