湖南科技大学课程教案
（章节、专题首页）

授课教师：王志喜　　职称：副教授　　单位：计算机科学与工程学院

课程名称	计算机图形图像技术
章节、专题	样条方法
教学目标及基本要求	初步掌握插值样条和逼近样条的表示方法以及样条物体的OpenGL实现。
教学重点	Hermite插值，Bézier样条，B-样条
教学难点	Hermite插值，Bézier样条，B-样条
教学内容与时间分配	（1）柔性物体与样条方法（0.7课时） （2）三次样条插值（0.7课时） （3）Bézier样条（1.5课时） （4）B-样条（1.6课时） 共计4.5课时。
习　　题	见9.7.1节（基础知识题）。

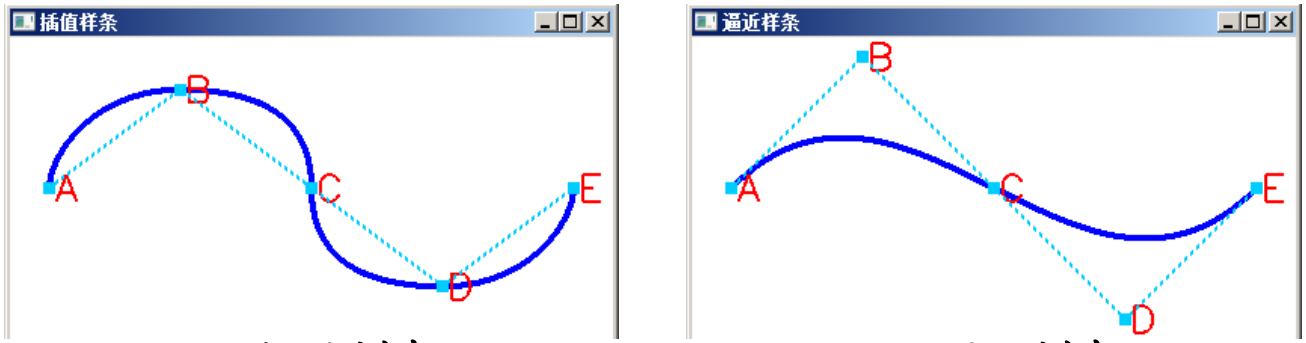
第9章　样条方法※

9.1　样条的含义和表示方法

9.1.1　样条的含义

1. 柔性物体和样条

- 柔性物体：柔性物体是在一定运动状态下或在它们接近其他物体时会改变其表面形状的物体。它们的曲面表面很难用常规形状表示。
- 样条方法：通过一组指定点集来生成平滑曲线的柔性带，如图9-1所示。



[1] 插值样条　　[2] 逼近样条

图9-1　样条曲线

2. 样条曲线和样条曲面

- 样条曲线：由多项式曲线段连接而成的曲线，在每段的边界处满足指定的连续条件。如图9-1所示。
- 样条曲面：用两组正交样条曲线描述。如图9-2所示。

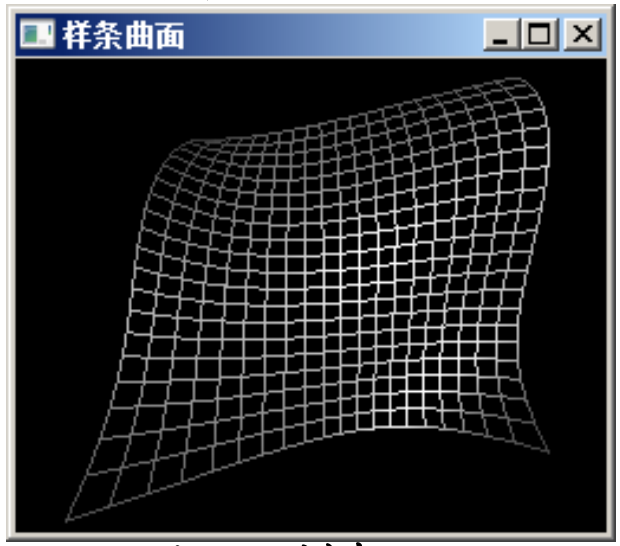
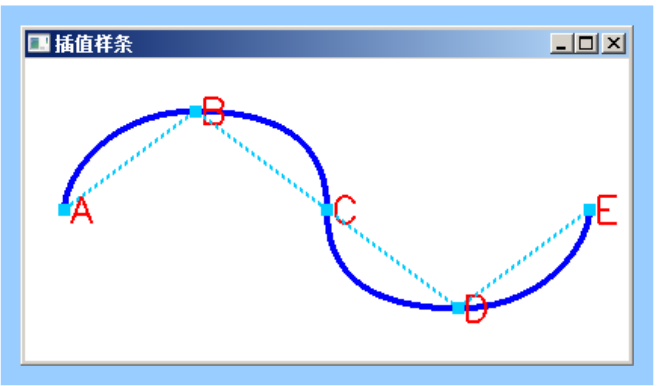
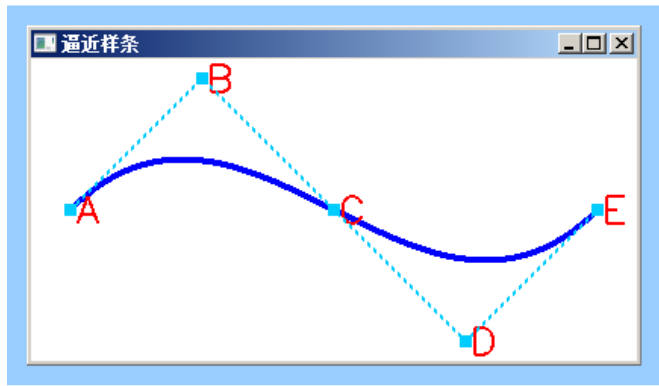


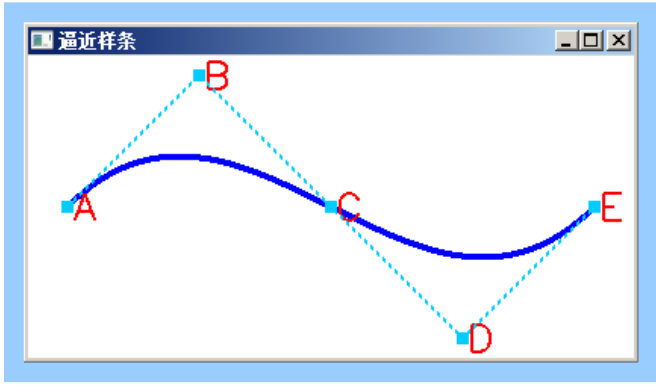
图9-2　样条曲面





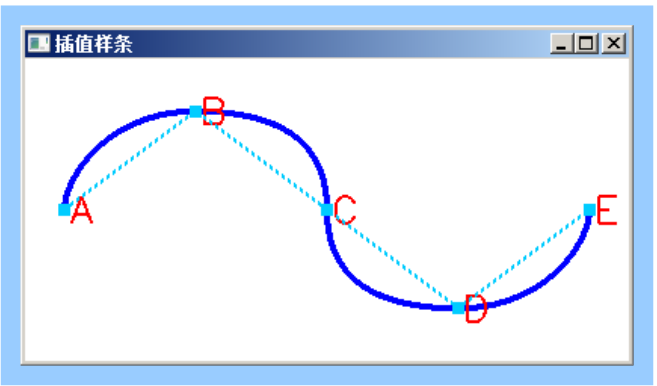
3. 控制多边形

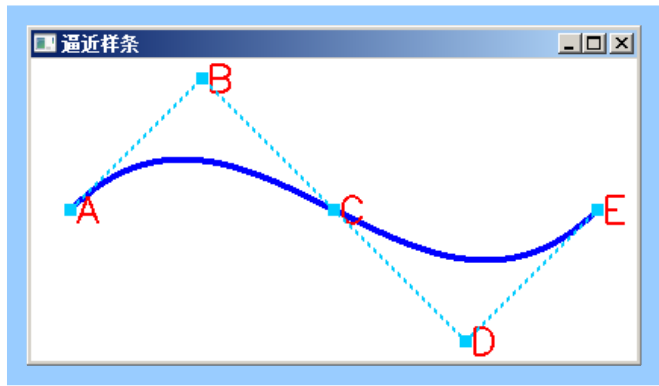
- 控制点：一组坐标点，这些点给出了曲线的大致形状。例如，图9-1[2]中的 $\{A,B,C,D,E\}$ 。
- 控制多边形（控制图、特征多边形）：连接有一定次序的控制点的线段序列。例如，图9-1[2]中的 $ABCDE$ 。



4. 插值样条和逼近样条

- 插值样条：控制点不仅给出了曲线的大致形状，并且选取的多项式使得曲线通过每个控制点。如图9-1[1]所示。通常用于数值化绘图或指定动画路径。
- 逼近样条：控制点只是给出了曲线的大致形状，选取的多项式使得曲线不一定通过每个控制点。如图9-1[2]所示。一般作为设计工具用于构造物体形状。

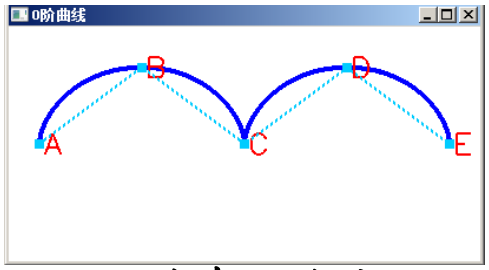




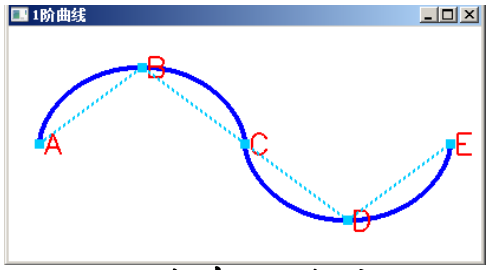
5. 参数连续性条件

假设如图9-3所示的3条曲线均由两个曲线段连接而成，且两个曲线段的交点都是C。

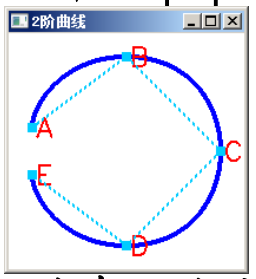
- 0阶参数连续性 C^0：曲线相连。如图9-3[1]所示。
- 1阶参数连续性 C^1：交点处有相同的一阶导数。如图9-3[2]所示。
- 2阶参数连续性 C^2：交点处有相同的二阶导数。如图9-3[3]所示。



[1] 0阶参数连续性



[2] 1阶参数连续性



[3] 2阶参数连续性

图9-3　参数连续性条件

9.1.2 样条的表示方法

1. 样条的3种等价表示方法

给定多项式的次数和控制点位置后，指定一个具体的样条表达式有3种等价的方法。

- 边界条件表示。列出一组加在样条上的边界条件。
- 样条基本矩阵表示。列出描述样条特征的矩阵。
- 基函数（又称混合函数、调和函数）表示。列出一组基函数，确定如何组合指定的几何约束条件，以计算曲线路径上的位置。

2. 样条路径的表达式

为了说明这三种等价描述，假设样条路径关于 x 分量的三次参数多项式表达式为

$$x(u)=a_x u^3+b_x u^2+c_x u+d_x \quad 0\le u\le 1$$

写成矩阵形式，得

$$x(u)=\begin{pmatrix}u^3 & u^2 & u & 1\end{pmatrix}\begin{pmatrix}a_x\\ b_x\\ c_x\\ d_x\end{pmatrix}$$

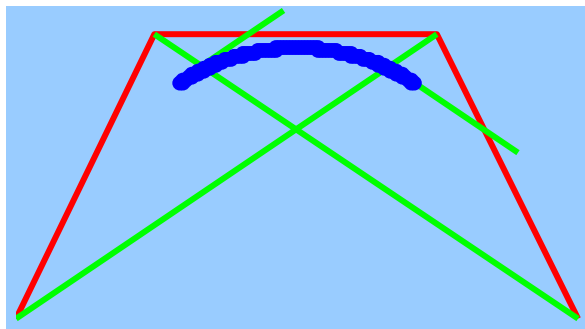
其中，$U=\begin{pmatrix}u^3 & u^2 & u & 1\end{pmatrix}$ 是参数矩阵，$C=\begin{pmatrix}a_x & b_x & c_x & d_x\end{pmatrix}^T$ 是系数矩阵。

9.1.3 一个样条路径示例

1. 边界条件表示

已知4个控制点的 x 坐标分别是 x_0、x_1、x_2 和 x_3，三次均匀B-样条的样条路径关于 x 分量在这4个控制点上的边界条件是

$$\begin{cases} x(0)=\dfrac{1}{6}(x_0+4x_1+x_2) \\ x(1)=\dfrac{1}{6}(x_1+4x_2+x_3) \\ x'(0)=\dfrac{1}{2}(x_2-x_0) \\ x'(1)=\dfrac{1}{2}(x_3-x_1) \end{cases}$$



$$x(u)=\begin{pmatrix}u^3 & u^2 & u & 1\end{pmatrix}\begin{pmatrix}a_x\\ b_x\\ c_x\\ d_x\end{pmatrix}$$

2. 样条基本矩阵表示

对 $x(u)$ 的矩阵形式求导数，得

$$x'(u)=\begin{pmatrix}3u^2 & 2u & 1 & 0\end{pmatrix}\begin{pmatrix}a_x\\ b_x\\ c_x\\ d_x\end{pmatrix}$$

从而

$$\begin{pmatrix}x(0)\\ x(1)\\ x'(0)\\ x'(1)\end{pmatrix}=\begin{pmatrix}0 & 0 & 0 & 1\\ 1 & 1 & 1 & 1\\ 0 & 0 & 1 & 0\\ 3 & 2 & 1 & 0\end{pmatrix}\begin{pmatrix}a_x\\ b_x\\ c_x\\ d_x\end{pmatrix}$$

$$\begin{cases} x(0)=\frac{1}{6}(x_0+4x_1+x_2) \\ x(1)=\frac{1}{6}(x_1+4x_2+x_3) \\ x'(0)=\frac{1}{2}(x_2-x_0) \\ x'(1)=\frac{1}{2}(x_3-x_1) \end{cases}$$

将边界条件写成矩阵形式，得

$$\begin{pmatrix} x(0) \\ x(1) \\ x'(0) \\ x'(1) \end{pmatrix} = \frac{1}{6}\begin{pmatrix} 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ -3 & 0 & 3 & 0 \\ 0 & -3 & 0 & 3 \end{pmatrix}\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

由此可得方程

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix}\begin{pmatrix} a_x \\ b_x \\ c_x \\ d_x \end{pmatrix} = \frac{1}{6}\begin{pmatrix} 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ -3 & 0 & 3 & 0 \\ 0 & -3 & 0 & 3 \end{pmatrix}\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$\begin{pmatrix} x(0) \\ x(1) \\ x'(0) \\ x'(1) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix}\begin{pmatrix} a_x \\ b_x \\ c_x \\ d_x \end{pmatrix}$$

解得

$$C=\begin{pmatrix}a_x\\ b_x\\ c_x\\ d_x\end{pmatrix}=\begin{pmatrix}0&0&0&1\\ 1&1&1&1\\ 0&0&1&0\\ 3&2&1&0\end{pmatrix}^{-1}\times\frac{1}{6}\begin{pmatrix}1&4&1&0\\ 0&1&4&1\\ -3&0&3&0\\ 0&-3&0&3\end{pmatrix}\begin{pmatrix}x_0\\ x_1\\ x_2\\ x_3\end{pmatrix}$$

$$=\frac{1}{6}\begin{pmatrix}-1&3&-3&1\\ 3&-6&3&0\\ -3&0&3&0\\ 1&4&1&0\end{pmatrix}\begin{pmatrix}x_0\\ x_1\\ x_2\\ x_3\end{pmatrix}$$

$$\begin{pmatrix}0&0&0&1\\ 1&1&1&1\\ 0&0&1&0\\ 3&2&1&0\end{pmatrix}\begin{pmatrix}a_x\\ b_x\\ c_x\\ d_x\end{pmatrix}=\frac{1}{6}\begin{pmatrix}1&4&1&0\\ 0&1&4&1\\ -3&0&3&0\\ 0&-3&0&3\end{pmatrix}\begin{pmatrix}x_0\\ x_1\\ x_2\\ x_3\end{pmatrix}$$

从而

$$x(u)=U\times C=\begin{pmatrix}u^3 & u^2 & u & 1\end{pmatrix}\frac{1}{6}\begin{pmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{pmatrix}\begin{pmatrix}x_0\\ x_1\\ x_2\\ x_3\end{pmatrix}$$

其中

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

是三次均匀B-样条的基本矩阵。

3. 基函数表示

$$x(u)=\begin{pmatrix}u^3 & u^2 & u & 1\end{pmatrix}\frac{1}{6}\begin{pmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{pmatrix}\begin{pmatrix}x_0\\ x_1\\ x_2\\ x_3\end{pmatrix}$$

$$=\frac{1}{6}(1-u)^3x_0+\frac{1}{6}(3u^3-6u^2+4)x_1+$$

$$\frac{1}{6}(-3u^3+3u^2+3u+1)x_2+\frac{1}{6}u^3x_3$$

其中，$N_{03}(u)=\frac{1}{6}(1-u)^3$、$N_{13}(u)=\frac{1}{6}(3u^3-6u^2+4)$、$N_{23}(u)=\frac{1}{6}(-3u^3+3u^2+3u+1)$ 和 $N_{33}(u)=\frac{1}{6}u^3$ 是三次均匀B-样条的基函数。

9.1.3 样条曲面

定义一个样条曲面可以通过使用在某个空间区域中的一个控制点网格指定两组正交的样条曲线来实现。

假设在某一个空间区域中给定了 $(m+1)\times(n+1)$ 个控制点 $\{P_{ij}\,|\,i=0,\ldots,m,j=0,\ldots,n\}$，则样条曲面上的任何一个位置都可以用样条曲线基函数的乘积来计算。计算方法为

$$p(u,v)=\sum_{i=0}^{m}\sum_{j=0}^{n}p_{ij}\times BF_i(u)\times BF_j(v)$$

其中 $u_{\min}\le u\le u_{\max}$，$v_{\min}\le v\le v_{\max}$。

9.2 三次样条插值

三次样条插值方法大多用于建立物体运动路径或提供实体表示和动画，有时也用来设计物体形状。

三次多项式在灵活性和计算速度之间提供了一个合理的折中方案。它比更高次多项式需要更少的计算时间和存储空间，模拟任意曲线形状时比低次多项式更灵活。

9.2.1 三次样条插值的描述

假设有 $n+1$ 个控制点，坐标分别为 $P_k=(x_k,y_k,z_k)$，$k=0,\ldots,n$。拟合每一对控制点的插值曲线段的参数方程为

$$\begin{cases} x(u)=a_x u^3+b_x u^2+c_x u+d_x \\ y(u)=a_y u^3+b_y u^2+c_y u+d_y \\ z(u)=a_z u^3+b_z u^2+c_z u+d_z \end{cases} \qquad 0\le u\le 1$$

写成矩阵形式，可得

$$\begin{pmatrix} x(u) & y(u) & z(u) \end{pmatrix}=\begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix}\begin{pmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{pmatrix}$$

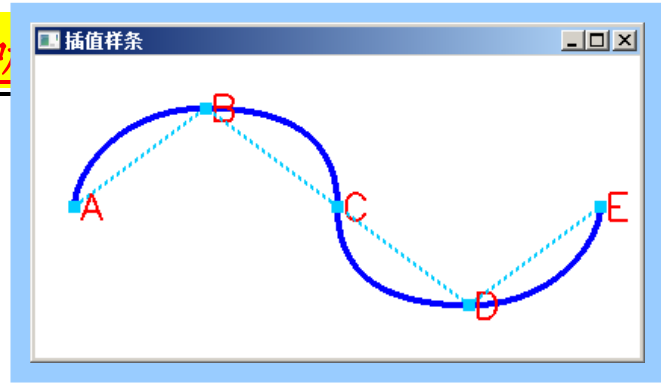
为了描述方便，将插值曲线段的参数方程改写成向量形式。令 $a=(a_x,a_y,a_z)$、$b=(b_x,b_y,b_z)$、$c=(c_x,c_y,c_z)$、$d=(d_x,d_y,d_z)$，就可以得到下列向量形式的参数方程。

$$p(u)=au^3+bu^2+cu+d,\ 0\le u\le 1$$

写成矩阵形式，可得

$$p(u)=\begin{pmatrix}u^3 & u^2 & u & 1\end{pmatrix}\begin{pmatrix}a\\ b\\ c\\ d\end{pmatrix}$$

其中，$U=\begin{pmatrix}u^3 & u^2 & u & 1\end{pmatrix}$是参数矩阵，$C=\begin{pmatrix}a & b & c & d\end{pmatrix}^T$ 是系数矩阵。



9.2.2 Hermite插值

1. 边界条件表示

设 P_k 和 P_{k+1} 之间的曲线段是三次参数函数 $p(u)(0 \le u \le 1)$，则边界条件为

$$\begin{cases} p(0) = P_k \\ p(1) = P_{k+1} \\ p'(0) = dP_k \\ p'(1) = dP_{k+1} \end{cases}$$

例如，对于图9-1[1]中的曲线段 $p = \overset{\frown}{BC}$ 可以给出下列形式的边界条件。

$$\begin{cases} p_x(0) = x_B \\ p_x(1) = x_C \\ p_x'(0) = dx_B \\ p_x'(1) = dx_C \end{cases} \quad \begin{cases} p_y(0) = y_B \\ p_y(1) = y_C \\ p_y'(0) = dy_B \\ p_y'(1) = dy_C \end{cases} \quad \begin{cases} p_z(0) = z_B \\ p_z(1) = z_C \\ p_z'(0) = dz_B \\ p_z'(1) = dz_C \end{cases}$$

2. 基本矩阵表示

$$p(u)=\begin{pmatrix}u^3 & u^2 & u & 1\end{pmatrix}\begin{pmatrix}a\\ b\\ c\\ d\end{pmatrix} \qquad 0\le u\le 1$$

求导数，得

$$p'(u)=\begin{pmatrix}3u^2 & 2u & 1 & 0\end{pmatrix}\begin{pmatrix}a\\ b\\ c\\ d\end{pmatrix} \qquad 0\le u\le 1$$

$$p'(u)=\begin{pmatrix}3u^2 & 2u & 1 & 0\end{pmatrix}\begin{pmatrix}a\\b\\c\\d\end{pmatrix}$$

将边界条件代入上述方程，得

$$\begin{pmatrix}P_k\\P_{k+1}\\dP_k\\dP_{k+1}\end{pmatrix}=\begin{pmatrix}p(0)\\p(1)\\p'(0)\\p'(1)\end{pmatrix}=\begin{pmatrix}0&0&0&1\\1&1&1&1\\0&0&1&0\\3&2&1&0\end{pmatrix}\begin{pmatrix}a\\b\\c\\d\end{pmatrix}$$

解得

$$C=\begin{pmatrix}a\\b\\c\\d\end{pmatrix}=\begin{pmatrix}0&0&0&1\\1&1&1&1\\0&0&1&0\\3&2&1&0\end{pmatrix}^{-1}\begin{pmatrix}P_k\\P_{k+1}\\dP_k\\dP_{k+1}\end{pmatrix}$$

$$=\begin{pmatrix}2&-2&1&1\\-3&3&-2&-1\\0&0&1&0\\1&0&0&0\end{pmatrix}\begin{pmatrix}P_k\\P_{k+1}\\dP_k\\dP_{k+1}\end{pmatrix}$$

$$p(u)=U\times C=\begin{pmatrix}u^3 & u^2 & u & 1\end{pmatrix}\begin{pmatrix}2 & -2 & 1 & 1\\ -3 & 3 & -2 & -1\\ 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 0\end{pmatrix}\begin{pmatrix}P_k\\ P_{k+1}\\ dP_k\\ dP_{k+1}\end{pmatrix}$$

其中

$$M_H=\begin{pmatrix}2 & -2 & 1 & 1\\ -3 & 3 & -2 & -1\\ 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 0\end{pmatrix}$$

是Hermite插值样条的基本矩阵，称为Hermite矩阵。

3. 基函数表示

$$p(u)=P_k\times(2u^3-3u^2+1)+P_{k+1}\times(-2u^3+3u^2)+$$
$$dP_k\times(u^3-2u^2+u)+dP_{k+1}\times(u^3-u^2)$$

其中 $H_0(u)=2u^3-3u^2+1$ 、 $H_1(u)=-2u^3+3u^2$ 、 $H_2(u)=u^3-2u^2+u$ 和 $H_3(u)=u^3-u^2$ 是Hermite插值样条的基函数。

4. 参数方程表示

$$\begin{cases} x(u)=x_k\times H_0(u)+x_{k+1}\times H_1(u)+dx_k\times H_2(u)+dx_{k+1}\times H_3(u) \\ y(u)=y_k\times H_0(u)+y_{k+1}\times H_1(u)+dy_k\times H_2(u)+dy_{k+1}\times H_3(u) \\ z(u)=z_k\times H_0(u)+z_{k+1}\times H_1(u)+dz_k\times H_2(u)+dz_{k+1}\times H_3(u) \end{cases}$$
$$(0\le u\le 1)$$

$$\begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix}\begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} P_k \\ P_{k+1} \\ DP_k \\ DP_{k+1} \end{pmatrix}$$

9.3 Bézier样条

Bézier样条是法国工程师Bézier使用逼近样条为雷诺汽车公司设计汽车外形而开发的。Bézier样条在各种CAD系统、大多数图形系统、相关绘图和图形软件包中有广泛应用。

9.3.1 Bézier基函数

1. 基函数的定义

已知 $n \geq 1$， $0 \leq k \leq n$， $u \in [0,1]$。由公式 $B_{k,n}(u) = C_n^k u^k (1-u)^{n-k}$ 定义的 $B_{k,n}(u)$ 称为 n 次Bézier样条基函数。其中，$C_n^k = \dfrac{n!}{k!(n-k)!}$ 是二项式系数。

2. 基函数的2个特性

- 非负性：$B_{k,n}(u) \geq 0$。
- 权性：$\sum_{k=0}^{n} B_{k,n}(u) = (u + (1-u))^n = 1$。

9.3.2 Bézier曲线

1. 定义

已知 $n+1$ 个控制点 $\{P_k=(x_k,y_k,z_k)\,|\,k=0,\ldots,n\}$，则 P_0 与 P_n 之间的逼近Bézier多项式函数为

$$p(u)=\sum_{k=0}^{n}P_k\times B_{k,\,n}(u)\qquad 0\le u\le 1$$

写成参数方程就是

$$\begin{cases}x(u)=\sum\limits_{k=0}^{n}x_k\times B_{k,n}(u)\\ y(u)=\sum\limits_{k=0}^{n}y_k\times B_{k,n}(u)\qquad 0\le u\le 1\\ z(u)=\sum\limits_{k=0}^{n}z_k\times B_{k,n}(u)\end{cases}$$

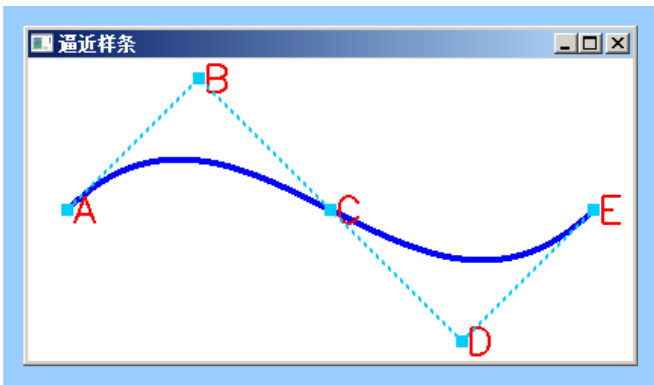
$B_{k,n}(u)=C_n^k u^k(1-u)^{n-k}$

2. 性质

（1）起点位置。显然，当 $k>0$ 时 $B_{k,n}(0)=0$，所以 $p(0)=P_0B_{0,n}(0)$。由 $B_{0,n}(u)=(1-u)^n$ 可得 $p(0)=P_0$。如图9-1[2]中的A。

（2）终点位置。显然，当 $k<n$ 时 $B_{k,n}(1)=0$，所以 $p(1)=P_nB_{n,n}(1)$。由 $B_{n,n}(u)=u^n$ 可得 $p(1)=P_n$。如图9-1[2]中的E。

（3）起点处切线。当 $k>1$ 时，由 $B_{k,n}'(u)$ 含有因子 u^{k-1} 可知 $B_{k,n}'(0)=0$，所以 $p'(0)=P_0B_{0,n}'(0)+P_1B_{1,n}'(0)$。由 $B_{0,n}'(u)=-n(1-u)^{n-1}$ 和 $B_{1,n}'(u)=n(1-u)^{n-2}(1-nu)$ 可得 $p'(0)=n(P_1-P_0)$，即 P_0 处的切线为 P_0P_1。如图9-1[2]中的AB。



（4）终点处切线。当 $k<n-1$ 时，由 ${B_{k,n}}'(u)$ 含有因子 $(1-u)^{n-k-1}$ 可知 ${B_{k,n}}'(1)=0$，所以 $p'(1)=P_nB_{n,n}'(1)+P_{n-1}B_{n-1,n}'(1)$。由 ${B_{n,n}}'(u)=nu^{n-1}$ 和 ${B_{n-1,n}}'(u)=nu^{n-2}(n-1-nu)$ 可得 $p'(1)=n(P_n-P_{n-1})$，即 P_n 处的切线为 $P_{n-1}P_n$。如图9-1[2]中的DE。

$$B_{k,n}(u)=C_n^k u^k(1-u)^{n-k}$$

9.3.3 三次Bézier曲线

1. 基函数

$p(u)=P_0\times B_{0,3}(u)+P_1\times B_{1,3}(u)+P_2\times B_{2,3}(u)+P_3\times B_{3,3}(u)$，其中

$$\begin{cases}B_{0,3}(u)=(1-u)^3\\B_{1,3}(u)=3u(1-u)^2\\B_{2,3}(u)=3u^2(1-u)\\B_{3,3}(u)=u^3\end{cases}$$

2. 矩阵形式

$$p(u)=\begin{pmatrix}u^3 & u^2 & u & 1\end{pmatrix}\begin{pmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 3 & 0 & 0\\ 1 & 0 & 0 & 0\end{pmatrix}\begin{pmatrix}P_0\\ P_1\\ P_2\\ P_3\end{pmatrix}=U\times M_{\text{Bez}}\times M_{\text{geom}}$$

其中 M_{Bez} 是三次Bézier样条的基本矩阵。

$$\begin{aligned}p(u)&=P_0\times(1-u)^3+P_1\times 3u(1-u)^2+P_2\times 3u^2(1-u)+P_3\times u^3\\&=P_0\times(-u^3+3u^2-3u+1)+P_1\times(3u^3-6u^2+3u)+P_2\times(-3u^3+3u^2)+P_3\times u^3\end{aligned}$$

3. 示例

【问题】给定四个控制点 $P_0=(0,0)$、$P_1=(1,1)$、$P_2=(2,-1)$ 和 $P_3=(3,0)$，请构造一条三次Bézier曲线，并计算参数为0、1/3、2/3和1时的值。该曲线的图像如图9-4所示。

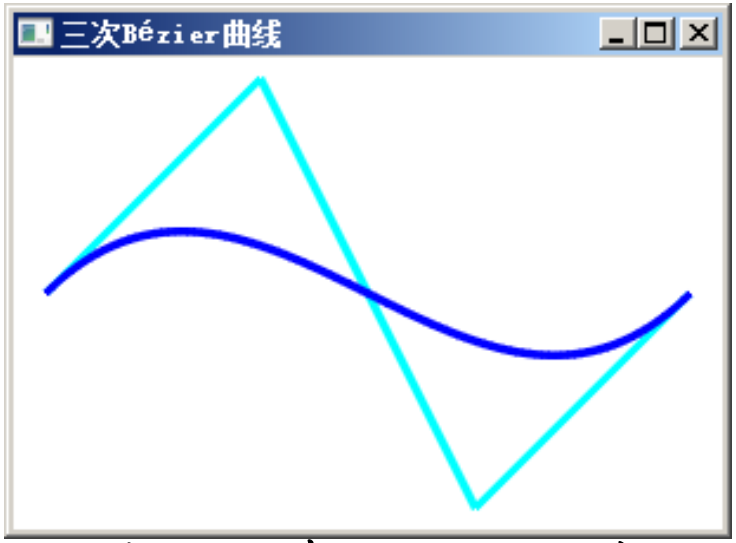


图9-4　一条三次Bézier曲线

【解答】

$$
\begin{aligned}
p(u) &= \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 2 & -1 \\ 3 & 0 \end{pmatrix} \\
&= \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} 0 & 6 \\ 0 & -9 \\ 3 & 3 \\ 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 3u & 6u^3 - 9u^2 + 3u \end{pmatrix}
\end{aligned}
$$

所以 $p(0) = (0,0)$ ， $p(1/3) = (1,2/9)$ ， $p(2/3) = (2,-2/9)$ ， $p(1) = (3,0)$ 。

9.3.4 Bézier曲面

1. 含义

如果在某一个空间区域中给定了 $(m+1)\times(n+1)$ 个控制点 $\{P_{ij}\,|\,i=0,\ldots,m,j=0,\ldots,n\}$，则将下列形式的参数曲面称为 $m\times n$ 次Bézier曲面。

$$p(u,v)=\sum_{i=0}^{m}\sum_{j=0}^{n}p_{ij}B_{i,m}(u)B_{j,n}(v)\qquad 0\le u,v\le 1$$

2. 示例

使用两组正交的Bézier曲线描述。如图9-5所示，其中左侧是网格线曲面，右侧是填充曲面，使用了4×4个控制点。

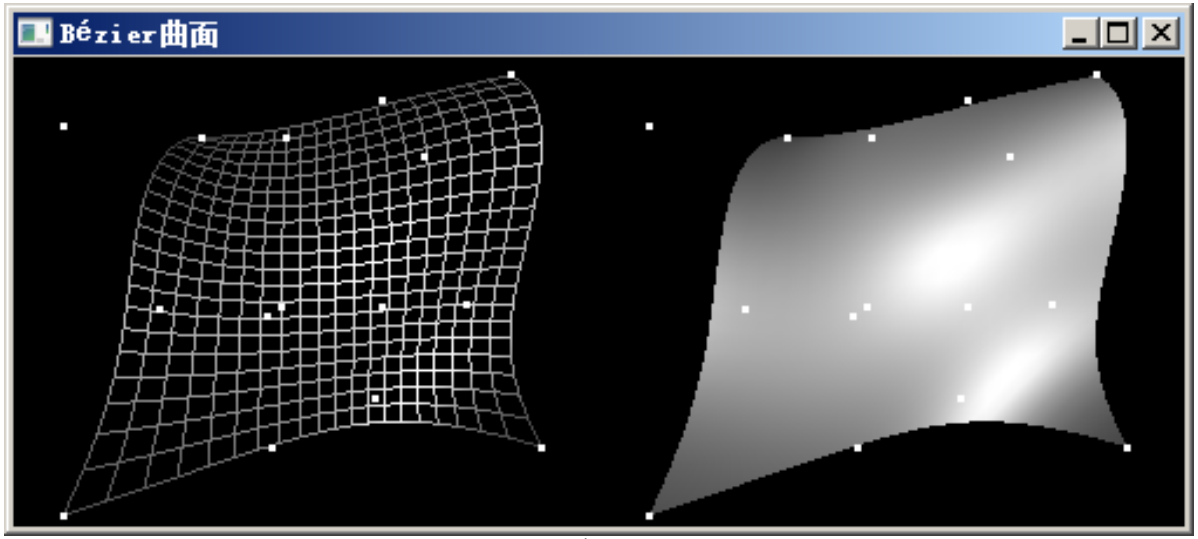


图9-5　一个Bézier曲面

9.4 Bézier样条的OpenGL实现※

【注】为了方便，将在本章编写的一些通用函数组织在文件gll.h中。下面列出了gll.h的预处理部分。

```
// gll.h
#pragma once
#include<stdio.h> // sprintf等
#include<stdlib.h> // 随机数函数等
#include<string.h> // 字符串函数
#include<stdarg.h> // 可变参数
#include<iso646.h> // not, and, or, etc.
#include<complex.h> // 复数函数, 需C99支持
#include<math.h> // floor, ceil, etc.
#include<tgmath.h> // 通用数学函数, 需C99支持
#include<gl/freeglut.h> // GL, GLU和GLUT函数
```

```
#ifndef Complex // gll.h的复数类型, 需C99支持
typedef double _Complex Complex;
#endif
```

9.4.1 需要的函数和调用

1. glMap1*()

【调用形式】

- void glMap1d(GL_MAP1_VERTEX_3, GLdouble u1, GLdouble u2, GLint stride, GLint order, GLdouble *points)
- void glMap1f(GL_MAP1_VERTEX_3, GLfloat u1, GLfloat u2, GLint stride, GLint order, GLfloat *points)

【功能】指定计算Bézier曲线坐标值的控制点和参数区间。

【参数】

- u1和u2是参数起始值和终止值。
- stride是相邻控制点的距离。
- order和points是控制点个数和控制点数组。

【说明】控制点数组还可能包含其他属性分量，计算相邻控制点和相邻行的距离时需考虑所有分量。

2. glMapGrid1*()

【函数原型】

- void glMapGrid1d(GLint un, GLdouble u1, GLdouble u2);
- void glMapGrid1f(GLint un, GLfloat u1, GLfloat u2);

【功能】定义一个一维网格，将参数区间[u1, u2]均匀分成un份，对于Bézier曲线，参数区间一般取[0, 1]。

3. glEvalMesh1()

【函数原型】void glEvalMesh1(GLenum mode, GLint i1, GLint i2);

【功能】用与“for(i=i1; i<=i2; ++i) ...”等价的办法自动计算(u1+i*du)对应的坐标值并生成对象（du是u参数间隔），第一个参数可取GL_POINT或GL_LINE。

4. glMap2*()

【调用形式】

- void glMap2d(GL_MAP2_VERTEX_3, GLdouble u1, GLdouble u2, GLint ustride, GLint uorder, GLdouble v1, GLdouble v2, GLint vstride, GLint vorder, GLdouble *points)
- void glMap2f(GL_MAP2_VERTEX_3, GLfloat u1, GLfloat u2, GLint ustride, GLint uorder, GLfloat v1, GLfloat v2, GLint vstride, GLint vorder, GLfloat *points)

【功能】指定计算Bézier曲面坐标值的控制点和参数区间。

【参数】

- u1和u2是u参数的起始值和终止值，v1和v2是v参数的起始值和终止值。
- ustride是相邻控制点（$P(i,j+1)$和$P(i,j)$）的距离，vstride是相邻行（$P(i+1,j)$和$P(i,j)$）的距离。
- uorder是控制点列数，vorder是控制点行数。
- points是控制点数组。

【说明】控制点数组还可能包含其他属性分量，计算相邻控制点和相邻行的距离时需考虑所有分量。

5. glMapGrid2*()

【函数原型】

- void glMapGrid2d(GLint un, GLdouble u1, GLdouble u2, GLint vn, GLdouble v1, GLdouble v2);
- void glMapGrid2f(GLint un, GLfloat u1, GLfloat u2, GLint vn, GLfloat v1, GLfloat v2);

【功能】定义一个二维网格，将参数区间[u1, u2][v1, v2]均匀分成un×vn份。

6. glEvalMesh2()

【函数原型】void glEvalMesh2(GLenum mode, int i1, int i2, int j1, int j2);

【功能】用与“for(i=i1; i<=i2; ++i) { for(j=j1; j<=j2; ++j) ... }”等价的办法自动计算(u1+i*du, v1+j*dv)对应的坐标值并生成对象（du是u参数间隔，dv是v参数间隔），第一个参数可取GL_POINT、GL_LINE或GL_FILL。

7. 计算函数的启用和禁用

- glEnable(GL_MAP1_VERTEX_3)。启用Bézier曲线坐标值的计算函数。
- glDisable(GL_MAP1_VERTEX_3)。禁用Bézier曲线坐标值的计算函数。
- glEnable(GL_MAP2_VERTEX_3)。启用曲面坐标值的计算函数。
- glDisable(GL_MAP2_VERTEX_3)。禁用曲面坐标值的计算函数。

9.4.2 实现方法

保存在文件gll.h中。

1. 绘制Bézier曲线

```
void gllBezCurve(const float pts[], int npts, int un)
{   // 控制点数组, 控制点个数, 参数区间分割数
    glMap1f(GL_MAP1_VERTEX_3, 0, 1, 3, npts, pts);
    // 坐标值计算方法(GL_MAP1_VERTEX_3, 参数起始值, 参数终止值,
    // 相邻控制点距离(含其他分量), 控制点个数, 控制点数组
    glEnable(GL_MAP1_VERTEX_3); // 启用坐标值计算函数
    glMapGrid1f(un, 0, 1); // 定义一维网格, 将[0, 1]均匀分成un份
    glEvalMesh1(GL_LINE, 0, un); // 计算坐标值并生成对象
    glDisable(GL_MAP1_VERTEX_3); // 禁用坐标值计算函数
}
```

2. Bézier曲面的实现方法

```
void gllBezSurface(GLenum mode, const float pts[], int uorder, int vorder,
                   int un, int vn)
{   // 绘制方式, 控制点数组, 控制点列数, 控制点行数,
    // u参数区间分割数, v参数区间分割数
    glMap2f(GL_MAP2_VERTEX_3, 0, 1, 3, uorder, 0, 1, 3 * uorder, vorder,
            pts);
    // 坐标值计算方法(GL_MAP2_VERTEX_3, u参数起始值, u参数终止值,
    // 控制点P(i, j+1)和P(i, j)的距离, 控制点列数,
    // v参数起始值, v参数终止值, 控制点P(i+1, j)和P(i, j)的距离,
    // 控制点行数, 控制点数组
    glEnable(GL_MAP2_VERTEX_3); // 启用坐标值计算函数
    glMapGrid2f(un, 0, 1, vn, 0, 1);
    // 定义一个二维网格, 将[0, 1][0, 1]均匀分成un×vn份
    glEvalMesh2(mode, 0, un, 0, vn); // 计算坐标值并生成对象
    glDisable(GL_MAP2_VERTEX_3); // 禁用坐标值计算函数
}
```

9.4.3 举例说明

1. 显示控制点

本章后续很多例子都需要显示控制点，这里将显示控制点的程序段组织成一个函数，保存在文件gll.h中。

```
void gllPoints(const float pts[], int npts)
{   const float *F = pts, *L = pts + 3 * npts;
    glBegin(GL_POINTS);
    for(; F != L; F += 3) glVertex3fv(F);
    glEnd();
} // 借鉴了C++ STL的常见做法
```

2. Bézier曲线

下列程序演示了Bézier曲线的绘制方法。运行结果如图9-6所示。

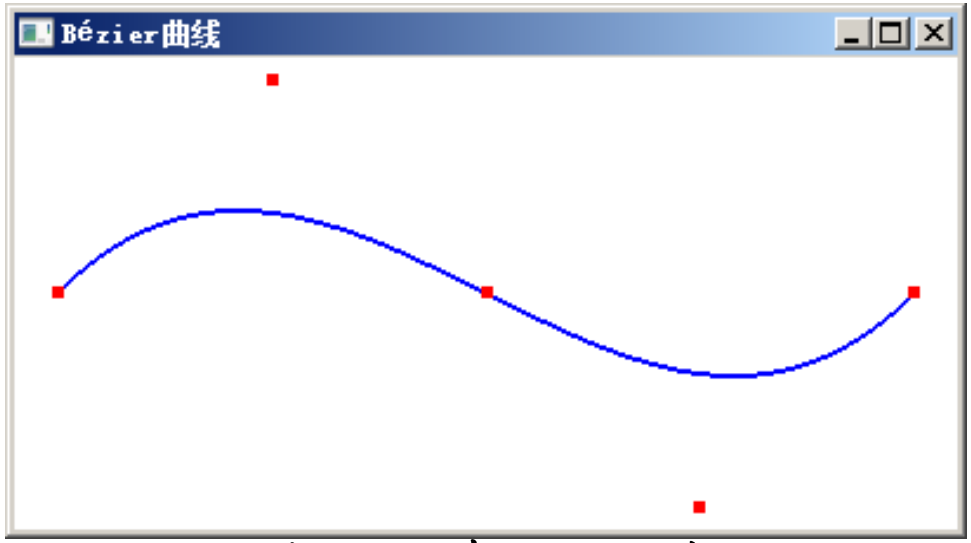


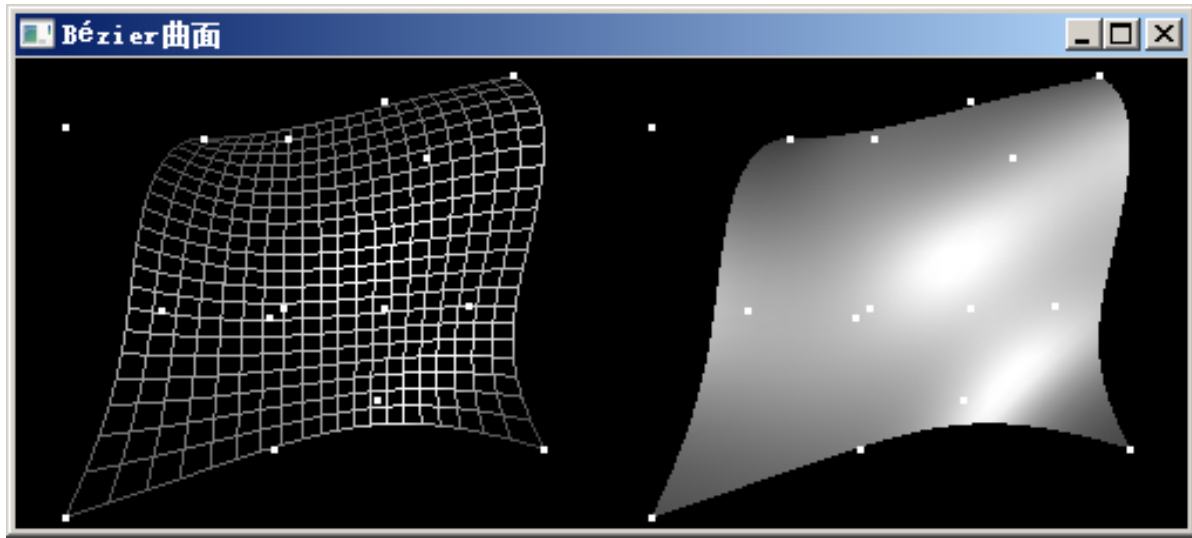
图9-6　一条Bézier曲线

```
// BezCurve.c
#include"gll.h"
void Reshape(int w, int h) // 窗口变化回调函数
{   glViewport(0, 0, w, h); // 根据指定位置和大小创建视口
    glMatrixMode(GL_PROJECTION); // 投影矩阵栈
    glLoadIdentity(); // 当前矩阵改为单位矩阵
    gluOrtho2D(-2.2, 2.2, -1.1, 1.1); // 裁剪窗口，比[-2, 2][-1, 1]稍大
    glMatrixMode(GL_MODELVIEW); // 视图造型矩阵栈
    glLoadIdentity(); // 当前矩阵改为单位矩阵
}
```

```
void Paint() // 场景绘制函数
{   glClearColor(1, 1, 1, 1); // 白色背景
    glClear(GL_COLOR_BUFFER_BIT); // 清除颜色缓冲
    // 控制点数组
    float pts[][3] = {{-2, 0, 0}, {-1, 1, 0}, {0, 0, 0}, {1, -1, 0}, {2, 0, 0}};
    int npts = 5, un = 100; // 控制点个数, 参数区间分割数
    glColor3f(0, 0, 1), glLineWidth(2); // 曲线颜色和线宽
    gllBezCurve((float *)pts, npts, un); // Bézier曲线
    glColor3f(1, 0, 0), glPointSize(5); // 控制点颜色和大小
    gllPoints((float *)pts, npts); // 控制点
    glFlush();
}
int main(int argc, char *argv[])
{   glutInit(&argc, argv);
    glutInitWindowSize(400, 200); // 程序窗口大小(4 : 2)
    glutCreateWindow("Bézier曲线");
    glutDisplayFunc(Paint);
    glutReshapeFunc(Reshape);
    glutMainLoop();
}
```

3. Bézier曲面

下列程序演示了Bézier曲面的绘制方法，分别绘制了一个网格线曲面和一个填充曲面，并绘制了它们的控制点以体验控制点和曲面的关系。运行结果如图9-5所示。



```
// BezSurf.c
#include"gll.h"
void Viewport(int x, int y, int w, int h) // 创建视口，指定投影变换
{   glViewport(x, y, w, h); // 根据指定位置和大小创建视口
    glMatrixMode(GL_PROJECTION); // 投影矩阵栈
    glLoadIdentity(); // 当前矩阵改为单位矩阵
    gluPerspective(30, (float)w / h, 1, 1000);
    // 透视变换(视角30度，宽高比，近平面，远平面)
    glTranslatef(0.25, 0.35, -8.5); // 场景右移0.25，上移0.35，远移8.5
    glMatrixMode(GL_MODELVIEW); // 视图造型矩阵栈
    glLoadIdentity(); // 当前矩阵改为单位矩阵
}
void Hint() // 初始化(材质属性和光照属性等)
{   glPointSize(3); // 点的大小
    glMaterialfv(GL_FRONT, GL_DIFFUSE, (float[]) {1, 1, 1, 1}); // 白色材质
    glEnable(GL_AUTO_NORMAL); // 自动计算表面法向量
    glEnable(GL_NORMALIZE); // 自动单位化法向量
    glEnable(GL_LIGHT0); // 启用0号光源
    glEnable(GL_LIGHTING); // 启用光照效果
}
```

```
void Paint() // 场景绘制函数
{   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    // 清除颜色缓冲和深度缓冲
    int w = glutGet(GLUT_WINDOW_WIDTH) / 2; // 视口宽度
    int h = glutGet(GLUT_WINDOW_HEIGHT); // 视口高度
    float pts[4][4][3] = // 控制点坐标
    {   {{ -1.5, -1.5, 4}, { -0.5, -1.5, 2}, {0.5, -1.5, -1}, {1.5, -1.5, 2}},
        {{ -1.5, -0.5, 1}, { -0.5, -0.5, 3}, {0.5, -0.5, 0}, {1.5, -0.5, -1}},
        {{ -1.5, 0.5, 4}, { -0.5, -0.5, 0}, {0.5, 0.5, 3}, {1.5, 0.5, 4}},
        {{ -1.5, 1.5, -2}, { -0.5, 1.5, -2}, {0.5, 1.5, 0}, {1.5, 1.5, 1}}
    };
    Viewport(0, 0, w, h); // 左侧用线框模型
    // 构造Bezier曲面(绘制方式, 控制点数组, 列数, 行数, u分割数, v分割数)
    gllBezSurface(GL_LINE, (float *)pts, 4, 4, 20, 20);
    gllPoints((float *)pts, 4 * 4); // 绘制控制点
    Viewport(w, 0, w, h); // 右侧用填充模型
    gllBezSurface(GL_FILL, (float *)pts, 4, 4, 20, 20);
    gllPoints((float *)pts, 4 * 4); // 绘制控制点
    glFlush();
}
```

```
int main(int argc, char *argv[])
{   glutInit(&argc, argv);
    glutInitWindowSize(500, 200); // 程序窗口大小(5 : 2)
    glutCreateWindow("Bézier曲面");
    Hint(); // 初始化(材质属性和光照属性等)
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

9.5 B-样条

9.5.1 B-样条基函数

1. 定义

已知 $2 \le d \le n+1$，给定参数 $u(u_{\min} \le u \le u_{\max})$ 的一个分割 $T = \{u_i\}_{i=0}^{n+d}$，其中 $u_i \le u_{i+1}$。由下列递归公式定义的 $N_{k,d}(u)$ 称为分割 T 的 d 阶B-样条基函数。

$$\begin{cases} N_{k,1}(u) = \begin{cases} 1 & u_k \le u < u_{k+1} \\ 0 & \text{else} \end{cases} \\ N_{k,d}(u) = \dfrac{u-u_k}{u_{k+d-1}-u_k} N_{k,d-1}(u) + \dfrac{u_{k+d}-u}{u_{k+d}-u_{k+1}} N_{k+1,d-1}(u) \end{cases}$$

其中，$T = \{u_i\}_{i=0}^{n+d}$ 称为节点向量，u_i 称为节点。

【注】如果在计算中遇到分母为0的情况，则定义0/0=0。

2. 基函数的性质

（1）局部性。当 $u \notin [u_k, u_{k+d})$ 时，$N_{k,d}(u)=0$，只有当 $u_k \le u < u_{k+d}$ 时，才有 $N_{k,d}(u)>0$。这是因为由递推公式可知，要计算 $N_{k,d}(u)$，必须计算 $N_{k,1}(u)$、$N_{k+1,1}(u)$、…、$N_{k+d-1,1}(u)$ 等1阶基函数。而当 $u \notin [u_k, u_{k+d})$ 时，这些1阶基函数都等于0。

（2）权性。B-样条基函数满足

$$\sum_{k=0}^{n} N_{k,d}(u)=1 \qquad u_{d-1} \le u \le u_{n+1}$$

（3）分段多项式。$N_{k,d}(u)$ 是次数不高于 $d-1$ 的分段多项式。其中每两个相邻节点之间的区间就是一个分段区间。

（4）连续性。$N_{k,d}(u)$ 具有 C^{d-2} 参数连续性。

$$\begin{cases} N_{k,1}(u)=\begin{cases}1 & u_k \le u < u_{k+1} \\ 0 & \text{else}\end{cases} \\ N_{k,d}(u)=\dfrac{u-u_k}{u_{k+d-1}-u_k}N_{k,d-1}(u)+\dfrac{u_{k+d}-u}{u_{k+d}-u_{k+1}}N_{k+1,d-1}(u) \end{cases}$$

9.5.2 B-样条曲线

1. 定义

给定 $n+1$ 个控制点 $\{P_0,\ldots,P_n\}$，如下定义B-样条曲线的坐标位置。

$$p(u)=\sum_{k=0}^{n} P_k N_{k,d}(u)$$

其中，$2\le d\le n+1$，参数区间为 $[u_{\min},u_{\max}]$，曲线定义区间为 $[u_{d-1},u_{n+1}]$。

【注】B-样条曲线不一定定义在整个参数区间内。例如，对于给定4个控制点的三次均匀B-样条曲线，节点向量可以设置为 $\{0,1,2,3,4,5,6,7\}$，即参数区间为 $[0,7]$。但曲线定义使用的参数范围是 $[u_{d-1},u_{n+1}]$，即 $[3,4]$。

2. 性质

（1）分段参数多项式。$p(u)$ 是参数 u 的次数不高于 $d-1$ 的分段多项式（$n-d+2$ 段）。每个曲线段受 d 个控制点影响。

（2）连续性。$p(u)$ 具有 C^{d-2} 参数连续性。

9.5.3 均匀B-样条曲线

1. 节点向量的分类

- 均匀的。两相邻节点之间的距离为常数，如{2, 2.5, 3, 3.5, 4}。
- 开放均匀的。除了两端的节点值重复 d 次以外，其余节点间距是均匀的，如{0, 0, 1, 2, 3, 4, 4}（$d=2$，$n=4$）。
- 非均匀的。可以对节点向量和间距指定任何值，如{0, 1, 3, 4, 7}。

2. 均匀B-样条节点向量

（1）一般均匀节点向量。如{-1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2}。

（2）标准均匀节点向量。介于0、1之间，如{0, 0.2, 0.4, 0.6, 0.8, 1}。

（3）自然、方便的均匀节点向量。起始值为0，间距为1。例如，对5个控制点的二次均匀B-样条，有 $n=4$，$d=3$，节点值个数为 $n+d+1=8$，节点向量可以设置为{0, 1, 2, 3, 4, 5, 6, 7}。

3. 开放均匀B-样条节点向量

（1）特点。两端节点值重复 d 次，其余节点间距均匀。

（2）例子。对6个控制点的三次开放均匀B-样条，有 $n=5$，$d=4$，$n+d+1=10$，节点向量可以设置为 $\{0,0,0,0,1,2,3,3,3,3\}$。

（3）节点向量的生成。可用公式

$$u_i=\begin{cases}0 & 0\le i<d\\ i-d+1 & d\le i\le n\\ n-d+2 & i>n\end{cases}$$

生成。其中，$0\le i\le n+d$。

4. 周期性基函数

均匀B-样条有周期性基函数。令 $\Delta u=u_{i+1}-u_i$，则

$$N_{k,d}(u)=N_{k+1,d}(u+\Delta u)\text{ 或 }N_{k,d}(u)=N_{k-1,d}(u-\Delta u)$$

【注】对于开放均匀B-样条，不能利用这种周期性。

9.5.4 B-样条曲线举例

1. 问题

给定3个控制点 $P_0(0,0)$、$P_1(0.5,1)$、$P_2(1,0)$，请构造一条均匀二次B-样条曲线。该曲线图像如图9-7所示。

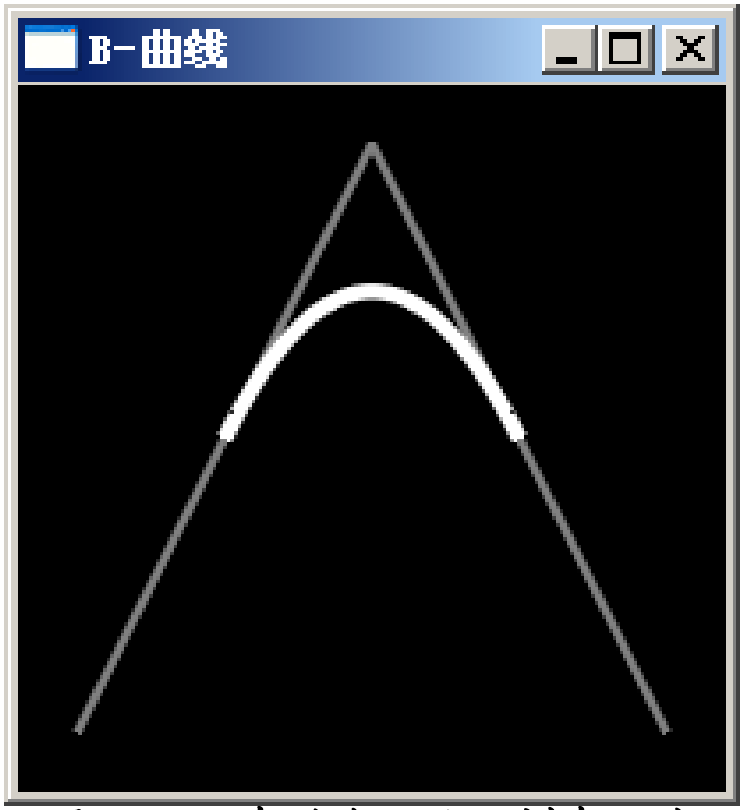


图9-7 一条均匀二次B-样条曲线

2. 解答

使用自然的均匀节点向量，此时参数值 $n=2,d=3$，节点值个数为 $n+d+1=6$，因此设置节点向量为 $\{0,1,2,3,4,5\}$。

首先计算第一个基函数：$N_{03}(u),0\le u<3$。

（1）当 $0\le u<1$ 时，$N_{12}(u)=0,N_{11}(u)=0,N_{01}(u)=1$。

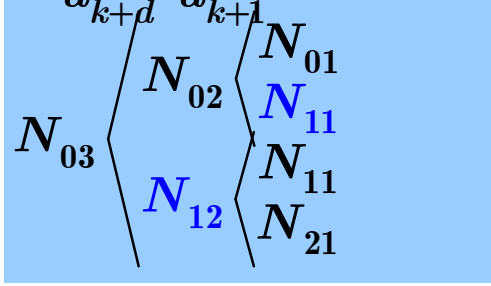
因为

$$N_{02}(u)=\frac{u-u_0}{u_1-u_0}N_{01}(u)=u$$

所以

$$N_{03}(u)=\frac{u-u_0}{u_2-u_0}N_{02}(u)=\frac{1}{2}u^2$$

$$N_{k,d}=\frac{u-u_k}{u_{k+d-1}-u_k}N_{k,d-1}+\frac{u_{k+d}-u}{u_{k+d}-u_{k+1}}N_{k+1,d-1}$$



（2）当 $1 \le u < 2$ 时， $N_{11}(u) = 1, N_{01}(u) = 0$ 。

由周期性，有 $N_{12}(u) = u - 1$ 。

因为

$$N_{02}(u) = \frac{u_2 - u}{u_2 - u_1} N_{11}(u) = 2 - u$$

所以

$$\begin{aligned} N_{03}(u) &= \frac{u - u_0}{u_2 - u_0} N_{02}(u) + \frac{u_3 - u}{u_3 - u_1} N_{12}(u) \\ &= \frac{1}{2} u(2 - u) + \frac{1}{2}(u - 1)(3 - u) \end{aligned}$$

（3）当 $2 \le u < 3$ 时， $N_{02}(u) = 0$ 。

由周期性，有 $N_{12}(u) = 3 - u$ ，所以

$$N_{03}(u) = \frac{u_3 - u}{u_3 - u_1} N_{12}(u) = \frac{1}{2}(3 - u)^2$$

$$N_{k,d} = \frac{u - u_k}{u_{k+d-1} - u_k} N_{k,d-1} + \frac{u_{k+d} - u}{u_{k+d} - u_{k+1}} N_{k+1,d-1}$$

$$N_{03} \left\langle \begin{matrix} N_{02} \\ N_{12} \end{matrix} \right\rangle \left\langle \begin{matrix} N_{01} \\ N_{11} \\ N_{11} \\ N_{21} \end{matrix} \right.$$

总结上述结果得

$$N_{03}(u)=\begin{cases}\dfrac{1}{2}u^2 & 0\le u<1\\ \dfrac{1}{2}u(2-u)+\dfrac{1}{2}(u-1)(3-u) & 1\le u<2\\ \dfrac{1}{2}(3-u)^2 & 2\le u<3\end{cases}$$

然后由周期性计算其余2个基函数。

$$N_{13}(u)=\begin{cases}\frac{1}{2}(u-1)^2 & 1\le u<2\\ \frac{1}{2}(u-1)(3-u)+\frac{1}{2}(u-2)(4-u) & 2\le u<3\\ \frac{1}{2}(4-u)^2 & 3\le u<4\end{cases}$$

$$N_{23}(u)=\begin{cases}\frac{1}{2}(u-2)^2 & 2\le u<3\\ \frac{1}{2}(u-2)(4-u)+\frac{1}{2}(u-3)(5-u) & 3\le u<4\\ \frac{1}{2}(5-u)^2 & 4\le u<5\end{cases}$$

$$N_{03}(u)=\begin{cases}\frac{1}{2}u^2 & 0\le u<1\\ \frac{1}{2}u(2-u)+\frac{1}{2}(u-1)(3-u) & 1\le u<2\\ \frac{1}{2}(3-u)^2 & 2\le u<3\end{cases}$$

多项式曲线 $p(u)$ 的参数范围是：$u_{d-1} \le u \le u_{n+1}$，即 $2 \le u \le 3$，考虑到多项式曲线 $p(u)$ 具有一阶参数连续性（C^{d-2}），可以如下定义 $p(u)$。

当 $2 \le u \le 3$ 时，

$$
\begin{aligned}
p(u) &= P_0 \times \frac{1}{2}(3-u)^2 + P_1 \times \left[\frac{1}{2}(u-1)(3-u) + \frac{1}{2}(u-2)(4-u)\right] + \\
&\quad P_2 \times \frac{1}{2}(u-2)^2 \\
&= (0,0) \times \left(\frac{1}{2}u^2 - 3u + \frac{9}{2}\right) + (0.5,1) \times \left(-u^2 + 5u - \frac{11}{2}\right) + \\
&\quad (1,0) \times \left(\frac{1}{2}u^2 - 2u + 2\right) \\
&= \left(\frac{1}{2}u - \frac{3}{4}, -u^2 + 5u - \frac{11}{2}\right)
\end{aligned}
$$

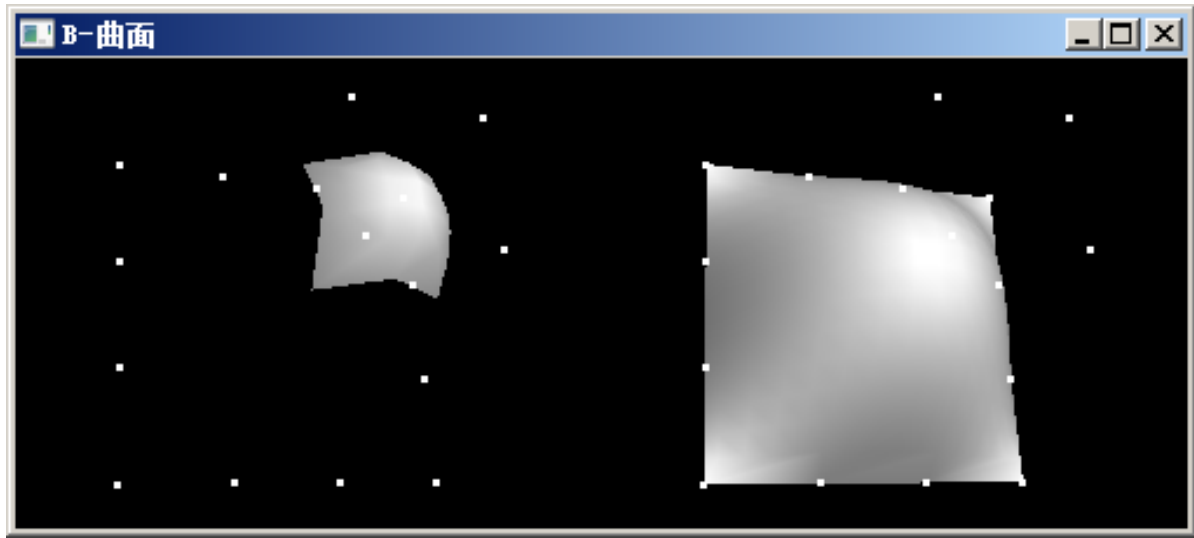
9.5.6 B-样条曲面

如果在某一个空间区域中给定了 $(m+1)\times(n+1)$ 个控制点 $\{P_{ij}\,|\,i=0,\ldots,m,j=0,\ldots,n\}$，则下列形式的参数曲面为 $k\times h$ 阶B-样条曲面。

$$p(u,v)=\sum_{i=0}^{m}\sum_{j=0}^{n}p_{ij}N_{i,k}(u)N_{j,h}(v)$$

其中，$2\le k\le m+1$，$2\le h\le n+1$，参数范围是 $u_{\min}\le u\le u_{\max}$，$v_{\min}\le v\le v_{\max}$，曲面定义范围是 $u_{k-1}\le u\le u_{m+1}$，$v_{h-1}\le v\le v_{n+1}$。

图9-8给出的两个4×4阶B-样条曲面均使用了4×4个控制点。



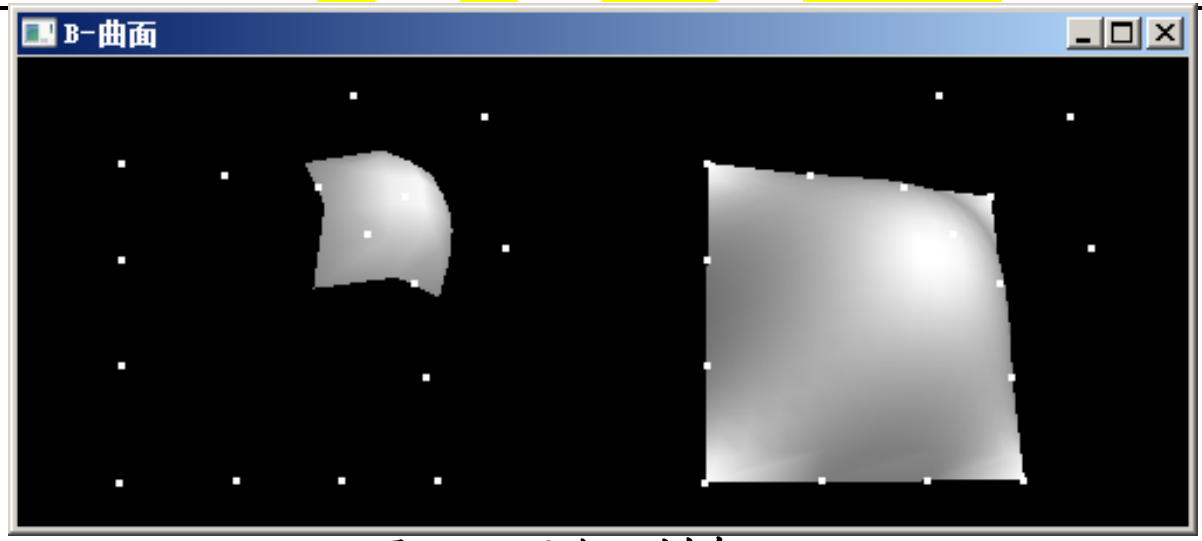


图9-8　两个B-样条曲面

9.5.7 有理B-样条

如果在某一个空间区域中给定了 $n+1$ 个控制点 $\{P_0,\ldots,P_n\}$，则称下列形式的参数曲线为 d 阶有理B-样条曲线。

$$p(u)=\sum_{k=0}^{n}\omega_k P_k N_{k,d}(u)\Big/\sum_{k=0}^{n}\omega_k N_{k,d}(u)$$

其中，$2\le d\le n+1$，参数区间为 $[u_{\min},u_{\max}]$，曲线定义区间为 $[u_{d-1},u_{n+1}]$。ω_k 是控制点的加权因子。ω_k 越大，曲线越靠近控制点 P_k。

一般图形软件包（例如OpenGL）通常使用非均匀节点向量来构造有理B-样条。这种样条称为NURBS（非均匀有理B-样条）。

$$\sum_{k=0}^{n}N_{k,d}(u)=1 \qquad u_{d-1}\le u\le u_{n+1}$$

9.6 B-样条的OpenGL实现※

9.6.1 需要的函数

1. gluNewNurbsRenderer()

【函数原型】GLUnurbs *gluNewNurbsRenderer();

【功能】创建一个NURBS对象，返回指向该对象的指针。

2. gluDeleteNurbsRenderer()

【函数原型】void gluDeleteNurbsRenderer(GLUnurbs *obj);

【功能】删除NURBS对象。

3. gluBeginCurve()

【函数原型】void gluBeginCurve(GLUnurbs *obj);

【功能】开始NURBS曲线的定义。

4. gluEndCurve()

【函数原型】void gluEndCurve(GLUnurbs *obj);

【功能】结束NURBS曲线的定义。

5. gluNurbsCurve()

【函数原型】void gluNurbsCurve(GLUnurbs *obj, GLint nknots, GLfloat *knots, GLint stride, GLfloat *pts, GLint order, GLenum type)

【功能】定义一条NURBS曲线。

【参数】

- obj是NURBS对象。
- nknots和knots是节点值个数和节点向量。
- stride是相邻控制点的偏移量。
- pts是控制点数组，坐标必须与曲线类型一致。
- order是NURBS曲线的阶数（次数+1）。
- type是曲线类型。GL_MAP1_VERTEX_3表示B-曲线，GL_MAP1_VERTEX_4表示有理B-曲线。

$n+d+1$

6. gluBeginSurface()

【函数原型】void gluBeginSurface(GLUnurbs *obj);

【功能】NURBS曲面定义开始。

7. gluEndSurface()

【函数原型】void gluEndSurface(GLUnurbs *obj);

【功能】结束NURBS曲面的定义。

8. gluNurbsSurface()

【函数原型】void gluNurbsSurface(GLUnurbs *obj, GLint nsknots, GLfloat *sknots, GLint ntknots, GLfloat *tknots, GLint sstride, GLint tstride, GLfloat *pts, GLint sorder, GLint torder, GLenum type);

【功能】定义NURBS曲面。

【参数】

- obj是NURBS对象。
- nsknots和sknots是s节点值个数和s节点向量。
- ntknots和tknots是t节点值个数和t节点向量。
- sstride和tstride是相邻行控制点偏移量和相邻控制点偏移量。
- pts是控制点数组，坐标必须与曲线类型一致。
- sorder和torder是s阶数和t阶数。
- type是曲面类型。GL_MAP2_VERTEX_3表示B-曲面，GL_MAP2_VERTEX_4表示有理B-曲面。

$2 \le k \le m+1$，$2 \le h \le n+1$，参数范围是 $u_{\min} \le u \le u_{\max}$，$v_{\min} \le v \le v_{\max}$，曲面定义范围是 $u_{k-1} \le u \le u_{m+1}$，$v_{h-1} \le v \le v_{n+1}$。

9. gluNurbsProperty()

【函数原型】void gluNurbsProperty(GLUnurbs *obj, GLenum property, GLfloat value);

【功能】控制存储在NURBS对象中的属性，这些属性影响NURBS对象的绘制方式。

【说明】这里只介绍property使用GLU_SAMPLING_TOLERANCE的情形，即调用形式为

```
void gluNurbsProperty(
    GLUnurbs *obj,
    GLU_SAMPLING_TOLERANCE,
    GLfloat value
)
```

该调用形式用于将对象obj的采样间隔规定为value像素，即用于表示曲线的折线中每条线段的长度不超过value像素。

9.6.2 实现方法

保存在文件gll.h中。

1. 生成均匀节点向量

```
void gllUniknots(float knots[], int npts, int order)
{   // 节点向量, 控制点个数, 阶数
    int nknots = npts + order; // 节点值个数
    for(int i = 0; i < nknots; i++) knots[i] = i; // 生成均匀节点向量
}
```


$n+d+1$

2. 生成开放均匀节点向量

节点向量的生成公式为

$$u_i = \begin{cases} 0 & 0 \le i < d \\ i - d + 1 & d \le i \le n \\ n - d + 2 & i > n \end{cases}$$

```
void gllOpenUniknots(float knots[], int npts, int order)
{   // 节点向量, 控制点个数, 阶数
    int i, nknots = npts + order; // 节点值个数
    for(i = 0; i < order; i++) knots[i] = 0; // 0≤i<d
    for(i = order; i < npts; i++) knots[i] = i - order + 1; // d≤i≤n
    for(i = npts; i < nknots; i++) knots[i] = npts - order + 1; // i>n
}
```

3. 绘制B-样条曲线

```
void gllBCurve(float knots[], int npts, float pts[], int order, float value)
{   // 节点向量, 控制点个数, 控制点数组, 阶数, 采样间隔(像素)
    int nknots = npts + order; // 节点值个数
    GLUnurbsObj *nobj = gluNewNurbsRenderer(); // 创建NURBS对象
    gluNurbsProperty(nobj, GLU_SAMPLING_TOLERANCE, value);
    // 采样间隔为value像素
    gluBeginCurve(nobj); // 开始NURBS曲线的定义
    gluNurbsCurve(nobj, nknots, knots, 3, pts, order, GL_MAP1_VERTEX_3);
    // NURBS曲线(NURBS对象, 节点值个数, 节点向量, 相邻控制点距离,
    // 控制点数组, 阶数, 曲线类型)
    // GL_MAP1_VERTEX_3为B曲线, GL_MAP1_ VERTEX_4为有理B曲线
    gluEndCurve(nobj); // 结束NURBS曲线的定义
    gluDeleteNurbsRenderer(nobj); // 删除NURBS对象
}
```


$n+d+1$

4. 绘制均匀B-样条曲线

```
void gllUniBCurve(int npts, float pts[], int order, float value)
{   // 控制点个数, 控制点数组, 阶数, 采样间隔(像素)
    float knots[npts + order]; // 节点向量
    gllUniknots(knots, npts, order); // 生成均匀节点向量
    gllBCurve(knots, npts, pts, order, value); // 绘制B样条曲线
}
```

5. 绘制开放均匀B-样条曲线

```
void gllOpenUniBCurve(int npts, float pts[], int order, float value)
{   // 控制点个数, 控制点数组, 阶数, 采样间隔(像素)
    float knots[npts + order]; // 节点向量
    gllOpenUniknots(knots, npts, order); // 生成开放均匀节点向量
    gllBCurve(knots, npts, pts, order, value); // 绘制B样条曲线
}
```


$n+d+1$

6. 绘制B-样条曲面

```
void gllBSurface(float sknot[],float tknot[], int nspts, int ntpts, float pts[],
                 int sorder, int torder)
{   // s节点向量, t节点向量, 控制点行数, 控制点列数, 控制点数组,
    // s阶数, t阶数
    int nsknots = nspts + sorder; // s节点值个数
    int ntknots = ntpts + torder; // t节点值个数
    GLUnurbsObj *nobj = gluNewNurbsRenderer(); // 创建NURBS对象
    gluBeginSurface(nobj); // 曲面定义开始
    gluNurbsSurface(nobj, nsknots, sknot, ntknots, tknot, 3 * ntpts, 3, pts,
            sorder, torder, GL_MAP2_VERTEX_3);
    // B曲面(NURBS对象, s节点值个数, s节点向量, t节点值个数, t节点向量,
    // 相邻行控制点距离, 相邻控制点距离, 控制点数组, s阶数, t阶数,
    // 曲面类型)
    // GL_MAP2_VERTEX_3为B曲面, GL_MAP2_VERTEX_4为有理B曲面
    gluEndSurface(nobj); // 结束曲面的绘制
    gluDeleteNurbsRenderer(nobj); // 删除NURBS对象
}
```

7. 绘制均匀B-样条曲面

```
void gllUniBSurface(int nspts, int ntpts, float pts[], int sorder, int torder)
{   // 控制点行数, 控制点列数, 控制点数组, s阶数, t阶数
    float sknot[nspts + sorder], tknot[ntpts + torder]; // s节点向量, t节点向量
    gllUniknots(sknot, nspts, sorder); // 生成均匀s节点向量
    gllUniknots(tknot, ntpts, torder); // 生成均匀t节点向量
    gllBSurface(sknot, tknot, nspts, ntpts, pts, sorder, torder); // 绘制B-曲面
}
```

8. 绘制开放均匀B-样条曲面

```
void gllOpenUniBSurface(int nspts, int ntpts, float pts[], int sorder, int torder)
{   // 控制点行数, 控制点列数, 控制点数组, s阶数, t阶数
    float sknot[nspts + sorder], tknot[ntpts + torder]; // s节点向量, t节点向量
    gllOpenUniknots(sknot, nspts, sorder); // 生成开放均匀s节点向量
    gllOpenUniknots(tknot, ntpts, torder); // 生成开放均匀t节点向量
    gllBSurface(sknot, tknot, nspts, ntpts, pts, sorder, torder); // 绘制B-曲面
}
```

9.6.3 举例说明

1. B-样条曲线

下列程序绘制了一条均匀B-样条曲线和一条开放均匀B-样条曲线；并绘制了它们的控制点以比较两者的不同。运行结果如图9-9所示。

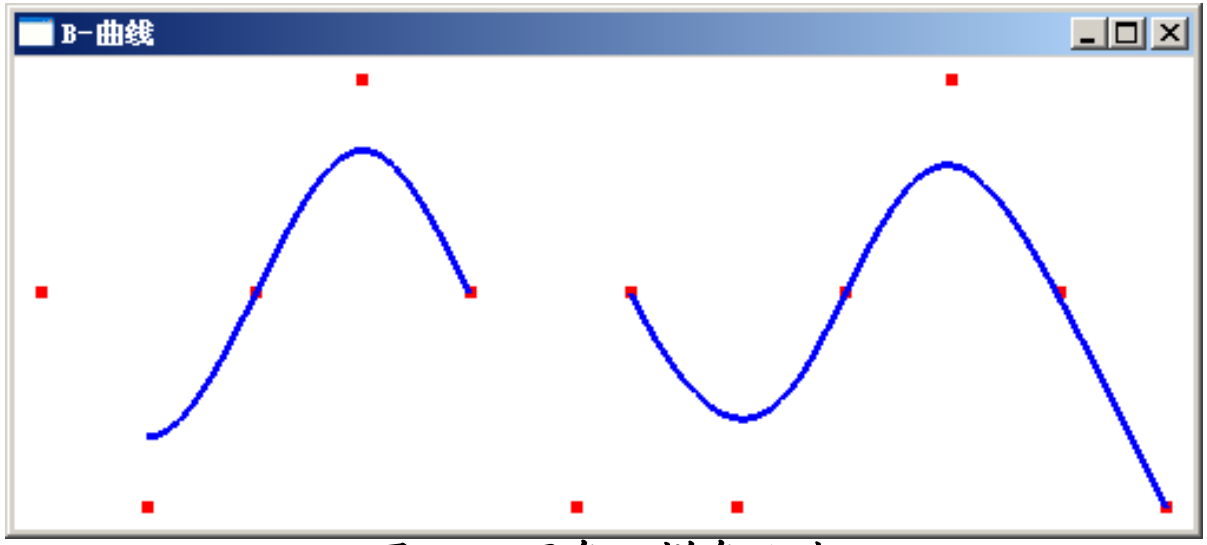


图9-9 两条B-样条曲线

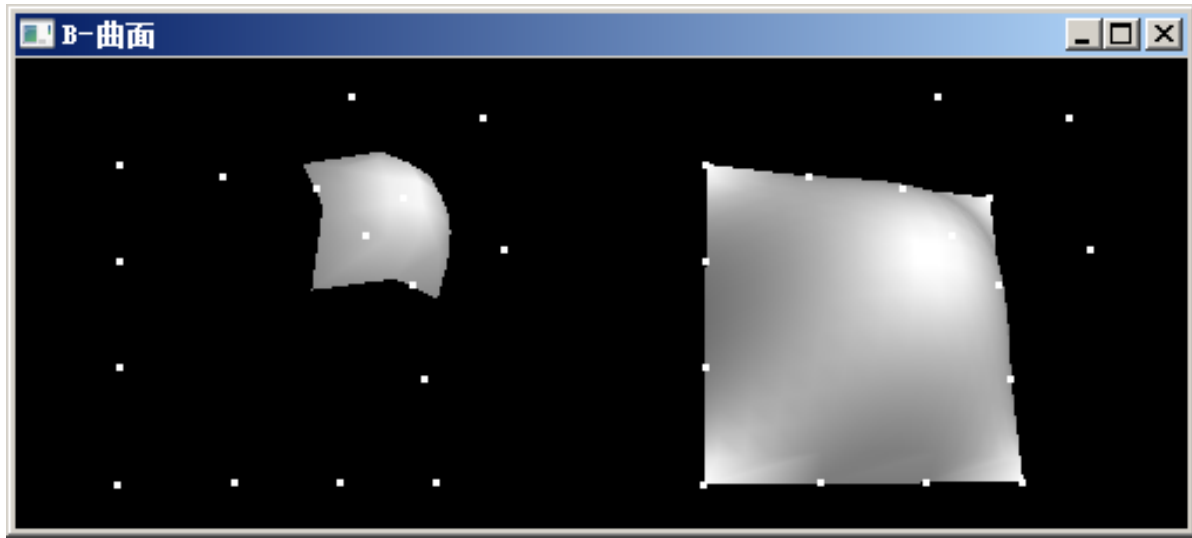
```
// BCurve.c
#include"gll.h"
void Viewport(int x, int y, int w, int h) // 创建视口，指定投影变换
{   glViewport(x, y, w, h); // 根据指定位置和大小创建视口
    glMatrixMode(GL_PROJECTION); // 投影矩阵栈
    glLoadIdentity(); // 当前矩阵改为单位矩阵
    gluOrtho2D(-0.25, 5.25, -2.2, 2.2); // 裁剪窗口，比[0, 5][-2, 2]稍大
    glMatrixMode(GL_MODELVIEW); // 视图造型矩阵栈
    glLoadIdentity(); // 当前矩阵改为单位矩阵
}
```

```
void Paint() // 场景绘制函数
{   glClearColor(1, 1, 1, 1); // 白色背景
    glClear(GL_COLOR_BUFFER_BIT); // 清除颜色缓冲
    int w = glutGet(GLUT_WINDOW_WIDTH) / 2; // 视口宽度
    int h = glutGet(GLUT_WINDOW_HEIGHT); // 视口高度
    float pts[][3] = // 控制点坐标
    {{0, 0, 0}, {1, -2, 0}, {2, 0, 0}, {3, 2, 0}, {4, 0, 0}, {5, -2, 0}};
    int npts = 6, order = 4; // 控制点个数,阶数
    Viewport(0, 0, w, h); // 左视区，均匀B-曲线
    glColor3f(1, 0, 0), glPointSize(5); // 控制点颜色和大小
    gllPoints((float *)pts, npts); // 控制点
    glColor3f(0, 0, 1), glLineWidth(2.5); // 曲线颜色和线宽
    gllUniBCurve(npts, (float *)pts, order, 1); // 均匀B-曲线
    Viewport(w, 0, w, h); // 右视区，开放均匀B-曲线
    glColor3f(1, 0, 0), glPointSize(5); // 控制点颜色和大小
    gllPoints((float *)pts, npts); // 控制点
    glColor3f(0, 0, 1), glLineWidth(2.5); // 曲线颜色和线宽
    gllOpenUniBCurve(npts, (float *)pts, order, 1); // 开放均匀B-曲线
    glFlush();
}
```

```
int main(int argc, char *argv[])
{   glutInit(&argc, argv);
    glutInitWindowSize(500, 200); // 程序窗口大小(5 : 2)
    glutCreateWindow("B-曲线");
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

2. B-样条曲面

下列程序绘制了一个均匀B-样条曲面和一个开放均匀B-样条曲面；并绘制了它们的控制点以比较两者的不同。运行结果如图9-8所示。



```
// BSurf.c
#include"gll.h"
void Viewport(int x, int y, int w, int h) // 创建视口，指定投影变换
{   glViewport(x, y, w, h); // 根据指定位置和大小创建视口
    glMatrixMode(GL_PROJECTION); // 投影矩阵栈
    glLoadIdentity(); // 当前矩阵改为单位矩阵
    // 透视变换(视角30度，宽高比，近平面，远平面)
    gluPerspective(30, (float)w / h, 1, 1000);
    glTranslatef(0.5, 0.5, -14); // 场景右移0.5，上移0.5，远移14
    glRotatef(30, -1, 1, 0); // 场景绕(-1, 1, 0)方向旋转30度
    glMatrixMode(GL_MODELVIEW); // 视图造型矩阵栈
    glLoadIdentity(); // 当前矩阵改为单位矩阵
}
```

```
void Hint() // 初始化(材质属性和光照属性等)
{   glPointSize(3); // 点的大小
    glMaterialfv(GL_FRONT, GL_DIFFUSE, (float[]) {1, 1, 1, 1}); // 白色材质
    glEnable(GL_AUTO_NORMAL); // 自动计算表面法向量
    glEnable(GL_NORMALIZE); // 自动单位化法向量
    glEnable(GL_LIGHT0); // 启用0号光源
    glEnable(GL_LIGHTING); // 启用光照效果
}
```

```
void Paint() // 场景绘制函数
{   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    // 清除颜色缓存和深度缓存
    int w = glutGet(GLUT_WINDOW_WIDTH) / 2; // 视口宽度
    int h = glutGet(GLUT_WINDOW_HEIGHT); // 视口高度
    float pts[4][4][3] = // 控制点坐标
    {   {{ -3, -3, -3}, { -3, -1, -3}, { -3, 1, -3}, { -3, 3, -3}},
        {{ -1, -3, -3}, { -1, -1, 3}, { -1, 1, 3}, { -1, 3, -3}},
        {{1, -3, -3}, {1, -1, 3}, {1, 1, 3}, {1, 3, -3}},
        {{3, -3, -3}, {3, -1, -3}, {3, 1, -3}, {3, 3, -3}}
    };
```

```
    Viewport(0, 0, w, h); // 左侧, 均匀B-曲面
    gllUniBSurface(4, 4, (float *)pts, 4, 4); // 均匀B-曲面
    gllPoints((float *)pts, 4 * 4); // 绘制控制点
    Viewport(w, 0, w, h); // 右侧, 开放均匀B-曲面
    gllOpenUniBSurface(4, 4, (float *)pts, 4, 4); // 开放均匀B-曲面
    gllPoints((float *)pts, 4 * 4); // 绘制控制点
    glFlush();
}
int main(int argc, char *argv[])
{   glutInit(&argc, argv);
    glutInitWindowSize(500, 200); // 程序窗口大小(5 : 2)
    glutCreateWindow("B-曲面");
    Hint(); // 初始化(材质属性和光照属性等)
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

9.7 练习题

9.7.1 基础知识题

1. 请指出插值样条和逼近样条的区别。

2. 假设在控制点p_k和p_{k+1}之间的曲线段是参数三次函数$p(u)$，Hermite曲线段的边界条件是什么？请解释所使用符号的含义。

3. 请写出Bézier样条曲线基函数的定义。

4. 请写出B-样条曲线基函数的定义。

5. 使用Hermite方法求$P_0(0, 0)$和$P_1(5, 6)$之间的曲线段$p(u)$（$0 \le u \le 1$），并计算参数值为0.5时的结果，其中$p(0)=(0, 0)$，$p(1)=(5, 6)$，$p'(0)=(3, 8)$，$p'(1)=(5, 1)$。

6. 给定控制点$A(0,, 0, 0)$、$B(1, 1, 1)$、$C(2, -1, -1)$和$D(3, 0, 0)$，请构造一条三次Bézier曲线，并计算参数为0、1/3、2/3和1时的值。

7. 给定控制点$A(0, 0, 0)$、$B(50, 60, 0)$和$C(100, 10, 0)$，请构造一条二次均匀B-样条曲线。

8 给定控制点$A(0, 0, 0)$、$B(1, 0, 1)$、$C(2, 0, 0)$和$D(3, 0, 0)$，请使用边界条件定义的方法构造一条三次均匀B-样条曲线，并计算参数为0、1/3、2/3和1时的值。

9.7.2 阶段实习题

1. 编写1个程序绘制由四个控制点生成的三次Bézier曲线。

2. 编写1个程序绘制由四个控制点生成的三次均匀B-样条曲线。

3. 任意给定至少4个控制点，编写1个函数绘制由这些控制点生成的三次均匀B-样条曲线。完成以后，分别指定5个控制点、8个控制点和13个控制点，使用该函数完成这3条三次均匀B-样条曲线的绘制。

4. 任意给定至少4×4个控制点，编写1个函数绘制由这些控制点生成的三次均匀B-样条曲面。完成以后，分别指定5×4个控制点、8×7个控制点和13×10个控制点，使用该函数完成这3个三次均匀B-样条曲面的绘制。

5. 构造三次样条函数$s(x)$去模拟一只飞鸟外形的上部，测得的数据如下。

x	0.9	1.3	1.9	2.1	2.6	3.0	3.9	4.4	4.7	5.0	6.0
y	1.3	1.5	1.85	2.1	2.6	2.7	2.4	2.15	2.05	2.1	2.25
x	7.0	8.0	9.2	10.5	11.3	11.6	12.0	12.6	13.0	13.3	
y	2.3	2.25	1.95	1.4	0.9	0.7	0.6	0.5	0.4	0.25	

边界条件为自然边界条件，即$s''(0.9)=s''(13.3)=0$。请使用OpenGL画出该图形，要求至少按照300×300分辨率绘制。