Chương 5
Hiển thị các đối tượng hình học trong không gian ba chiều

Trong đồ hoạ máy tính khi muốn xây dựng một đường cong tổng quát khi chưa biết phương trình toán học của nó người ta sử dụng một tập hợp các điểm điều khiển cho trước (control points). Giả sử ta dùng n+1 điểm điều khiển P_0, P_1, P_2,..., P_n, khi đó một đường cong C được tạo ra theo một trong hai cách sau:

- Nội suy các điểm điều khiển: C bắt đầu tại P0 và đi qua các điểm điều khiển trung gian theo thứ tự P_0, P_1, P_2,..., P_n. C kết thúc tại P_n
- Xấp xỉ các điểm điều khiển: C không nhất thiết phải đi qua các điểm điều khiển nhưng hình dạng của nó được quyết định bởi các điểm điều khiển.

Trong chương này ta sẽ nghiên cứu hai phương pháp cho phép vẽ các đường và mặt cong trong không gian đó là Bezier và B-Spline. Cả hai phương pháp đều sử dụng cách lấy xấp xỉ các điểm điều khiển.

1. Đường cong Bezier

Lí thuyết đường cong và mặt Bezier được phát minh bởi một kĩ sư người Pháp có tên là Pierre Bézier trong quá trình thiết kế mẫu xe ôtô. Điểm mạnh của lí thuyết này là tính dễ dàng và thuận tiện trong việc biểu diễn các đường cong và mặt cong.

1.1 Cách xây dựng đường cong Bezier

Giả sử một đường cong Bezier C được tạo ra từ n+1 điểm điều khiển P0, P1,..., Pn, kí hiệu toạ độ của mỗi điểm điều khiển là $Pk(x_k, y_k, z_k)$ trong đó $0 \le k \le n$. Tập hợp các điểm điều khiển ta gọi là đa giác điều khiển (control polygon). Khi đó các điểm trên đường cong Bezier C được tính theo công thức:

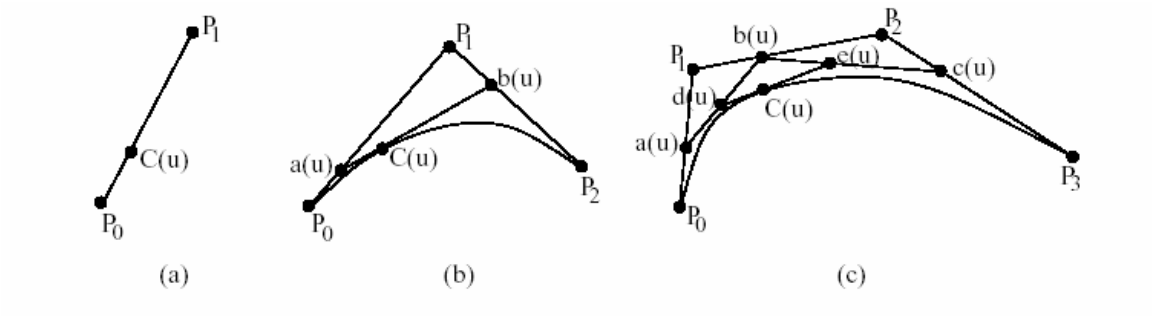
$$C(u)=\sum_{k=0}^{n} BEZ_{k,n}(u)\, P_k, \qquad (0 \le u \le 1)$$

Trong đó hàm $BEZ_{k,n}(u)=C(n,k)u^k(1-u)^{n-k}=\frac{n!}{k!(n-k)!}u^k(1-u)^{n-k}$

- Để dễ hình dung ta xét trường hợp đơn giản nhất khi chỉ có 2 điểm điều khiển P_0 và P_1 khi đó các điểm thuộc đường cong C được xác định bởi:

$C(u)=BEZ_{0,1}.P_0 + BEZ_{1,1}.P_1=(1-u)P_0 + uP_1 \qquad (0 \le u \le 1)$

Đường cong C lúc này chính là đoạn thẳng P_0P_1 như trong hình vẽ 5.1(a)



Hình 5.1: Đường cong Bezier

Ta thấy C(u) là tuyến tính theo tham số u và ta gọi đó là đường cong Bezier bậc 1

- Trường hợp có 3 điểm điều khiển P_0, P_1, P_2 như trong hình 5.1(b), ta dễ dàng tính được:

$BEZ_{0,2}=(1-u)^2$

$BEZ_{1,2}=2(1-u)u$

$BEZ_{2,2}=u^2$

Do đó phương trình của C là:

$$C(u)=(1-u)^2P_0 + 2(1-u)uP_1 + u^2P_2$$

C(u) lúc này được gọi là đường cong Bezier bậc 2

Công thức trên còn được xây dựng một cách tuần tự như sau:

+ Với mỗi giá trị $0\le u\le 1$ ta tính giá trị a(u) giữa hai điểm P_0 và P_1

$a(u)=(1-u)P_0 + uP_1$

+ Tính b(u) giữa hai điểm P_1 và P_2

$b(u)=(1-u)P_1 + uP_2$

+ Cuối cùng tính C(u) giữa hai điểm a(u) và b(u)

$C(u)=(1-u)a(u)+ub(u)= (1-u)^2P_0 + 2(1-u)uP_1 + u^2P_2$

- Tương tự khi có 4 điểm điều khiển P_0, P_1, P_2, P_3 như trong hình 5.1(c), ta tính được:

$BEZ_{0,3}=(1-u)^3$

$BEZ_{1,3}=3(1-u)^2u$

$BEZ_{2,3}=3(1-u)u^2$

$BEZ_{3,3}=u^3$

Do đó phương trình của C là:

$$C(u)=(1-u)^3P_0 + 3(1-u)^2uP_1 + 3(1-u)u^2P_2 +u^3P_3$$

C(u) là một hàm bậc 3 theo biến u và được gọi là đường cong Bezier bậc 3

Công thức trên còn có thể xây dựng một cách tuần tự như mô tả trong hình 5.1(c).

1.2 Tính chất của đường cong Bezier

- Từ công thức xây dựng đường cong Bezier ta dễ dàng nhận xét:

$C(0)=P_0$

$C(1)=P_n$

Do đó P_0 và P_n chính là điểm đầu và điểm cuối của đường cong

- Nếu lấy đạo hàm bậc nhất của C(u) tại điểm đầu và điểm cuối ta có

$C'(0)=-nP_0 + nP_1$

$C'(1)=-nP_{n-1} + nP_n$

Do đó độ dốc tại điểm đầu của C(u) nằm dọc theo đường thẳng qua hai điểm điều khiển đầu tiên và độ dốc tại điểm cuối của C(u) nằm dọc theo đường thẳng qua hai điểm điều khiển cuối cùng.

- Vì $\sum_{k=0}^{n} BEZ_{k,n}(u)=1$ và $BEZ_{k,n}(u)\ge 0$ do đó mọi điểm nằm trên C(u) đều thuộc tập lồi các điểm điều khiển (convex hull of the control points)

- Bậc của đường cong tăng cùng với số điểm điều khiển, cụ thể đường cong Bezier C(u) với n+1 điểm điều khiển là một đa thức bậc n của u. Do đó khi số điểm điều khiển lớn quá trình tính toán sẽ phức tạp.

1.3 Lệnh OpenGL hiển thị đường cong Bezier

- Trước hết ta phải định nghĩa hàm C(u) bằng lệnh:

glMap1*{fd}(GLenum target, TYPE u1, TYPE u2, GLint stride,GLint order,*
*const TYPE*points);*

Trong đó *target* chỉ định kiểu dữ liệu của điểm điều khiển, có thể nhận giá trị GL_MAP1_VERTEX_3 hoặc GL_MAP1_VERTEX_4. Hai tham số *u1* và *u2* chỉ khoảng giá trị của *u*. Tham số *stride* chỉ khoảng cách (giá trị floating-point) giữa hai số liệu điểm điều khiển liên tiếp. Tham số *order* có giá trị bằng bậc của đường Bezier cộng thêm 1. Tham số *point* chỉ đến mảng chứa các điểm điều khiển.

Tiếp theo đó sử dụng lệnh **glEnable(***GLenum target***)** để kích hoạt C(u)

- Sau khi định nghĩa hàm C(u) ta dùng lệnh **glEvalCoord1f(**u**) t**hay cho lệnh **glVertex()** để hiển thị các điểm của đường cong Bezier

Ví dụ 1: Thực hiện chương trình bezcurve.c

Ví dụ 2: Thực hiện chương trình bezcurveMesh.c

Chương trình này giống với chương trình ngoại trừ câu lệnh glMapGrid1f(30,0.0,1.0), lệnh này chia khoảng giá trị của tham số u[0,1] thành 30 khoảng bằng nhau. Sau đó ta sử dụng lệnh glEvalMesh1(GL_LINE, 0, 30) thay thế cho lệnh **glEvalCoord1f()** khi hiển thị đường cong.

Ví dụ 3: Thực hiện chương trình bezcurve2.c

2. Mặt cong Bezier

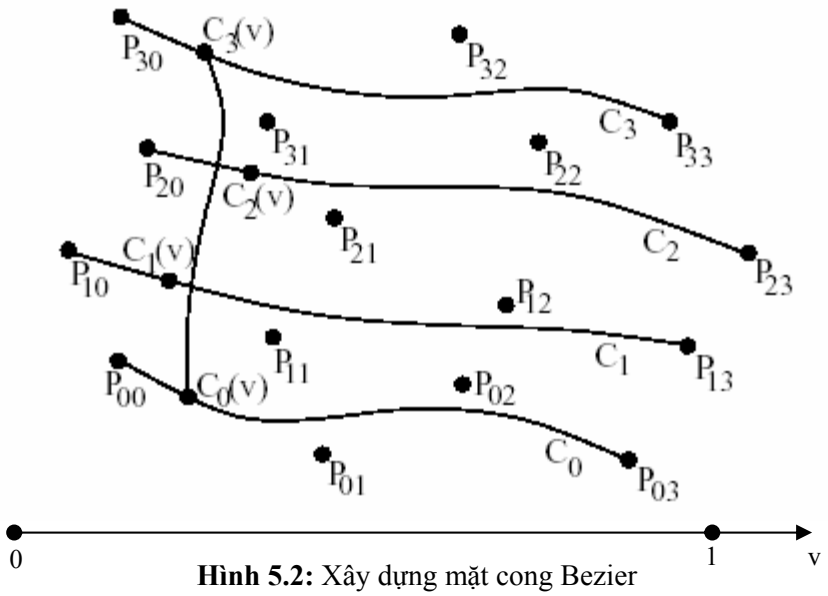
2.1 Cách xây dựng mặt cong Bezier

Mặt cong Bezier là mở rộng của đường cong Bezier. Giả sử ta có một mảng (n+1)x(m+1) các điểm điều khiển P_{ij}, $0\le i\le n$, $0\le j\le m$ ta gọi đó là khối đa diện điều khiển (control polyhedron). Khi đó các điểm trên mặt cong Bezier S được tính theo công thức:

$$S(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m} BEZ_{i,n}(u)BEZ_{j,m}(v)\,P_{ij}, \qquad (0\le u,v\le 1)$$

Để có mặt cong Bezier từ các đường cong Bezier ta hãy coi mảng (n+1)x(m+1) các điểm điều khiển P_{ij} như là n+1 mảng một chiều khác nhau, mỗi mảng gồm m+1 điểm điều khiển. Xây dựng đường cong Bezier từ n+1 mảng điểm điều khiển đó ta sẽ được n+1 đường cong Bezier. Ta kí hiệu đường cong ứng với mảng điểm điều khiển thứ i là C_i, $0\le i\le n$ và phương trình tham số của C_i là $C_i(v)$, $0\le v\le 1$.

Nói cách khác với mỗi giá trị $0\le v\le 1$ ta có n+1 điểm nằm tương ứng trên các đường cong C_i, ta kí hiệu tập các điểm đó là $\{C_i(v)\}_{i=0..n}$. Nếu ta tiếp tục sử dụng n+1 điểm đó làm đa giác điều khiển ta sẽ thu được một đường cong Bezier. Tưởng tượng khi v tăng từ 0 đến 1 ta sẽ được một lưới các đường cong Bezier, đó chính là mặt cong Bezier như minh hoạ trong hình 5.2.



Hình 5.2: Xây dựng mặt cong Bezier

2.2 Lệnh OpenGL hiển thị mặt cong Bezier

Ta sử dụng hai lệnh glMap2*() và glEvalCoord2*() để định nghĩa và hiển thị mặt cong Bezier.

glMap2*{fd}(GLenum target, TYPE u1, TYPE u2, GLint ustride,*
GLint uorder, TYPE v1, TYPE v2, GLint vstride,
GLint vorder, TYPE points);

Trong đó target giống như trong lệnh **glMap2*()** và được sử dụng trong lệnh **glEnable()**. Hai cặp giá trị (*u1*,*u2*) và (*v1*,*v2*) chỉ định miền giá trị của hai tham số *u* và *v*. Tham số *stride* chỉ khoảng cách (giá trị floating-point) giữa hai số liệu điểm điều khiển liên tiếp. Tham số *order* có giá trị bằng bậc của đường Bezier cộng thêm 1. Tham số *point* chỉ đến mảng chứa các điểm điều khiển. Hai tham số *ustride và vstride* xác định khoảng cách giữa hai điểm điều khiển xét theo tham biến *u* và tham biến *v*.Hai tham số *uorder* và *vorder* có giá trị bằng bậc của đường Bezier xét theo lần lượt tham số *u* và *v* cộng thêm 1. Tham số *point* chỉ đến mảng chứa các điểm điều khiển.

glEvalCoord2*{fd}(TYPE u, TYPE v);*

Cho phép hiển thị các điểm của mặt cong thay cho lệnh glVertex*().

Ví dụ 1: Thực hiện chương trình bezsurf.c
Ví dụ 2: Thực hiện chương trình bezmesh.c
Ví dụ 3: Thực hiện chương trình bezboat.c

3. Đường cong B-spline

Giống như đường cong và mặt Bezier, đường cong và mặt B-spline cũng sử dụng phương pháp xấp xỉ các điểm điều khiển.

Điểm mạnh của phương pháp B-spline so với phương pháp Bezier đó là:

- Bậc của đa thức B-spline có thể thiệt lập một cách độc lập với số lượng các điểm điều khiển.

- B-spline cho phép điều khiển cục bộ (local control) nghĩa là khi ta thay đổi vị trí một điểm điều khiển thì vị trí hai điểm điều khiển liền kề sẽ thay đổi tương ứng giúp ta có thể điều chỉnh một phần nào đó của đường cong và mặt cong. Ngược lại với phương pháp Bezier khi ta thay đổi vị trí một điểm điều khiển, hình dáng của cả đường cong hoặc mặt cong sẽ bị thay đổi điều đó gây khó khăn khi ta muốn chỉnh sửa hình vẽ.

Điểm hạn chế của phương pháp B-spline là nó phức tạp hơn phương pháp Bezier.

3.1 Cách xây dựng đường cong B-spline

Giả sử ta có n+1 điểm điều khiển P0, P1,..., Pn, kí hiệu toạ độ của mỗi điểm điều khiển là $P_i(x_i, y_i, z_i)$ trong đó $0 \le i \le n$. Tập hợp các điểm điều khiển ta gọi là đa giác điều khiển (control polygon). Khi đó các điểm trên đường cong B-Spline được tính theo công thức:

$$C(u)=\sum_{i=0}^{n} N_{i,m}(u)\, P_i, \quad t_{min} \le u \le t_{max}, \quad 2 \le m \le n+1$$

Ta có thể lựa chọn miền giá trị của tham số u. Hàm $N_{i,m}(u)$ được gọi là hàm B-spline là một đa thức có bậc là m-1. Giá trị của tham số m có thể chọn là một trong số các giá trị từ 2 đến n+1. Trong thực tế ta có thể thiết lập m=1 nhưng khi đó chỉ hiển thị các điểm điều khiển.

Trước khi định nghĩa hàm $N_{i,m}(u)$ ta phải xây dựng các khoảng giá trị của tham biến u, hàm $N_{i,m}(u)$ sẽ được định nghĩa trên từng khoảng đó. Muốn vậy ta định nghĩa r+1 điểm chia $t_0 \le t_1 \le ... \le t_r$, mỗi điểm như vậy được gọi là một điểm nút. Tập hợp các điểm nút $T=\{t_0, t_1, ..., t_r\}$ được gọi là Vecto các điểm nút (knot vector).

Các điểm nút tạo thành một dãy số không giảm và có thể một vài điểm nút có giá trị bằng nhau.

Hàm $N_{i,m}(u)$ được định nghĩa một cách đệ quy theo m như sau:

$$N_{i,1}(u) = \begin{cases} 1 & t_i \le u \le t_{i+1} \\ 0 & u \notin [t_i, t_{i+1}] \end{cases}$$

Các hàm B-spline với m>1 được định nghĩa bởi công thức sau

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

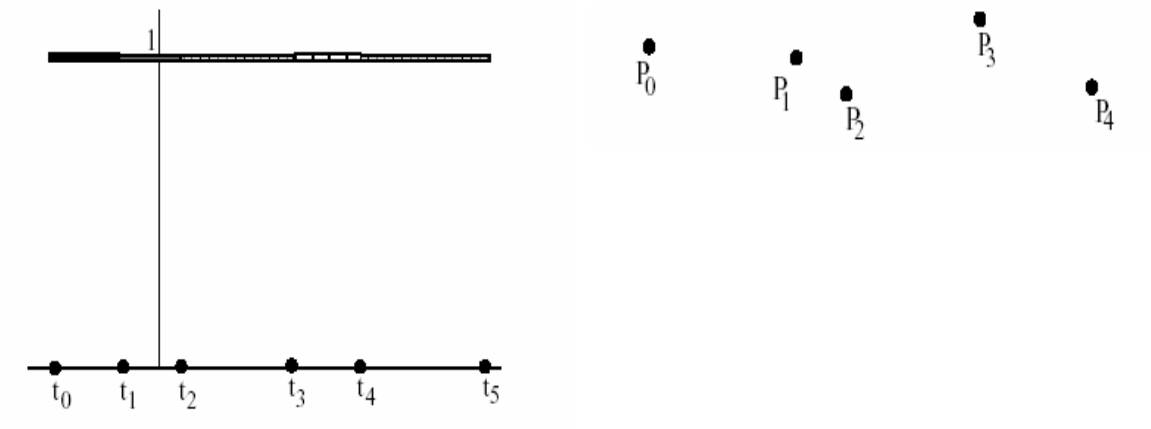
Nhìn vào công thức tính trên ta thấy để tính được $N_{i,m}(u)$ ta cần các nút $t_0, t_1, ..., t_{i+m}$ trong vectơ nút. Vậy khi i=n ta cần $t_0, t_1, ..., t_{n+m}$ trong véc tơ nút, chính vì lí do đó mà ta phải chọn từ đầu vectơ nút sao cho khoảng giá trị của tham số u được chia thành n+m khoảng bởi n+m+1 điểm chia hay nói cách khác r=n+m.

Để dễ hình dung cách xây dựng đường cong B-spline ta xét khi m=1,2,3.

+ Khi m=1, hàm B-spline $N_{i,1}$ sẽ có bậc bằng 0, phương trình đường cong B-spline có dạng:

$$C(u)=\sum_{i=0}^{n} N_{i,1}(u)\, P_i = N_{0,1}P_0 + N_{1,1}P_1 + ... + N_{n,1}P_n \quad (t_0 \le u \le t_{n+1})$$

Theo định nghĩa ở trên ta có khi $t_0 \le u \le t_1$ chỉ có duy nhất hàm $N_{0,1}=1$ còn các hàm B-spline khác đều bằng 0 do đó $C(u)=P_0$. Tương tự như vậy khi xét lần lượt các khoảng của tham số u ta thấy trên khoảng $[t_i, t_{i+1}]$ chỉ có duy nhất hàm $N_{i,1}$ có giá trị bằng 1, còn các hàm B-spline khác có giá trị bằng 0. Vậy khi m=1 ta có đường cong C(u) chính là các điểm điều khiển rời rạc. Hình 5.3 dưới đây minh hoạ đồ thị của các hàm $N_{i,1}$ ($0 \le i \le 4$) và đường cong C(u).



Hình 5.3: Đồ thị các hàm B-spline $N_{i,1}$ và đường cong C(u) là các điểm điều khiển

+ Khi m=2, hàm B-spline $N_{i,2}$ sẽ có bậc bằng 1, phương trình đường cong B-spline có dạng:

$C(u)=\sum_{i=0}^{n} N_{i,2}(u)\, P_i = N_{0,2}P_0 + N_{1,2}P_1 + ...+ N_{n,2}P_n \quad (t_1 \le u \le t_{n+1})$

Ta xét hàm B-spline đầu tiên $N_{0,2}$

$$N_{0,2}(u)=\frac{u-t_0}{t_1-t_0}N_{0,1}(u)+\frac{t_2-u}{t_2-t_1}N_{1,1}(u)$$

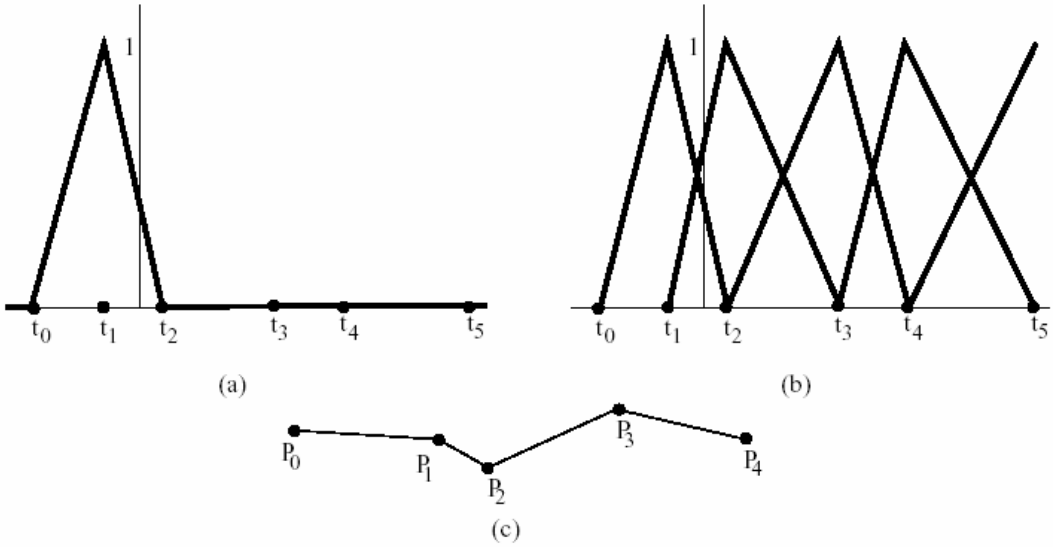
Ta nhận thấy $N_{0,2}$ được tính dựa vào $N_{0,1}$ và $N_{1,1}$. Hệ số của $N_{0,1}$ là $\frac{u-t_0}{t_1-t_0}$, đây là phương trình của một hàm số bậc nhất tăng trên đoạn $[t_0,t_1]$, giá trị của hàm bằng 0 khi $u=t_0$ và bằng 1 khi $u=t_1$ (ta gọi đây là nửa bên trái của $N_{0,2}$). Tương tự hệ số của $N_{1,1}$ là một hàm số bậc nhất giảm trên đoạn $[t_1,t_2]$, giá trị của hàm bằng 1 khi $u=t_1$ và bằng 0 khi $u=t_2$ (ta gọi đây là nửa bên phải của $N_{0,2}$). Phương trình của $N_{0,2}(u)$ có thể viết lại như sau:

$$N_{0,2}(u)=\begin{cases} 0 & u \le t_0 \\ \dfrac{u-t_0}{t_1-t_0} & t_0 \le u \le t_1 \\ \dfrac{t_2-u}{t_2-t_1} & t_1 \le u \le t_2 \\ 0 & u \ge t_2 \end{cases}$$

Tổng quát ta có công thức sau:

$$N_{i,2}(u)=\begin{cases} 0 & u \le t_i \\ \dfrac{u-t_i}{t_{i+1}-t_i} & t_i \le u \le t_{i+1} \\ \dfrac{t_{i+2}-u}{t_{i+2}-t_{i+1}} & t_{i+1} \le u \le t_{i+2} \\ 0 & u \ge t_{i+2} \end{cases}$$

Hình 5.4(a) minh hoạ đồ thị của hàm $N_{0,2}$, hình 5.4(b) minh hoạ đồ thị các hàm $N_{i,2}$ $(0 \le i \le 3)$



Hình 5.4

Ta nhận thấy rằng trên đoạn $[t_i,t_{i+1}]$ $(1 \le i \le n)$ có hai nửa đồ thị, nửa bên trái của $N_{i,2}$ và nửa bên phải của $N_{i-1,2}$. Do đó phương trình của C(u) trên đoạn $[t_i,t_{i+1}]$ là:

$$C(u)=\frac{t_{i+1}-u}{t_{i+1}-t_i} P_{i-1} + \frac{u-t_i}{t_{i+1}-t_i} P_i$$

Hai hệ số của P_{i-1} và P_i đều không âm và có tổng bằng 1 do đó C(u) chính là đoạn thẳng $P_{i-1}P_i$.

Chứng tỏ khi m=2 đường cong B-spline chính là đường gấp khúc $P_0P_{1....}P_n$ như minh hoạ trong hình 5.4(c).

+ Khi m=3, hàm B-spline $N_{i,3}$ sẽ có bậc bằng 2, phương trình đường cong B-spline có dạng:

$$C(u)=\sum_{i=0}^{n} N_{i,3}(u)\, P_i = N_{0,3}P_0 + N_{1,3}P_1 + ...+ N_{n,3}P_n \quad (t_2 \le u \le t_{n+1})$$

Ta xét hàm B-spline đầu tiên $N_{0,3}$

$$N_{0,3}(u)=\frac{u-t_0}{t_2-t_0} N_{0,2}(u)+\frac{t_3-u}{t_3-t_1} N_{1,2}(u)$$

$N_{0,3}(u)$ là hàm hợp của hai hàm $N_{0,2}(u)$ và $N_{1,2}(u)$ trên đoạn $[t_0,t_3]$, thay các giá trị đã biết $N_{0,2}(u)$ và $N_{1,2}(u)$ vào ta có:

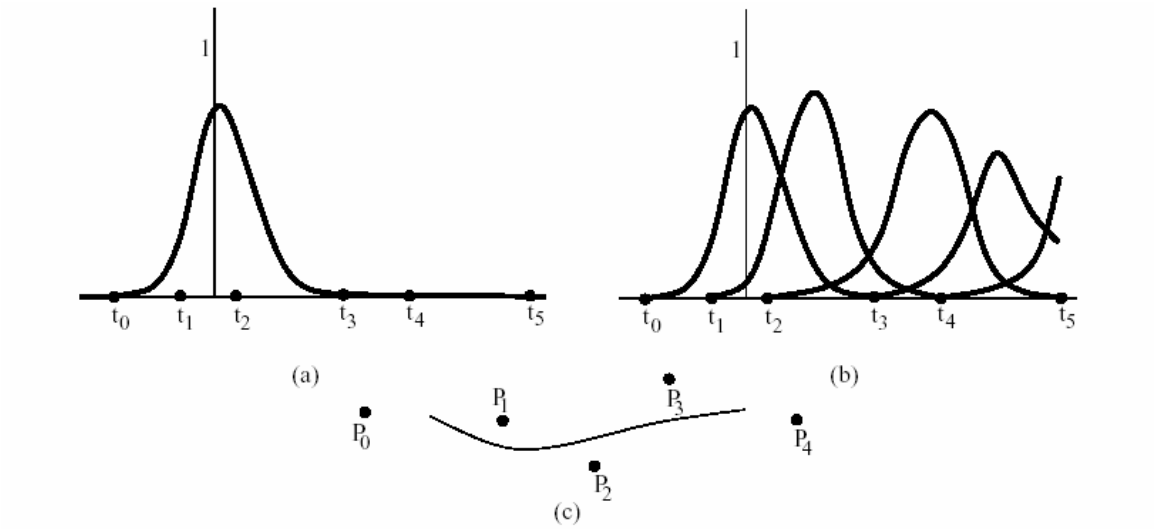
$$N_{0,3}(u) = \begin{cases} 0 & u \le t_0 \\ \frac{u-t_0}{t_2-t_0} * \frac{u-t_0}{t_1-t_0} & t_0 \le u \le t_1 \\ \frac{u-t_0}{t_2-t_0} * \frac{t_2-u}{t_2-t_1} + \frac{t_3-u}{t_3-t_1} * \frac{u-t_1}{t_2-t_1} & t_1 \le u \le t_2 \\ \frac{t_3-u}{t_3-t_1} * \frac{t_3-u}{t_3-t_2} & t_2 \le u \le t_3 \\ 0 & u \ge t_3 \end{cases}$$

Giá trị của hàm $N_{0,3}(u)$ hoặc là bằng 0 hoặc là một hàm bậc 2 theo biến u.

Tổng quát ta có công thức sau:

$$N_{i,3}(u)=\begin{cases}0 & u\le t_i\\ \dfrac{u-t_i}{t_{i+2}-t_i}*\dfrac{u-t_i}{t_{i+1}-t_i} & t_i\le u\le t_{i+1}\\ \dfrac{u-t_i}{t_{i+2}-t_i}*\dfrac{t_{i+2}-u}{t_{i+2}-t_{i+1}}+\dfrac{t_{i+3}-u}{t_{i+3}-t_{i+1}}*\dfrac{u-t_{i+1}}{t_{i+2}-t_{i+1}} & t_{i+1}\le u\le t_{i+2}\\ \dfrac{t_{i+3}-u}{t_{i+3}-t_{i+1}}*\dfrac{t_{i+3}-u}{t_{i+3}-t_{i+2}} & t_{i+2}\le u\le t_{i+3}\\ 0 & u\ge t_{i+3}\end{cases}$$

Hình 5.5(a) minh hoạ đồ thị của $N_{0,3}(u)$, phần đồ thị nằm trên trục hoành ta co thể chia thành ba phần tạm gọi là phần bên trái, phần ở giữa và phần bên phải. Hình 5.5(b) minh hoạ các đoạn đồ thị khác 0 của $N_{i,3}$, $0\le i\le 2$, phần bên trái và phần ở giữa của $N_{3,3}(u)$, phần bên trái của $N_{4,3(u)}$.



Hình 5.5

Xét trên đoạn $[t_i,t_{i+1}]$, $2\le i\le n$ bao gồm ba phần đồ thị: Phần bên trái của đồ thị $N_{i,3}$, phần giữa của đồ thị $N_{i-1,3}$ và phần bên phải của đồ thị $N_{i-2,3}$.

Do đó trên đoạn $[t_i,t_{i+1}]$, $2\le i\le n$ ta có:

$$C(u)=\left(\frac{t_{i+1}-u}{t_{i+1}-t_{i-1}}*\frac{t_{i+1}-u}{t_{i+1}-t_i}\right)P_{i-2}+\left(\frac{u-t_{i-1}}{t_{i+1}-t_{i-1}}*\frac{t_{i+1}-u}{t_{i+1}-t_i}+\frac{t_{i+2}-u}{t_{i+2}-t_i}*\frac{u-t_i}{t_{i+1}-t_i}\right)P_{i-1}+\left(\frac{u-t_i}{t_{i+2}-t_i}*\frac{u-t_i}{t_{i+1}-t_i}\right)P_i$$

Hệ số của P_{i-2} là phần bên phải của $N_{i-2,3}$, hệ số của P_{i-1} là phần giữa của $N_{i-1,3}$, hệ số của P_i là phần bên trái của $N_{i,3}$ và tổng ba hệ số này bằng 1 với mọi u thuộc $[t_i,t_{i+1}]$. Đồ thj của C(u)là một đuờng cong được minh hoạ trên hình 5.5(c).

Ví dụ: Thực hiện chương trình bSpline.c

3.2 Lệnh GLU hiển thị đường cong B-spline

Trước hết ta tạo một đối tượng NURBS (Non-Uniform Rational B-Spline) và trả về một con trỏ tới một đối tượng mới bằng lệnh:

*GLUnurbsObj** **gluNewNurbsRenderer** *(void;*

Thiết lập các thuộc tính của đối tượng NURBS bằng lệnh:

gluNurbsProperty*(GLUnurbsObj *nobj, GLenum property,GLfloat value)*

(Tham khảo các tham số và ý nghĩa trong OpenGL ebook)

Bắt đầu một đường cong NURBS bằng lệnh:

gluBeginCurve *(GLUnurbsObj *nobj)*

Khởi tạo và biểu diễn một đường cong NURBS ta dùng lệnh:

gluNurbsCurve *(GLUnurbsObj *nobj, GLint uknot_count,*
*GLfloat *uknot, GLint u_stride, GLfloat *ctlarray,*
GLint uorder, GLenum type)

Lệnh trên định nghĩa một đường cong NURBS cho đối tượng *nobj*, tham số *uknot_count* chỉ định số lượng điểm nút, con trỏ **uknot* chỉ đến mảng chứa các nút, tham số *u_stride* chỉ khoảng cách giữa hai điểm điều khiển, con trỏ **ctlarray* chỉ đến mảng các điểm điều khiển, tham số uorder xác định bậc của các hàm B-spline, tham số *type* xác định kiểu của các điểm điều khiển(GL_MAP1_VERTEX_3 hoặc GL_MAP1_VERTEX_4)

Kết thúc biểu diễn một đường cong NURBS bằng lệnh:

gluEndCurve *(GLUnurbsObj *nobj)*

Ví dụ: Thực hiện chương trình splineCurveOrder3.c, ví dụ này tạo một đường cong B-spline bậc 3 và cho phép di chuyển các điểm điều khiển cũng như các nút.

Ví dụ: Thực hiện chương trình splineCurveOrder4.c, ví dụ này tạo một đường cong B-spline bậc 4 và cho phép di chuyển các điểm điều khiển cũng như các nút.

4. Mặt cong B-spline

4.1 Cách xây dựng mặt cong B-spline

Phương trình mặt cong B-spline có dạng như sau:

$$S(u,v)=\sum_{i=0}^{n_u}\sum_{j=0}^{n_v} Ni,m1(u)Nj,m2(v)Pij$$

Trong đó Pij là giá trị điểm điều khiển trong ma trận hai chiều (nu+1)*(nv+1). Các giá trị m1-1 và m2-1 thiết lập bậc của các hàm B-spline theo hai biến u và v.

4.2 Lệnh GLU hiển thị mặt cong B-spline (NURBS)

Ngoài các lệnh giống như khi hiển thị đường cong NURBS, mặt cong NURBS sử dụng hai lệnh sau để bắt đầu và kết thúc một mặt cong NURBS

gluBeginSurface *(GLUnurbsObj *nobj)*

gluEndSurface *(GLUnurbsObj *nobj)*

Khởi tạo và biểu diễn mặt cong NURBS bằng lệnh:

gluNurbsSurface *(GLUnurbsObj *nobj, GLint uknot_count,*
*GLfloat *uknot, GLint vknot_count, GLfloat *vknot,*
*GLint u_stride, GLint v_stride, GLfloat *ctlarray,*
GLint uorder, GLint vorder, GLenum type)

Các tham số của lệnh tương tự như lệnh **gluNurbsCurve**().