BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC MỎ - ĐỊA CHẤT

BÁO CÁO HỌC THUẬT

Tên đề tài:

NGHIÊN CỨU ỨNG DỤNG ĐƯỜNG MẶT TRONG THIẾT KẾ BẢN VẼ

Người thực hiện: Vũ Hữu Tuyên

Khoa: Khoa Học Cơ Bản

Hà Nội, tháng 6 /2022

NGHIÊN CỨU ỨNG DỤNG ĐƯỜNG MẶT TRONG THIẾT KẾ BẢN VẼ

GVC. TS. VŨ HỮU TUYÊN

vuhuutuyen@humg.edu.vn

LỜI NÓI ĐẦU

Hiện nay đồ hoạ máy tính (Computer Graphics) là một trong những chương trình thông dụng nhất, nó đã góp phần quan trọng làm cho giao tiếp giữa con người và máy tính trở nên thân thiện hơn. Thật vậy, giao diện kiểu văn bản (text) đã được thay thế hoàn toàn bằng giao diện đồ hoạ, cùng với công nghệ đa phương tiện (multimedia) đã đưa ngành Công Nghệ Thông Tin sang một phiên bản mới.

Trong báo cáo tác giả tìm hiểu cơ sở lý thuyết về đường cong và mặt cong 3D trong tổng thể lý thuyết. Cụ thể, đề cập đến thuật toán xây dựng đường cong Bezier, đường cong Spline, mô hình các bề mặt,…Từ các thuật toán này, chúng ta dần làm quen với các kỹ thuật phức tạp hơn trong kỹ thuật đồ họa.

Với sự cần thiết của kỹ thuật đồ họa như trên, nghiên cứu đường cong, mặt cong trong 3D sẽ làm sáng tỏ ý nghĩa của việc ứng dụng kỹ thuật này trong tương lai.

1. ĐƯỜNG CONG

1.1. Đường cong CURVE

Trong các ứng dụng của đồ hoạ máy tính, hầu như các thực thể là đường cong mềm và mặt cong, chúng dùng để mô tả thế giới thực: nhà cửa, xe cộ, núi non…. hay xây dựng nên các thực thể đang được thiết kế. Nhưng ta thấy sử dụng các phương trình đường cong không thể hiện được hình ảnh thực hay ý tưởng của người thiết kế, còn nếu ta dùng tập hợp các điểm thì thường cần nhiều dung lượng nhớ để lưu trữ cũng như tốc độ tính toán. Ta có quỹ đạo chuyển động của một điểm trong không gian thì tạo thành đường cong. Trong chương này sẽ đưa ra phương pháp tổng thể về những mô hình toán học để biểu diễn và xây dựng các loại đường và mặt cong trong không gian 3D trên máy tính.

Ta có quỹ đạo chuyển động của một điểm trong không gian thì tạo thành đường cong. Trong chương này sẽ đưa ra phương pháp tổng thể về những mô hình toán học để biểu diễn và xây dựng các loại đường và mặt cong trong không gian 3D trên máy tính.

Điểm biểu diễn đường cong (curve presents points)

Ta thấy qua hai điểm vẽ được một đường thẳng. Qua ba điểm vẽ được một đường cong trong mặt phẳng. Qua bốn điểm vẽ được một đường cong trong không gian. Dùng các phương trình đường cong như Hypebol, parabol... thì tính toán phức tạp và không thể hiện được hình ảnh thực hay ý tưởng của người thiết kế.

Chọn đường cong như thế nào để phù hợp với máy tính? Biểu diễn điểm và kiểm soát đường cong -Points represent-and control-the curve. Đường cong là các đối tượng cơ bản thường là kết quả của tiến trình thiết kế và các điểm đóng vai trò là công cụ để kiểm soát và mô hình hoá đường cong. Cách tiếp cận này là cơ sở của lĩnh vực thiết kế mô hình hình học nhờ máy tính (Computer Aided Geometric Design - CAGD)

Các cách để biểu diễn đường cong:

Tường minh (Explicit functions):

$y = f(x), z = g(x)$

Không tường minh (Implicit equations):

$f(x,y,z) = 0$

Biểu diễn các đường cong tham biến (Parametric representation):

$x = x(t), y = y(t), z = z(t)$ trong đó $t \in [0\ \ 1]$

Hạn chế:

Hệ đồ hoạ ứng dụng chỉ mô tả bó hẹp trong đoạn nào đấy.

Đường cong bậc cao với mỗi giá trị của x ta luôn có 2 tập giá trị của y (thực tế chỉ cần 1).

Chúng ta cần biểu diễn đường cong mềm (chỉ biểu diễn đường "cong gẫy")

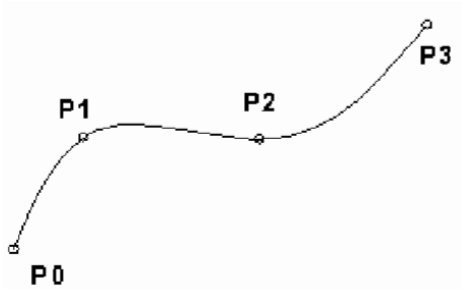
Đường cong đa thức bậc ba tham biến

Phải đảm bảo là đường cong không gian với 3 trục toạ độ x, y, z. Tránh được những tính toán phức tạp và những phần nhấp nhô ngoài ý muốn xuất hiện ở những đường đa thức bậc cao.

Công thức mô tả:

Tường minh : $y = f^3(x), z = g^3(x)$

Không tường minh: $f^3(x,y,z) = 0$



Hình 1.1. Đường cong đa thức bậc ba

Biểu diễn các đường cong tham biến (Parametric representation):

$x = f^3(u)$, $y = f^3(u)$, $z = f^3(u)$ trong đó $u \in [0\ \ 1]$

Theo Lagrange:

$x = a_1 + b_1u + c_1u_2 + d_1u_3$

$y = a_2 + b_2u + c_2u_2 + d_2u_3$

$z = a_3 + b_3u + c_3u_2 + d_3u_3$

Ở đây ba phương trình với 12 ẩn số

Với 4 điểm p_0, p_1, p_2, p_3 phương trình xác định (vì 4 điểm thì xác định 1 đường cong trong không gian).

Mỗi 1 điểm cho ta cặp 3 giá trị: Cả thảy có 12 phương trình, thay vào 3 phương trình trên ta tính được 12 ẩn

Ghi chú: rõ ràng có sự thay đổi một chút về đường cong thì ta lại phải giải lại hệ phương trình để tính các tham số cho đường cong, dẫn đến tính toán chậm.

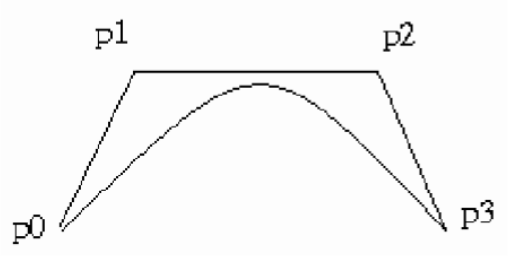
1.2. Đường cong Bezier

Việc sử dụng điểm với các vector kiểm soát được độ dốc của đường cong tại những điểm mà nó đi qua. Tuy nhiên không được thuận lợi cho việc thiết kế tương tác, không tiếp cận với các độ dốc của đường cong bằng các giá trị số (Hermite).

Paul Bezier, nhân viên hãng RENAULT vào năm 1970 đi đầu trong việc ứng dụng máy tính cho việc xây dựng các bề mặt. Hệ thống UNISURF của ông được áp dụng trong thực tế vào năm 1972 được thiết kế và kiểm xe Mezesez hay Renaut.

Bezier đã sử dụng đa giác kiểm soát cho đường cong tại những đỉnh của đa giác và tiếp tuyến tại đó (p_0,p_1,p_2,p_3).

Ta có p_0, p_3 tương đương với p_0, p_1 trên đường Hermite, điểm trung gian p_1, p_2 được xác định bằng 1/3 theo độ dài của vector tiếp tuyến tại điểm p_0 và p_3.



Hình 1.2. Đa giác kiểm soát Bezier

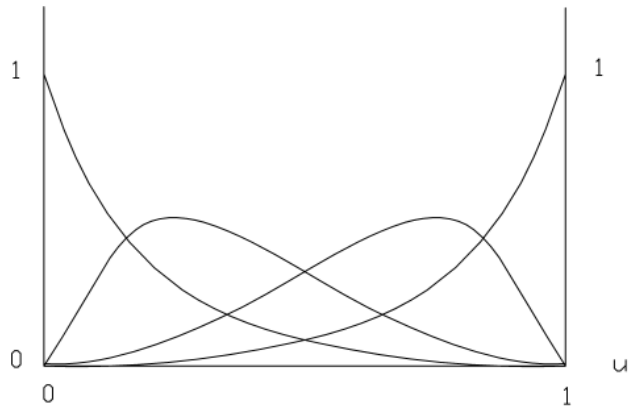
$p_0' = 3(p_1 - p_0)$

$p_1' = 3(p_3 - p_2)$

$p = p(u) = p_0(1-3u^2+2u^3) + p_1(3u^2-2u^3) + p_0'(u-2u^2+u^3) + p_1'(-u^2 + u^3)$

$p = p(u) = p_0(1 - 3u + 3u^2 - u^3) + p_1(3u-6u^2-3u^3) + p_2(3u^2 - 3u^3) + p_3u^3$

$$p = p(u) = [\ 1\ u\ u^2\ u^3\ ] \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$



Hình 1.3. Hàm hợp của đường cong Bezier

Ưu điểm:

- Dễ dàng kiểm soát hình dạng của đường cong hơn vector tiếp tuyến tại p0’ và p1’ của Hermite.
- Nằm trong đa giác kiểm soát với số điểm trung gian tuỳ ý (số bậc tuỳ ý), có số bậc =số điểm kiểm soát -1.
- Đi qua điểm đầu và điểm cuối của đa giác kiểm soát, tiếp xúc với cặp hai vector của đầu cuối đó.

Biểu thức Bezier-Bernstain

Đường Bezier cũng có thể được biết đến như biểu thức Bezier Bernstain bởi kỹ thuật mà Bezier sử dụng là áp dụng công thức hoá các vector trong phép tính đa giác xấp xỉ được Bernstain phát triển gần đây. Phép toán đại số được xác định như sau:

$0 \leq u \leq 1$

0!=1, ui =1 khi i = 0

$P(u) = \sum_{i=0}^{n} B_{i,n}(u)P_i$

$B_{i,n}(u) = C(n,i)u^i(1-u)^{n-i}$

$C(n,i) = \frac{n!}{i!(n-i)!}$

Trong đó P_0.....P_n: vector vị trí của đa giác (n+1) đỉnh.

1.3. Đường cong B-spline

1.3.1. Đường cong bậc ba Spline

Trong công thức của Bezier, chúng ta sử dụng hàm hợp liên tục để xác định điểm kiểm soát tương đối. Với các điểm nội suy thì mức độ tương đối sẽ khác nhau mà trong đó một chuỗi các phần tử nhỏ sẽ kết hợp với nhau tạo ra đường cong đa hợp. Theo tính toán thì đường bậc ba sẽ đa thức bậc thấp nhất có thể để biểu diễn một đường cong trong không gian và chuỗi điểm Hermite sẽ phù hợp nhất đối với việc xây dựng nên đường cong đa hợp này.

Việc yêu cầu người sử dụng đưa vào các vector tiếp tuyến tại mỗi điểm trong tập hợp các điểm là cực kỳ bất tiện cho nên thường trong các đường bậc ba đa hợp ta sử dụng các điều kiện biên liên tục trong phép đạo hàm bậc một và hai tại điểm nối giữa. và đường cong được xác định như trên gọi là đường spline bậc ba với phép đạo hàm liên tục bậc hai. Giá trị đạo hàm của đường cong sẽ xác định độ cong tại mỗi điểm nút và nó cũng đưa ra điều kiện biên cho mỗi đoạn trên đường cong.

Vậy đường bậc ba spline có ưu điểm là không phải xác định độ dốc của đường tại các nút nhưng nhược điểm của nó là chỉ tạo ra sự thay đổi toàn cục khi ta thay đổi vị trí của điểm.

Đường cong – Spline đi qua n điểm cho trước mà mỗi đoạn là các đường cong bậc ba độc lập có độ dốc và độ cong liên tục tại mỗi điểm kiểm soát hay điểm nút. Với n điểm ta có (n-1) đoạn với mỗi đoạn gồm bốn vector hệ số hay 4(n-1) cho n-1 đoạn, và 2(n-1) điều kiện biên và (n2) điều kiện về độ dốc cùng (n-2) về độ cong.

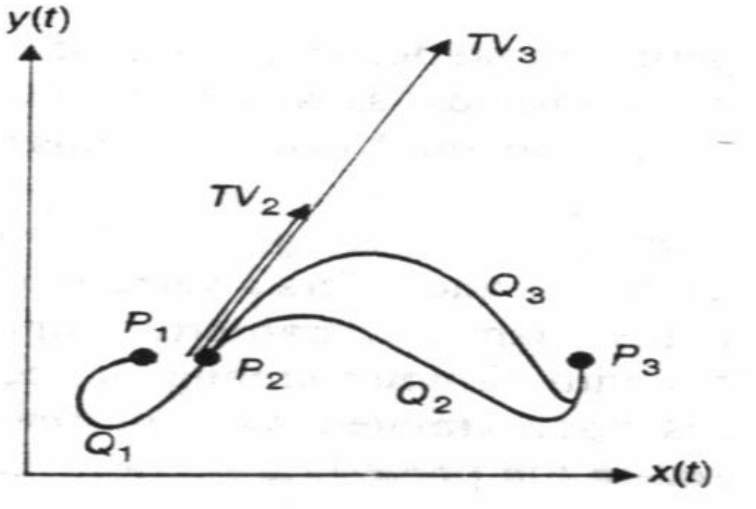
Để xây dựng nên đường spline có tham số với n điểm nút ta có một dãy các giá trị tham số mà ta gọi là vector nút.

u_0.....u_{n-1} trong đó $u_{i+1} > u_i$

Cần lựa chọn tại mỗi nút, cách lựa chọn đơn giản nhất là theo cách đơn điệu có nghĩa là với giá trị 0 tại điểm đầu và tăng lên 1 tại những điểm kế tiếp. tuy vậy phương pháp này dẫn đến độ cong không mong muốn tại các điểm vì vậy việc tham số hoá sẽ đưa vào chiều dài, nhưng phương pháp này cũng không được chính xác khi mà đường cong chưa xác định chiều dài. Tuy nhiên thông thường người ta sử dụng việc tích luỹ của các dây cung với:

u_0=0và $u_{i+1} = u_i + d_{i+1}$ trong đó di: là khoảng cách giữa 2 điểm p_{i-1} và p_i

Trong các trường hợp đường cong có bậc lớn hơn ba có thể dùng cho đường spline. Thông thường đường spline bậc n sẽ được xây dựng trên các phần nhỏ liên tục của các biến độc lập.



Hình 1.4. Kết nối hai đường cong

Hình trên cho thấy hai đoạn cong có chung điểm nối mà đường cong liên tục tại điểm đó, việc biểu diễn tính liên tục của đường cong thông qua chữ cái C-Cuntinue. C_0 để đảm bảo không có sự gián đoạn giữa hai đoạn cong. C_1 tính liên tục bậc nhất hay đạo hàm bậc nhất tại điểm nối. C_2 đạo hàm bậc hai liên tục của đường cong tại điểm nối.

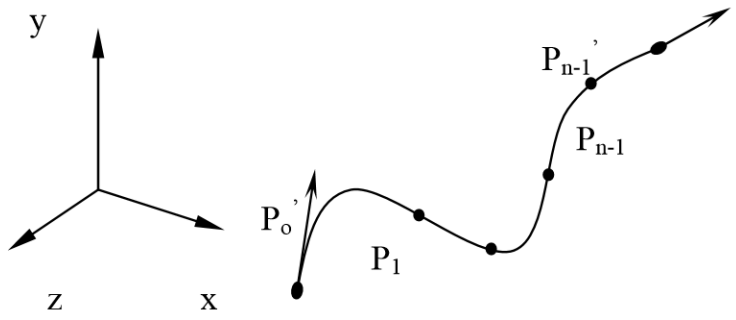
Giả sử khi biểu diễn đường cong mềm thông qua các đoạn cong q_1, q_2, q_3 (mỗi đoạn có 4 vector hệ số) cần thoả mãn: Liên tục tại điểm nối hay $C_0^1 = C_0^2$.

Độ dốc (hay vector tiếp tuyến) tại điểm nối (điểm cuối của q_1 và đầu q_2) là như nhau: $C_1^1 = C_1^2$ (đạo hàm bậc nhất)

Thoả mãn liên tục trên tại điểm nối (đạo hàm bậc 2 liên tục tại điểm nối) $C_2^1 = C_2^2$

Việc kết hợp các đoạn cong Hermite bậc ba để mô tả một đường cong mềm theo kiểu phân đoạn spline là phương pháp đơn giản nhất hay còn gọi là phương pháp Hermite nội suy. Với phương pháp này thì tham biến ui cho mỗi đoạn cong i của tập các đoạn cong Hermite sẽ biến đổi trong khoảng từ 0 đến 1 và luôn tồn tại đạo hàm bậc nhất của các đoạn cong tại các điểm nối. Phương trình cho mỗi đoạn cong được sử dụng lúc này là phương trình đường cong bậc ba Hermite:

$$p = p(u) = [\ 1\ u\ u^2 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & 2 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ p_1 \\ p'_0 \\ p'_1 \end{bmatrix}$$



Hình 1.5. Phân đoạn của đường cong Spline – Hermite

Theo Hermite các đoạn là các đường cong, tính liên tục của đạo hàm bậc hai tại các điểm nối có thể dễ dàng đạt được bằng cách đặt $P''_{i-1}(u_{i-1}=1)$ là đạo hàm bậc hai tại điểm cuối của đoạn (i-1) bằng với $P''_i(u_i=0)$ đạo hàm bậc hai tại điểm đầu của đoạn thứ i.

$P''_{i-1}(1)= P''_i(0)$

Có phương trình:

$P_i(u) = k_{0i} + k_{1i}u + k_{2i}u^2 + k_{3i}u^3$

Đạo hàm bậc hai sẽ là:

$P''_i(u) = 2k_{2i} + 6k_{3i}u$

$P''_{i-1}(1)= P''_i(0)$ nên $2k_{2(i-1)} + 6k_{3(i-1)}u= 2k_{2i}$

Vì điểm cuối của đoạn i-1 trùng với điểm đầu của đoạn thứ i($P_i(0)=P_{i-1}(1)$)

Theo Hermite:

$k_2 = 3(p_1 - p_0) - 2p_0' - p_1'$ và $k_3 = 2(p_0 - p_1) + p_0' + p_1'$

$2(3(Pi - P_{i-1}) - 2P'_{i-1} - P'_i) + 6(2(P_{i-1} - P_i) + P'_{i-1} + P'_i) = 2(2(P_{i-1} - P_i) + P'_{i-1} + P'_i)$

Hay: $P'_{i-1} + 4P'_i + P'_{i+1} = 3(P_{i+1} - P_i)$ (*)

Với phương trình (*) này thì phương trình dạng tổng quát của đường cong Spline là tập của các đoạn cong Hermite sẽ xác định với điều kiện ban đầu cho là tập các điểm kiểm soát của đường cong và hai vector tiếp tuyến tại hai điểm đầu cuối của đường cong đó. Sử dụng (*) ta có thể tính được các giá trị của các vector tiếp tuyến tại từng điểm kiểm soát của đường cong.

$$\begin{bmatrix} 1 & 0 & . & . & . & . \\ 1 & 4 & 1 & 0 & . & . \\ 0 & 1 & 4 & 1 & 0 & . \\ . & . & . & . & . & . \\ . & . & 0 & 1 & 4 & 1 \\ . & . & . & . & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} P_0' \\ P_1' \\ . \\ . \\ P_{n-2}' \\ P_{n-1}' \end{bmatrix} = \begin{bmatrix} P_0' \\ 3(P_2 - P_0) \\ . \\ . \\ 3(P_{n-1} - P_{n-3}) \\ P_{n-1}' \end{bmatrix}$$

Tương đương với:

$$\begin{bmatrix} P_0' \\ P_1' \\ . \\ . \\ P_{n-2}' \\ P_{n-1}' \end{bmatrix} = \begin{bmatrix} 1 & 0 & . & . & . & . \\ 1 & 4 & 1 & 0 & . & . \\ 0 & 1 & 4 & 1 & 0 & . \\ . & . & . & . & . & . \\ . & . & 0 & 1 & 4 & 1 \\ . & . & . & . & 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} P_0' \\ 3(P_2 - P_0) \\ . \\ . \\ 3(P_{n-1} - P_{n-3}) \\ P_{n-1}' \end{bmatrix}$$

1.3.2. Đường Spline

Với Bezier hay spline đều không cho ta thay đổi đường cong một cách cục bộ, việc thay đổi vị trí các điểm kiểm soát hay các vector tiếp tuyến không chỉ ảnh hưởng trực tiếp đến độ dốc của đường cong lân cận quanh điểm kiểm soát mà còn kéo theo ảnh hưởng đến các phần còn lại của đường cong. Đường Bezier thêm vào đó là khi tính xấp xỉ ở bậc cao sẽ rất phức tạp còn khi liên kết nhiều đoạn Bazier hay Hermite bậc thấp (bậc ba) có thể đem lại ích lợi khi tính toán nhưng yếu tố ràng buộc về tính liên tục của đạo hàm bậc cao tại các điểm nối không cho điều khiển cục bộ như mong muốn.

Việc kết hợp luôn phiên các đoạn cong tổng hợp, thông qua các đa thức tham số xác định riêng rẽ trên một số điểm kiểm soát lân cận với số bậc tuỳ ý không phụ thuộc vào số lượng các điểm kiểm soát, cho phép tạo nên đường cong trơn

mềm B-spline. Đường cong này đã khắc phục được các nhược điểm mà các dạng đương cong trước chưa đạt được. Có nghĩa là khi dịch chuyển điểm kiểm soát của đương cong thì chỉ một vài phân đoạn lân cận của điểm kiểm soát đó bị ảnh hưởng chứ không phải toàn bộ đường cong.

Với n+1 số điểm kiểm soát P_i ta có:

$P(u) = \sum_{i=0}^{n} N_{i,k}(u).P_i$

Trong đó $N_{i,k}(u)$ là hàm hợp B-Spline bậc k-1 và sự khác biệt giữa B-spline và Bezier sẽ được thể hiện trên đó. Trong đường Bezier bậc của đa thức được xác định bởi số đoạn cong trên đường cong đó, còn với B-spline bậc được thoả mãn độc lập với số điểm kiểm soát của đường.

Hơn nữa hàm hợp của Bezier khác 0 trên toàn bộ khoảng của tham số u còn B-spline chỉ khác 0 trên đoạn ngắn của các tham số. Mỗi đoạn trên hàm hợp chỉ tương ứng với một điểm thì chỉ dẫn tới sự thay đổi cục bộ trong khoảng mà trên đó tham số của hàm hợp khác 0. Biểu diễn toán học của B-spline, với hàm B-spline có bậc k-1 xác định thì:

$$N_{i,k}(u) = \frac{(u - U_{i+1-k})}{(U_i - U_{i+1-k})} N_{i-1,k-1}(u) + \frac{(U_{i+1} - u)}{(U_{i+1} - U_{i+2-k})} N_{i,k-1}(u)$$

$$N_{i,1}(u) = \begin{cases} 1 & u \in [u_i, u_{i+1}] \\ 0 & \end{cases}$$

Trong đó ui là giá trị tại nút pi với biến số là u được gọi là các vector nút.

Tất cả các giá trị nút đồng thời xác định trên vector nút và các nút nguyên thường sử dụng dễ dàng. Trong trường hợp này các hàm hợp bậc k sẽ khác 0 trong khoảng k của vector nút và toàn bộ các giá trị trên vector cho một tập hợp điểm bằng n+1+k.

Không như Bezier, đường B-spline không đi qua hai điểm đầu và cuối trừ khi hàm hợp được dùng là tuyến tính.

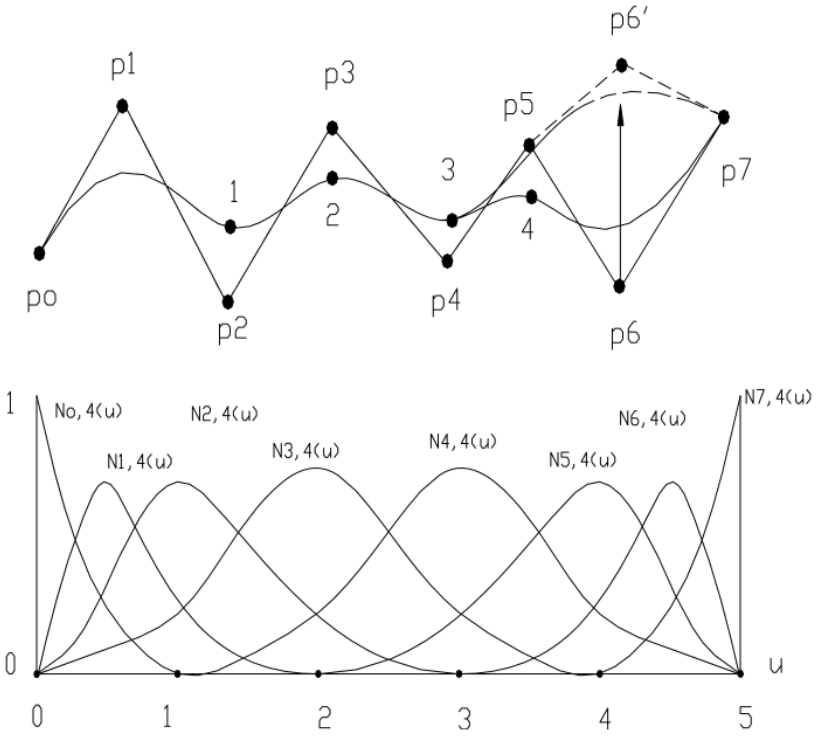
Đường B-spline có thể được tạo qua hai điểm đầu, cuối và tiếp xúc với vector đầu và cuối của đa giác kiểm soát. Bằng cách thêm vào các nút tại vị trí của các nút cuối của vector tuy nhiên các giá trị giống nhau không nhiều hơn bậc của đường cong.

Giống như đường cong Bezier, tính chất bao lồi của đa giác kiểm soát và tính chất chuẩn được thỏa mãn. Vậy có:

$\sum_{i=0}^{n} N_{i,k}(u) = 1$

Trong đường cong B-spline, số lượng các nút, bậc của đường cong và số điểm điều khiển luôn có các quan hệ ràng buộc:

$0 \leq u \leq n - k + 2$



Hình 1.6. Đường cong B-Spline

Vậy việc xác định các vector nút sẽ phụ thuộc vào sự phân loại của chính bản thân chúng và điều đó sẽ ảnh hưởng đến hình dạng của đường cong được mô tả. Phân loại sẽ dựa trên loại của đường cong như sau:

Đều tuần hoàn (periodic)

Không tuần hoàn (open or unperodic)

Không đều (non-uniform)

a. B-Spline đều và tuần hoàn

Vector nút là đều khi giá trị của chúng cách đều nhau một khoảng ∇ xác định. Ví dụ: [0 1 2 3 4 5] với ∇ xác định = 1

[-2-1/2 1 5/2 4] với ∇ xác định = 3/2

[-1-0.6 -0.2 0.2 0.6 1] với ∇ xác định = 0.4

Trong các bài toán thực tế, thông thường thì khoảng xác định của tham biến nằm trong khoảng từ 0 đến 1 hay từ 0^0 đến 360^0 thì việc chọn giá trị của các vector nút được chuẩn hoá trong khoảng [0 1] hay [0^0 360^0] đó.

[00.2 0.4 0.6 0.81] với ∇ xác định = 0.2

[$0^0120^0240^0360^0$] với ∇ xác định = 120^0

Bậc (k-1)	Cấp (k)	Vector nút (m=n+k)	Khoảng tham số (k-1)≤t≤(n+1)
1	2	[0 1 2 3 4 5 6 7]	1≤t≤6
2	3	[0 1 2 3 4 5 6 7 8]	2≤t≤6
3	4	[0 1 2 3 4 5 6 7 8 9]	3≤t≤6

Các vector nút gọi là đều và tuần hoàn khi các hàm B-spline đối với mỗi phân đoạn có thể chuyển đổi lẫn nhau. Bảng trên chỉ ra sự thay đổi của miền tham số và vector nút khác nhau của các đường cong B-spline khi bậc của đường cong thay đổi. Số lượng của vector nút được qui định bởi biểu thức m-n+k và số lượng các điểm kiểm soát tính qua biểu thức (n+1) bằng 6.

Tính chất:

Ảnh hưởng của mỗi hàm cơ sở được giới hạn trong k đoạn là cấp của đường cong cần thể hiện. Vậy chúng ta sử dụng đường cong bậc ba thì ảnh hưởng của hàm cơ sở trải dài trên bốn đoạn của đường cong.

Đường B-spline tuần hoàn không đi qua các điểm đầu và cuối của đa giác kiểm soát ngoại trừ với đường bậc 1 (k=2) mà khi đó đường cong chuyển dạng thành đường thẳng.

Ví dụ về các đường B-spline tuần hoàn có các bậc khác nhau có cùng các điểm và đa giác kiểm soát.

Khi k=2 đường cong bậc một trùng với các cạnh của đa giác kiểm soát.

Khi k=3, đường cong B-spline bậc 2, bắt đầu tại trung điểm của cạnh thứ nhất và kết thúc tại trung điểm của cạnh cuối cùng của đa giác kiểm soát.

b. B-Spline không tuần hoàn (Open – Non Uniform)

Một vector không tuần hoàn hoặc mở là vector nút có giá trị nút tại các điểm đầu cuối lặp lại với số lượng các giá trị lặp lại này bằng chính cấp k của đường cong và các giá trị nút trong mỗi điểm lặp này là bằng nhau

Nếu một trong hai điều kiện này hoặc cả hai điều kiện không được thoả mãn thì vecto nút là không đều.

Ví dụ, xét một đa giác kiểm soát với bốn đỉnh. Các đường cong B-spline cấp 2,3,4 được xây dựng dựa trên đa giác kiểm soát có số lượng các nút m=n+k sẽ có vector nút như sau:

Cấp (k)	Số lượng nút (m = n + k)	Vector nút không tuần hoàn
2	6	[0 0 1 2 3 3]
3	7	[0 0 0 1 2 2 2]
4	8	[0 0 0 0 1 1 1 1]

Các biểu thức phải được thoả mãn đối với nút u_i trên vector nút không tuần hoàn bắt đầu tại u_0.

Danh sách các vector nút không tuần hoàn đã đưa ra ở mục này đều thoả mãn các biểu thức sau:

$u_i = 0$ $1=<i<=k$

$u_i = i-k$ $k+1<i<=n+1$

$u_i = n-k+2$ $n+1<i<=n+k+1$

Các vector nút không tuần hoàn cung cấp các hàm cơ sở được định nghĩa trong một miền tham số phức tạp và không có sự mất mát như với loại vector tuần hoàn và vì vậy đường cong Bspline loại này luôn đi qua các điểm đầu và cuối của đa giác kiểm soát.

Ví dụ: Hàm hợp bậc ba tính xấp xỉ cho 8 khoảng sẽ xác định trên vector nút là 00001234555. Ở đây chúng ta còn thấy sự thay đổi cục bộ trên đường cong khi ta thay đổi vị trí mỗi điểm.

Đường cong Bezier là trường hợp đặc biệt của B-spline không tuần hoàn, trong đó số lượng các đỉnh sử dụng bằng với cấp của đường cong. Vector nút trong trường hợp này là:

[0 0 ..... 0 1 1 ..... 1 1]

kk

Đường cong B-spline bậc ba với bốn điểm kiểm soát và vector không tuần hoàn [0 0 0 0 1 1 1 1] cũng chính là đường cong Bezier.

c. B-Spline không đều

Trong vector nút không tuần hoàn, giá trị các nút xuất hiện tại các biên được lặp lại và các nút bên trong các bước nút bằng nhau. Nếu một trong hai điều kiện này hoặc cả hai điều kiện này không được thoả mãn thì vector nút là không đều.

Ví dụ các nút không đều có thể tạo ra bằng cách đặt các giá trị lặp lại đối với các nút ở khoảng giữa [0 1 2 3 3 4 5]

Hay tạo ra bước nhảy không bằng nhau giữa các nút [0.0 0.2 0.5 0.75 1.0]

Các vector nút loại đều cho phép người sử dụng dễ hình dung và xử lý trong các phép toán nhưng trong một số các trường hợp bước nút không đều lại có những ưu điểm đặc biệt. Ví dụ như trong việc điều khiển hình dạng của đường cong trong tiến trình thiết kế khi các sai lệch không mong muốn có thể xuất hiện mà việc sử dụng đường cong B-spline đều với các dữ liệu điểm có các khoảng cách tương đối lớn mà không đều nhau.

Kết luận

- B-spline là một dòng của Bezier
- Thực tế khi ta chọn bậc k cho tập hợp k điểm thì thi B-spline chuyển thành Bezier
- Khi bậc của đa thức giảm sự ảnh hưởng cục bộ của mỗi điểm nút càng rõ ràng hơn.
- Khi tồn tại ảnh hưởng cục bộ càng lớn và đường cong phải đi qua điểm đó.
- Chúng ta có thể thay đổi hình dạng đường cong B-spline bằng cách:
    - Thay đổi kiểu vector nút: đều tuần hoàn, mở, không đều
    - Thay đổi cấp k của đường cong
    - Thay đổi số đỉnh và vị trí các đỉnh đa giác kiểm soát
    - Sử dụng các điểm kiểm soát trùng nhau

2. MÔ HÌNH BỀ MẶT VÀ CÁC PHƯƠNG PHÁP XÂY DỰNG

2.1. Các khái niệm cơ bản

Mặt cong (surface): Là quỹ đạo chuyển động của một đường cong tạo nên. Biểu diễn tham biến cho mặt cong:

Dựa vào việc xây dựng và tạo bề mặt toán học trên những điểm dữ liệu. Dựa trên việc xây dựng nên bề mặt phụ thuộc vào biến số có khả năng thay đổi một cách trực diện thông qua các tương tác đồ hoạ.

Ta có:

$x=x(u,v,w)$ $\quad u,v,w \in [0, 1]$

$y=y(u,v,w)$ $\quad u + v + w = 1$

$z=z(u,v,w)$

$Q(u,v,w) = Q[\ x=x(u,v,w)\ y=y(u,v,w)\ z=z(u,v,w)\ ]$

Biểu diễn theo mảnh:

Biểu diễn miếng tứ giác - quadrilatera Patches

Biểu diễn miếng tam giác - Triangular Patches

2.2. Biểu diễn mảnh tứ giác

Phương trình:

$x=x(u,v)$

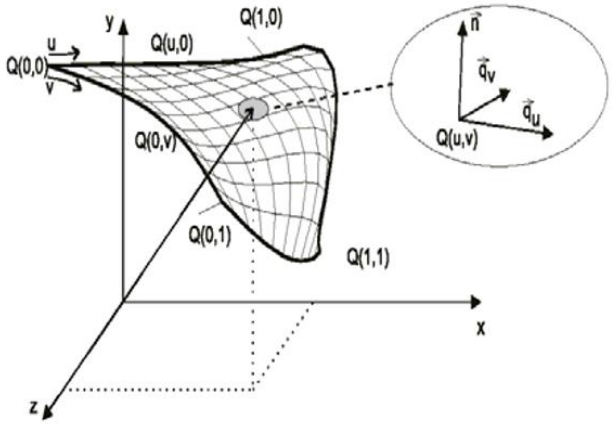
$y=y(u,v)$ $\quad u,v \in [\ 0, 1]$

$z=z(u,v)$

$Q(u,v) = Q[\ x=x(u,v)\ y=y(u,v)\ z=z(u,v)\ ]$

Thành phần u,v là các tham biến

Các điểm Q(0,0) Q(0,1), Q(1,0), Q(1,1) là cận của mảnh, các đường cong Q(1,v), Q(0,v), Q(u,0), Q(u,1) là các biên của mảnh. Đạo hàm riêng tại điểm Q(u,v) xác định vector tiếp tuyến theo hướng u, v.

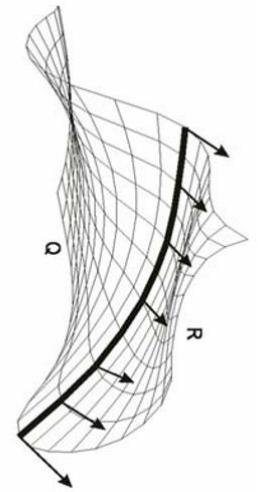


Hình 2.1. Biểu diễn mảnh tứ giác

$\partial Q(u,v)/\partial u = Q[\partial x(u,v)/\partial u, \partial y(u,v)/\partial u, \partial z(u,v)/\partial u]$

$\partial Q(u,v)/\partial v = Q[\partial x(u,v)/\partial v, \partial y(u,v)/\partial v, \partial z(u,v)/\partial v]$

2.2.1. Kết nối mảnh tử giàc



Hình 2.2. Kết nối mảnh tứ giác

Thực thể hình học biểu diễn thông qua các mảnh cùng dạng, các mảnh có thể nối với nhau theo các hướng u,v khi hai mảnh cùng hướng đó. Nếu mọi điểm trên biên của hai mảnh bằng nhau, hay hai biên bằng nhau. Hai mảnh liên tục bậc C_0. Nếu hai biên bằng nhau và đạo hàm bằng nhau trên cùng một hướng thì hai mảnh gọi là kết nối bậc C_1.

2.2.2. Hệ tọa độ Barycentric Coordinates

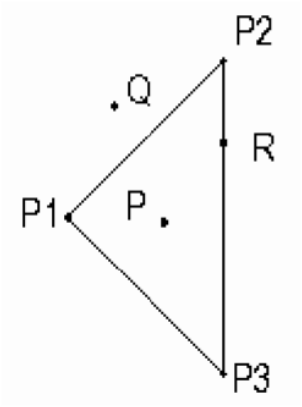
Tập các điểm $P_1, P_2 \dots P_n$, tập các tổ hợp của các điểm đó

$k_1P_1 + k_2P_2 + k_3P_3 \dots + k_nP_n$

Với $k_1 + k_2 + k_3 + \dots + k_n = 1$

Các điểm tạo thành không gian affine với các giá trị tọa độ nates $k_1,k_2,k_3,..k_n$được gọi là hệ toạ độ barycentric.

2.2.3. Tam giác – Triangula



Hình 2.3. Mảnh tam giác

Trong tam giác các điểm có dạng P_1, P_2, P_3

Hệ số: k_1, k_2, $k_3 \in [0, 1]$

$$k_1 + k_2 + k_3 = 1$$

$$P = k_1P_1 + k_2P_2 + k_3P_3$$

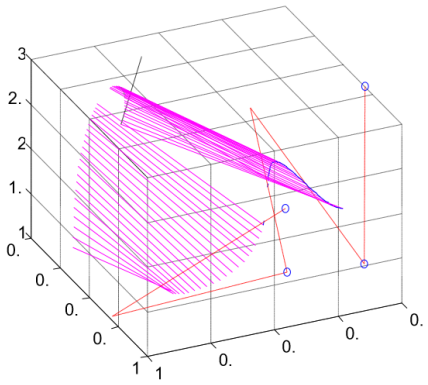
Nếu Hệ số $k_i > 1$ hoặc <0 điểm P sẽ nằm ngoài tam giác (Q)

Nếu Hệ số $k_i = 1$ hoặc $=0$ điểm P sẽ nằm trên cạnh tam giác (R)

2.3. Mô hình hóa các mặt cong (Surface Patches)

2.3.1. Mặt kẻ (Ruled Surface)

Bề mặt được xây dựng bằng cách cho trượt một đoạn thẳng trên hai đường cong. Các mặt kẻ nhận được bằng phép nội suy tuyến tính từ hai đường cong biên cho trước tương ứng với hai biên đối diện của mặt kẻ $P_1(u)$ và $P_2(u)$.



Hình 2.4. Mô hình bề mặt kẻ

Phương trình mặt kẻ:

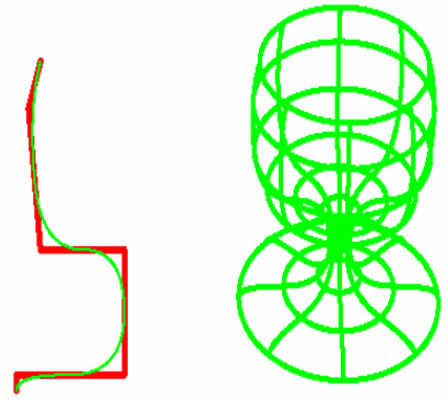
$Q(u,v) = P_2(u)v + P_1(u)(1-v)$

Nếu hai đường cong cho trước tương ứng là $P_1(v)$ và $P_2(v)$

Thì mặt kẻ có phương trình:

$$Q(u,v) = P_1(v)(1-u) + P_2(v)u = [(1-u)]\begin{bmatrix} P_1(v) \\ P_2(v) \end{bmatrix}$$

2.3.2. Mặt tròn xoay (Revolution surface)



Hình 2.5. Mô hình mặt tròn xoay

Mặt được xây dựng bởi đường thẳng hay một đường cong phẳng, quanh một trục trong không gian.

Giả sử đường cong phẳng có dạng: P(t)=[x(t) y(t) z(t)] 0≤t≤tmax

Ví dụ: quay quanh trục x một thực thể nằm trên mặt phẳng xoy, phương trình bề mặt là *Q(t, f) = [x(t) y(t) cosfz(t) sinf]*

$0 \le \varphi \le 2\pi$

Ví dụ: Mặt tròn xoay

P_1[1 1 0] và P_2[6 2 0] nằm trong mặt phẳng xoy. Quay đường thẳng quanh trục ox sẽ được một mặt nón. Xác định điểm của mặt tại t=0.5, f =p/3.

Phương trình tham số cho đoạn thẳng từ P_1 tới P_2 là:

$P(t) = [\ x(t)\ \ y(t)\ \ z(t)\ ] = P1 + (P1 - P2)t \qquad 0 \le t \le 1$

Với các thành phần Đề-các:

$x(t) = x1 + (x2- x1)t = 1+5t$

$y(t) = y1 + (y2- y1)t = 1+t$

$z(t) = z1 + (z2- z1)t = 0$

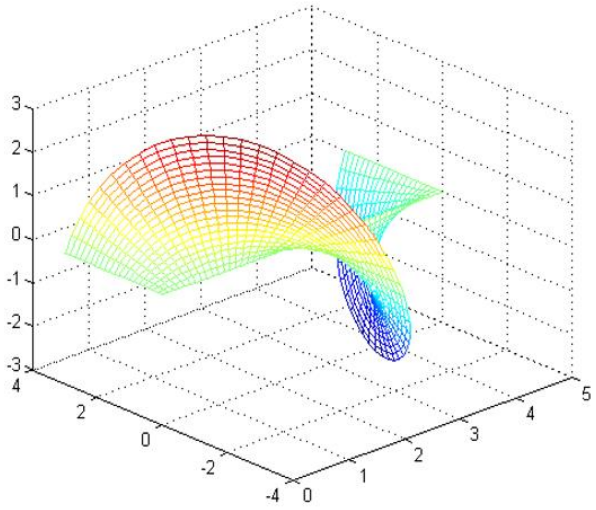
Dùng phương trình:

$$Q(1/2, p/3) = [\ 1+5t(1+t)\cos f\ (1+t)\sin f\ ]$$

$$= \left[\frac{7}{2} \quad \frac{3}{2}\cos\frac{\pi}{3} \quad \frac{3}{2}\sin\frac{\pi}{3}\right]$$

$$= \left[\frac{7}{2} \quad \frac{3}{4} \quad \frac{3\sqrt{3}}{4}\right]$$

2.3.3. Mặt trượt (Sweep Surface)



Hình 2.6. Mô hình mặt trượt

Sweep surface là mặt được tạo bởi bằng cách trượt một thực thể.

Ví dụ: một đường thẳng, đa giác, một đường cong, một hình… dọc theo một đường trong không gian.

Q(u,v) = P(u)*[T(v)]

P(u) thực thể cần trượt

[T(v)] là ma trận biến đổi ([T(v)] có thể là ma trận tịnh tiến, quay, hay tỉ lệ …hoặc là kết hợp của nhiều phép biến đổi đó).

Ví dụ:

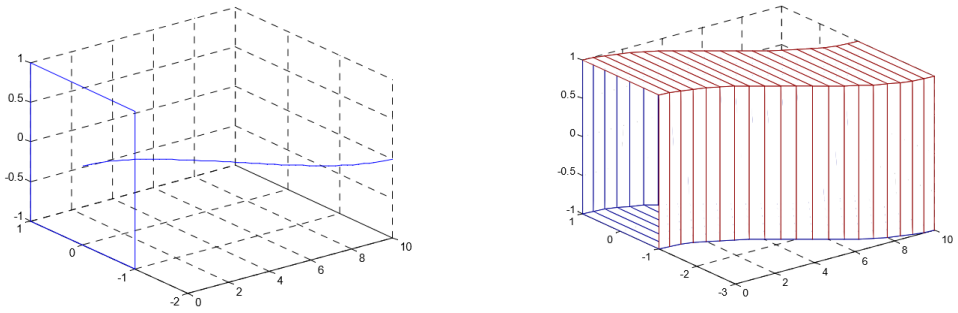
P1[000], P2[03 0]

P(t) = P1 + (P2 – P1) *u = [0 3u01]

$0 \leq u,v \leq 1$

$$|T(v)| = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(2\Pi v) & \sin(2\Pi v) & 0 \\ 0 & -\sin(2\Pi v) & \cos(2\Pi v) & 0 \\ 10v & 0 & 0 & 1 \end{bmatrix}$$

Ví dụ về mặt trượt (Sweept Extrusion)



Hình 2.7. Hình thành mặt trượt

Hình vuông xác định bởi 4 đỉnh:

P_1[0 -10], P_2[0-1-1]

P_3[01 -1], P_4[01 1]

Đường cong trượt:

x= 10vy= cos(Pv) – 1

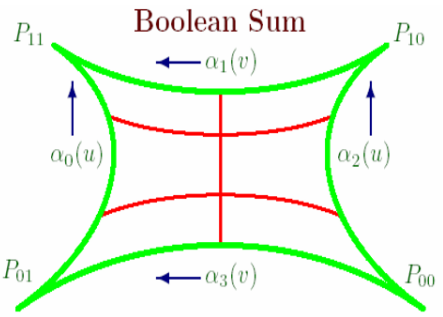
$$|P(u) = \begin{bmatrix} P1 \\ P2 \\ P3 \\ P4 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 1 & 1 \\ 0 & -1 & -1 & 1 \\ 0 & 1 & -1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & -1 & 1 & 1 \end{bmatrix} \qquad |T(v)| = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 10v & \cos(\Pi v) - 1 & 0 & 1 \end{bmatrix}$$

Quay 1 góc khi trượt:

$$\begin{bmatrix} \cos(\varphi) & \sin(\varphi) & 0 & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 10v & \cos(\Pi v) - 1 & 0 & 1 \end{bmatrix}$$

2.3.4. Mặt nội suy trên bốn đường biên(Boolean sum surface)

Mặt được xây dựng trên 4 điểm và các đường cong biên, S(u,v) mặt nội suy trên 4 đường biên.



Hình 2.8. Mô hình mặt cong Boolean Sum

$S(u, v) = S_1(u, v) + S_2(u, v) - P(u,v)$

Với:

$P(u,v) = (1-u)(1-v)P_{00} + (1-u)vP_{01} + u(1-v)P_{10} + uvP_{11}$

$S_1(u,v) = va_0(u) + (1-v)a_2(u)$

$S_2(u,v) = ua_1(v) + (1-u)a_3(v);$

P là các đỉnh của mảnh 4

ai(u) là các phương trình đường biên

Ví dụ về mặt Boolean Sum:

Với u = 0

$S(0,v) = S_1(0,v) + S_2(0,v) - P(0, v)$

$= v\ a_0(0) + (1 - v)a_2(0) + 0a_1(v) + 1\ a_3(v) - (1 - v)P_{00} - v\ P_{01}$

$= v\ P_{01} + (1 - v)P_{00} + a3(v) -(1 - v)P_{00} - v\ P_{01}$

$= a_3(v)$

2.4. Mặt từ các đường cong

2.4.1. Mặt cong bậc ba Hermite

Mặt cong tham biến được tạo bởi bề mặt qua tại 4 điểm dữ liệu tại 4 góc và các đường cong có phương trình bậc ba qua chúng, như vậy 16 vector điều kiện hay tương đương với 48 giá trị đại số cần thiết để xác định các hệ số phương trình.

Khi những hệ số là 4 điểm dữ liệu góc và 8 vector tiếp tuyến tại các điểm đó theo các hướng u,v tương ứng cùng 4 vector xoắn thì mặt cong tạo thành là mặt cong Hermite. Phương trình có dạng:

$$Q(u,v)=\sum_{i=0}^{3}\sum_{j=0}^{3}C_{ij}u^{i}v^{j} \qquad 0\le u,v\le 1$$

$Q(u, v) = [U][C][V]^T \quad 0 \le u, v \le 1$

Với: $u = [\, u^3 u^2 u1] \,, v = [v^3 \;\; v^2 \;\; v \;\; 1]$

Và ma trận hệ số [C] là:

$$[C]=\begin{bmatrix} C_{00} & C_{01} & C_{02} & C_{03} \\ C_{10} & C_{11} & C_{12} & C_{13} \\ C_{20} & C_{21} & C_{22} & C_{23} \\ C_{30} & C_{31} & C_{32} & C_{33} \end{bmatrix}$$

Cuối cùng ta thu được các hệ số theo phương trình mới có dạng:

$Q(u, v) = [U][M_H] [B] [M_H]^T [V]^T \; u,v \in [0,1]$

$$[M_H]=\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Và B là ma trận điều kiện biên:

$$[B]=\left[\begin{array}{cc|cc} P_{00} & P_{01} & P_{v00} & P_{v01} \\ P_{10} & P_{11} & P_{v10} & P_{v11} \\ \hline P_{u00} & P_{u01} & P_{uv00} & P_{uv01} \\ P_{u10} & P_{u11} & P_{uv10} & P_{uv11} \end{array}\right]$$

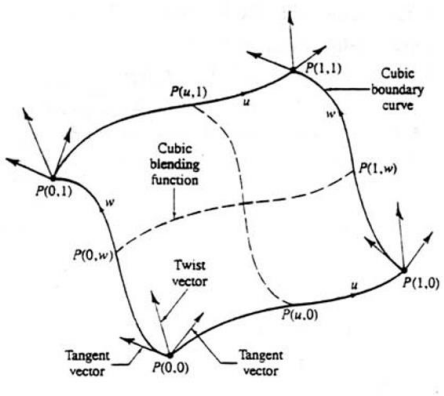
Hay với dạng thức rút gọn của ma trận [B] theo các ma trận điều kiện biên tương ứng:

[B] ma trận các giá tị tham số

$[P_u]$, $[P_v]$ các vector tiếp tuyến theo u,v tương ứng.

$[P_{uv}]$ ma trận xoắn trên u,v

$$[B] = \left[\begin{array}{c|c} [P] & [P_v] \\ \hline [P_u] & [P_{uv}] \end{array} \right]$$



Hình 2.9. Mặt cong Hermite và các điểm dữ liệu

Các vector tiếp tuyến và vector xoắn của bề mặt cong được biểu diễn qua phương trình sau:

$$Q_u(u,v) = [U]\,[M_H]^u\,[B]\,[M_H]^T\,[V]^T$$

$$Q_v(u,v) = [U]\,[M_H]^v\,[B]\,[M_H]^T\,[V]^T$$

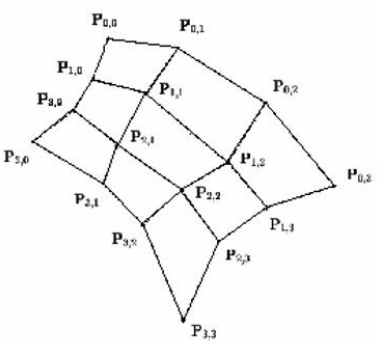
$$Q_{uv}(u,v) = [U]\,[M_H]^{uv}\,[B]\,[M_H]^T\,[V]^T$$

Với:

$$[M_H]^U \;\&\; [M_H]^V = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 6 & -6 & 3 & 3 \\ -6 & 6 & -4 & -2 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2.4.2. Mặt cong Bezier

Mảnh-patch Bézier



Hình 2.10. Mặt cong Bezier

Mảnh Bezier được hình thành trên phép trượt của đường cong Bezier.Việc xây dựng nên mảnh Bezier dưới các điểm kiểm soát, tạo nên đa diện kiểm soát.

$\{ P_{i,j}: 0 \le i \le n, 0 \le j \le m\}$

Phương trình tổng quát của mặt cong tham biến Bezier có dạng:

$P(u,v) = \sum_{j=0}^{m} \sum_{i=0}^{n} P_{i,j} B_{i,j}(u) B_{j,m}(v)$ trong đó u,v ∈ [0,1]

Mảnh Bezier bậc ba: Mặt cong Bezier bậc ba là mặt phổ biến nhất trong CG, vì đi độ đơn giản của nó. Nó hình thành trên 4x4 điểm kiểm soát, công thức có dạng:

$$Q(u,v) = \sum_{i=0}^{3} \sum_{j=0}^{3} B_{i,j}(u) B_{m,j}(v) P_{ij}$$

Đa thức Bernstein có dạng:

$B_0(t) = (1-t)^3$

$B_1(t) = 3t\ (1-t)^2$

$B_2(t) = 3t^2(1-t)$

$B_3(t) = t^3$

Tính chất của mảnh Bézier:

- Tính bao lồi: Mặt cong Bezier luôn nằm trong đa diện lồi của các điểm kiểm soát.
- Mặt cong đi qua 4 điểm cận P_{00}, P_{01},P_{10},P_{11} hay chính xác $Q(0,0)=P_{00}$,
    - $Q(0,1)=P_{01}$, $Q(1,0)=P_{10}$, $Q(1,1)=P_{11}$
- Đường cong biên của Mặt Bezier là đường cong Bezier
- Mặt cong là liên tục và đạo hàm riêng các bậc tồn tại của nó cũng liên tục.
- Đạo hàm riêng của mặt cong có dạng:

$Q(u,v) = [U]\ [N]\ [B]\ [M]^T\ [V]^T$

$\partial Q(0,0)/\ \partial u = 3(P_{01} - P_{00})$

$\partial Q(0,0)/\ \partial u = 3(P_{01} - P_{00})$

$\partial Q(0,1)/\ \partial u = 3(P_{03} - P_{02})$

$\partial Q(1,0)/\ \partial u = 3(P_{13} - P_{03})$

Q(u,v) là mọi điểm nằm trên mặt cong và

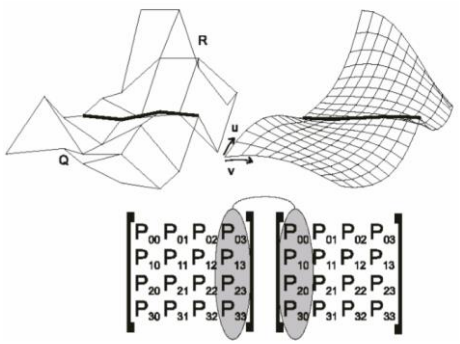
$[V] = [v^3\ \ v^2\ \ v\ \ 1]$

$[U] = [u^3\ \ u^2\ \ u\ \ 1]$

[N] và [M] được biểu diễn = $\begin{bmatrix} -1 & 3 & -1 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$

Nối 2 miếng Bezier bậc 3 (Bi-cubic)

$$Q(u,v) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} B_{00} & B_{01} & B_{02} & B_{03} \\ B_{10} & B_{11} & B_{12} & B_{13} \\ B_{20} & B_{21} & B_{22} & B_{23} \\ B_{30} & B_{31} & B_{32} & B_{33} \end{bmatrix} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$



Hình 2.11. Nối hai mảnh Bezier bậc ba

Hai mảnh Q và R cùng chung tham biến tại biên (Giả sử u), hai đường cong biên phải bằng nhau Q(1,v)=R(0,v). Hệ số của cột cuối ma trận Q = cột đầu ma trận R, tương tự: nếu theo hướng của v thì hàng sẽ thay cột ma trận.

Bậc của mặt cong theo mỗi hướng của tham biến bằng số điểm kiểm soát trừ 1.Tính liên tục hay đạo hàm của mặt theo mỗi tham biến bằng số điểm kiểm soát trừ 2. Hình dạng của mặt biến đổi theo các cạnh của đa giác kiểm soát. Mặt lưới chỉ đi qua các điểm góc cạnh của đa giác kiểm soát, nó chỉ nằm trong phần giới hạn bởi lưới của đa giác lồi kiểm soát và không thay đổi dưới tác động của các phép biến đổi affine. Mỗi đường biên của mặt Bezier là một đường cong Bezier với mặt cong bậc ba Bezier các đường cong biên luôn đảm bảo là các đường Bezier bậc 3. Như vậy lưới đa giác cho bề mặt sẽ là 4 x 4.

2.4.3. Mặt cong B-Spline

Phương trình mặt B-spline:

$$Q(u,w)=\sum_{i=1}^{n}\sum_{j=1}^{m}N_{i,k}(u).M_{j,h}(w).P_{i,j}$$

$$N_{i,k}(u)=\begin{cases}1 & x_i \le u < x_{i+1}\\ 0 & otherwise\end{cases}$$

$$Ni,k(u)=\frac{(u-x_i)N_{i,k-1}(u)}{x_{i+k-1}-x_i}+\frac{(x_{i+k}-u).N_{i+1,k-1}(u)}{x_{i+k}-x_{i+1}}$$

$$\begin{cases}x_i=0(1\le i\le k)\\ x_i=i-k(k+1\le i\le n)\\ x_i=n-k+1(n+1\le i\le n+k)\end{cases}$$

P_{ij} là điểm kiểm soát

N và M là đa thức B-spline

Với các mặt cong mở mặt cong phụ thuộc vào các nút vector

Đặc điểm của mặt cong B-Spline

- Số bậc caonhất của bề mặt theo mỗi hướng thì bằng số điểm kiểm soát -1 theo hướng đó.
- Đạo hàm riêng của phương trình bề mặt theo mỗi tham biến có bậc bằng số điểm kiểm soát theo tham biến đó trừ 2.
- Bề mặt B-spline thì không chịu ảnh hưởng của phép biến đổi anfine. Bề mặt sẽ thay đổi nếu ta thay đổi đa giác kiểm soát.
- Ảnh hưởng của một điểm kiểm soát đơn được giới hạn bởi + - k/2 h/2 khoảng đối với mỗi tham số.
- Nếu số đỉnh của đa giác kiểm soát bằng số bậc theo mỗi tham biến và không có điểm kép nào thì mặt B-spline sẽ chuyển thành mặt Bezier.
- Nếu các đa giác kiểm soát có dạng tam giác thì lưới đa giác kiểm soát sẽ có hình dáng gần giống với bề mặt cong.
- Mỗi mặt B-Spline luôn nằm trong bao lồi của đa giác kiểm soát .
- Mỗi mặt B-Spline có dáng điệu luôn bám theo hình dáng của đa giác kiểm soát.

Tóm tắt:

Việc tạo ra các đường cong theo ý muốn cũng là vấn đề thường gặp khi làm việc với đồ hoạ máy tính. Chúng ta khảo sát cách tiếp cận vẽ đường cong bằng

Hermite, Bezier và B-spline. Các cách tiếp cận này dựa trên cơ sở vẽ đường cong bằng một tập điểm mô tả hình dáng của đường cong gọi là tập điểm kiểm soát. Khi thay đổi tập điểm này, hình dáng của đường cong sẽ thay đổi theo. Cách tiếp cận này cho thấy sự thuận lợi và linh hoạt khi cần phải vẽ các đường cong phức tạp và do đó nó được dùng nhiều trong thiết kế.

Một nhược điểm trong cách vẽ đường cong bằng Bezier là khi một phần của đường cong đã đạt yêu cầu, nhưng khi hiệu chỉnh phần còn lại sẽ mất đi phần đã đạt yêu cầu, hay việc nối trơn các đường cong sẵn có. Để khắc phục các vấn đề này ta có cách tiếp cận cải tiến vẽ đường cong bằng B-spline.

Tương tự như vậy việc biểu diễn các mặt cong trong đồ hoạ máy tính cũng là một vấn đề cần thiết để mô tả đối tượng trong thế giới thực. Chúng ta khảo sát về các phương pháp biểu diễn mặt cong thông qua phương trình tham số. Trong đó, phương trình tham số của một mặt có dạng là một phương trình tham số hai biến p(u,v) và một điểm bất kỳ trên mặt sẽ được biểu diễn dưới dạng p(u,v)= (x(u,v), y(u,v), z(u,v)). Chúng ta khảo sát một số mặt đơn giản như: mặt kẻ, mặt tròn xoay, mặt trượt và mặt Boolean Sum.

Trên cơ sở các đường cong bằng Hermite, Bezier và B-spline chúng ta cũng xây dựng được các mặt Hermite, Bezier và B-spline.

III: SỬ DỤNG THƯ VIỆN OPENGL VẼ ĐƯỜNG CONG VÀ MẶT CONG TRONG 3D

3.1. Chuẩn bị

- Cài đặt công cụ Visual Studio 2013

- Cài đặt thư viện Open GL

3.2. Yêu cầu

- Vẽ đường cong Bezier và mặt cong Bezier trong 3D

3.3. Code và kết quả

Phần code sử dụng các thư viện trong OpenGL đã được thiết lập sẵn thuật toán.

3.3.1. Vẽ đường cong Bezier với điểm điều khiển cho trước

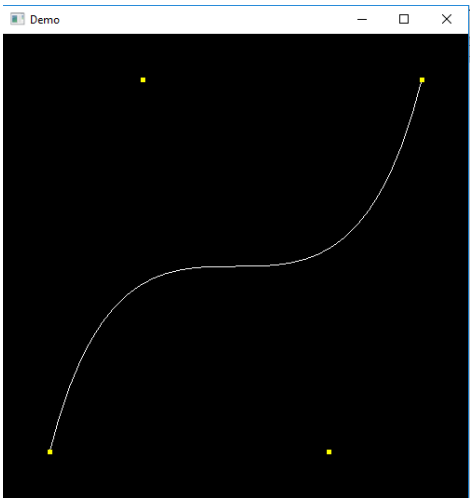
Code:

```
#include <GL/glut.h>                    // GLUT
#include <GL/glu.h>                     // GLU
#include <GL/gl.h>                     // OpenGL
#include <stdlib.h>
```

```
GLfloat ctrlpoints[4][3] = { { -4.0, -4.0, 0.0 }, { -2.0, 4.0, 0.0 }, { 2.0, -4.0, 0.0
}, { 4.0, 4.0, 0.0 } };
void init(void)
{
	glClearColor(0.0, 0.0, 0.0, 0.0);
	glShadeModel(GL_FLAT);
	glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 4, &ctrlpoints[0][0]);
	glEnable(GL_MAP1_VERTEX_3);
}
void display(void)
{
	int i;
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.0, 1.0, 1.0);
	glBegin(GL_LINE_STRIP);
	for (i = 0; i <= 30; i++)
		glEvalCoord1f((GLfloat)i / 30.0);
	glEnd(); /* The following code displays the control points as dots. */
	glPointSize(5.0);
	glColor3f(1.0, 1.0, 0.0);
	glBegin(GL_POINTS);
	for (i = 0; i < 4; i++)
		glVertex3fv(&ctrlpoints[i][0]);
	glEnd();
	glFlush();
}
void reshape(int w, int h)
{
	glViewport(0, 0, (GLsizei)w, (GLsizei)h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	if (w <= h)
		glOrtho(-5.0, 5.0, -5.0*(GLfloat)h / (GLfloat)w, 5.0*(GLfloat)h /
(GLfloat)w, -5.0, 5.0);
	else
		glOrtho(-5.0*(GLfloat)w / (GLfloat)h, 5.0*(GLfloat)w /
(GLfloat)h, -5.0, 5.0, -5.0, 5.0);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}
int main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500, 500);
```

```
	glutInitWindowPosition(100, 100);
	glutCreateWindow("Demo");
	init();
	glutDisplayFunc(display);
	glutReshapeFunc(reshape);
	glutMainLoop();
	return 0;
}
```

Kết quả:



3.3.2. Vẽ mặt cong Bezier với tập điều khiển cho trước

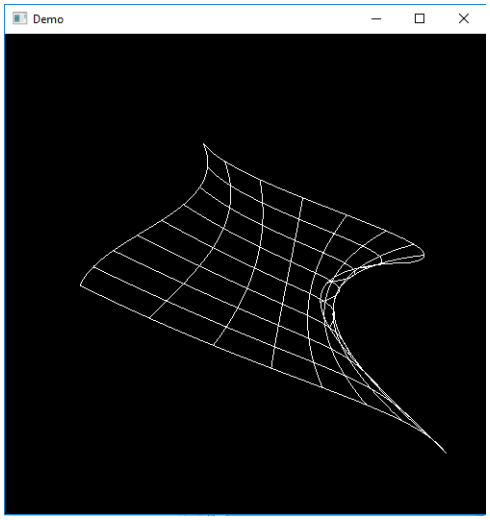
Code:

```
#include <windows.h>
#include <stdlib.h>
#include <math.h>
#include <gl/gl.h>
#include <gl/glu.h>
#include <gl/glut.h>
GLfloat ctrlpoints[4][4][3] = {
	{ { -1.5, -1.5, 4.0 }, { -0.5, -1.5, 2.0 }, { 0.5, -1.5, -1.0 }, { 1.5, -1.5, 2.0 }
},
	{ { -1.5, 0.5, 4.0 }, { -0.5, 0.5, 0.0 }, { 0.5, 0.5, 3.0 }, { 1.5, 0.5, 4.0 } },
	{ { -1.5, -0.5, 1.0 }, { -0.5, -0.5, 3.0 }, { 0.5, -0.5, 0.0 }, { 1.5, -0.5, -1.0 }
},
	{ { -1.5, 1.5, -2.0 }, { -0.5, 1.5, -2.0 }, { 0.5, 1.5, 0.0 }, { 1.5, 1.5, -1.0 } }
};
void display(void)
{
	int i, j;
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glColor3f(1.0, 1.0, 1.0);
```

```
        glPushMatrix();
        glRotatef(85.0, 1.0, 1.0, 1.0);
        for (j = 0; j <= 8; j++)
        {
                glBegin(GL_LINE_STRIP);
                for (i = 0; i <= 30; i++)
                glEvalCoord2f((GLfloat)i / 30.0, (GLfloat)j / 8.0);
                glEnd();
                glBegin(GL_LINE_STRIP);
                for (i = 0; i <= 30; i++)
                glEvalCoord2f((GLfloat)j / 8.0, (GLfloat)i / 30.0);
                glEnd();
        }
        glPopMatrix();
        glFlush();
}
void init(void)
{
        glClearColor(0.0, 0.0, 0.0, 0.0);
        glMap2f(GL_MAP2_VERTEX_3, 0, 1, 3, 4, 0, 1, 12, 4,
&ctrlpoints[0][0][0]);
        glEnable(GL_MAP2_VERTEX_3);
        glEnable(GL_DEPTH_TEST);
        glShadeModel(GL_FLAT);
}
void reshape(int w, int h)
{
        glViewport(0, 0, (GLsizei)w, (GLsizei)h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        if (w <= h)
                glOrtho(-4.0, 4.0, -4.0*(GLfloat)h / (GLfloat)w, 4.0*(GLfloat)h /
(GLfloat)w, -4.0, 4.0);
        else
                glOrtho(-4.0*(GLfloat)w / (GLfloat)h, 4.0*(GLfloat)w /
(GLfloat)h, -4.0, 4.0, -4.0, 4.0);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
}
int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowSize(500, 500);
        glutInitWindowPosition(100, 100);
```

```
        glutCreateWindow("Demo");
        init();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}
```

Kết quả:



4.2.3. Ứng dụng trong khoa học cơ bản

Có thể nói cùng với lý thuyết topo, hình học phân hình đã cung cấp cho khoa học một công cụ khảo sát tự nhiên vô cùng mạnh mẽ, vật lý học và toán học thế kỷ XX đối đầu với sự xuất hiện của tính hỗn độn trong nhiều quá trình có tính quy luật của tự nhiên. Từ sự đối đầu đó, trong những thập niên tiếp theo đã hình thành một lý thuyết mới chuyên nghiên cứu về các hệ phi tuyến, gọi là lý thuyết hỗn độn. Sự khảo sát các bài toán phi tuyến đòi hỏi rất nhiều công sức trong việc tính toán và thể hiện các quan sát một cách trực quan, do đó sự phát triển của lý thuyết này bị hạn chế rất nhiều. Chỉ gần đây với sự ra đời của lý thuyết fractal và sự hỗ trợ đắt lực của máy tình, các nghiên cứu chi tiết về sự hỗn độn mới được đẩy mạnh. Vai trò của hình học phân hình trong lĩnh vực này thể hiện một cách trực quan các cư xử kỳ dị của các tiến trình được khảo sát, qua đó tìm ra được các đặc trưng hoặc các cấu trúc tương tự nhau trong các ngành khoa học khác nhau. Hình học phân hình đã được áp dụng vào nghiên cứu lý thuyết từ tính, lý thuyết các phức chất trong hoá học, lý thuyết tái định chuẩn và phương trình Yang & Lee của vật lý, các nghiệm của các hệ phương trình phi tuyến được giải dựa trên phương pháp xấp xỉ liên tiếp của Newton trong giải tích số,… Các kết quả thu được giữ vai trò rất quan trọng trong các lĩnh vực tương ứng.

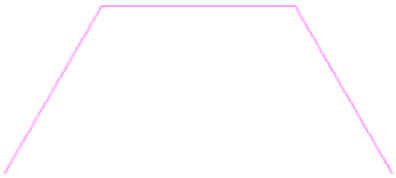
4.3.3. Họ đường Sierpinski

Còn gọi là ‘tam-giác Sierpinski’, bắt-đầu từ một hình tam-giác đều, đáy phẳng, nối liền trun- điểm của ba cạnh thành một hình tam-giác đều nhỏ rồi móc bỏ tam-giác này, Cứ tiếp-tục phân-chia như vậy cho những tam-giác nhỏ hơn cho đến khi không thề phân-chia được nữa. Có thể áp dụng tương tự với khối hình vuông, hoặc khối hình hộp

Đường Sierpinski được trình bày sau đây là một đường cong rất đặc biệt, bởi vì có rất nhiều cách phát sinh ra nó với các khởi động ban đầu hoàn toàn khác nhau nhưng lại kết thúc ở việc sinh ra một loại đường cong duy nhất.

Chúng ta đã quen với phương pháp đầu tiên để phát sinh ra tam giác Sierpinski bằng cách sử dụng kỹ thuật initiator / generator được mô tả ở các phần trước. Đối với đường này, initiator là một đoạn thẳng.

Generator đối với đường cong này và các đường được sinh ra ở mức 1, 2 và 3 được minh hoạ như sau:



Generator của tam giác Sierpinski mức 1

Generator của tam giác Sierpinski mức 2

Generator của tam giác Sierpinski mức 3

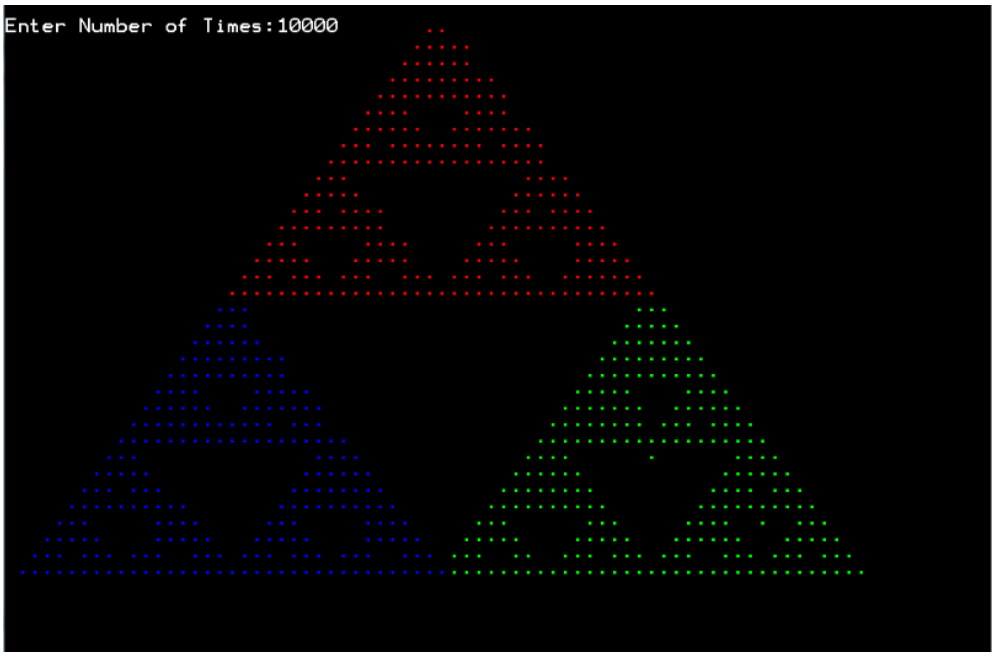
Để phát sinh ra đường này ta dùng kỹ thuật giống như các đường họ Von Kock và Peano.

Đoạn mã của hàm Generator như sau:

```
void Generator_SierpinskiCurve(CDC *pDC, double X1, double Y1, double X2,
double Y2, int Level, int Sign,int NumLines, doubleLinelen, double Angles[],
COLORREF color)
{
CPen pen;
pen.CreatePen(PS_SOLID,1,color);
CPen* pOldPen=pDC->SelectObject(&pen);
 double *XPoints,*YPoints;
 int i;
int Init_Sign;
double Turtle_Theta,TurtleX,TurtleY,TurtleR;
XPoints=new double[NumLines+1];
YPoints=new double[NumLines+1];
TurtleR=sqrt((X2-X1)*(X2-X1)+(Y2-Y1)*(Y2-Y1))*Linelen;
 XPoints[0]=X1;  YPoints[0]=Y1;
XPoints[NumLines]=X2;
YPoints[NumLines]=Y2;
Turtle_Theta=Point(X1,Y1,X2,Y2);  TurtleX=X1;  TurtleY=Y1;
Turn(Angles[0]*Sign,Turtle_Theta);
for(i=1;i<NumLines;i++)
{
Step(TurtleX,TurtleY,TurtleR,Turtle_Theta);
XPoints[i]=TurtleX;   YPoints[i]=TurtleY;
Turn(Angles[i]*Sign,Turtle_Theta);
}
 --Level;
Sign*=-1;
if(Level)
```

```
{
Init_Sign=Sign;
for(i=0; i<NumLines; i++)
{
X1=XPoints[i];
Y1=YPoints[i];
X2=XPoints[i+1];
Y2=YPoints[i+1];
Generator_SierpinskiCurve(pDC, X1, Y1, X2, Y2, Level,
Init_Sign, NumLines, Linelen, Angles, color);
Init_Sign*= -1;
}
}
else
for(i=0;i<NumLines;i++)
{
 pDC->MoveTo((int)XPoints[i],(int)YPoints[i]);
 pDC->LineTo((int)XPoints[i+1],(int)YPoints[i+1]);
}
pDC->SelectObject(pOldPen);
delete []XPoints;
delete []YPoints;
}
```

Với Num-line = 3 và mảng Angle là [60, -60, 0].



4.3.4. Mặt Maldelbrot

Trong nhiều thập niên của thế kỷ XX, các nhà toán học đã để tâm nghiên cứu đến một loại biểu thức phức tạp xác định bởi:

$z_{n+1} = z_n^2 + c$, trong đó z_i C, i N & c C (1)

Để đơn giản hoá vấn đề, trước hết ta xét trường hợp c = 0 và z_0 R. Khi đó có 3 trường hợp sau:

+ $z_0 = 1$: khi đó $z_n = 1, \forall n \geq 1$.
+ $z_0 < 1$: khi đó $z_n \to 0$ khi $n \to \infty$.
+ $z_0 > 1$: khi đó $z_n \to \infty$ khi $n \to \infty$.

Ở đây tốc độ tiến đến 0 hay tiến đến của dãy (z_n) được quyết định bởi giá trị ban đầu z_0 của dãy. Trong trường hợp $z_0 < 1$, giá trị z0 càng nhỏ thì dãy (z_n) tiến đến 0 càng nhanh. Ngược lại khi $z_0 > 1$, giá trị z_0 càng lớn thì dãy (z_n) càng tiến nhanh ra ∞ .

Trong trường hợp tổng quát, dãy (z_n) được xác định bởi công thức (1) ở trên rất khó khảo sát về mặt lý thuyết.

4.3.5. Mặt Julia

Là tập đối tượng fractal có mối quan hệ chặt chẽ với mặt Maldelbrot. Khi phóng to một phần của mặt Maldelbrot ta có được một hình rất giống với mặt Julia

Đối với biểu thức $z_{n+1} = z_n^2 + c$, ngoài hướng đã khảo sát như đã trình bày trong phần tập Mandelbrot, còn có hướng khảo sát khác bằng cách cho c cố định và xem xét dãy (z_n) ứng với mỗi giá trị khác của z_0. Theo hướng này chúng ta sẽ thu được 1 lớp các đối tượng fractal mới được gọi là tập Julia.

Tập Julia và tập Mandelbrot là hai lớp các đối tượng fractal có mối liên hệ rất chặt chẽ với nhau. Một tính chất đáng chú ý là tập Mandelbrot có thể xem như một loại "bản đồ" Mandelbrot có thể cho ra các dạng tập Julia đầy sức lôi cuốn. Các vị trí như vậy được quan sát thấy ở gần biên của tập Mandelbrot. Nhất là gần các chỏm nhọn. Ngoài ra khi phóng to một phần của tập Mandelbrot, ta sẽ thu được một hình rất giống với tập Julia được tạo bởi giá trị của tâm phần được phóng to.

Công thức toán học:

Để thể hiện tập Julia trên màn hình máy tính, ta vẫn sử dụng các công thức như trong phần tập Mandelbrot, như là:

$x_{n+1} = x_n^2 - y_n^2 + p$

$y_{n+1} = 2x_n y_n + q$

Ngoài ra các tính chất đã nêu về giới hạn của dãy (z_0) vẫn được sử dụng cho tập Julia.

Thuật toán thể hiện tập Julia:

Điểm khác biệt so với tập Mandelbrot ở đây là giá trị p và q được giữ cố định, mặt phẳng màn hình biến đổi thành mặt phẳng phức thu hẹp biểu diễn các giá trị của x_0 với:

- Trục x biểu diễn phần thực của số phức z_0.

- Trục y biểu diễn phần ảo của số phức z_0.

Ngoài ra còn có sự phân lớp các giá trị của z_0 như sau:

Lớp 1:

Bao gồm các giá trị (z_0) có $| z_k | < 2$, với $0 \leq k \leq N$ trong đó N là hằng số hữu hạn. Tức là lớp 1 gồm các giá trị z_0 làm cho dãy (z_0) không tiến ra vô cực.

Lớp 2:

Bao gồm các giá trị (z_0) có $| z_n | > 2$, với $n \geq k$, $k \in Z+$, tức là gồm các giá trị làm cho dãy (z_n) tiến ra vô cực.

Ngược lại với tập Mandelbrot, khi thể hiện tập Julia trên màn hình, chúng ta quan tâm đến các giá trị z_0 làm cho dãy (z_n) không hội tụ đến vô cực. Do đó kỹ thuật tô màu của tập Julia vẫn là kỹ thuật xoay vòng nhưng hoàn toàn ngược lại với kỹ thuật tô màu tập Mandelbrot. Trong kỹ thuật tô màu này:

- Các điểm ảnh tương ứng với các giá trị z_0 thuộc lớp 1, sẽ được gán màu tùy thuộc độ lớn của $| z_l|$ với l là ngưỡng quyết định hội tụ của dãy (z_n) đã nêu trong định nghĩa về lớp 1. - Các điểm ảnh tương ứng với giá trị z_0 thuộc lớp 2 sẽ được gán màu trùng với màu nền của bảng màu đang sử dụng.

Với các thay đổi như vậy, tập Julia sẽ được thể hiện bằng thuật toán trình bày như sau:

Bước 1:
Xuất phát với 1 giá trị khởi đầu $z_0 = (x_0, y_0)$ và giá trị cố định c = (p, q).
Bước 2:
Kiểm tra z_0 thuộc lớp 1 hay 2.
Bước 3:
Tô màu điểm ảnh tương ứng với z_0 theo kỹ thuật tô màu được nêu ở trên.

Bước 4:

Chọn giá trị z_0 mới và lặp lại bước 1 cho đến khi đã quét hết tất cả các giá trị z_0 cần khảo sát.

Sử dụng ký hiệu đã được xác định khi trình bày thuật toán Mandelbrot chúng ta có thuật toán tạo tập Julia một cách chi tiết được viết dưới dạng sau:

```
for(Col = 0; Col<Max_Col; ++Col)
{
for(Row=0; Row<= Max_Row; ++Row)
{
x0 = xmin + Col* Δx0;
y0 = ymax - Row* Δy0;
X=Y= 0;
Count =1;
While((Count < Max_Iterations) & (X+Y < 4))
{
X = xn2 ;
Y = yn2 ;
y0 = 2x0y0   + q ;
x0 =  X – Y + p ;
++Count ;
}
if(Count > Max_Iterations)
```

Tô màu điểm ảnh (Col, Row) bởi màu có số hiệu (X + Y) (Max_Color – 1) mod (Max_Color +1) ;

else Tô màu điểm ảnh (Col, Row) bởi màu nền của bảng màu hiện tại;

}

Sự khác biệt chủ yếu giữa thuật toán này với thuật toán Mandelbrot là sự thay đổi vai trò của z_0 và c. Giá trị c = (p,q) được giữ cố định trong khi z_0 thay đổi. Các đại lượng x_0, y_0, x_0, y_0 được xác định theo cách hoàn toàn giống với các đại lượng p, q, p, q trong thuật toán tạo tập Mandelbrot tức là:

$$\Delta x = \frac{x_{\max} - x_{\min}}{Max\_Col}$$

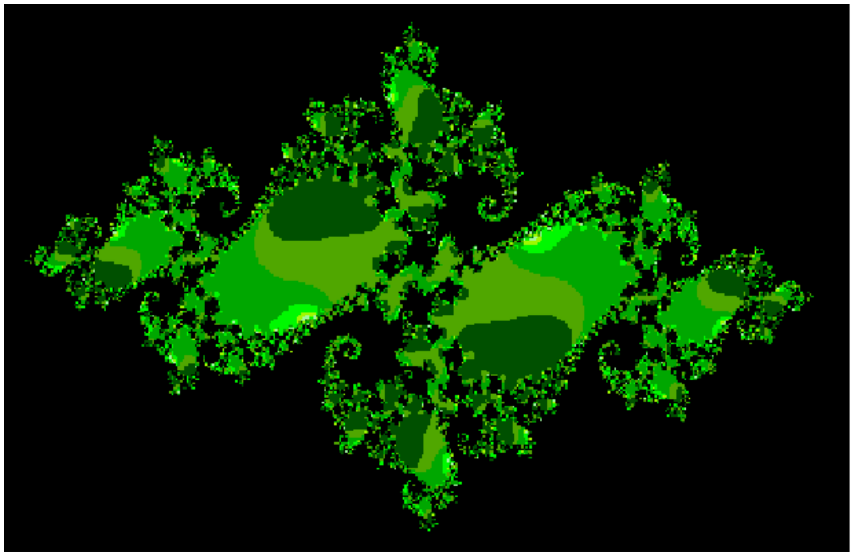
$$\Delta y = \frac{y_{\max} - y_{\min}}{Max\_Row}$$

$x_0 = x_{min} + Col * x0;$

$y_0 = y_{max} – Row * y0;$

Còn có một điểm cần chú ý là các giá trị x_0, y_0 vừa đại diện cho z_0 ban đầu và cũng đại diện cho dãy (z_n) trong vòng lặp kiểm tra z_0 thuộc lớp 1 hay 2.

Kết quả như sau:



4.3.6. Mặt Phonix

Họ đường cong Phoenix do Shigehiro Ushiki ở đại học Tokyo tìm ra. Phương trình của dường cong được xác định bởi

$$Z_{n+1} = z_n^2 + p + q.z_{n-1}$$

Trong đó:

$Z_i \in C$ i, N.

$p = (p, 0) \in C$.

$q = (q, 0) \in C$.

Phương trình được khai triển thành các phần thực và ảo của z_n có dạng: $x_{n+1} = x_n^2 - y_n^2 + p + q.x_{n-1}$

$$y_{n+1} = 2x_n.y_n + q.y_{n-1}$$

với: $x_{n+1} = Re(z_{n+1})$;

$y_{n+1} = Im(z_{n+1})$.

Khi đó việc thể hiện đường cong này lên màn hình gần giống với việc thể hiện tập Julia. Tuy nhiên có hai điểm thay đổi quan trọng:

Thay đổi 1:

- Trục x của màn hình biểu thị phần ảo của số phức z_0.

- Trục y của màn hình biểu thị phần thực của số phức z_0.

Thay đổi 2:

Thay đổi về thuật toán tô màu. Ở đây với các điểm thuộc lớp 1 (theo định nghĩa đã nêu ở phần về tập Julia) chúng ta sẽ sử dụng 3 loại màu tuỳ theo ngưỡng hội tụ:

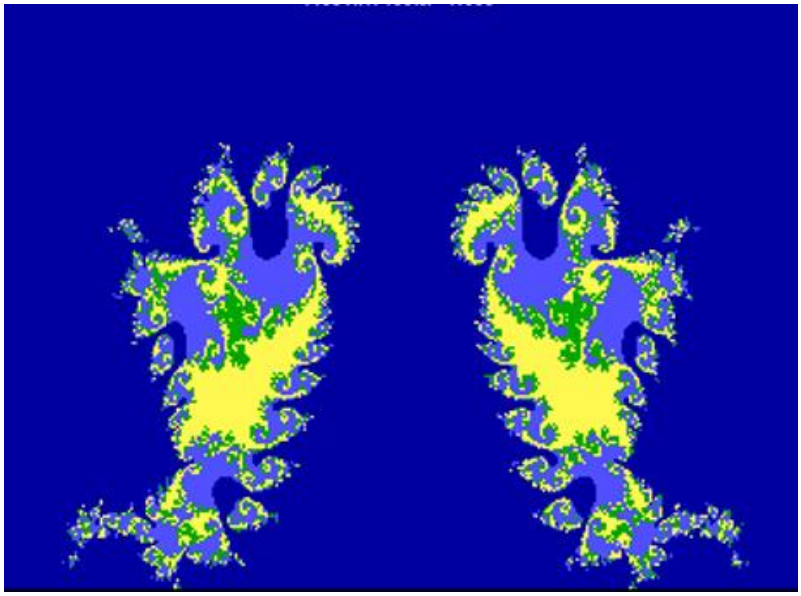
Màu 1: được sử dụng để tô các điểm z_0 cho ra giá trị $| z_k | < 2$ với tối đa k = 32 lần lặp.

Màu 2: được sử dụng để tô các điểm z_0 cho ra giá trị $| z_k | < 2$ với số lần lặp từ 33 đến 64.

Màu 3: được sử dụng để tô các điểm z_0 cho ra giá trị $| z_k | < 2$ với số lần lặp vượt quá 64 lần.

Còn đối với các điểm thuộc lớp 2, chúng ta sẽ tô chúng bằng một màu khác với màu nền hiện tại.

Đây là ảnh của đường cong Phoenix



KẾT LUẬN VÀ KIẾN NGHỊ

Qua Báo cáo về đường cong và mặt cong ta nắm vững khái niệm, thuật toán và cách thức xây dựng lên các đối tượng cấu tạo từ đường cong và mặt cong trong 3D, và ứng dụng của nó trên kĩ thuật đồ họa.

Thông qua báo cáo này giúp sinh viên nắm bắt những kĩ thuật đồ họa đang được sử dụng hàng ngày trong đời sống. Qua đó thấy được sự hữu ích cũng như tương lai của ngành kĩ thuật đồ họa.