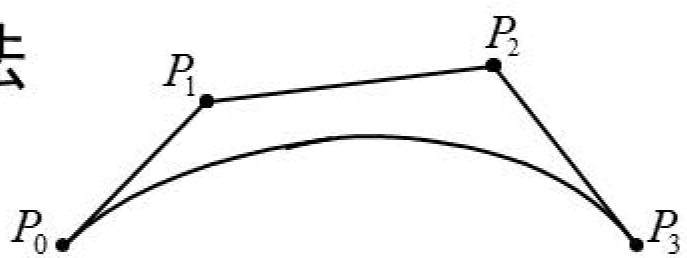
第4-2章抛物样条曲线

课程目标

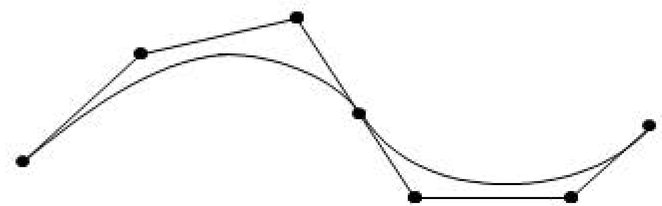
- 过三点定义一段抛物线
- 抛物线的加权合成
- 抛物线的端点条件
- 抛物样条线的性质
- 抛物线绘图函数

抛物样条曲线

- ✓ 用插值的方法生成通过给定型值点的样条曲线
- ✓ 二次Bezier曲线和二次B样条曲线采用逼近方法



Bézier曲线曲面



B样条曲线曲面

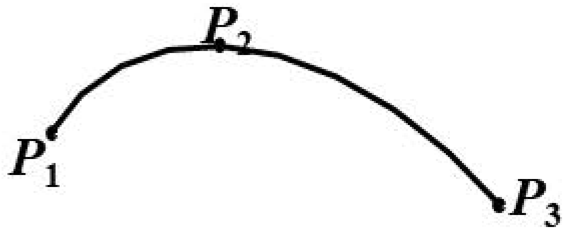
过三点定义一段抛物线

有不在同一直线上的三点：P_1，P_2，P_3，通过三点定义一条抛物线。

抛物线的矢量表达式写成：

$$P(t) = A_1 + A_2 t + A_3 t^2 \quad (0 \le t \le 1)$$

只要确定：A_1，A_2和A_3，就确定抛物线表达式



过三点的二次曲线

过三点定义一段抛物线

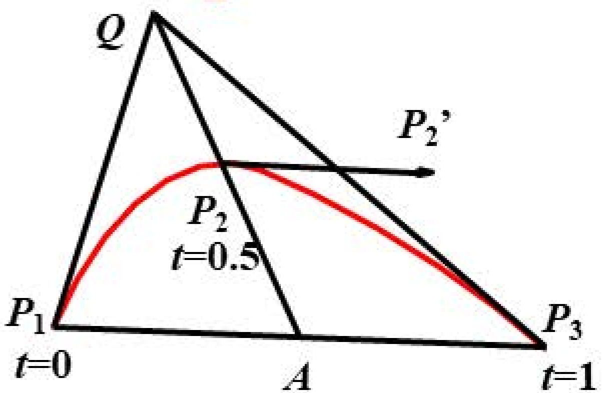
有三个独立的条件

抛物线过P_1，P_2，P_3三点，并且：

①抛物线以P_1为始点。即$t=0$时曲线过P_1

②抛物线以P_3为终点。即$t=1$时曲线过P_3

③当$t=0.5$时曲线过P_2，且切矢量等于P_3-P_1



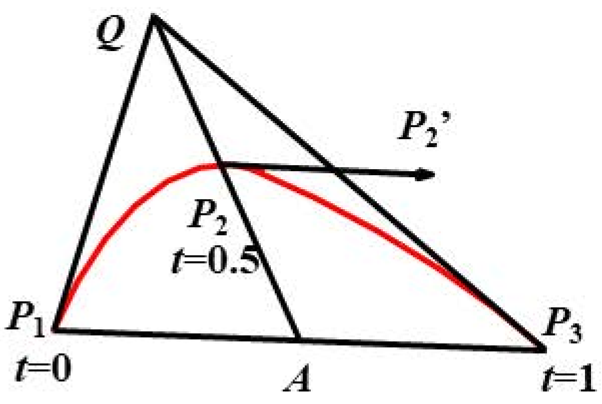
过三点定义一段抛物线

由条件可得

A为P_1P_3的中点，$AP_2=P_2Q$

抛物线P_1处与P_1Q相切，P_3处与QP_3相切

抛物线在P_2处的切矢P'_2与P_1P_3平行



过三点定义一段抛物线

根据设定的条件，可列出三个方程：

t=0时 $P(0)=A_1=P_1$

t=1时 $P(1)=A_1+A_2+A_3=P_3$

t=0.5时 $P(0.5)=A_1+0.5A_2+0.25A_3=P_2$

过三点定义一段抛物线

解以上三个联立方程：

$A_1=P_1$

$P_3=A_1+A_2+A_3=P_1+A_2+A_3$

$\therefore A_2=P_3-P_1-A_3$

$P_2=A_1+0.5A_2+0.25A_3$

$\therefore 4P_2=4A_1+2A_2+A_3=4P_1+2(P_3-P_1-A_3)+A_3=2P_1+2P_3-A_3$

$\therefore A_3=2P_1+2P_3-4P_2$

以上式回代到$A_2=P_3-P_1-A_3$中，得：

$A_2=4P_2-P_3-3P_1$

过三点定义一段抛物线

得到三个系数A_1，A_2，A_3分别为

$A_1 = P_1$

$A_2 = 4P_2-P_3-3P_1$

$A_3 = 2P_1+2P_3-4P_2$

把系数的值，代入到抛物线表达式中

$$P(t)= A_1 + A_2t + A_3t^2$$

$$= P_1 +(4P_2-P_3-3P_1)t+(2P_1+2P_3-4P_2)t^2$$

$$= (2t^2-3t+1)P_1+(4t-4t^2)P_2+(2t^2-t)P_3$$

$(0\leq t\leq 1)$

过三点定义一段抛物线

矩阵形式

$$P(t)=\begin{bmatrix} t^2 & t & 1 \end{bmatrix}\begin{bmatrix} 2 & -4 & 2 \\ -3 & 4 & -1 \\ 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

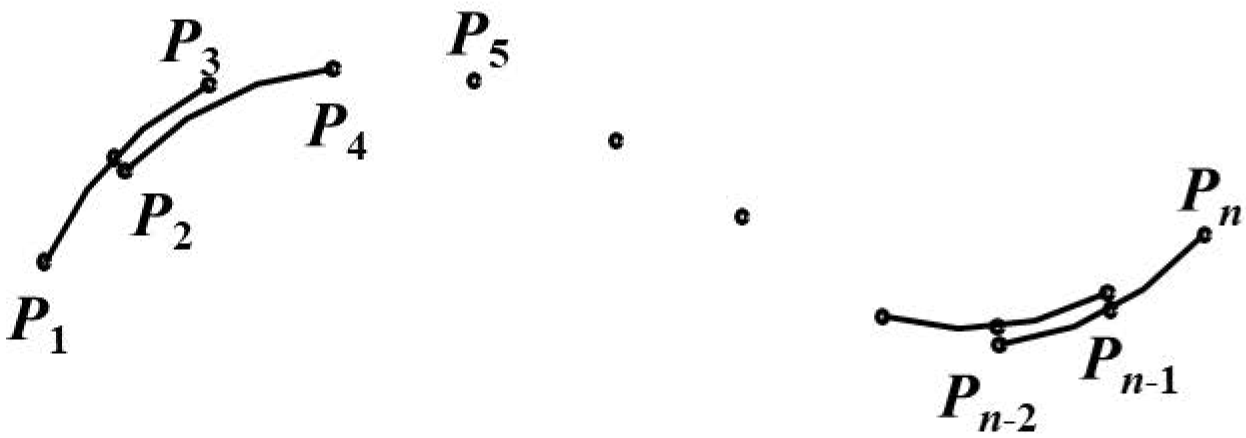
根据参变量t的取值，一一计算出曲线上的数据点，顺次连线绘出图形。

抛物线的加权合成

设有一离散型值点列Pi(i=1,2, …,n)

每经过相邻三点作一段抛物线

有n个型值点，抛物线段一共可以作出n-2条。



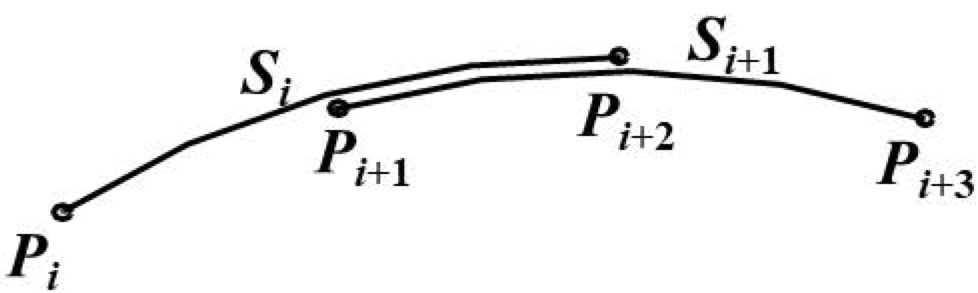
抛物线的加权合成

第i条抛物线段经过P_i，P_{i+1}，P_{i+2}，表达式：

$$S_i(t_i)=(2t_i^2-3t_i+1)P_i+(4t_i-4t_i^2)P_{i+1}+(2t_i^2-t_i)P_{i+2} \qquad (0\leqslant t_i\leqslant 1)$$

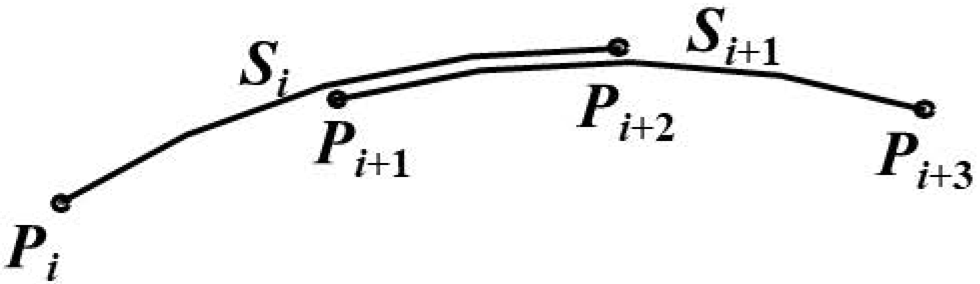
第i+1条抛物线段经过P_{i+1}，P_{i+2}，P_{i+3}，表达式：

$$S_{i+1}(t_{i+1})=(2t_{i+1}^2-3t_{i+1}+1)P_{i+1}+(4t_{i+1}-4t_{i+1}^2)P_{i+2}+(2t_{i+1}2-t_{i+1})P_{i+3} \qquad (0\leqslant t_{i+1}\leqslant 1)$$



抛物线的加权合成

- ✓ 两段曲线的搭接区间是不可能重合。
- ✓ S_i和S_{i+1}两条抛物线，P_{i+1}和P_{i+2}两点间为搭接区间，不太可能会自然地重合成一条曲线。



抛物线的加权合成

- ✓ 整个型值点须用一条光滑的曲线连接。
- ✓ 在S_i和S_{i+1}两条曲线的共同区间内，须让它们结合成一条曲线，办法就是加权合成。

抛物线的加权合成

- ✓ 加权合成，要选择两个合适的**权函数**。
- ✓ 设两个权函数分别为$f(T)$和$g(T)$，加权合成后曲线为$P_{i+1}(t)$

$$P_{i+1}(t)=f(T)\cdot S_i(t_i)+g(T)\cdot S_{i+1}(t_{i+1})$$

抛物线的加权合成

权函数f(T)和g(T)是简单一次函数，有互补性

$$f(T) = 1-T$$

$$g(T) = T \qquad (0 \le T \le 1)$$

$$P_{i+1}(t)=f(T)\cdot S_i(t_i)+g(T)\cdot S_{i+1}(t_{i+1})$$

改写为：

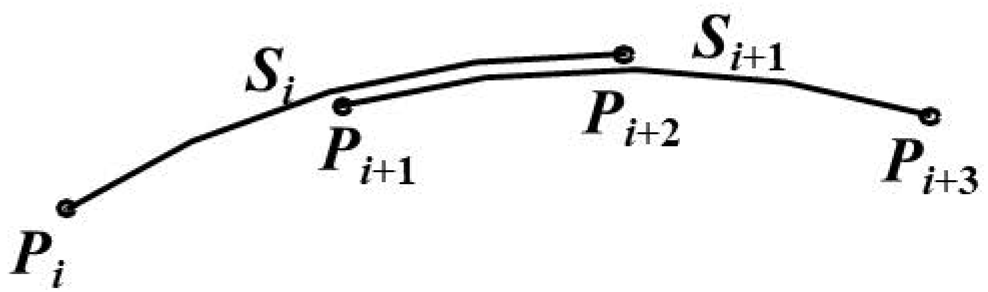
$$P_{i+1}(t) = (1-T)\cdot S_i(t_i) + T\cdot S_{i+1}(t_{i+1})$$

抛物线的加权合成

$P_{i+1}(t) = (1-T)\cdot S_i(t_i) + T\cdot S_{i+1}(t_{i+1})$

三个参变量:T、t_i和t_{i+1}，选择t为统一后的参变量

$0.5 \le ti \le 1$	$t_i = 0.5 + t \quad 0 \le t \le 0.5$
$0 \le t_{i+1} \le 0.5$	$t_{i+1} = t$
$0 \le T \le 1$	$T = 2t$



抛物线的加权合成

根据新的参变量t改写后的形式：

$$P_{i+1}(t)=(1-2t)\cdot S_i(t+0.5)+2t\cdot S_{i+1}(t)$$

其中：

$$1-2t=f(T)$$

$$2t=g(T)$$

$$S_i(t+0.5)=(2t^2-t)P_i+(1-4t^2)P_{i+1}+(2t^2+t)P_{i+2}$$

$$S_{i+1}(t)=(2t^2-3t+1)P_{i+1}+(4t-4t^2)P_{i+2}+(2t^2-t)P_{i+3}$$

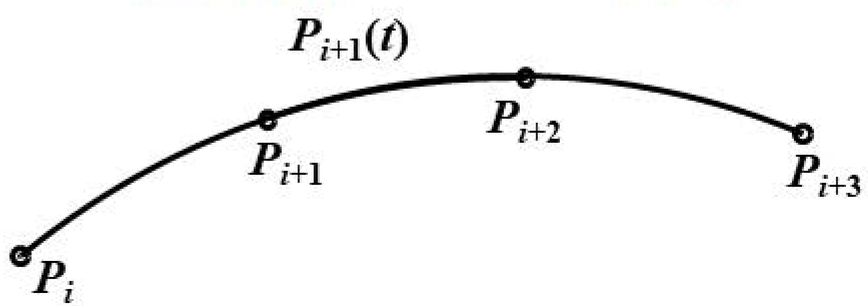
抛物线的加权合成

把以上四式代入式，展开、整理后可得：

$$P_{i+1}(t)=(-4t^3+4t^2-t)P_i+(12t^3-10t^2+1)P_{i+1}+(-12t^3+8t^2+t)P_{i+2}+(4t^3-2t^2)P_{i+3}$$

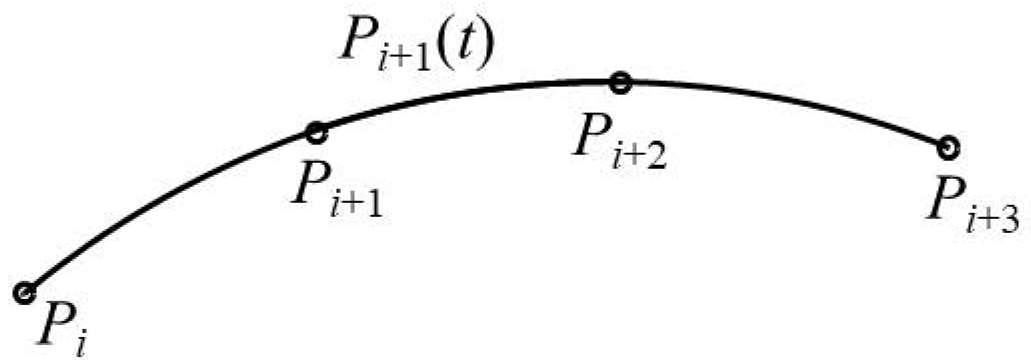
$(i = 1, 2, \ldots, n-3) \qquad (0 \le t \le 0.5)$

该式的实质：每相邻的四个点可以决定中间的一段抛物样条曲线



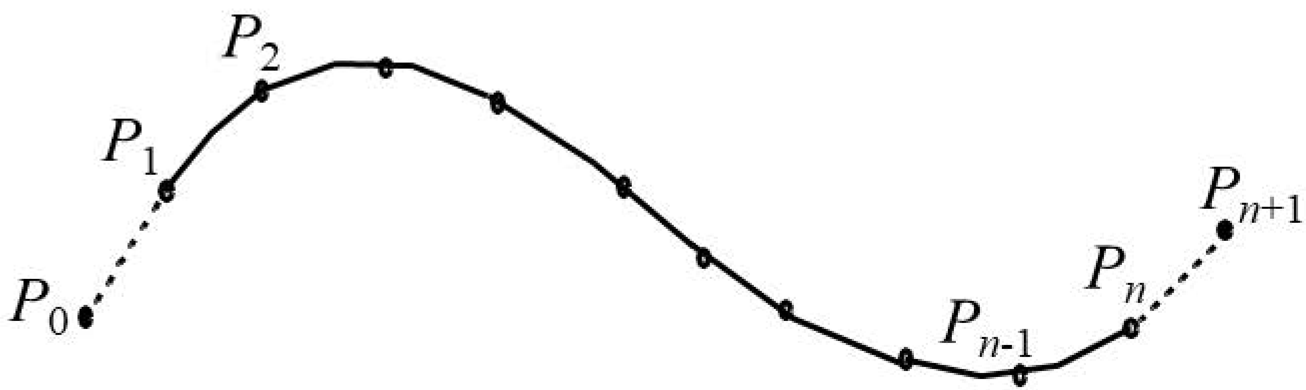
抛物线的端点条件

- ✓ 设离散点列P_i有n个型值点，即i=1,2,…,n
- ✓ 可以加权合成生成n-3段抛物样条曲线。
- ✓ 但n个型值点之间应有n-1个区段。
- ✓ 因其点列的首、尾两段曲线P_1P_2和$P_{n-1}P_n$段，由于缺乏连续相邻的四点而无法产生。



抛物线的端点条件

- ✓ 为了要产生**首尾**两段曲线，在原点列的端点各加一个辅助点P_0和P_{n+1}



抛物线的端点条件

✓ P_0和P_{n+1}两点如何加上去，依据什么原则，这就是 的“端点条件”。

✓ 常用的三种方法

- 已知两端的切矢P'_1和P'_n
- 自由端条件
- 形成封闭曲线

抛物线的端点条件

①已知两端的切矢P'_1和P'_n

- 由P_1、P_2、P_3确定的抛物线，过P_2点曲线的切矢

  $P'_2=P_3-P_1$，即：$P_1=P_3-P'_2$

- 则两端的切矢P'1和P'n

  $P'_1=P_2-P_0$　　∴ $P_0=P_2-P'_1$

  $P'_n=P_{n+1}-P_{n-1}$　　∴ $P_{n+1}=P_{n-1}+P'_n$

- 即可以确定辅助点P_0和P_{n+1}的坐标位置。

抛物线的端点条件

②自由端条件

- 所补的点P_0和P_{n+1}与原两端点P_1和P_n分别重合

$$P_0=P_1$$

$$P_{n+1}=P_n$$

- 适用于对曲线的两端没有什么特殊的要求。

抛物线的端点条件

③形成封闭曲线

- 在n个型值点之间形成封闭曲线，要生成n段曲线段。
- 补点工作中要加三个点，首先让首尾两点重合，然后各向前后延长一点。
- 即：

$$P_{n+1}=P_1$$

$$P_0=P_n$$

$$P_{n+2}=P_2$$

抛物样条曲线的性质

✓ 拿什么标准作为评价曲线光滑程度的指标?

- 是否光滑：两段曲线在节点处的导数是否相等?
- 光滑程度：导数的阶次
- 连续的阶次愈高，曲线愈光滑，设计工作增加难度、提高成本
- 一般应用，C^1连续就可以

抛物样条曲线的性质

✓假设两个曲线段$P_{i+1}(t)$和$P_{i+2}(t)$在节点P处相连，曲线在节点P处达到C^1连续。

- 在P点处，　$P_{i+1}(t)$的参变量$t=0.5$
- 在P点处，　$P_{i+2}(t)$的参变量$t=0$
- 假如$P'_{i+1}(0.5)= P'_{i+2}(0)$成立，那么曲线在节点P处达到$C^1$连续。

抛物样条曲线的性质

$$P'_{i+1}(t)=(-12t^2+8t-1)P_i+(36t^2-20t)P_{i+1}+$$
$$(-36t^2+16t+1)P_{i+2}+(12t^2-4t)P_{i+3}$$
$$=P_{i+3}-P_{i+1} \qquad (t=0.5)$$

$$P'_{i+2}(t)=(-12t^2+8t-1)P_{i+1}+(36t^2-20t)P_{i+2}+$$
$$(-36t^2+16t+1)P_{i+3}+(12t^2-4t)P_{i+4}$$
$$=P_{i+3}-P_{i+1} \qquad (t=0)$$

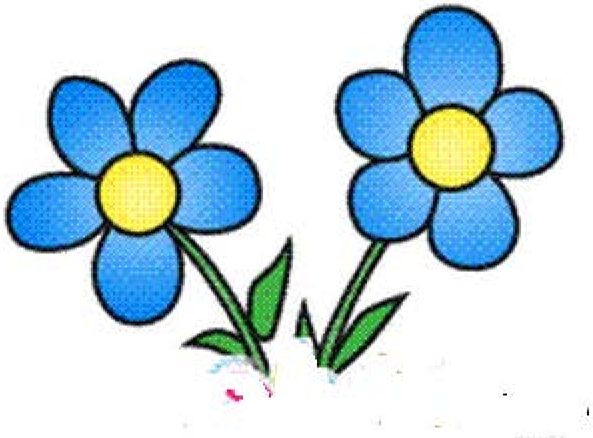
- 可得 $P'_{i+1}(0.5)=P'_{i+2}(0)$
- 抛物样条曲线可以达到C^1连续。

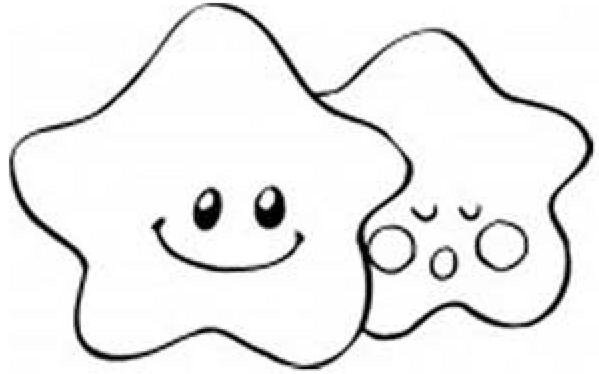
抛物线绘图函数

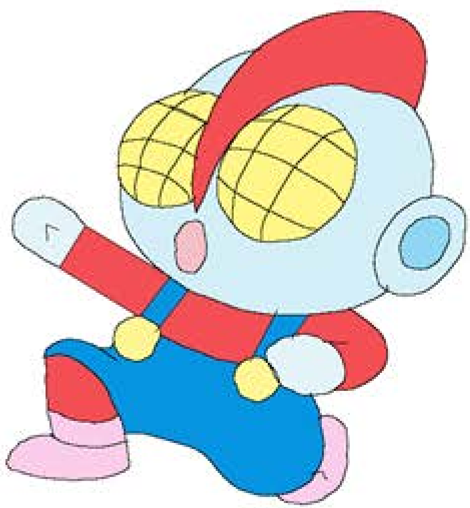
```
typedef struct point {  int x,y;}POINT;
void Parabola(POINT *p,int n)
{//n为型值点数，k为插值数，即是把参变量t区间细分的份数。
   int x,y,i,j,k=10;      double t1,t2,t3,t,a,b,c,d; t=0.5/k;
   p[0].x=p[1].x;p[0].y=p[1].y;
   p[n+1].x=p[n].x;p[n+1].y=p[n].y;
   moveto(p[1].x,p[1].y);
   for(i=1;i<n;i++){
     for(j=1;j<k;j++){
       t1=j*t;t2=t1*t1;t3=t2*t1;
       a=4.0*t2-t1-4.0*t3; b=1.0-10.0*t2+12.0*t3;
       c=t1+8.0*t2-12.0*t3; d=4.0*t3-2.0*t2;
       x=(int)(a*p[i-1].x+b*p[i].x+c*p[i+1].x+d*p[i+2].x);
       y=(int)(a*p[i-1].y+b*p[i].y+c*p[i+1].y+d*p[i+2].y);
       lineto(x,y);
     }
     lineto(p[i+1].x,p[i+1].y);
   }
}
```

抛物线绘图函数

```
void  parspl (int p[], int n, int k, int e) {
//P为型值点的坐标数组，n为型值点数，k为插值数，即是把参变量t区间细分的份数。
  int  x, y, i, j, m=n;
  float t1, t2, t3, t, a, b, c, d;
  if (e==1) //自由端 {
    p[0].x = p[1].x; p[0].y = p[1].y;
    p[n+1].x = p[n].x; p[n+1].y = p[n].y;
  }
  else  //画封闭曲线 {
    p[0].x = p[n].x; p[0].y = p[n].y;
    p[m].x = p[1].x; p[m].y = p[1].y;
    p[m+1].x = p[2].x; p[m+1].y = p[2].y;
  }
```








Snoopy

玫瑰花

西班牙程序员 Roman Cortes 用纯 javascript 脚本编写了红色玫瑰花

把这个精彩的程序移植到了 VC 上

玫瑰花

```
#include <graphics.h>
#include <conio.h>
#include <math.h>


// 定义全局变量
int rosesize = 500;
int h = -250;


// 定义结构体
struct DOT
{
    double x;
    double y;
    double z;
    double r;   // 红色
    double g;   // 绿色
    // b(蓝色) 通过 r 计算
};


// 计算点
bool calc(double a, double b, double c, DOT& d)
{
    double j, n, o, w, z;

    if (c > 60)             // 花柄
    {
        d.x = sin(a * 7) * (13 + 5 / (0.2 + pow(b * 4, 4))) - sin(b) * 50;
        d.y = b * rosesize + 50;
        d.z = 625 + cos(a * 7) * (13 + 5 / (0.2 + pow(b * 4, 4))) + b * 400;
        d.r = a * 1 - b / 2;
        d.g = a;
        return true;
    }

    double A = a * 2 - 1;
    double B = b * 2 - 1;
    if (A * A + B * B < 1)
    {
        if (c > 37)         // 叶
        {
            j = (int(c) & 1);
            n = j ? 6 : 4;
            o = 0.5 / (a + 0.01) + cos(b * 125) * 3 - a * 300;
            w = b * h;

            d.x = o * cos(n) + w * sin(n) + j * 610 - 390;
            d.y = o * sin(n) - w * cos(n) + 550 - j * 350;
            d.z = 1180 + cos(B + A) * 99 - j * 300;
            d.r = 0.4 - a * 0.1 + pow(1 - B * B, -h * 6) * 0.15 - a * b * 0.4 + cos(a + b) / 5 + pow(cos((o * (a + 1) + (B > 0 ? w : -w)) / 25), 30) * 0.1 * (1 - B * B);
            d.g = o / 1000 + 0.7 - o * w * 0.000003;
            return true;
        }
        if (c > 32)         // 花萼
        {
```

玫瑰花

```
            c = c * 1.16 - 0.15;
            o = a * 45 - 20;
            w = b * b * h;
            z = o * sin(c) + w * cos(c) + 620;

            d.x = o * cos(c) - w * sin(c);
            d.y = 28 + cos(B * 0.5) * 99 - b * b * b * 60 - z / 2 - h;
            d.z = z;
            d.r = (b * b * 0.3 + pow((1 - (A * A)), 7) * 0.15 + 0.3) * b;
            d.g = b * 0.7;
            return true;
        }

        // 花
        o = A * (2 - b) * (80 - c * 2);
        w = 99 - cos(A) * 120 - cos(b) * (-h - c * 4.9) + cos(pow(1 - b, 7)) * 50 + c * 2;
        z = o * sin(c) + w * cos(c) + 700;

        d.x = o * cos(c) - w * sin(c);
        d.y = B * 99 - cos(pow(b, 7)) * 50 - c / 3 - z / 1.35 + 450;
        d.z = z;
        d.r = (1 - b / 1.2) * 0.9 + a * 0.1;
        d.g = pow((1 - b), 20) / 4 + 0.05;
        return true;
    }

    return false;
}


// 主函数
int main()
{
    // 定义变量
    short* zBuffer;
    int     x, y, z, zBufferIndex;
    DOT     dot;

    // 初始化
    initgraph(640, 480);                // 创建绘图窗口
    setbkcolor(WHITE);                  // 设置背景色为白色
    cleardevice();                      // 清屏

    // 初始化 z-buffer
    zBuffer = new short[rosesize * rosesize];
    memset(zBuffer, 0, sizeof(short) * rosesize * rosesize);

    for (int j = 0; j < 2000 && !_kbhit(); j++) // 按任意键退出
    {
        for (int i = 0; i < 10000; i++)         // 减少是否有按键的判断
            if (calc(double(rand()) / RAND_MAX, double(rand()) / RAND_MAX, rand() % 46 / 0.74, dot))
            {
                z = int(dot.z + 0.5);
                x = int(dot.x * rosesize / z - h + 0.5);
                y = int(dot.y * rosesize / z - h + 0.5);
                if (y >= rosesize) continue;
```

```
                zBufferIndex = y * rosesize + x;

                if (!zBuffer[zBufferIndex] || zBuffer[zBufferIndex] > z)
                {
                    zBuffer[zBufferIndex] = z;

                    // 画点
                    int r = ~int((dot.r * h));              if (r < 0) r = 0;   if (r > 255) r = 255;
                    int g = ~int((dot.g * h));              if (g < 0) g = 0;   if (g > 255) g = 255;
                    int b = ~int((dot.r * dot.r * -80));    if (b < 0) b = 0;   if (b > 255) b = 255;
                    putpixel(x + 50, y - 20, RGB(r, g, b));
                }
            }

        Sleep(1);
    }

    // 退出
    delete[]zBuffer;
    _getch();
    closegraph();
    return 0;
}
```