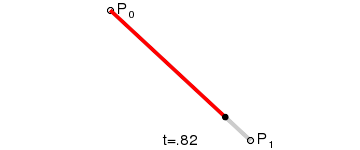
**Bezier曲线的原理**

Bezier曲线是应用于二维图形的曲线。曲线由顶点和控制点组成，通过改变控制点坐标可以改变曲线的形状。

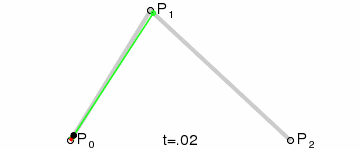
一次Bezier曲线公式：



C:\Users\wyz\Documents\My Knowledge\temp\c0b175a5-31ef-4bef-83cd-9227e781a0cc_4_files\6e53eb2c-440e-4501-afe2-87c3f61ac947.png

一次Bezier曲线是由P0至P1的连续点，描述的一条线段

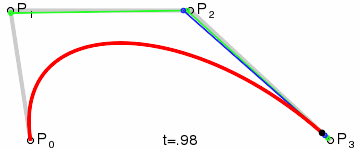
二次Bezier曲线公式：



C:\Users\wyz\Documents\My Knowledge\temp\c0b175a5-31ef-4bef-83cd-9227e781a0cc_4_files\0c6e449e-0cc1-4ccf-83f8-75a831be22c7.png

二次Bezier曲线是 P0至P1 的连续点Q0和P1至P2 的连续点Q1 组成的线段上的连续点B(t)，描述一条抛物线。

三次Bezier曲线公式：



C:\Users\wyz\Documents\My Knowledge\temp\c0b175a5-31ef-4bef-83cd-9227e781a0cc_4_files\9e08532b-7767-49c1-9dcf-bd9ec8ecda50.png

 阶贝塞尔曲线可如下推断。给定点**P**0、**P**1、…、**P**n，其贝塞尔曲线即

{\mathbf  {B}}(t)=\sum _{{i=0}}^{n}{n \choose i}{\mathbf  {P}}_{i}(1-t)^{{n-i}}t^{i}={n \choose 0}{\mathbf  {P}}_{0}(1-t)^{n}t^{{0}}+{n \choose 1}{\mathbf  {P}}_{1}(1-t)^{{n-1}}t^{{1}}+\cdots +{n \choose n-1}{\mathbf  {P}}_{n}(1-t)^{{1}}t^{{n-1}}+{n \choose n}{\mathbf  {P}}_{n}(1-t)^{{0}}t^{n}{\mbox{ , }}t\in [0,1]。

其中C:\Users\wyz\Documents\My Knowledge\temp\c0b175a5-31ef-4bef-83cd-9227e781a0cc_4_files\da33e828-c3d9-45aa-a95e-61063b83ed8b.jpg为（a+b）的n次方展开后，第i项的系数，即C:\Users\wyz\Documents\My Knowledge\temp\c0b175a5-31ef-4bef-83cd-9227e781a0cc_4_files\f7135024-dc90-4dcc-b4b4-73decb4a0350.jpg

例如：

。

**二次Bezier曲线的实现**

**[cpp]** [view plaincopy](http://blog.csdn.net/jimi36/article/details/7792103)

1. #ifndef CBEZIERCURVE\_H\_
2. #define CBEZIERCURVE\_H\_
4. #include <vector>
6. class CBezierCurve
7. {
8. public:
9. CBezierCurve();
10. ~CBezierCurve();
12. void SetCtrlPoint(POINT& stPt);
14. bool CreateCurve();
16. void Draw(CDC\* pDC);
18. private:
19. // 主要算法，计算曲线各个点坐标
20. void CalCurvePoint(float t, POINT& stPt);
22. private:
23. // 顶点和控制点数组
24. std::vector<POINT> m\_vecCtrlPt;
25. // 曲线上各点坐标数组
26. std::vector<POINT> m\_vecCurvePt;
27. };
29. #endif

**[html]** [view plaincopy](http://blog.csdn.net/jimi36/article/details/7792103)

1. #include <math.h>
2. #include "BezierCurve.h"
4. CBezierCurve::CBezierCurve()
5. {
6. }
8. CBezierCurve::~CBezierCurve()
9. {
10. }
12. void CBezierCurve::SetCtrlPoint(POINT& stPt)
13. {
14. m\_vecCtrlPt.push\_back(stPt);
15. }
17. void CBezierCurve::CreateCurve()
18. {
19. // 确保是二次曲线，2个顶点一个控制点
20. assert(m\_vecCtrlPt.size() == 3);
22. // t的增量， 可以通过setp大小确定需要保存的曲线上点的个数
23. float step = 0.01;
24. for (float t = 0.0; t <= 1.0; t += step)
25. {
26. POINT stPt;
27. CalCurvePoint(t, stPt);
28. m\_vecCurvePt.push\_back(stPt);
29. }
30. }
32. void CBezierCurve::Draw(CDC\* pDC)
33. {
34. // 画出曲线上个点，若不连续可以用直线连接各点
35. int nCount = m\_vecCurvePt.size();
36. for (int i = 0; i < nCount; ++i)
37. {
38. pDC->SetPixel(m\_vecCurvePt[i], 0x000000);
39. }
40. }
42. void CBezierCurve::CalCurvePoint(float t, POINT& stPt)
43. {
44. // 确保是二次曲线，2个顶点一个控制点
45. assert(m\_vecCtrlPt.size() == 3);
47. // 计算曲线点坐标，此为2次算法，改变此处可以实现多次曲线
48. float x = (float)m\_vecCtrlPt[0].x \* pow(1 - t, 2)   +
49. (float)m\_vecCtrlPt[1].x \* t \* (1 - t) \* 2 +
50. (float)m\_vecCtrlPt[2].x \* pow(t, 2);
51. float y = (float)m\_vecCtrlPt[0].y \* pow(1 - t, 2)   +
52. (float)m\_vecCtrlPt[1].y \* t \* (1 - t) \* 2 +
53. (float)m\_vecCtrlPt[2].y \* pow(t, 2);
54. stPt.x =x;
55. stPt.y= y;
56. }

来源： <<http://blog.csdn.net/jimi36/article/details/7792103>>

**二、Bezier曲面**

1. **1、定义**

[clip_image002[20]](http://www.tanglei.name/wp-content/uploads/2012/10/clip_image00220.png)

其中，[clip_image004[7]](http://www.tanglei.name/wp-content/uploads/2012/10/clip_image0047.png),S(*u*,*v*)称为Bezier曲面，*Bi*,*n*(*t*)称为Bernstein基函数，P*i*,*j*称为控制顶点。