

合肥工业大学

《计算机图形学》实验报告

学 号 2017218007

姓 名 文 华

专 业 班 级 物联网工程 17-2 班

指 导 老 师 闵 海

合肥工业大学

2020 年 12 月

目录

| | | |
|-------|--------------------------|----|
| 1 | 实验 2：直线的生成 | 1 |
| 1.1 | 实验要求和目的 | 1 |
| 1.2 | 实验课时 | 1 |
| 1.3 | 实验环境 | 1 |
| 1.4 | 实验内容 | 1 |
| 1.5 | 核心代码 | 3 |
| 1.6 | 实验结果 | 7 |
| 1.6.1 | DDA 算法 | 10 |
| 1.6.2 | Mid-Bresenham 算法 | 11 |
| 1.7 | 心得与体会 | 12 |
| 2 | 实验 4：BSpline 曲线绘制 | 13 |
| 2.1 | 实验要求和目的 | 13 |
| 2.2 | 实验课时 | 13 |
| 2.3 | 实验环境 | 13 |
| 2.4 | 实验内容 | 13 |
| 2.5 | 核心代码 | 16 |
| 2.6 | 实验结果 | 18 |
| 2.6.1 | B-样条算法 | 19 |
| 2.6.2 | Bezeir 算法 | 22 |
| 2.7 | 心得与体会 | 24 |
| 附录 | | 25 |
| | BSpline 曲线控制点的测试数据 | 25 |
| | 数据 1 | 25 |
| | 数据 2 | 27 |
| | 数据 3 | 29 |
| | 数据 4 | 30 |
| | 数据 5 | 31 |

| | |
|-----------|----|
| 数据 6..... | 33 |
| 数据 7..... | 36 |
| 数据 8..... | 38 |

1 实验 2：直线的生成

1.1 实验要求和目的

理解直线生成的原理；掌握典型直线生成算法；掌握步处理、分析实验数据的能力；

编程实现 DDA 算法、Bresenham 中点算法；对于给定起点和终点的直线，分别调用 DDA 算法和 Bresenham 中点算法进行批量绘制，并记录两种算法的绘制时间；利用 excel 等数据分析软件，将试验结果编制成表格，并绘制折线图比较两种算法的性能。

1.2 实验课时

3 学时

1.3 实验环境

本试验提供自带实验平台

- 开发环境：Visual C++ 6.0
- 实验平台：Free_Curve（自制平台）

1.4 实验内容

本实验提供名为 Experiment_Frame_One 的平台，该平台提供基本绘制、设置、输入功能，学生在此基础上实现 DDA 算法和 Mid_Bresenham 算法，并进行分析。

- 平台界面：如图 1.4.1 所示
- 设置：通过 view->setting 菜单进入，如图 1.4.2 所示
- 输入：通过 view->input...菜单进入，如图 1.4.3 所示
- 实现算法：

- DDA 算法：void CExperiment_Frame_OneView::DDA(int X0, int Y0, int X1, int Y1)

- Mid_Bresenham 算法：void CExperiment_Frame_OneView::Mid_Bresenham(int X0, int Y0, int X1, int Y1)

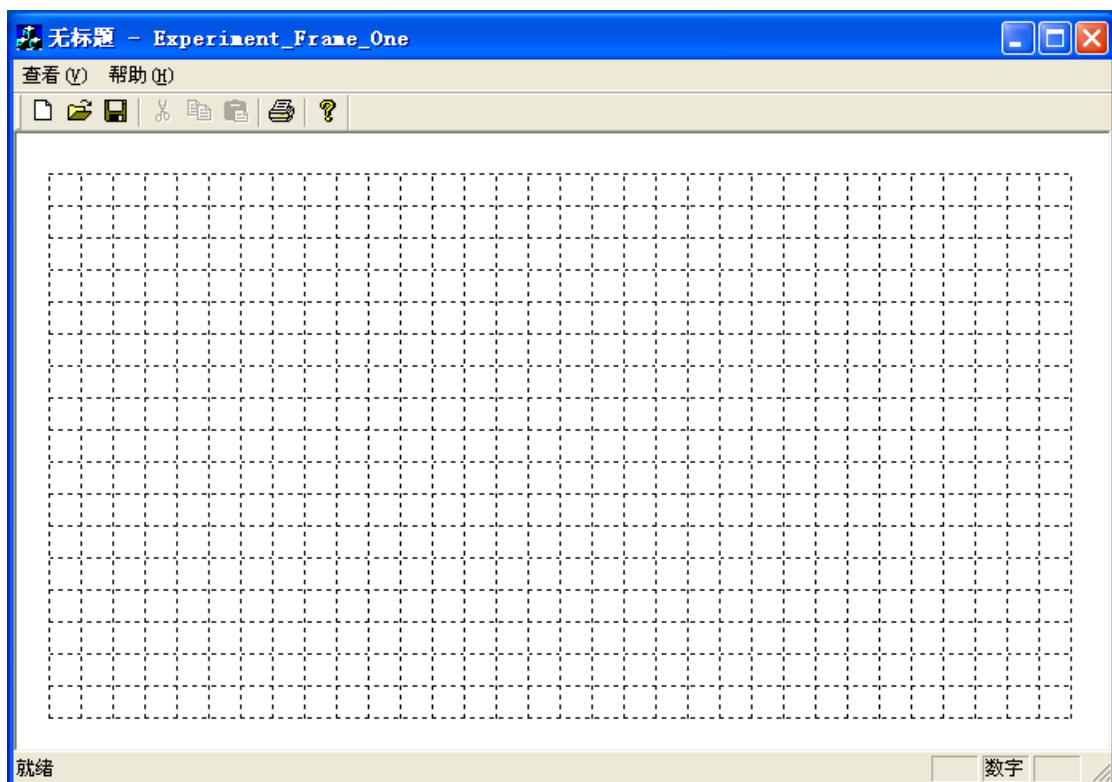


图 1.4.1 总界面



图 1.4.2 设置界面

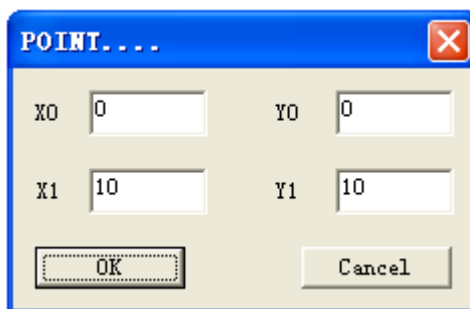


图 1.4.3 输入界面

1.5 核心代码

本次实验的核心代码如下所示。

```
//-----算法实现-----//
//绘制像素的函数DrawPixel(x, y);
void CExperiment_Frame_OneView::DDA(int X0, int Y0, int X1, int Y1)
{
    //-----请实现DDA算法-----//
    float k, b;
    k = float((Y1 - Y0)) / float(X1 - X0);
    b = float(X1 * Y0 - X0 * Y1) / float(X1 - X0);

    if (fabs(k) <= 1) {
        int x;
        float y;
        x = X0;
        y = Y0;
        while (x <= X1) {
            DrawPixel(x, round(y));
            x++;
            y += k;
        }
    }
    else {
        float x;
        int y;
        x = X0;
        y = Y0;
        while (y <= Y1) {
            DrawPixel(x, round(y));
            y++;
            x += 1 / k;
        }
    }
}
```

```

        return;
    }

void CExperiment_Frame_OneView::Mid_Bresenham(int x0, int y0, int x1, int y1)
{
    // AllocConsole();
    //-----实现Mid-Bresenham画线算法-----//
    if (x1 < x0) {
        int temp = x1;
        x1 = x0;
        x0 = temp;
        temp = y1;
        y1 = y0;
        y0 = temp;
    }

    int dx = x1 - x0;
    int dy = y1 - y0;

    bool y_flag = false; // 用于判断|k|与1的关系
    if (int_abs(y1 - y0) > int_abs(x1 - x0)) {
        y_flag = true;
    }

    if (y_flag) {
        int temp = dx;
        dx = dy;
        dy = temp;
    }

    int twoD1 = -2 * dy;
    int twoD2 = 2 * dx - 2 * dy;

    int dd = 1;

    int D = dx - 2 * dy;

    bool flag = false;
    if ((y1 - y0)*(x1 - x0) < 0) { // 若k<0, 则flag为true
        flag = true;
    }
}

```

```

if (flag) {
    D = -dx - 2 * dy;
    dd = -1;
    twoD2 = -2 * dx - 2 * dy;
}

int x = x0;
int y = y0;
if (!y_flag && !flag) { // 0 < k <= 1的情况
    while (x <= x1) {
        DrawPixel(x, y);

        if (D >= 0) {
            D += twoD1;
        }
        else {
            y += dd;
            D += twoD2;
        }

        x++;
    }
}
else if (!y_flag && flag) { // -1 < k < 0的情况
    while (x <= x1) {
        DrawPixel(x, y);

        if (D >= 0) {
            y += dd;
            D += twoD2;
        }
        else {
            D += twoD1;
        }

        x++;
    }
}
else if (y_flag && !flag) { // 1 < k 的情况
    while (y <= y1) {
        DrawPixel(x, y);

        if (D >= 0) {
            D += twoD1;
        }
        else {

```



```

        x += dd;
        D += twoD2;
    }

    y++;
}
}
else if (y_flag && flag) { // k < -1 的情况
    y = y0;
    dd = 1;
    dx = x1 - x0;
    dy = y1 - y0;
    twoD1 = -2 * dx;
    twoD2 = -2 * dx - 2 * dy;
    D = -2 * dx - dy;
    while (y >= y1) {
        _cprintf("%d %d\n", x, y);
        DrawPixel(x, y);

        _cprintf("D: %d\n", D);
        if (D >= 0) {
            D += twoD1;
        }
        else {
            x += dd;
            D += twoD2;
        }

        y--;
    }
}

}

/*
// 参考书上的代码
void CExperiment_Frame_OneView::Mid_Bresenham(int X0, int Y0, int X1, int Y1)
{
    //-----请实现Mid_Bresenham算法-----//
    AllocConsole();
    int dX = abs(X1 - X0);
    int dY = abs(Y1 - Y0);
    int X = X0, Y = Y0;
    int twoDX = 2 * dX;
    int twoDY = 2 * dY;
    int D = 2 * dY - dX; // D>=0则yi+1 = yi + 1

```

```

int s1, s2, interchange = 0; // interchange == 1 表示直线斜率大于45度
if (X1 - X0 >= 0) {
    s1 = 1;
}
else {
    s1 = -1;
}
if (Y1 - Y0 >= 0) {
    s2 = 1;
}
else {
    s2 = -1;
}
if (dY > dX) {
    int temp = dX;
    dX = dY;
    dY = temp;
    interchange = 1;
}
for (int i = 0; i <= dX; i++) {
    DrawPixel(X, Y);
    if (D > 0) {
        if (interchange) { X += s1; }
        else {
            Y += s2;
            D -= twoDX;
        }
    }
    if (interchange) { Y += s2; }
    else { X += s1; }
    D += twoDY;
}
}
*/

```

1.6 实验结果

程序的启动界面，如图 1.6.1 所示。

程序的查看选项，如图 1.6.2 所示。

程序的坐标输入，如图 1.6.3 所示。

程序的算法选择，如图 1.6.4 所示。

DDA 算法在不同的坐标下的输出，如图 1.6.5 至图 1.6.7 所示。

Mid-Bresenham 算法在不同的坐标下的输出，如图 1.6.8 至图 1.6.9 所示。

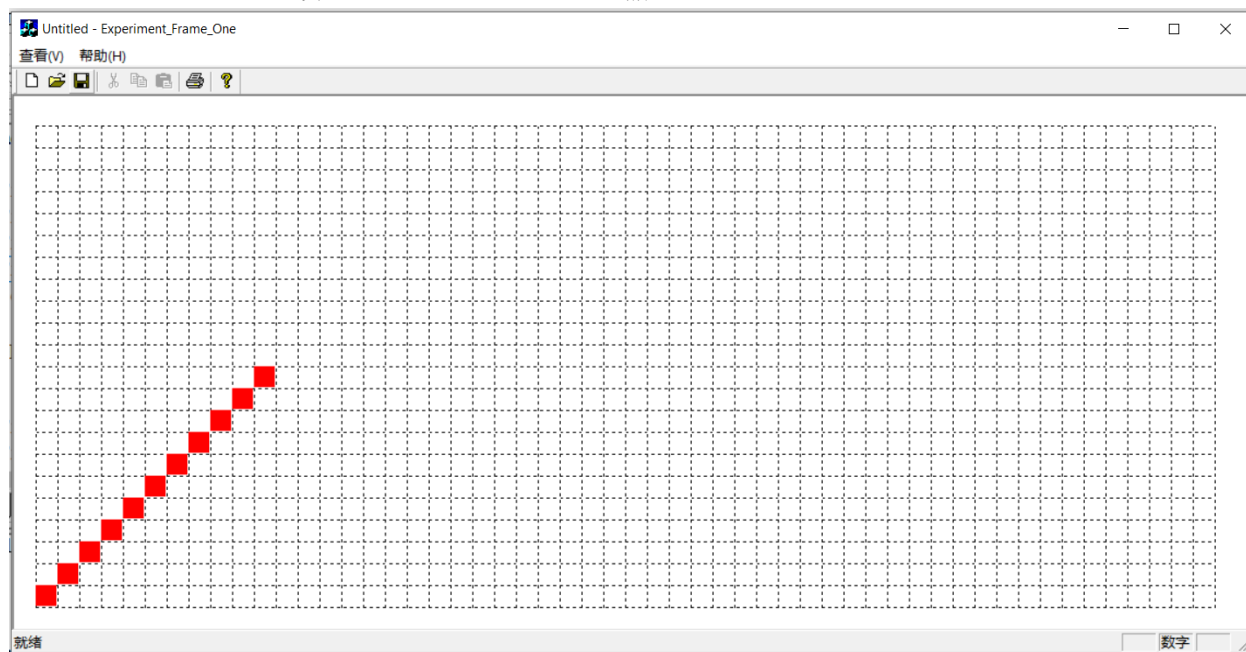
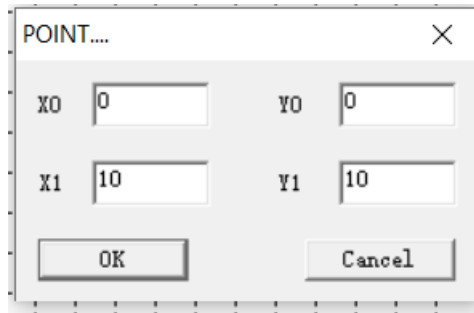


图 1.6.1 启动界面——(0,0) & (10,10)



图 1.6.2 查看选项

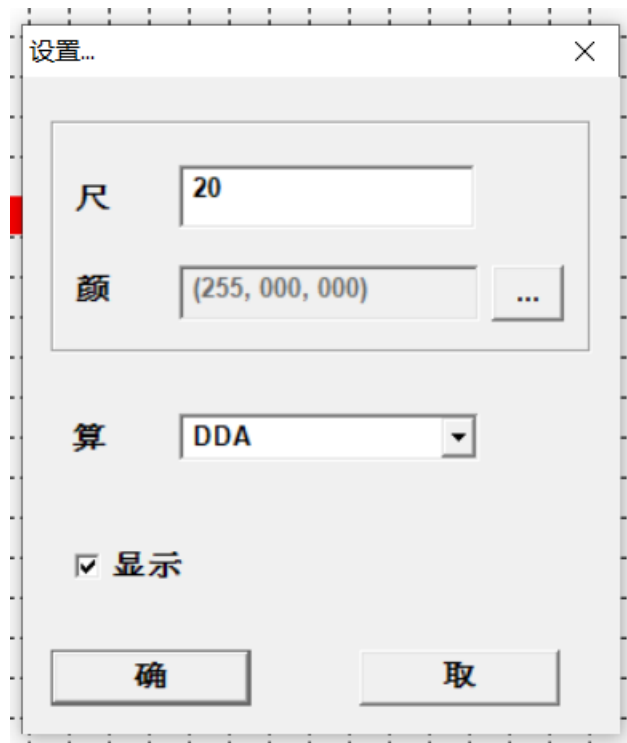


A dialog box titled "POINT...." with a close button (X) in the top right corner. It contains four input fields arranged in a 2x2 grid. The top row has "X0" and "Y0", both with the value "0". The bottom row has "X1" and "Y1", both with the value "10". At the bottom of the dialog are two buttons: "OK" on the left and "Cancel" on the right.

| | | | |
|----|----|----|----|
| X0 | 0 | Y0 | 0 |
| X1 | 10 | Y1 | 10 |

OK Cancel

图 1.6.3 坐标输入



A dialog box titled "设置..." (Settings) with a close button (X) in the top right corner. It contains several settings: "尺" (Scale) with a text input field containing "20"; "颜" (Color) with a text input field containing "(255, 000, 000)" and a color selection button "..."; "算" (Algorithm) with a dropdown menu showing "DDA"; a checked checkbox labeled "显示" (Display); and two buttons at the bottom: "确" (Confirm) on the left and "取" (Cancel) on the right.

尺 20

颜 (255, 000, 000) ...

算 DDA

☒ 显示

确 取

图 1.6.4 算法选择

1.6.1 DDA 算法

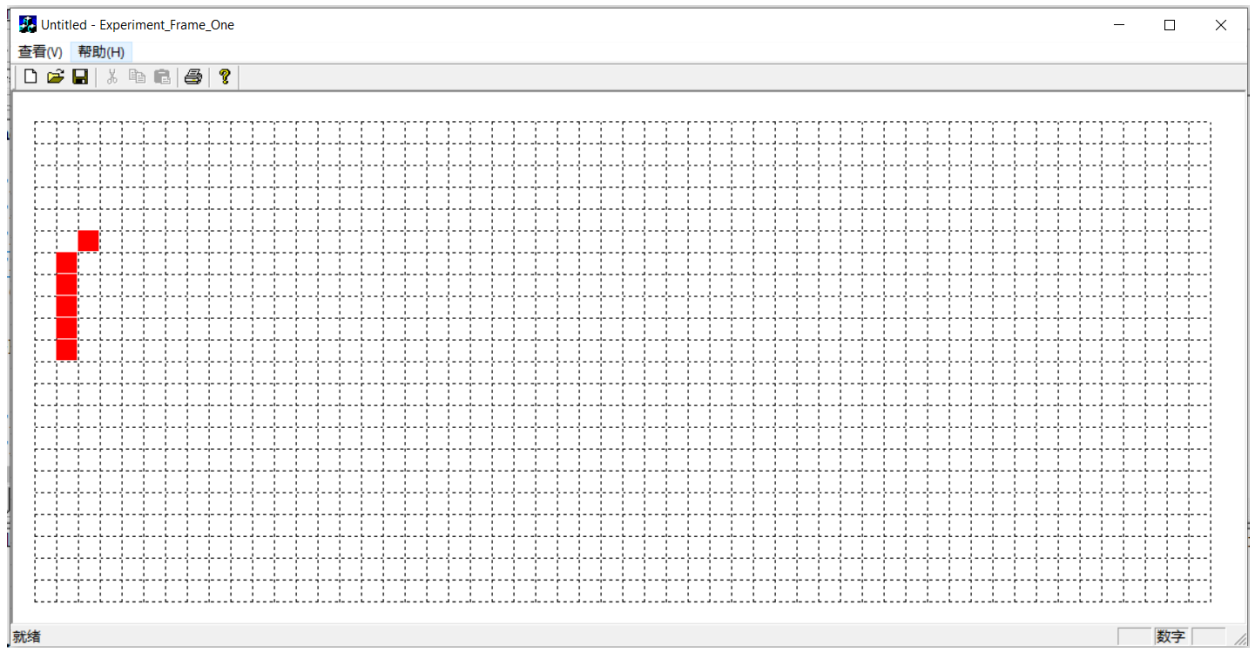


图 1.6.5 (1,11) & (2,16)

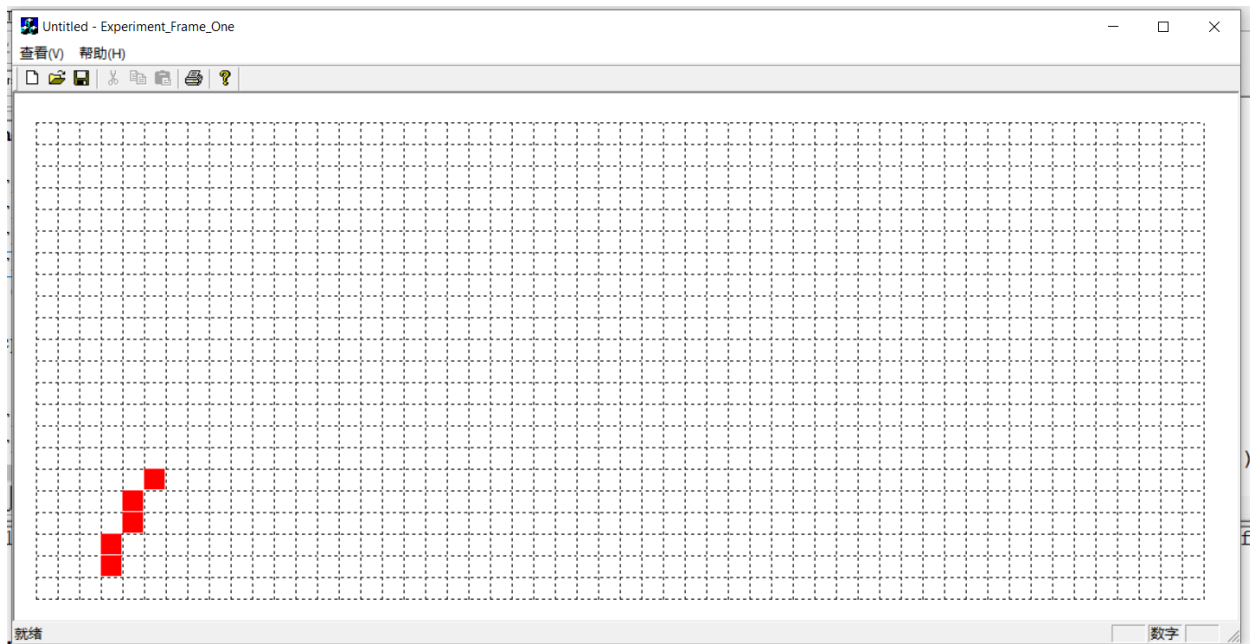


图 1.6.6 (3,1) & (5,5)

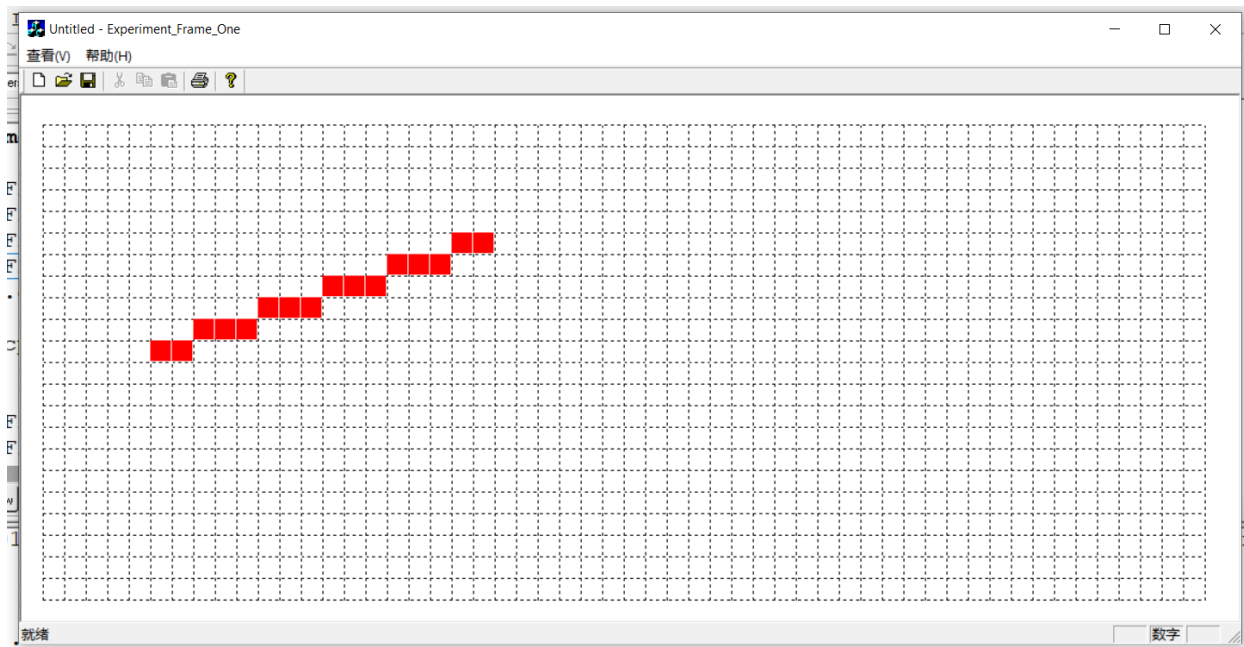


图 1.6.7 (5,11) & (20,16)

1.6.2 Mid-Bresenham 算法

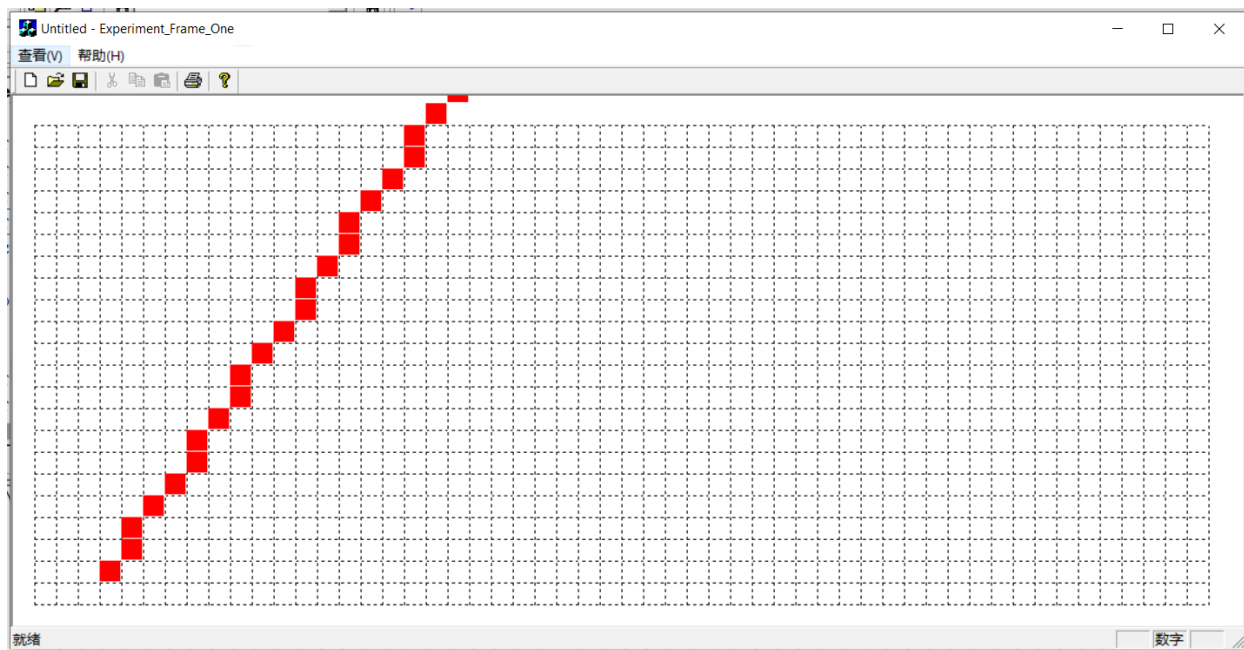


图 1.6.8 (3,1) & (56,75)

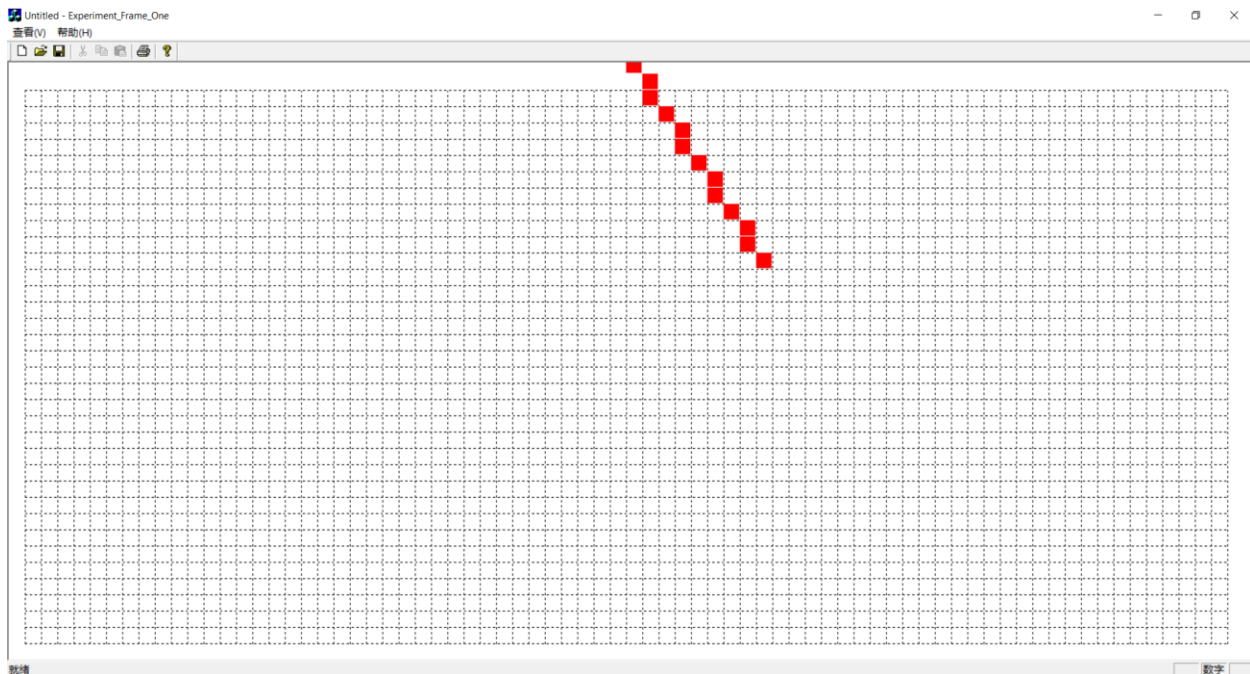


图 1.6.9 (45,23) & (16,67)

1.7 心得与体会

在这次的图形学实验中，我了解了图形界面的编程基础，也对 VC 中的 MFC 有了一定的了解，收获了继续学习 VC 图形编程的能力，增长了对于图形编程及其应用的浓厚兴趣；同时，加深了 DDA 算法与 Mid-Bresenham 算法的理解与感性认识。

2 实验 4: BSpline 曲线绘制

2.1 实验要求和目的

理解掌握自由曲线生成的基本原理和方法；编程实现三次 B 样条曲线：

- 均匀周期性 B 样条曲线
- 开放均匀 B 样条曲线

2.2 实验课时

4 学时

2.3 实验环境

本试验提供自带实验平台

- 开发环境：Visual C++ 6.0
- 实验平台：Free_Curve（自制平台）

2.4 实验内容

本实验提供名为 Free_Curve 的平台，该平台提供基本绘制、设置、输入功能，学生在此基础上实现：

[1]编码实现 BSpline 曲线基函数

[2]编码实现不同参数条件下的节点矢量的生成

- 平台界面：如图 2.4.1 所示
- 多边形输入，界面如图 2.4.1 所示：
 - 用户按【功能】→【输入……】菜单开始输入控制多边形；
 - 单击鼠标左键输入多边形顶点；
 - 点击鼠标右键结束控制多边形输入
- 参数设置：界面如图 2.4.2 所示
 - 用户按“【功能】→【设置……】”启动设置对话框
 - 设置内容：
 - ◆ 控制点设置(包括颜色、是否显示、控制点大小)
 - ◆ 控制多边形设置(包括控制多边形的颜色，是否显示)
 - ◆ BSpline 曲线设置
- 实现下列函数

- 实现 BSpline 曲线的基函数;

```
float BKM(float t, int k, int m, float nodes[])
```

参数含义参考代码注解;

- 节点矢量的计算:

```
bool Create_Nodes_Vector(int n,  
                          int m,  
                          int SplineType,  
                          float nodes[])
```

参数含义参考代码注解;

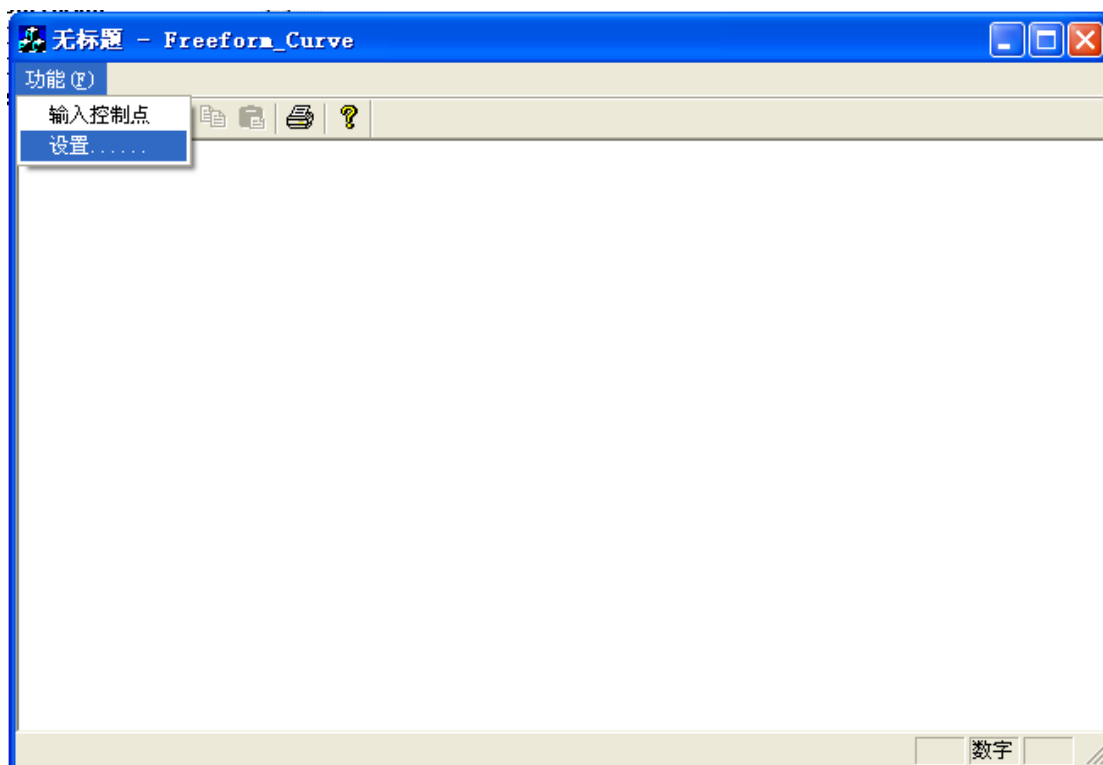


图 2.4.1 平台界面



图 2.4.2 设置界面

2.5 核心代码

本次实验的核心代码如下所示。

```
/******  
Function:    calEquation  
Description: 计算曲线方程  
Author:  
Calls:      buildInfo  
Input:  
            - start: int, 曲线起始控制点  
            - end: int, 曲线结束控制点  
            - p, 保存结果矩阵的数组指针  
Return:      Equation  
*****/  
EquationInfo Curve::calEquation(int start, int end, double(*p)[2]) {  
    const int* parr = NULL;          // 指向 对应系数矩阵 的指针  
  
    if (degree == 1)  
        parr = &(Curve::W1[0][0]);  
    else if (degree == 2) {  
        if (type == Bezier)  
            parr = &(Curve::W2[0][0][0]);  
        else  
            parr = &(Curve::W2[1][0][0]);  
    }  
    else {  
        if (type == Bezier)  
            parr = &(Curve::W3[0][0][0]);  
        else  
            parr = &(Curve::W3[1][0][0]);  
    }  
  
    int n = degree + 1;  
  
    // 矩阵乘  
    for (int i = 0; i < n; i++) {  
        for (int j = 0; j < 2; j++) {  
            int tot = 0;  
  
            for (int k = 0; k < n; k++) {  
                if (!j)  
                    tot += (int)ctrlPoints[start + k].x * *(parr + i * n + k);  
                else  
                    tot += (int)ctrlPoints[start + k].y * *(parr + i * n + k);  
            }  
        }  
    }  
}
```

```

        p[i][j] = (double)tot;
    }
}

// B- 样条曲线 前缀系数
if (this->type == Spline && this->degree != 1) {
    double w = this->BSplineW[degree];

    for (int i = 0; i < n; i++)
        p[i][0] *= w, p[i][1] *= w;
}

// 构造 info
EquationInfo info;

buildInfo(p, 0, n, info.nameX);
buildInfo(p, 1, n, info.nameY);

return info;
}

/*****
Function:    generateCurvePoints
Description: 采样生成样条曲线上的点
Author:
Calls:      calEquation
Input:
    - start: int, 曲线起始控制点 在 ctrlPoints 中的索引
    - end: int, 曲线结束控制点 在 ctrlPoints 中的索引

Return:
    - points: vector<CP2>, 生成的点
*****/
std::vector<CP2> Curve::generateCurvePoints(int start, int end)
{
    std::vector<CP2> points;

    if (end - start + 1 <= this->degree)           // 控制点个数 不足以计算 degree 阶曲线
        return points;

    int offset;
    if (type == Bezier)           // Bezier 曲线
        offset = this->degree;
    else                           // B - 样条曲线
        offset = 1;

```

```

double eps = 1.0 / this->precision;           // 参量 t 精度

double(*p)[2];                               // 结果矩阵, N * 2 矩阵
p = new double[this->degree + 1][2];

for (int i = start; i + this->degree <= end; i += offset) {           // 起始控制点索引 i
    EquationInfo info = calEquation(i, i + this->degree, p);           // 计算该段曲
线方程
    this->equations.push_back(info);

    for (double t = 0; t <= 1; t += eps) {           // 参数方程 参量 t
        CP2 cur;

        cur.x = p[this->degree][0];
        cur.y = p[this->degree][1];

        double ct = 1.0;           // t 的各个幂次, 初始化为 t^0
        for (int j = this->degree - 1; j > -1; j--) {
            ct *= t;
            cur.x += (ct * p[j][0]);
            cur.y += (ct * p[j][1]);
        }

        points.push_back(cur);
    }
}

// 防止 Bezier 曲线 精度过低曲线不连续
if (type == Bezier)
    points.push_back(ctrlPoints[end]);

delete[] p;
return points;
}

```

2.6 实验结果

程序的启动界面，如图 2.6.1 所示。

B-样条算法的实验结果，如图 2.6.2 至图 2.6.5 所示。其中，图 2.6.4 与图 2.6.5 为同一组控制点数据在曲线阶数分别为 3 和 1 时 B-样条算法的输出对比。

Bezier 算法算法的实验结果，如图 2.6.6 至图 2.6.8 所示。其中，图 2.6.7 与图 2.6.8 为同一组控制点数据在曲线阶数分别为 3 和 1 时 Bezeir 算法的输出对比。

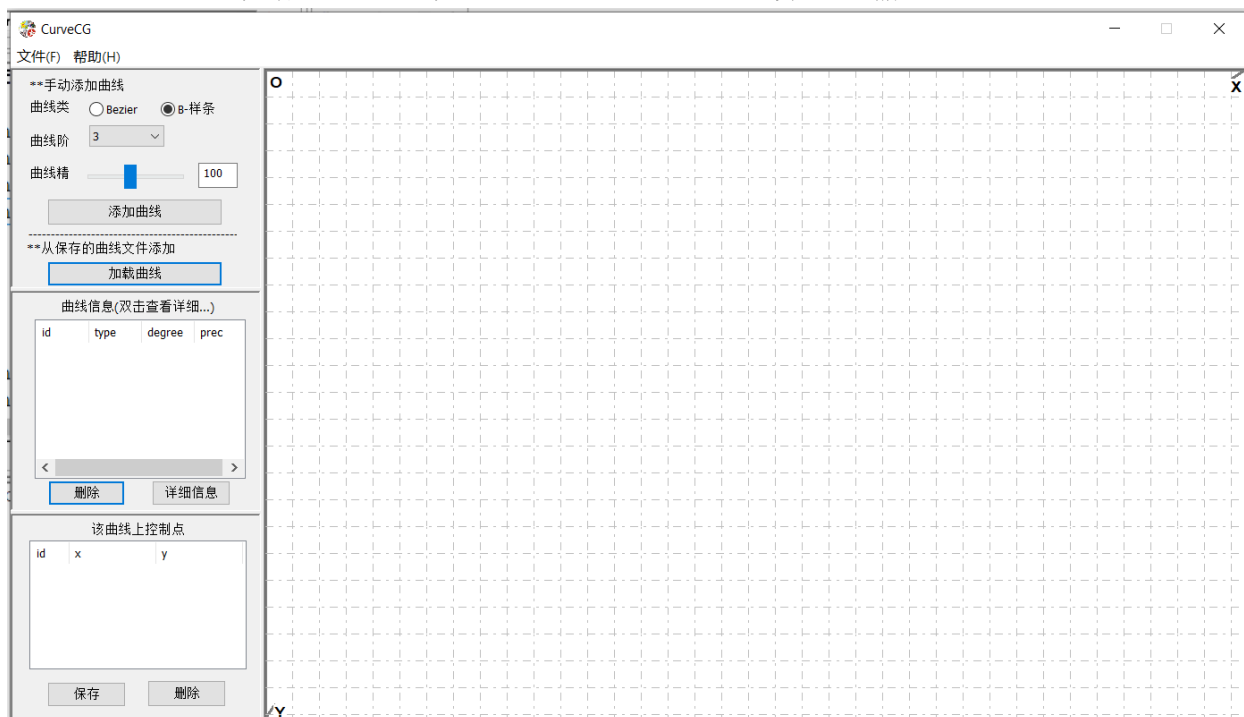


图 2.6.1 启动界面

2.6.1 B-样条算法

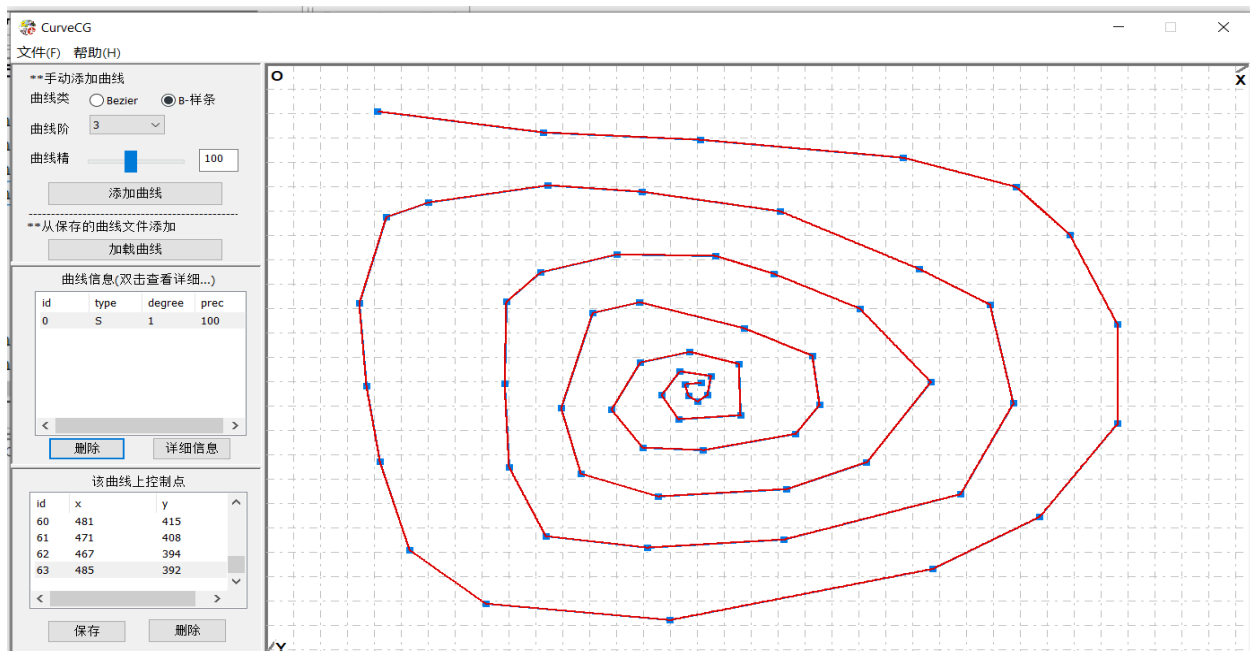


图 2.6.2 阶数：1，精度 100（数据 1）

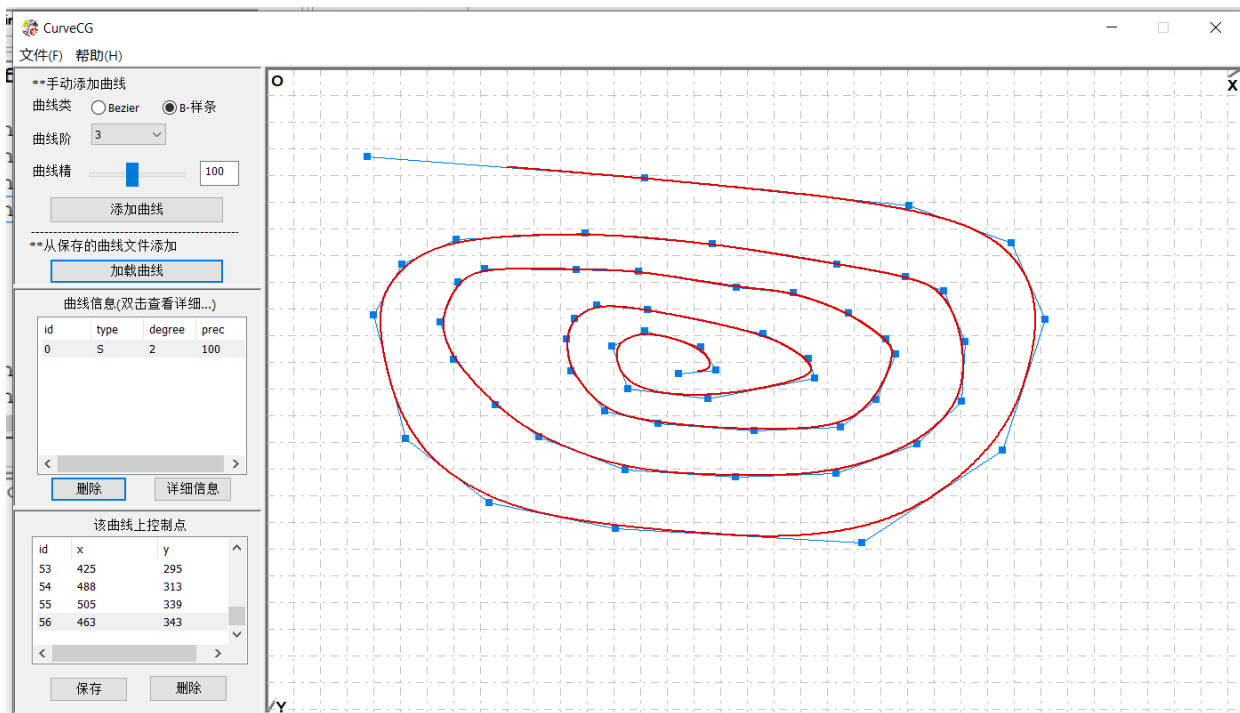


图 2.6.3 阶数：2，精度 100（数据 2）

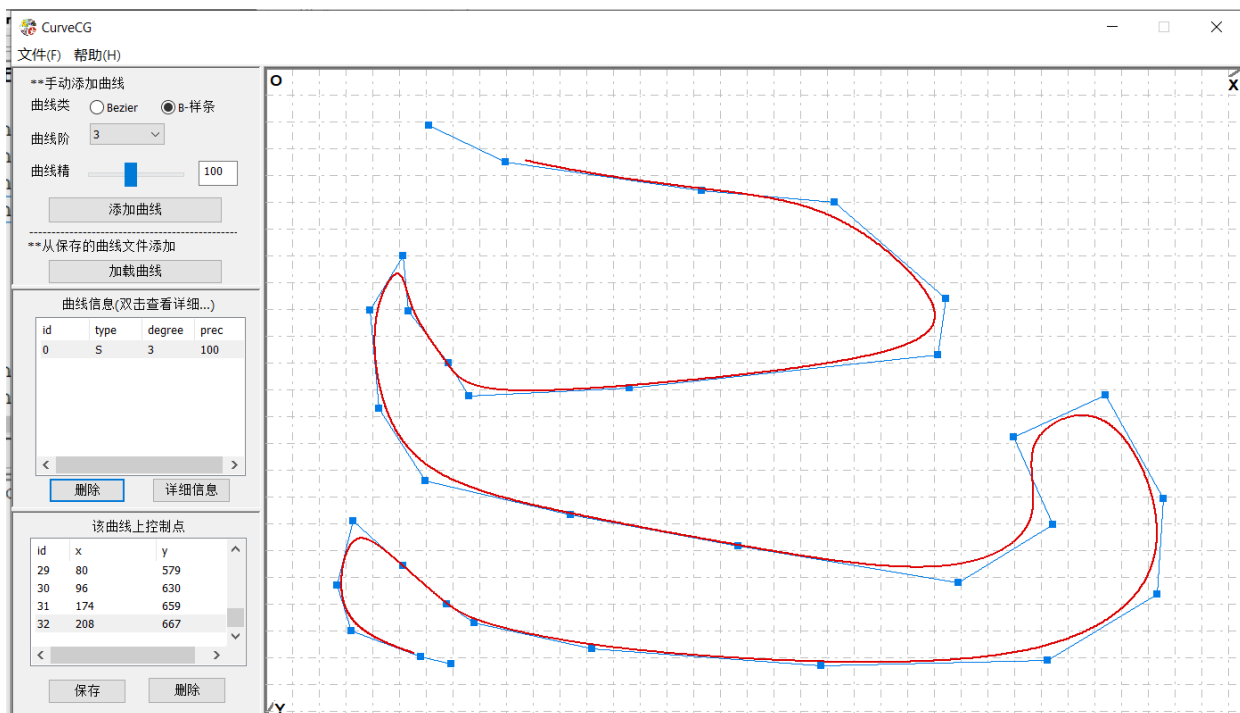


图 2.6.4 阶数：3，精度 100（数据 3）

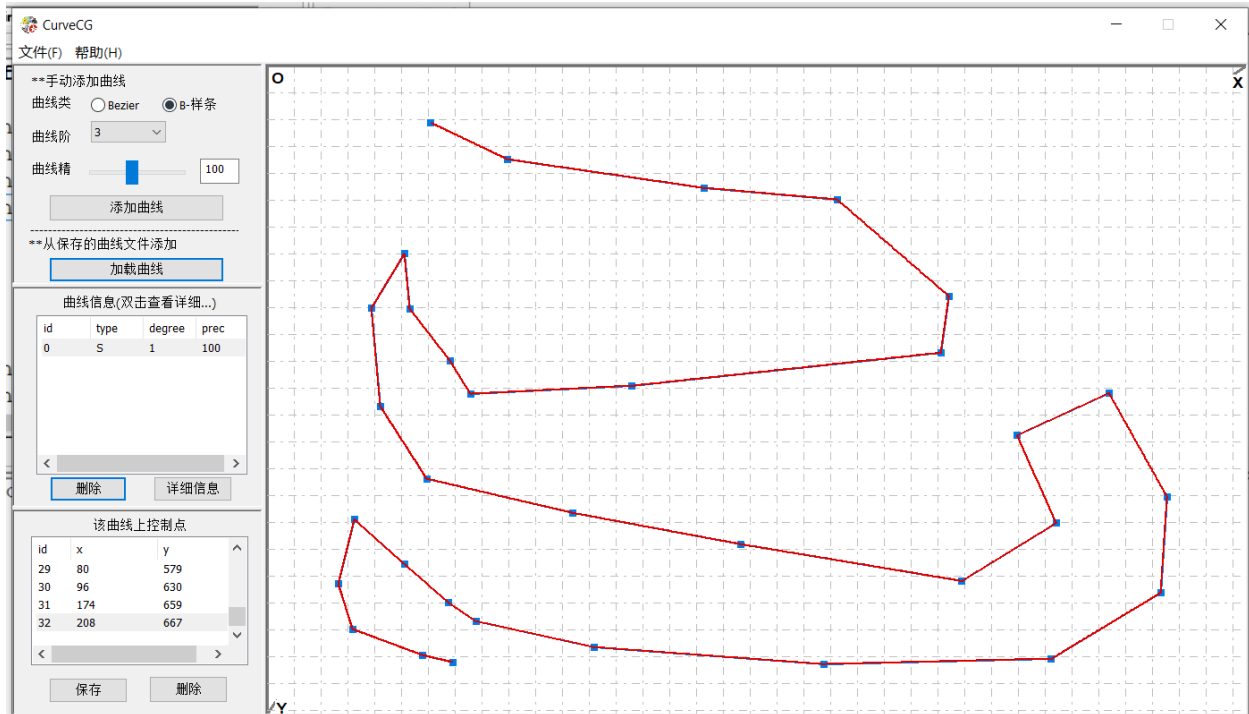


图 2.6.5 阶数：1，精度 100（数据 4）

2.6.2 Bezeir 算法

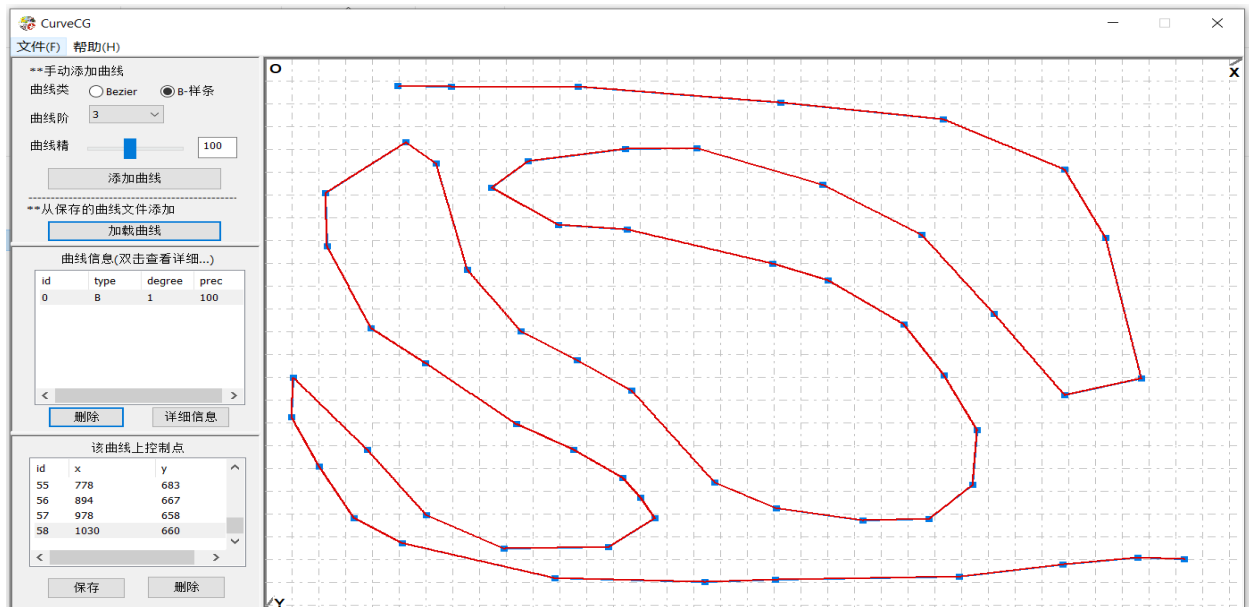


图 2.6.6 阶数：1，精度 100（数据 5）

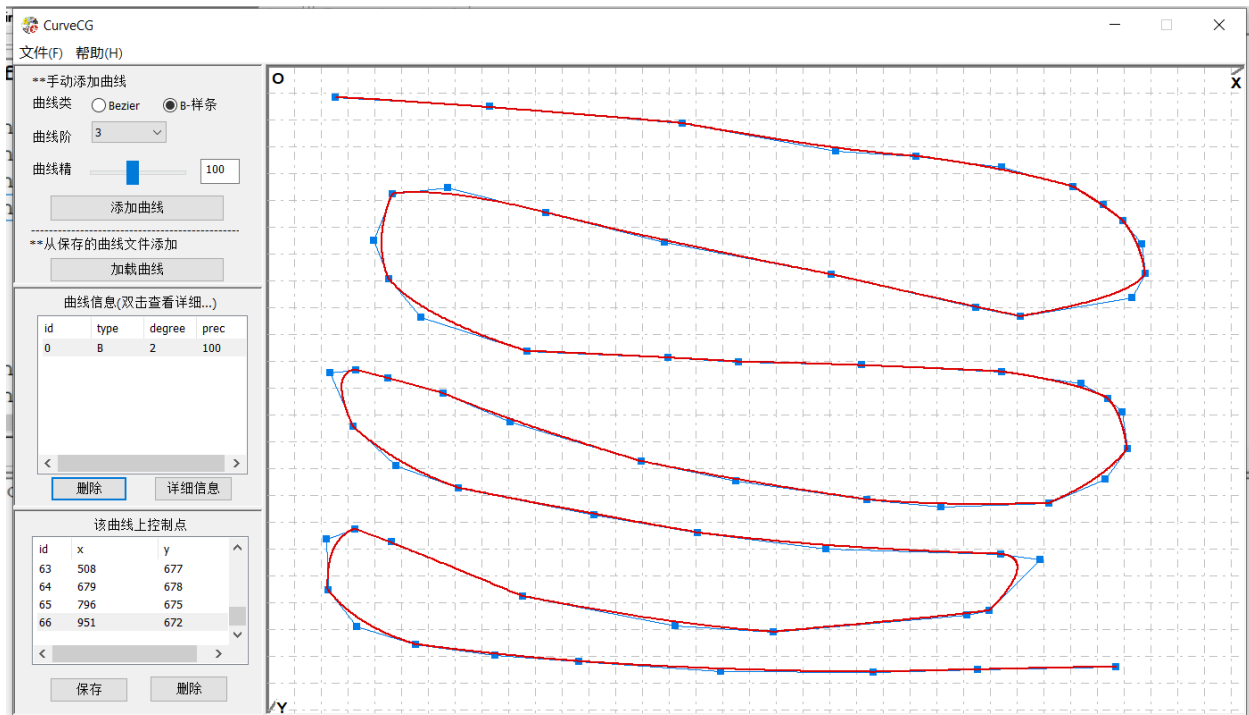


图 2.6.7 阶数：2，精度 100（数据 6）

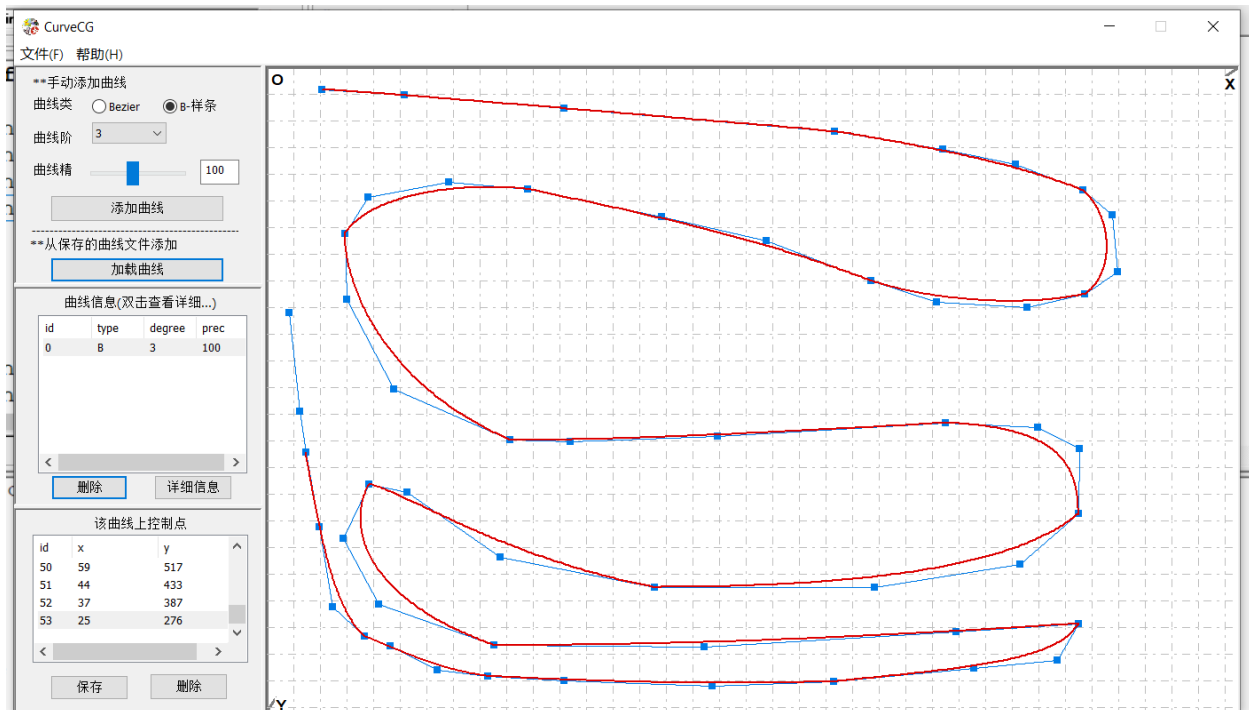


图 2.6.8 阶数：3，精度 100（数据 7）

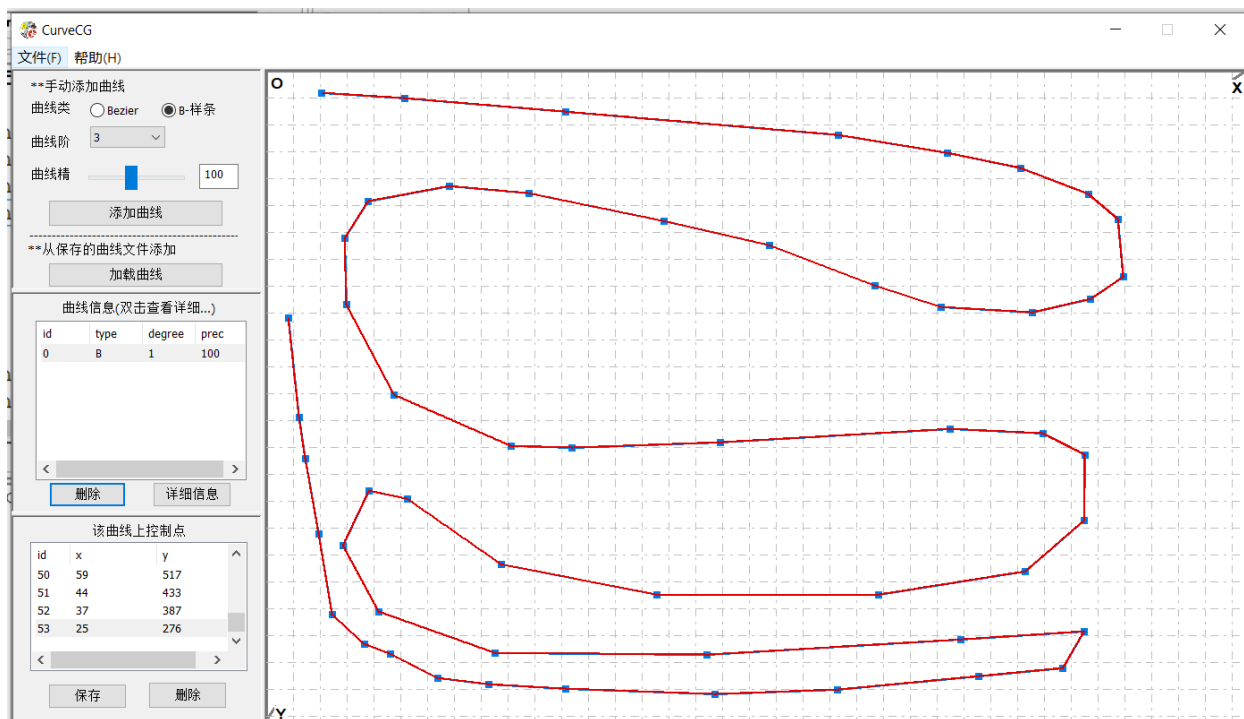


图 2.6.9 阶数：1，精度 100（数据 8）

2.7 心得与体会

B-样条曲线比 Bezeir 曲线需要更多信息(即，曲线的次数和一个节点向量)和更复杂的理论。但是它有更多优点来弥补这个缺点。

- 1) 一个 B-样条曲线可以是一个 Bezeir 曲线。
- 2) B-样条曲线满足 Bezeir 曲线有的所有重要性质。
- 3) B-样条曲线提供了比 Bezeir 曲线更灵活的控制。

例如，一个 B-样条曲线的次数与控制点数目是分开的。更准确地说，我们可以使用更低次曲线而仍然保持很多控制点。我们可以改变一个控制点位置而不会全局地改变整个曲线形状（局部修改性质）。因为 B-样条曲线满足强凸包性质，它们可以进行更精细的形状控制。而且，还有其他设计和编辑形状的技术比如改变节点。

但是，记住 B-样条曲线仍然是多项式曲线而多项式曲线不能表示许多有用的简单的曲线比如圆和椭圆。

附录

BSpline 曲线控制点的测试数据

数据 1

type:S,degree:1,pre:100

(124, 58)

(309, 84)

(484, 93)

(710, 115)

(836, 151)

(896, 210)

(949, 320)

(949, 442)

(862, 557)

(743, 621)

(450, 684)

(245, 664)

(160, 598)

(127, 489)

(112, 396)

(104, 294)

(134, 188)

(181, 170)

(314, 149)

(419, 157)

(573, 181)

(728, 252)

(807, 296)

(833, 417)

(774, 529)

(577, 585)
(425, 595)
(312, 581)
(271, 496)
(266, 393)
(268, 292)
(306, 256)
(391, 234)
(501, 236)
(566, 258)
(662, 301)
(741, 391)
(669, 490)
(580, 523)
(437, 532)
(351, 504)
(329, 423)
(364, 306)
(416, 293)
(533, 325)
(609, 359)
(617, 419)
(590, 455)
(487, 475)
(420, 472)
(385, 425)
(417, 367)
(472, 354)
(527, 369)

(529, 432)

(460, 437)

(441, 407)

(461, 378)

(496, 384)

(492, 407)

(481, 415)

(471, 408)

(467, 394)

(485, 392)

数据 2

type:S,degree:2,pre:100

(113, 99)

(425, 123)

(722, 154)

(837, 196)

(875, 282)

(827, 429)

(669, 533)

(392, 517)

(250, 488)

(156, 416)

(120, 277)

(152, 220)

(213, 192)

(358, 185)

(501, 197)

(641, 220)

(718, 234)

(761, 250)
(785, 307)
(781, 374)
(731, 422)
(640, 455)
(527, 459)
(403, 451)
(306, 414)
(257, 378)
(210, 327)
(195, 285)
(215, 240)
(245, 225)
(348, 226)
(418, 228)
(528, 246)
(592, 252)
(654, 275)
(696, 304)
(707, 321)
(685, 372)
(645, 403)
(548, 407)
(440, 399)
(380, 385)
(342, 340)
(337, 304)
(346, 281)
(371, 266)

(428, 271)

(558, 298)

(609, 326)

(616, 348)

(496, 371)

(406, 360)

(388, 312)

(425, 295)

(488, 313)

(505, 339)

(463, 343)

数据 3

type:S,degree:3,pre:100

(183, 64)

(269, 105)

(489, 137)

(638, 150)

(763, 258)

(754, 321)

(408, 358)

(228, 367)

(205, 330)

(160, 272)

(154, 210)

(117, 271)

(127, 381)

(179, 462)

(342, 500)

(530, 535)

(777, 576)
(883, 511)
(839, 413)
(942, 366)
(1007, 482)
(1000, 589)
(877, 663)
(623, 669)
(366, 650)
(234, 621)
(203, 600)
(154, 557)
(98, 507)
(80, 579)
(96, 630)
(174, 659)
(208, 667)

数据 4

type:S,degree:1,pre:100

(183, 64)
(269, 105)
(489, 137)
(638, 150)
(763, 258)
(754, 321)
(408, 358)
(228, 367)
(205, 330)
(160, 272)

(154, 210)
(117, 271)
(127, 381)
(179, 462)
(342, 500)
(530, 535)
(777, 576)
(883, 511)
(839, 413)
(942, 366)
(1007, 482)
(1000, 589)
(877, 663)
(623, 669)
(366, 650)
(234, 621)
(203, 600)
(154, 557)
(98, 507)
(80, 579)
(96, 630)
(174, 659)
(208, 667)

数据 5

type:B,degree:1,pre:100

(149, 37)
(209, 38)
(351, 38)
(578, 59)

(760, 81)
(896, 147)
(942, 237)
(982, 422)
(896, 444)
(817, 337)
(736, 233)
(625, 167)
(484, 119)
(404, 120)
(295, 136)
(254, 171)
(329, 220)
(406, 226)
(569, 271)
(631, 293)
(716, 351)
(761, 418)
(798, 490)
(793, 562)
(744, 607)
(670, 609)
(573, 593)
(504, 559)
(411, 438)
(350, 398)
(287, 360)
(227, 279)
(192, 139)

(158, 111)
(68, 178)
(70, 248)
(119, 356)
(180, 402)
(282, 482)
(346, 516)
(401, 553)
(421, 579)
(437, 606)
(385, 644)
(268, 646)
(181, 602)
(115, 516)
(32, 421)
(30, 473)
(61, 538)
(100, 606)
(154, 639)
(325, 685)
(493, 690)
(572, 687)
(778, 683)
(894, 667)
(978, 658)
(1030, 660)

数据 6

type:B,degree:2,pre:100

(76, 35)

(249, 45)
(465, 64)
(637, 95)
(727, 101)
(823, 113)
(903, 135)
(937, 155)
(959, 173)
(980, 199)
(984, 232)
(969, 259)
(844, 280)
(794, 270)
(632, 233)
(445, 197)
(312, 164)
(202, 136)
(140, 143)
(119, 195)
(136, 238)
(172, 281)
(291, 319)
(449, 326)
(528, 331)
(666, 334)
(823, 342)
(912, 355)
(942, 372)
(958, 387)

(964, 428)
(939, 462)
(876, 489)
(755, 493)
(672, 485)
(525, 464)
(419, 442)
(272, 398)
(197, 366)
(135, 349)
(99, 340)
(70, 343)
(96, 403)
(144, 447)
(214, 472)
(366, 502)
(482, 522)
(626, 540)
(822, 546)
(866, 552)
(809, 609)
(784, 614)
(567, 633)
(457, 626)
(286, 593)
(139, 532)
(98, 518)
(66, 529)
(68, 586)

(100, 627)

(166, 647)

(255, 659)

(349, 666)

(508, 677)

(679, 678)

(796, 675)

(951, 672)

数据 7

type:B,degree:3,pre:100

(62, 25)

(155, 31)

(335, 46)

(640, 72)

(762, 92)

(844, 109)

(920, 138)

(953, 166)

(959, 230)

(922, 255)

(857, 270)

(755, 264)

(681, 240)

(563, 195)

(445, 168)

(294, 137)

(205, 129)

(114, 146)

(88, 187)

(90, 261)
(143, 362)
(274, 419)
(342, 421)
(508, 415)
(765, 400)
(869, 405)
(916, 429)
(915, 502)
(849, 559)
(685, 585)
(437, 585)
(263, 551)
(158, 478)
(115, 469)
(86, 530)
(126, 604)
(256, 650)
(493, 652)
(777, 635)
(915, 626)
(891, 667)
(797, 676)
(639, 691)
(502, 696)
(335, 690)
(249, 685)
(192, 678)
(139, 651)

(110, 640)

(74, 607)

(59, 517)

(44, 433)

(37, 387)

(25, 276)

数据 8

type:B,degree:1,pre:100

(62, 25)

(155, 31)

(335, 46)

(640, 72)

(762, 92)

(844, 109)

(920, 138)

(953, 166)

(959, 230)

(922, 255)

(857, 270)

(755, 264)

(681, 240)

(563, 195)

(445, 168)

(294, 137)

(205, 129)

(114, 146)

(88, 187)

(90, 261)

(143, 362)

(274, 419)
(342, 421)
(508, 415)
(765, 400)
(869, 405)
(916, 429)
(915, 502)
(849, 559)
(685, 585)
(437, 585)
(263, 551)
(158, 478)
(115, 469)
(86, 530)
(126, 604)
(256, 650)
(493, 652)
(777, 635)
(915, 626)
(891, 667)
(797, 676)
(639, 691)
(502, 696)
(335, 690)
(249, 685)
(192, 678)
(139, 651)
(110, 640)
(74, 607)

(59, 517)

(44, 433)

(37, 387)

(25, 276)