

江南大学物联网工程学院实验报告

课程名称 《计算机图形学》 实验名称 实验 2 线段和多边形的扫描转换 实验日期 2017.11.12

班级 计科 1404 姓名 阎覃 学号 1030414414

实验报告要求 1. 实验目的 2. 实验内容 3. 实验步骤 4. 运行情况 5. 实验体会

1 实验内容

- 实现直线的 DDA 算法
- 实现直线的中点算法
- 实现基于扫描线的多边形扫描转换算法

2 实验步骤及运行情况

Problem 1 实现直线的 DDA 算法

数值微分法（DDA 法）根据直线的微分方程来绘制直线，它是最简单的一种画线方法。若一条直线的起点为 (x_1, y_1) ，终点为 (x_2, y_2) ，令

$$\begin{aligned}\Delta x &= x_2 - x_1 \\ \Delta y &= y_2 - y_1 \\ \Delta m &= \max(|\Delta x|, |\Delta y|)\end{aligned}$$

则可以得到递推公式为

$$\Delta x_{i+1} = x_i + \frac{\Delta x}{\Delta m} \quad (1)$$

$$\Delta y_{i+1} = y_i + \frac{\Delta y}{\Delta m} \quad (2)$$

因此，直线的 DDA 算法程序为

Entities/LineDDA2D.cpp

```
1 //
2 // Created by Ethan on 2017/11/16.
3 //
4
5 #include "LineDDA2D.h"
6
7 void LineDDA2D::drawMe() {
8     glBegin(GL_POINTS);
9
10    int x1 = sP.x, x2 = eP.x, y1 = sP.y, y2 = eP.y;
11
12    int dm = 0;
13    if (abs(x2 - x1) > abs(y2 - y1))
14        dm = abs(x2 - x1); // x 为记长方向
15    else
16        dm = abs(y2 - y1); // y 为记长方向
17
18    float dx = (float)(x2 - x1) / dm; // 当x为记长方向时, dx的值为1
19    float dy = (float)(y2 - y1) / dm; // 当y为记长方向时, dy的值为1
20    float x = x1 + 0.5f;
21    float y = y1 + 0.5f;
22    for (int i = 0; i < dm; i++) {
23        glVertex2i((int) x, (int) y);
24        x += dx;
25        y += dy;
```

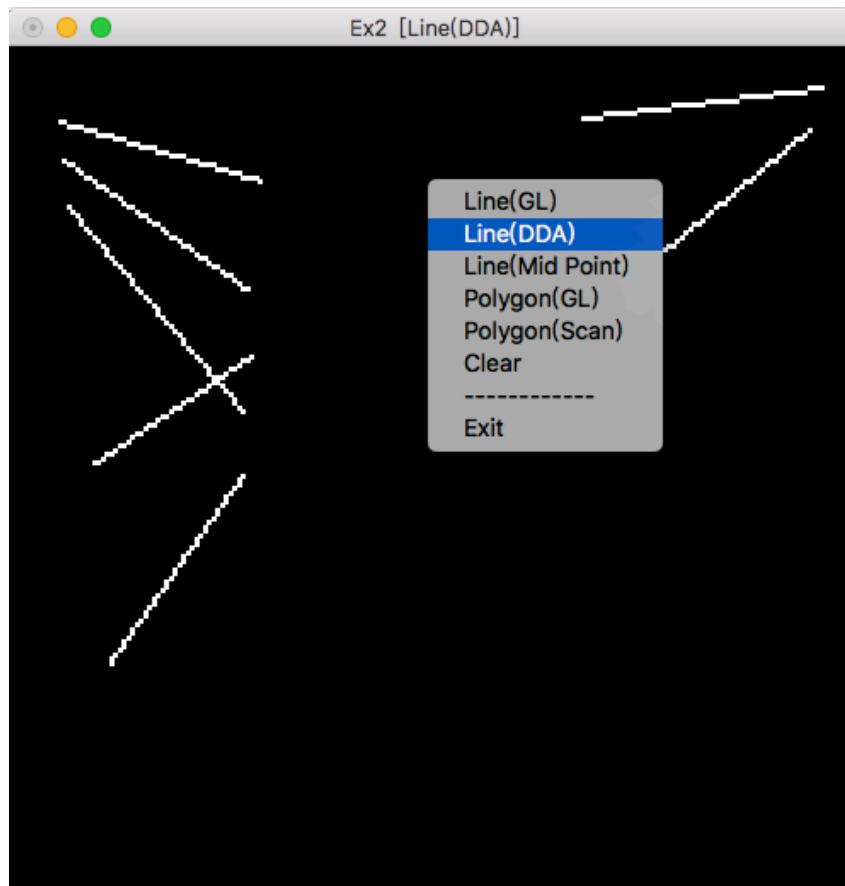
```

26     }
27
28     glEnd();
29 }

```

本程序参考书上代码 P42。

运行截图



Problem 2 实现直线的中点算法

若一条直线的起点为 (x_0, y_0) , 终点为 (x_1, y_1) , 直线方程为

$$F(x, y) = ax + by + c = 0$$

其中, $a = y_0 - y_1$, $b = x_1 - x_0$, $c = x_0y_1 - x_1y_0$ 。

若当前的点为 (x_p, y_p) , 则可以带入下一中点, 构造判别式:

$$\begin{aligned} d &= F(x_p + 1, y_p + 0.5) \\ &= a(x_p + 1) + b(y_p + 0.5) + c \end{aligned} \quad (3)$$

$d < 0$ 时取右上方点, $d \geq 0$ 取右方点。

为了提高效率, 可以采用增量算法。

当斜率为正:

当 $d \geq 0$ 时, $d' = d + a$

当 $d < 0$ 时, $d' = d + a + b$

当斜率为负:

当 $d \geq 0$ 时, $d' = d - a$

当 $d < 0$ 时, $d' = d - a + b$

因此, 直线的中点算法程序为

Entities/LineMidP2D.cpp

```

1  //
2  // Created by Ethan on 2017/11/16.
3  //
4
5  #include <math.h>
6  #include "LineMidP2D.h"
7
8  #define swap(x, y) {int tmp; tmp = x;x = y; y = tmp;}
9
10 void LineMidP2D::drawMe() {
11     glBegin(GL_POINTS);
12
13     int x0 = sP.x, x1 = eP.x, y0 = sP.y, y1 = eP.y;
14     int a, b, d, x, y, tag = 0;
15     if (abs(x1 - x0) < abs(y1 - y0)) {
16         swap(x0, y0);
17         swap(x1, y1);
18         tag = 1;
19     }
20     if (x0 > x1) {
21         swap(x0, x1);
22         swap(y0, y1);
23     }
24
25     a = y0 - y1;
26     b = x1 - x0;
27     d = a + b / 2;
28     if (y0 < y1) {
29         x = x0;
30         y = y0;
31         while (x < x1) {
32             if (d < 0) {
33                 x++;
34                 y++;
35                 d = d + a + b;
36             } else {
37                 x++;
38                 d += a;
39             }
40             if (tag) glVertex2i(y, x);
41             else glVertex2i(x, y);
42         }
43     } else {
44         x = x1;
45         y = y1;
46         while (x > x0) {
47             if (d < 0) {
48                 x--;
49                 y++;
50                 d = d - a + b;
51             } else {
52                 x--;
53                 d -= a;
54             }
55             if (tag) glVertex2i(y, x);
56             else glVertex2i(x, y);
57         }
58     }
59 }
60

```

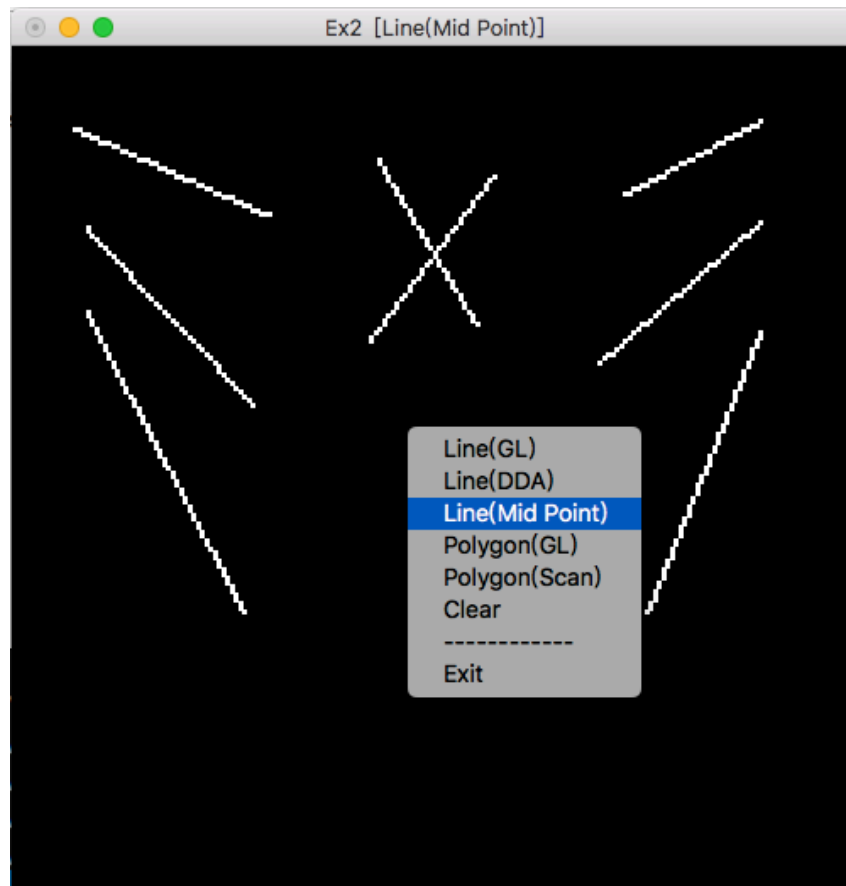
```

61   glEnd();
62 }

```

本程序参考书上代码 P49, P50。

运行截图



Problem 3 实现基于扫描线的多边形扫描转换算法

定义边节点:

```

1  class Edge {
2  public:
3      int ymax;
4      float x;
5      float dx;
6      Edge *next;
7  };

```

其中, ymax 为该边最大的 y 坐标, x 为 y 值最小的端点的 x 坐标, dx 为斜率。

定义一个边表数组 ET, 存放所有的边。数组大小为扫描线的数目。定义一个链表 AET, 即活动边表, 表示正在处理的扫描线。

具体的算法如下:

1. 初始化 ET, 将各边按照该边最小的 y 坐标存放至数组内对应位置。
2. 初始化 AET 为空链表。
3. 扫描线从下向上扫描, y 从 0 增加至最大值, 每次加 1。

(a) 取出 ET 中当前扫描线的所有边并按 x 递增的顺序加入 AET。

- (b) AET 中的边两两配对并填色。
- (c) 删除 AET 中满足 $y=y_{\max}$ 的边。
- (d) 更新 AET 中边的 x 值, 进入下一循环

对于本程序坐标系的范围是-100 至 100, 所以要对所有的 y 值进行偏移以便消去负数放入数组中。

Entities/PolygonScan2D.cpp

```

1  //
2  // Created by Ethan on 2017/11/16.
3  //
4
5  #include <queue>
6  #include <App.h>
7  #include "PolygonScan2D.h"
8
9  //定义用于边表ET和活动边表AET的通用类Edge
10 class Edge {
11 public:
12     int ymax;
13     float x;
14     float dx;
15     Edge *next;
16 };
17
18
19 void PolygonScan2D::drawMe() {
20
21     // 计算最高点的y坐标
22     int maxY = 0, minY = 1000000;
23     for (auto &point : points) {
24         if (point.y > maxY) {
25             maxY = point.y;
26         }
27         if (point.y < minY) {
28             minY = point.y;
29         }
30     }
31     // 初始化ET和AET
32
33     int h = maxY - minY;
34
35     Edge **ET = new Edge *[h + 1];
36
37     for (int i = 0; i < h + 1; i++) {
38         ET[i] = new Edge();
39         ET[i]→next = nullptr;
40     }
41     //活动边表
42     Edge *AET = new Edge();
43     AET→next = nullptr;
44
45     glBegin(GL_POINTS);
46
47     //建立边表ET
48     for (int i = 0; i < points.size(); i++) {
49         //取出当前点1前后相邻的共4个点, 点1与点2的连线作为本次循环处理的
50         //边, 另外两个点点0和点3用于计算奇点
51         Point2D p0 = points[(i - 1 + points.size()) % points.size()];
52         Point2D p1 = points[i];
53         Point2D p2 = points[(i + 1) % points.size()];
54         Point2D p3 = points[(i + 2) % points.size()];
55
56         int x0 = (int) p0.x, x1 = (int) p1.x, x2 = (int) p2.x, x3 = (int)
57             p3.x;

```

```

56     int y0 = (int) p0.y, y1 = (int) p1.y, y2 = (int) p2.y, y3 = (int)
        p3.y;
57
58     y0 += -minY;
59     y1 += -minY;
60     y2 += -minY;
61     y3 += -minY;
62
63     //水平线直接舍弃
64     if (p1.y == p2.y)
65         continue;
66     //分别计算下端点y坐标、上端点y坐标、下端点x坐标和斜率倒数
67     int ymin = y1 > y2 ? y2 : y1;
68     int ymax = y1 > y2 ? y1 : y2;
69     float x = y1 > y2 ? x2 : x1; // y值最小的端点的x坐标
70     float dx = (float) (x1 - x2) / (y1 - y2); //斜率1/m
71     //奇点特殊处理, 若点2->1->0的y坐标单调递减则y1为奇点, 若点1->2->3
        的y坐标单调递减则y2为奇点
72     if (((y1 < y2) && (y1 > y0)) || ((y2 < y1) && (y2 > y3))) {
73         ymin++;
74         x += dx;
75     }
76     //创建新边, 插入边表ET
77     Edge *p = new Edge();
78     p->ymax = ymax;
79     p->x = x;
80     p->dx = dx;
81     p->next = ET[ymin]->next;
82     ET[ymin]->next = p;
83 }
84
85 //扫描线从下往上扫描, y坐标每次加1
86 for (int i = 0; i < h; i++) {
87     //取出ET中当前扫描行的所有边并按x的递增顺序(若x相等则按dx的递增
        顺序)插入AET
88     while (ET[i]->next) {
89         //取出ET中当前扫描行表头位置的边
90         Edge *pInsert = ET[i]->next;
91         Edge *p = AET;
92         //在AET中搜索合适的插入位置
93         while (p->next) {
94             if (pInsert->x > p->next->x) {
95                 p = p->next;
96                 continue;
97             }
98             if (pInsert->x == p->next->x && pInsert->dx > p->next->dx
                ) {
99                 p = p->next;
100                 continue;
101             }
102             //找到位置
103             break;
104         }
105         //将pInsert从ET中删除, 并插入AET的当前位置
106         ET[i]->next = pInsert->next;
107         pInsert->next = p->next;
108         p->next = pInsert;
109     }
110
111     //AET中的边两两配对并填色
112     Edge *p = AET;
113     while (p->next && p->next->next) {
114         for (int x = p->next->x; x < p->next->next->x; x++) {
115             glVertex2i(x, i + minY);
116         }
117         p = p->next->next;

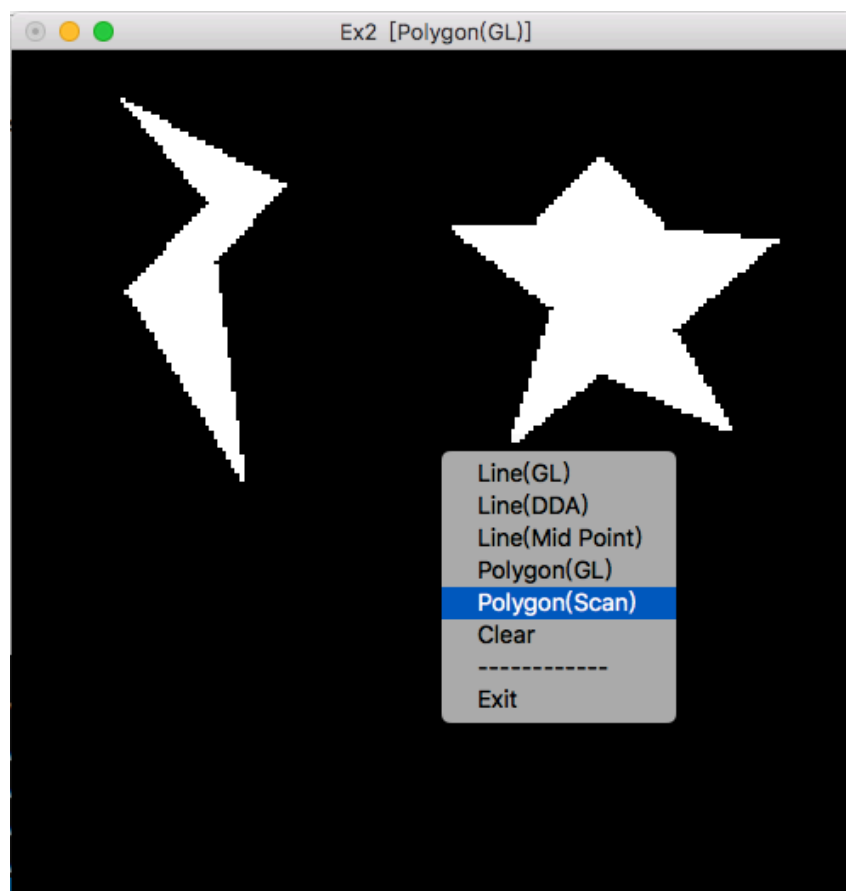
```

```

118     }
119
120     //删除AET中满足y=ymax的边
121     p = AET;
122     while (p->next) {
123         if (p->next->ymax == i) {
124             Edge *pDelete = p->next;
125             p->next = pDelete->next;
126             pDelete->next = nullptr;
127             delete pDelete;
128         } else {
129             p = p->next;
130         }
131     }
132
133     //更新AET中边的x值, 进入下一循环
134     p = AET;
135     while (p->next) {
136         p->next->x += p->next->dx;
137         p = p->next;
138     }
139
140 }
141 delete AET;
142 delete ET;
143 glEnd();
144 }

```

运行截图



3 实验体会

前两个直线的算法非常简单, 并且书上提供了代码, 很快就可以做出来。然而第三个书并没有提供代码, 只有文字说明算法的步骤。但是由于本人理解能力有限, 还是没能看懂。所以我在网上搜索了一下, 发现<http://www.twinklingstar.cn/2013/325/region-polygon-fill-scan-line/>这篇文章讲的比较详细清楚。之后又参照了相关代码, 完成了本次作业。

实验报告采用 \LaTeX 排版, 完整代码托管至 GitHub:
<https://github.com/Ethan-yt/JNU-CG-exp>

教师评价	优		良		中		及格		不及格		教师签名		日期	
------	---	--	---	--	---	--	----	--	-----	--	------	--	----	--