**成都信息工程大学计算机学院**

**课**

**程**

**实**

**验**

**报**

**告**

|  |  |
| --- | --- |
| **实验课程：** | **计算机图形学** |
| **实验项目：** | 图形填充与裁剪 |
| **指导教师：** |  |
| **学生姓名:** |  |
| **学生学号：** |  |
| **班 级：** |  |
| **实验地点：** |  |
| **实验时间：** |  |
| **实验成绩：** |  |

**一【上机实验内容】**

**课内实验：**

1. 利用多边形扫描线算法实现多边形的填充。
2. 利用Cohen-Sutherland算法实现直线段的裁剪。

**二【多边形填充：扫描线算法】（重点）(20分)**

请写出多边形填充扫描线算法的计算步骤。

按扫描线顺序（由下到上），计算扫描线与多边形各边的交点，把所有交点按x值递增顺序排序，第一个与第二个，第三个与第四个等等；每对交点代表扫描线与多边形的一个相交区间，把相交区间内的象素置成多边形颜色，即完成填充工作

若扫描到两条边相交顶点，检查相交顶点的两条边的另外两个顶点的y值。按这两个y值中大于交点y值的个数是0,1,2来决定是取零个、一个、还是两个。

对扫描线与多边形的相交区间取左闭右开。

可以用有效边表加速算法

**三【上机调试中出现的错误信息、错误原因及解决办法】(10分)**

（记录下你调试程序中出现的错误信息的英文提示，分析出错原因及解决办法）

错误信息：

无法打开源文件QtWidgets/Qapplication

出错原因：

未知

解决方法：

重启电脑

**四【上机实验中的其他它问题及心得】（重点）(10分)**

**有效边表的学习：**

一条扫描线往往只与少数几条边相交，甚至与整个多边形都不相交。若在处理每条扫描线时，把所有边都拿来与扫描线求交，则其中大多数计算都是徒劳无用的。为了提高效率，在处理一条扫描线时，仅对与它相交的多边形的边进行求交运算。把与当前扫描线相交的边称为有效边，并把它们按与扫描线交点 x 坐标递增的顺序存放在一个链表中，称此链表为有效边表。假定当前扫描线与多边形某一条边的交点的x坐标为x，则下一条扫描线与该边的交点只要加一个增量△x。每条边需要存以下信息

第1项存当前扫描线与边的交点坐标x值；

第2项存从当前扫描线到下一条扫描线间x的增量D x；

第3项存该边所交的最高扫描线号ymax；

第4项存指向下一条边的指针

边按照链地址法解决冲突的哈希表的形式存储，其中哈希值为该边较低端点ymin

扫描之前建立有效边表，扫描中删去之后不会相交的边

**五【源程序、测试数据及结果】**

（对程序添加注释，包括公式和文字解释）

（程序结果请截图）

1. 实现多边形的填充。**(30分 程序20分 结果图10分)**

void gfill(QImage\* image, GPoint\* point, QRgb value, int mlength) {

double\* x, \* y;//一笔画顺序存储点的坐标

x = new double[mlength];

y = new double[mlength];

for (int i = 0; i < mlength; i++)

{

x[i] = point[i].x;

y[i] = point[i].y;

}

int N = 9;

int n = 0;

double suitable\_x[10];

double min\_y = FindMin(y, N), max\_y = FindMax(y, N);

for (int now\_y = min\_y; now\_y <= max\_y; now\_y++)// 扫描线从下到上

{

SuitableX(x, y, N, now\_y, suitable\_x, n);//求交点

Sort(suitable\_x, n); //从小到大排序

for (int i = 0; i < n; i += 2)//确定相交区间

{

for (int now\_x = suitable\_x[i]; now\_x < suitable\_x[i + 1]; now\_x++)//对相交区间涂色

{

image->setPixel(now\_x, now\_y, value);

}

}

}

delete[] x;

delete[] y;

}



1. 实现直线段的裁剪。**(30分 程序20分 结果图10分)**

void gclip(QImage\* image, Rectangle\* rect, int x1, int y1, int x2, int y2, QRgb value) {

int code1 = Encode(x1, y1, rect);

int code2 = Encode(x2, y2, rect);

QPointF p1, p2;

p1.setX(x1);

p1.setY(y1);

p2.setX(x2);

p2.setY(y2);

while (code1 != 0 || code2 != 0)

{

//在直线之外，直接舍去，但是跨过三个区域的情况舍弃不了，需要在后面分割一次再分别舍去

if ((code1 & code2) != 0)

return;

//处理大于上界的点

if ((code1 & 8) != 0)

{

p1.setX(p1.x() - ((p1.y() - rect->ymax) \* (p1.x() - p2.y()) / (p1.y() - p2.y())));

p1.setY(rect->ymax);

code1 = Encode(p1.x(), p1.y(), rect);

}

else if ((code2 & 8) != 0)

{

p2.setX(p2.x() - ((p2.y() - rect->ymax) \* (p2.x() - p1.y()) / (p2.y() - p1.y())));

p2.setY(rect->ymax);

code2 = Encode(p2.x(), p2.y(), rect);

}

//处理小于下界的点

else if ((code1 & 4) != 0)

{

p1.setX(p1.x() - ((p1.y() - rect->ymin) \* (p1.x() - p2.x()) / (p1.y() - p2.y())));

p1.setY(rect->ymin);

code1 = Encode(p1.x(), p1.y(), rect);

}

else if ((code2 & 4) != 0)

{

p2.setX(p2.x() - ((p2.y() - rect->ymin) \* (p2.x() - p1.x()) / (p2.y() - p1.y())));

p2.setY(rect->ymin);

code2 = Encode(p2.x(), p2.y(), rect);

}

//处理大于右界的点

else if ((code1 & 2) != 0)

{

p1.setY(p1.y() - ((p1.x() - rect->xmax) \* (p1.y() - p2.y()) / (p1.x() - p2.x())));

p1.setX(rect->xmax);

code1 = Encode(p1.x(), p1.y(), rect);

}

else if ((code2 & 2) != 0)

{

p2.setY(p2.y() - ((p2.x() - rect->xmax) \* (p2.y() - p1.y()) / (p2.x() - p1.x())));

p2.setX(rect->xmax);

code2 = Encode(p2.x(), p2.y(), rect);

}

//处理小于左边界的点

else if ((code1 & 1) != 0)

{

p1.setY(p1.y() - ((p1.x() - rect->xmin) \* (p1.y() - p2.y()) / (p1.x() - p2.x())));

p1.setX(rect->xmin);

code1 = Encode(p1.x(), p1.y(), rect);

}

else if ((code2 & 1) != 0)

{

p2.setY(p2.y() - ((p2.x() - rect->xmin) \* (p2.y() - p1.y()) / (p2.x() - p1.x())));

p2.setX(rect->xmin);

code2 = Encode(p2.x(), p2.y(), rect);

}

}

Mid(image, p1.x(), p1.y(), p2.x(), p2.y(), value);

Mid(image, rect->xmin, rect->ymin, rect->xmin, rect->ymax, value);

Mid(image, rect->xmax, rect->ymin, rect->xmax, rect->ymax, value);

Mid(image, rect->xmin, rect->ymin, rect->xmax, rect->ymin, value);

Mid(image, rect->xmin, rect->ymax, rect->xmax, rect->ymax, value);

}

