**郑州轻工业大学**

**《计算机图形学》**

**课程****实验报告**

实验名称：填充算法

姓 名： 原彬贺

学 号： 542001020223

地 点： 实验楼302

指导老师： 黄 伟、陈启强

专业班级： 计算机科学与技术20-02班

实验成绩：

1. 实验目的
2. 理解理解多边形填充的目的。
3. 掌握多边形的四连通区域种子填充算法。
4. 掌握区域填充的扫描线算法。

二、实验内容

区域种子填充算法通过设置一个起始点，通过从该点出发先不同方向染色，直到其到达边界。

而扫描线填充算法则是先将一条水平线染色，再分别看其上下两条线是否染色，依次进行。

通过对比可知，扫描线填充法运行效率较高，并且所占空间较少，适合进行使用。

三、实验步骤及结果

区域种子：

void flood\_fill\_8(int[] pixels, int x, int y, int old\_color, int new\_color)

{

if(x<w&&x>0&&y<h&&y>0)

{

if (pixels[y\*w+x]==old\_color)

{

pixels[y\*w+x]== new\_color);

flood\_fill\_8(pixels, x,y+1,old\_color,new\_color);

flood\_fill\_8(pixels, x,y-1,old\_color,new\_color);

flood\_fill\_8(pixels, x-1,y,old\_color,new\_color);

flood\_fill\_8(pixels, x+1,y,old\_color,new\_color);

flood\_fill\_8(pixels, x+1,y+1,old\_color,new\_color);

flood\_fill\_8(pixels, x+1,y-1,old\_color,new\_color);

flood\_fill\_8(pixels, x-1,y+1,old\_color,new\_color);

flood\_fill\_8(pixels, x-1,y-1,old\_color,new\_color);

}

}

}

扫描线填充

Stack stack=new Stack();//堆栈 pixel\_stack初始化

Stack.push (point)； //(x,y)是给定的种子像素

while (!stack.empty())

{

p=(Point)(stack.pop());//出栈，从堆栈中取一像素作种子像素

x=p.x;

y=p.y;

savex=x;//保存种子点的横坐标x的值

while (pixels [y\*w+x]!= boundary\_color)

{

pixels [y\*w+x]= new\_color;

x++;

} //从种子像素开始向右填充到边界

xright=x–1; //保存线段的右端点

x=savex–1; //设定种子点往左填充的起点

while (pixels [y\*w+x]!= boundary\_color)

{

pixels [y\*w+x] = new\_color;

x=x–1;

}

//从种子像素开始向左填充到边界，以上两步完成区间填充。

xleft=x+1； //保存线段的左端点，加1是因为前面 循环时多减一次

x=xleft; //起点是上次的左端点

y=y+1; //开始处理上一条扫描线

while(x<=xright) //在上一条扫描线上检查是否需要填充

{

span\_need\_fill=false; //先设定为不需要填充

while (pixels [y\*w+x] ==old\_color&&x<=xright )

{

//待填充的线段

span\_need\_fill=true; //发现有旧象素，需要填充

x=x+1;

} //待填充的线段处理完，即遇到边界色，!=old\_color跳出

if (span\_need\_fill) //如果区间需要填充，则将其右端点作为种子点压进堆栈

{

p=new Point(x-1,y);

stack.push (p); //进栈

span\_need\_fill=false;

}

//继续向右检查以防有遗漏

while (pixels [y\*w+x] !=old\_color &&x<=xright )

x=x+1;

} //在上一条扫描线上检查完

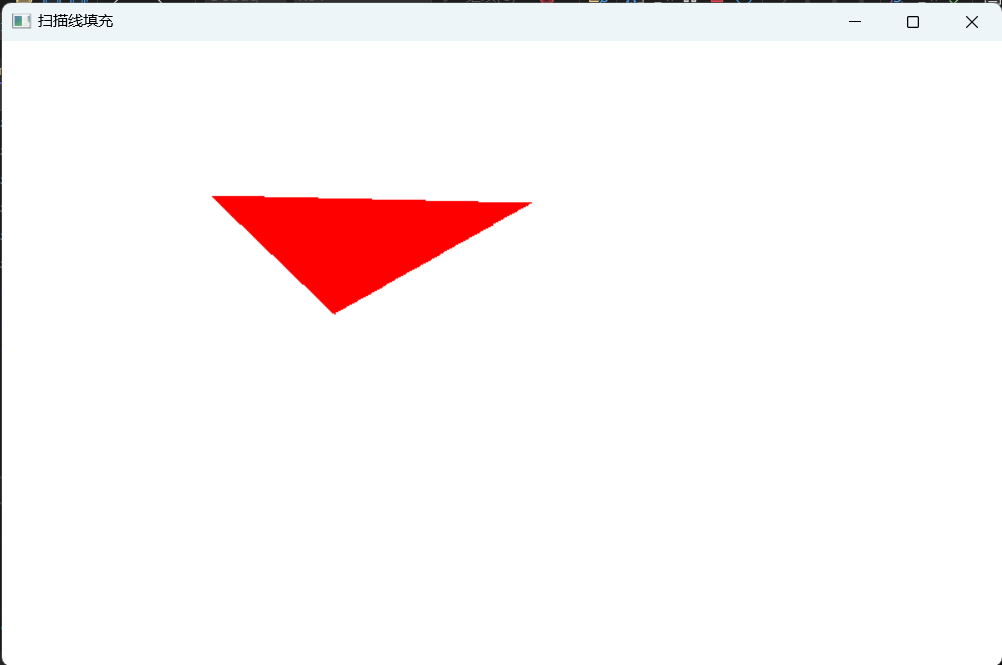
x＝xleft；

y=y–2; //形成下一条扫描线的y值

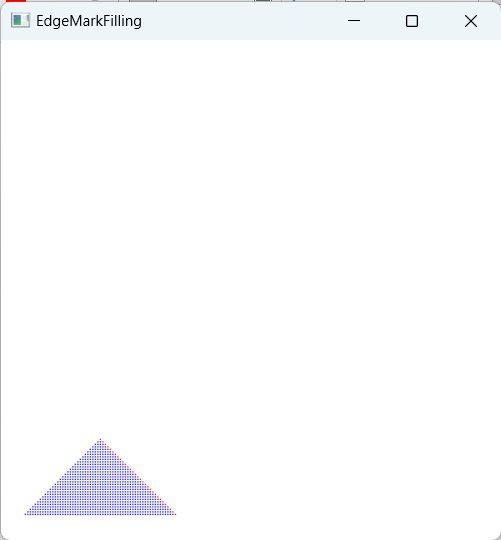
//在下一条扫描线上从左向右检查位于区间[xleft，xright]上的像素，其方法与在上一条扫描线上检查的情况完全一样，见书。

}//出栈完

四、实验结果及分析



图表 1种子填充



图表 2扫描线填充

五、实验总结

本次实验我通过实现种子填充算法与扫描线填充算法这两个图形学中的基本填充算法，种子填算法通过设置一个初始点，通过bfs或dfs进行逐渐染色，而扫描线填充法从一次一个点到一条线，极大的提高了染色效率。使我对计算机中颜色的填充有了更加充足的认识。虽然本次实验由于粗心导致程序无法运行，但是经过我的仔细排查，同时又通过网络来对错误进行排查，最终成功解决了问题。同时我也从更底层了解到颜色的填充过程，使我对计算机的了解更加深入，对我帮助很大，使我学到了很多知识，给我今后从事该方向有了更大的帮助。

|  |
| --- |
| **评语**（不要删除，指导老师填写评语，放置在实验报告后面单独成页） |
|  |