# 光栅图形学作业

张立博 2021012487

2023.3.21

## 1 代码逻辑

### 画线-Bresenham 算法

代码首先定义了起点坐标 $xA$、$yA$ 和终点坐标 $xB$、$yB$，并计算了 x 和 y 的增量方向 $sx$ 和 $sy$。然后使用 while 循环不断计算每个像素点的坐标，并将其绘制在图像上，直到达到终点坐标

在 while 循环内部，代码使用 Bresenham 算法中的增量计算方法来确定下一个要绘制的像素点的位置

具体来说，代码使用变量 $err$ 和 $e2$ 来表示当前像素点到直线的误差，$e2 = 2 * err$

然后根据直线的斜率关系来计算下一个要绘制的像素点的位置

如果 $e2 > -dy$，则 x 的值加上 sx; 如果 $e2 < dx$，则 y 的值加上 sy, 同时更新对应的 $err$

这样可以避免小数与除法，并且可以绘制斜率不存在和为 0 的直线段，同时不用单独对特殊情况进行处理

### 画圆-扫描转换算法

由于圆弧具有八对称性，所以只要扫描 1/8 圆弧就可以求出整个圆弧的像素集

方便起见，可以先考虑半径相同，以原点为圆心的圆，绘制时像素点进行平移

代码中以 $(0, R)$ 为起点，顺时针为方向绘制八分圆

代码首先定义了圆心坐标 $cx$、$cy$ 和半径 $radius$，并初始化变量 x、y 和 d。然后使用 while 循环不断计算圆上每个像素点的坐标，并将其绘制在图像上

在 while 循环内部，代码使用增量计算方法来确定下一个要绘制的像素点的位置

具体来说，代码使用变量 d 来表示当前像素点到圆形的误差，然后根据 d 和圆的形状关系来计算下一个要绘制的像素点的位置

代码将 d 初始化为 $1 - radius$ 预先计算两个增量值 $deltaE$ 和 $deltaSE$, 如果 d<0，则

增量值为 $deltaE$，否则增量值为 $deltaSE$
这种方法可以使浮点数改为整数，将乘法运算改为加法运算，提高算法的效率

### 填充-非递归版本

基于宽度优先搜索实现
代码首先初始化一个队列，将种子点入队；同时记录种子点像素的颜色为 $oldColor$，若队列非空，则进入 while 循环
在 while 循环内部，获取队列首元素坐标为当前坐标并令其出队
若当前坐标像素颜色不等于 $oldColor$, 则跳过剩下的部分进行判断是否开始下一次循环
否则将当前坐标像素染为填充颜色，同时判断其四周点是否在绘图区域内以及颜色是否为 $oldColor$, 若是则令其入队
这种基于宽度优先搜索的四联通填充算法可以避免递归实现，提高了效率

## 2 代码参考

未与同学进行讨论与网上借鉴

## 3 问题

1. 像素点位置确定有误: 光栅图形学中像素繁多，需要在代码中精确定位每一个像素坐标才能完成较为标准的图形
2. 通过误差确定下一个绘制的像素点: 需要较多通分和运算确定比较的关系式，同时要尽量用整数和乘除替换浮点数和加减