《计算机图形学》3 月报告

梁宇方，学号 171860695

邮箱 3509175458@qq.com

摘　要: 计算机图形学的实验部分的实验报告

1　实验概述

本次实验要求使用 Python3 语言编程，实现一个绘图系统。内容包括核心算法、命令行系统和图形界面系统的实现。

2　实验要求

根据所给的代码框架，完成三个.py 文件，即

1. cg_algorithm.py（基本完成）
2. cg_cli.py（基本完成）
3. cg_gui.py（未完成）

3　完成情况

1. 重置画布（原有）
2. 保存画布（原有）
3. 设置画笔颜色（原有）
4. 绘制线段
   a) DDA 算法（完成）
   b) Bresenham 算法（完成）
5. 绘制多边形（完成）
6. 绘制椭圆（完成）
7. 绘制曲线（未完成）
8. 图元平移（完成）
9. 图元旋转（完成）
10. 图元缩放（完成）
11. 对线段裁剪
    a) Cohen-Sutherland（完成）
    b) Liang-Barsky（未完成）

4 算法的理解

1. DDA 画直线算法
   朴素算法为计算直线方程 y=kx+b 中的 k 和 b，然后 x 每增加 1，y 增加 k；此方法的缺陷在于，当 k 值过大时画出来的直线会断断续续，并且无法画出 k=∞时的直线。DDA 算法为朴素算法的改进，在计算出|k|>1 或判断 k=无穷后，DDA 算法会切换步进的方向，改为 y 每增加 1，x 增加 1/k，有效的解决了朴素算法的两个缺陷
2. Bresenham 画直线算法
   DDA 算法中涉及浮点运算，Bresenham 算法的目的即尽可能地消除算法中地浮点运算以提高计算速度。Bresenham 算法的关键在于误差累积。假设|k|<1 即 dx>dy，那么此时 x 每增加 dx，y 增加 dy，即 x 每增加 dx/dy，y 增加 1，Bresenhan 记录下误差 e 的值，x 每加 1，e 增加 dx，当 e 第一次超过 dy 时即意味着 x 的增加超过了 dx/dy，此时给 y 增加 1，并让 e 减少 dy。算法的过程可以不涉及浮点运算，因为我们并不关心 k 的确切值而只关心 dx 和 dy 的相对大小，运算速度比 DDA 画直线算法更快
3. 中点椭圆算法
   使用的是中点椭圆算法中的 Bresenham 改进版本，通过解一个复杂的方程，求出误差 e 的迭代关系。从(0,b)处开始绘图在 k=-1 处交换坐标轴，直到绘图至(a,0)。因为椭圆是轴对称图形，可以同时在其他三个象限的对应位置进行绘制（因为约定了椭圆不能旋转，所以椭圆可以由四个参数完全确定。使用 Bresenham 的改进版本使得计算过程同样不涉及浮点运算。
4. Cohen-Sutherland 线段裁剪算法
   朴素的算法为判断线段的两个端点是否在窗口外，如果端点在窗口外，则寻找线段与窗口的一个焦点代替该端点。该算法低效率的原因是有大量不需裁剪的，在窗口内的直线也经历了繁复的计算。Cohen-Sutherland 将窗口划分的区域编码，通过位运算迅速判断端点所在的位置，对于完全在视窗外或者完全在视窗内的线段选择直接忽略。对于需要裁剪的线段，才进行计算，极大地提高了绘图效率

References:

[1] https://blog.csdn.net/u010429424/article/details/77834046 DDA 算法和 Bresenham 算法

[2] https://www.cnblogs.com/clairvoyant/p/5540023.html 中点椭圆算法

[3] https://blog.csdn.net/jxch /article/details/80726853 Cohen-Sutherland 直线裁剪算法