PA0: 光栅图形学实验报告

容逸朗 2020010869

1. draw 函数实现逻辑

1.1 直线段描画

函数已知线段的颜色及两端点，由此可以得到直线方程 $y = kx + b$。

首先讨论斜率不存在的情况 $(x_A = x_B)$，此时遍历 y_A 到 y_B 描点即可。

其他情况均有斜率，这时可以考虑 `Bresenham` 算法：

- 对于斜率属于 $[0, 1]$ 的情况：
  - 记误差初始值 $e = -0.5$
  - 由小至大遍历 x 轴
  - 假设 (x, y) 已被选取，此时只需¥考¥虑 $(x + 1, y)$ 与 $(x + 1, y + 1)$ 的情况
  - 若误差 $e \geq 0$ （更接近 $y + 1$） ，则 e 减少 1 、 y 增大 1，否则 y 不变。

其他情况如下表所示：

（步长指误差 e 每次增加的值，修正大小指达到误差修正條件后 e 改变的大小）

斜率	步增轴	步长	误差初始值	误差修正条件	修正大小
$k < -1$	y	k	$e = 0.5$	$e \leq 0$	-1
$-1 \leq k < 0$	x	k	$e = 0.5$	$e \leq 0$	-1
$0 \leq k \leq 1$	x	k	$e = -0.5$	$e \geq 0$	1
$k > 1$	y	k	$e = -0.5$	$e \geq 0$	1

观察上表，可以发现负斜率的情况乘上 -1 便可得到正斜率的情况，为此可以引入变量 f ，当斜率为负则为 -1 ，否则为 1 。

因此可以把情况减少为两类：

斜率	步增轴	步长	误差初始值	误差修正条件	修正大小
$\lvert k\rvert \leq 1$	x	k	$e = -0.5 * f$	$e * f \geq 0$	f
$\lvert k\rvert > 1$	y	k	$e = -0.5 * f$	$e * f \geq 0$	f

为了避免浮点数运算，对于 $\lvert k\rvert \leq 1$ 的情况（其他亦然）可以做变换 $e' = 2e\,dx$ ，此时步长变为 $k' = 2\,dy$ ，修正大小变为 $2f\,dx$ ，误差初始值为 $e' = -dx$ 。

斜率	步增轴	步长	误差初始值	误差修正条件	修正大小
$\|k\| \leq 1$	x	$2\,dy$	$e = -dx$	$e * f \geq 0$	$2f\,dx$
$\|k\| > 1$	y	$2\,dx$	$e = -dy$	$e * f \geq 0$	$2f\,dy$

利用上表的参数即可完成线段描画的工作。

1.2 圆描画

此函数的参数包括圆心位置、半径 R 和颜色。

- 为方便计算，可以假定目标圆圆心为圆点，先计算圆的相对位置，再通过平移操作得到实际描点的位置。
- 由于圆有对称性，因此只需要知道圆上任意一点，即可通过（当圆心定为圆点时的）四条对称轴：$y = 0, x = 0, y = x, y = -x$ ，得到其余 7 个点的相对位置。
- 最后通过中点画线法即可完成画圆任务：
  1. 取函数 $F(x, y) = x^2 + y^2 - R^2$ ，若某点 (x_0, y_0) 在圆上，则 $F(x_0, y_0) = 0$ 。
  2. 此时考察 $x = 0, y = R$ 处顺时针数 45 度的八等分圆。
  3. 若 (x, y) 在圆弧上，那么 $(x + 1, y)$ 和 $(x + 1, y - 1)$ 都可能在圆上。
  4. 故可构造判别式 $d = F(x + 1, y - 0.5) = (x + 1)^2 + (y - 0.5)^2 - R^2$ 。
  5. 若 $d < 0$ ，则中点在圆内，取点 $(x + 1, y)$ 更佳。
  6. 否则，若 $d \geq 0$ ，则中点在圆外，取点 $(x + 1, y - 1)$ 更接近实际情况。
  7. 重复上述操作 3 - 6 直至 $x \geq y$ 即完成任务。
  8. 每次重新代入 x 与 y 计算都是很麻烦的，例如对于第 5 条操作，对下一点的判别式为：$d_1 = F(x + 2, y - 0.5) = (x + 2)^2 + (y - 0.5)^2 - R^2 = d + 2x + 3$ ；
     同理，第 6 条操作也可化简为：$d_1 = F(x + 2, y - 1.5) = d + 2(x - y) + 5$ 。
  9. 为了避免浮点数运算，取判别式 $d' = 4d$ 即可。

1.3 区域填充

此函数的参数包括填充区域内一点 (x_0 , y_0) 和要填充之颜色。

1. 先得到此点之颜色（称为旧颜色），再将此点入队。
2. 若队非空，则从队头中拿出点，否则算法结束。
3. 向外查找与旧颜色相同的相邻点，并改变其颜色。
   - 为避免重复计算，可以从此点出发，向右侧不断查找可行点染色，直至不能染色，此时记下右边界 x_R 。
   - 然后从初始点向左做同样操作，记下左边界 x_L。
4. 此时区域被划分为上下两部分。（每部分之间不一定连续）
5. 先讨论上半部分（下半部分的操作相同）
   - 从左边界上方的点 $(x_L, y + 1)$ 开始，不断向右查找旧颜色的点，直至颜色不再相同，此时将右边界点入队。（此点应是旧颜色）
   - 此时向右查找，若找到某点是旧颜色，且尚未超出右边界，则从此点开始做上面的操作。

- 遍历完毕，对下半部分做同样操作，完成此步后回到操作 2 。

本部分需要特别注意边界判断的问题，因此改变坐标后应该先判断此点是否合法，否则可能出现段错误。

2. 其他问题

2.1 你在完成作业的时候和哪些同学进行了怎样的讨论?是否借鉴了网上/别的同学的代码?

完成此次作业并没有何任何同学进行讨论，代码内容主要是参考了课本上相关小节的内容。

2.2 你的代码有哪些未解决的 bug？如果给你更多时间来完成作业，你将会怎样进行调试?

代码完好，理论上不存在 bug。若有更多时间完成作业，我会利用更大的画布来测试所用算法的效能。

2.3 你对本次作业有什么建议? 文档或代码中有哪些需要我们改进的地方?

我认为本次作业的框架和指引都十分清晰，而且任务量恰到好处，暂时没有需要改进的地方。