个人信息和实验环境

杨雅儒, 2017011071, 计85

Ubuntu 18.04, gcc 7.5.0

问题回答

绘制逻辑

线的绘制

采用整数类型的 Bresenham 算法，**利用x、y互换的方式统一了斜率绝对值大于等于 1 和小于 1 的情况**。

具体来说：

- 首先处理 xA 与 xB 相等的情况（此时斜率不存在），此时一定都在整点上，直接 for 循环绘制即可；
- 然后判断是否需要互换横纵坐标（斜率绝对值大于 1 时需要互换），并且保证 xA <= xB；
- x 从 xA 开始循环到 xB，若 yB >= yA，则 e 从 -dx 开始且每次大于等于 0 时减去 2 * dx 同时 ++y；若 yB < yA，则 e 从 dx 开始且每次小于等于 0 时加上 2 * dx 同时 --y。

这样就处理完了所有情况，代码如下：

```
if (xA == xB) {
    if (yA > yB) {
        std::swap(yB, yB);
    }
    for (int y = yA; y <= yB; ++y) {
        img.SetPixel(xA, y, color);
    }
} else {
    // 记录是否 x, y 互换
    bool rev = (fabs((double)(yA - yB) / (xA - xB)) > 1);
    if (rev) {
        std::swap(xA, yA);
        std::swap(xB, yB);
    }
    if (xA > xB) {
        std::swap(xA, xB);
        std::swap(yA, yB);
    }
    // 此时一定有 |k| <= 1 且 xA <= xB
    int dx = xB - xA, dy = yB - yA, y = yA, e = ((dy >= 0) ? -dx : dx);
    for (int x = xA; x <= xB; ++x) {
        img.SetPixel((rev ? y : x), (rev ? x : y), color);
        e += 2 * dy;
        if (dy >= 0) {
            if (e >= 0) {
                e -= 2 * dx;
                ++y;
```

```
            }
        } else {
            if (e <= 0) {
                e += 2 * dx;
                --y;
            }
        }
    }
}
```

圆的绘制

采用整数类型的中点画圆法，利用对称性，只需要考虑 1/8 的圆弧。

具体来说：

- 从 (0, R) 开始，判别式初值 d = 5 - 4R，绘制初始点；
- 每次将 x++，并判断 d 与 0 的大小关系，若 d < 0 则 d += 4 * (2*x+3)；若 d >= 0 则 d += 4 * (2*(x-y)+5) 且 y--。循环直到 x > y，并在这个过程中以圆心坐标作为偏移绘制 (x, y)。

代码如下 (其中 circlepoints 表示对称地绘制八个点)：

```
int x = 0, y = radius;
// d 乘上 4 变为整数
int d = 5 - 4 * radius;
circlepoints(img, x, y);
while (x <= y) {
    if (d < 0) {
        d += 4 * (2*x+3);
    } else {
        d += 4 * (2*(x-y)+5);
        y--;
    }
    x++;
    circlepoints(img, x, y);
}
```

区域填充

采用非递归的扫描填充算法。

具体来说：

- 首先存储旧颜色 oldColor，并且从染色点向右找到连通且最靠右的旧颜色位置，将这个点坐标压入栈中；
- 每次从栈中弹出栈顶 (ox ,oy)，并且从该点开始向左将连通的旧颜色点都染成新颜色，令找到的最靠左的横坐标为 lx；
- 分别从 (lx, oy+1) 和 (lx, oy-1) 开始，向右找到与 ([lx,ox], oy) 这一线段连通且为旧颜色的水平线段，将这些线段最右边的点都压入栈中即可。

代码如下：

```
Vector3f oldColor = img.GetPixel(cx, cy);
if (oldColor == color) return;
// 注意每次压入栈中的为一段中最右边的像素点
while (cx < img.Width() - 1 && img.GetPixel(cx + 1, cy) == oldColor) ++cx;
std::stack<pair<int, int>> stk;
```

```
stk.push(make_pair(cx, cy));
while (!stk.empty()) {
    int ox = stk.top().first, oy = stk.top().second;
    stk.pop();
    int x = ox, y = oy;
    // 从 (x, y) 向左染色
    while (x > 0 && img.GetPixel(x - 1, y) == oldColor) {
        img.SetPixel(x, y, color);
        --x;
    }
    img.SetPixel(x, y, color);
    int lx = x;
    for (y = oy - 1; y <= oy + 1; y += 2) {
        // 枚举上下相邻两行，从 (lx, y) 开始向右扩展，找到每段的最右像素点
        // 注意若在 ox 之右发生断裂则不继续向右寻找(即目前已染色的段为 ([lx,ox], oy))
        x = lx;
        if (y < 0 || y > img.Height()) continue;
        while (x <= ox) {
            while (x <= ox && img.GetPixel(x, y) != oldColor) ++x;
            if (x <= ox) {
                while (x < img.Width() - 1 && img.GetPixel(x + 1, y) ==
oldColor) ++x;
                stk.push(make_pair(x, y));
                ++x;
            }
        }
    }
}
```

交流/讨论

并未与任何同学进行交流讨论，也没有借鉴任何网上或其它同学的代码，只参考了电子书和作业文档，并通过网络查阅了 B 样条和 Bezier 曲线的相关资料。

遇到的问题和未解决 bug

本次实验比较简单，没有遇到什么问题，从测试结果看也没有未解决的 bug。

如何编译并渲染

在 `code/` 目录下使用如下命令：

```
bash run_all.sh
```

结果将存放在 `code/output/` 目录下。