《计算机图形学》10 月报告

马珩峻 171860627

(南京大学 计算机科学与技术系, 南京 210093)

1 已完成的任务

算法部分：实现了所有图元的绘制，平移，旋转，缩放和线段的剪裁算法。
系统功能部分：实现了从控制台或者文件读取指令并识别、解析和执行指令，实现了图像的保存功能。

2 算法说明

2.1 线段绘制：

2.1.1 DDA 算法，通过直线方程 y=kx+b 或 x=my+n 计算 k 或 m（x 方向距离长时候选 k，y 方向距离长时候选 m），计算用 x 方向距离和 y 方向距离除 k 或 m 得到 dx,dy（必有一个为 1,一个小于 1），在为 1 的方向上步进，另一方向加偏移量，画出每次一步进的点，获得线段。

2.1.2 Bresenham 算法：在较长距离的方向上步进，通过计算距离差判断要取的下一个点的坐标，因为运算中只涉及乘 2 和整数加法运算，乘 2 课通过左移一位实现，所以该算法速度较快并且在硬件上容易实现。

2.2 多边形绘制：

把顶点依次两两组成线段顶点，通过遍历线段调用线段绘制方法实现多边形绘制

2.3 椭圆绘制：

中点圆算法：

椭圆可表示为 F(x, y) = b2x2 + a2y2 - a2b2 = 0。

因为椭圆在 x,y 方向上的步进会变化，dx 和 dy 的变化会随画点的位置变得某一个相对更显著，所以我们在切线斜率=1 的地方将椭圆划分为上下两部分分别处理，防止出现一些突变和跳跃的采样点。斜率判断用每一步的 dxdy 判断，b^2*dx<a^2*dy 时为上半部分

另外根据椭圆的对称性，我们只计算右上 1/4 部分的采样点。然后通过对称补全另外 3/4 部分的椭圆。

从短轴顶点开始，判别式 d 初始值为 b^2 + a^2 * (-b + 0.25) + 0.5

之后上半部分在 x 方向上步进，d<0 时候 d+=b^2 * (2 * dx + 3)，

d>=0 时 y 方向上减一个单位长度，d+=b^2 * (2 * dx + 3) + a^2 * (-2 * dy + 2)

到下半部分后在 y 方向上步进，d>0 时候 d+=a^2 * (-2 * dy + 2)，

d<=0 时 x 方向上加一个单位长度，d+=b^2 * (2 * dx + 3) + a^2 * (-2 * dy + 2)

2.4 曲线绘制：

2.4.1 Bezier 算法：

根据 Bezier 曲线公式：

$$B(t)=\sum_{i=0}^{n}\binom{n}{i}(1-t)^{n-i}t^{i}P_i(t\in[0,1])$$

在区间[0,1]上均匀取 1000 个分量带入公式计算点坐标并画点

2.4.2 B-spline 算法：

ppt 中并未对 B 样线的次数等作出细致要求，程序中默认画 3 次曲线。

节点向量在[0,1]区间上均分为 n+k+1 段。

根据 B 样条曲线公式：

$p(u)=\sum_{i=0}^{n}P_iN_{i,k}(u)$

其中$N_{i,k}(u)$是 k 次 B 样条基函数，定义为：

$$\begin{cases}N_{i,0}(u)=\begin{cases}1, & u_i<u<u_{i+1}\\ 0, & 其他情况\end{cases}\\ N_{i,k}(u)=\dfrac{(u-u_i)N_{i,k-1}(u)}{(u_{i+k}-u_i)}+\dfrac{(u_{i+k+1}-u)N_{i+1,k-1}(u)}{(u_{i+k+1}-u_{i+1})},u_k\le u\le u_{n+1}\end{cases}$$

实际计算时采用了 deboor 算法，通过递归函数 deBoorX 和 deBoorY 分别计算当前采样点的坐标并画点。

2.5 图元平移：

将图元的控制点坐标平移后重画。

平移点公式为：x+=dx, y+=dy

2.6 图元旋转：

将图元的控制点坐标选择后重画，此处有个特例，因为椭圆只存储了圆心和长短轴长度，为了实现椭圆旋转，需要一新属性 alpha 记录椭圆长轴与水平方向的夹角，然后画椭圆时候，采样点基于椭圆的圆心旋转 alpha 角度后再画，旋转椭圆时，先计算圆心旋转后的坐标，然后根据几何证明可知新的 alpha 等于旧的 alpha 加上旋转角度 theta。

图上点的旋转公式为：（通过极坐标转化公式可推导）

x = (originX - centerX) * cos(theta) - (originY - centerY) * sin(theta) + centerX

y = (originX - centerX) * sin(theta) + (originY - centerY) * cos(theta) + centerY

centerX,Y 为旋转中心，originX,Y 为原来的点坐标，theta 为旋转角

2.7 图元缩放：

将图元的控制点坐标按照缩放重新确定位置后重画。椭圆需要缩放长短轴长度。

点的缩放公式为：

x = centerX + (x - centerX) * scale

y = centerY + (y - centerY) * scale

centerX,Y 为缩放中心，scale 为缩放倍数

2.8 线段剪裁：

Cohen-Sutherland 算法：

将画布区域编码，根据线段端点所在区域给线段端点编码，求两端点编码的逻辑与和逻辑或结果，根据结果处理线段端点进行截断。

Liang-Barsky 算法：

用参数方程表示直线：

$$x=x_1+u\cdot(x_2-x_1)=x_1+\Delta x\cdot u$$

$$y=y_1+u\cdot(y_2-y_1)=y_1+\Delta y\cdot u$$

把被裁剪的线段看成是一条有方向的线段，把窗口的四条边分成两类：入边和出边

裁剪结果的线段起点是直线和两条入边的交点以及始端点三个点里最前面的一个点，即参数 u 最大的那个点；

裁剪线段的终点是和两条出边的交点以及端点最后面的一个点，取参数 u 最小的那个点。

3 系统功能说明

3.1 指令解析

指令解析器通过正则匹配实现，通过正则匹配指令的 pattern，并将参数部分捕获打包成列表返回，再根据指令的具体类型转化参数类型验证指令合法性并执行。

3.2 图像保存

另外因为 OpenGL 没有保存图像的功能函数，所以借用了 OpenCV 先将当前图像的缓存 buf 读出，用 numpy 把 buf resharp 为 OpenCV 的图像格式后用 OpenCV 的 imwrite 保存图像。

3.3 指令输入

另外因为如果在 OpenGL 的注册的 display 函数里等待输入的话会导致 OpenGL 的 MainLoop 因等待输入使得画布窗口假死，所以使用了子线程执行输入函数 inputThread 来将输入放入 orders 队列，而注册的 displayFunc 中若队列不为空则取出一个 order 进行解析执行。

References:

[1] 课程 ppt

[2] n 阶 bezier 曲线 通用公式说明和应用 https://blog.csdn.net/korekara88730/article/details/82505860

[3] B 样条曲线（B-spline Curves）https://blog.csdn.net/qq_40597317/article/details/81155571

[4] De Boor's algorithm https://en.wikipedia.org/wiki/De_Boor%27s_algorithm

[5] PyOpengl 学习(三)：绘制点、线、面（上） https://blog.csdn.net/BigBoySunshine/article/details/80308596