《计算机图形学》系统设计 11 月进展报告

张帅 151220162

目录

概要

本图形绘制系统采用面向对象设计，以 Qt 和 OpenGL 作为基础，目前已实现绘制、填充及编辑功能，可以绘制的图形包括直线、圆、椭圆、多边形，可以填充的图形包括圆和多边形。使用 Qt 提供界面交互，使用 OpenGL 提供绘制，界面简洁美观，操作方便。

本月新增功能：

1. 椭圆绘制、圆和多边形的填充，其中椭圆绘制使用了中点椭圆生成算法，多边形填充使用了多边形扫描填充算法，圆的填充算法为自写；
2. 将原先工程使用 Qt 重新实现，添加了更加用户友好的交互界面；
3. 添加了图形编辑功能。

1 算法原理介绍

1.1 中点椭圆生成算法

将椭圆按照象限分为 4 个部分，先绘制出第一象限的部分，然后根据对称性绘制出其余 3 个象限的部分。

第一象限的部分分为两个区域：区域 1：切线斜率小于 1；区域 2：切线斜率大于 1.

区域 1：

决策参数：$p1_k = f_{ellipse}\left(x_{k+1}, y_k - \frac{1}{2}\right) = r_y^2(x_k+1)^2 + r_x^2(y_k - \frac{1}{2})^2 - r_x^2 r_y^2.$

初始取值：$p1_0 = r_y^2 - r_x^2 r_y + \frac{r_x^2}{4}.$

更新公式：$\begin{cases} p1_{k+1} = p1_k + 2r_y^2 x_k + 3r_y^2, & p_k < 0 \\ p1_{k+1} = p1_k + 2r_y^2 x_k - 2r_x^2 y_k + 2r_x^2 + 3r_y^2, & p_k \geq 0 \end{cases}.$

区域 2：

决策参数：$p2_k = f_{ellipse}\left(x_k + \frac{1}{2}, y_k - 1\right) = r_y^2(x_k + \frac{1}{2})^2 + r_x^2(y_k - 1)^2 - r_x^2 r_y^2.$

初始取值：$p2_0 = r_y^2(x_0 + \frac{1}{2})^2 + r_x^2(y_0 - 1)^2 - r_x^2 r_y^2.$ (其中(x_0, y_0)为区域 1 最后一个点

更新公式：$\begin{cases} p2_{k+1} = p2_k - 2r_x^2 y_k + 3r_x^2, & p_k > 0 \\ p2_{k+1} = p2_k + 2r_y^2 x_k - 2r_x^2 y_k + 2r_y^2 + 3r_x^2, & p_k \leq 0 \end{cases}.$

1.2 多边形扫描填充算法

先后使用了有序边表与活化边表，得到各扫描行的填充范围，从而进行填充。

边表使用了如下数据结构：

```
struct Edge
{
    Edge() {}
    Edge(double xi, double dx, int ymax)
    {
```

```
        this->xi = xi;
        this->dx = dx;
        this->ymax = ymax;
    }
    double xi; //本扫描行该边的起始点横坐标
    double dx; //该边y每增加1，x的增加量，即斜率的倒数
    int ymax; //该边最高点的y值
};
```

a. 生成有序边表：扫描多边形中所有边，跳过所有点 y 值相同的水平边，将所有边最下方的点加入到有序边表中，填入对应的 xi，根据该边起始点计算出 dx。若该点在上下侧，则 ymax 为该边的最高点 y 值；若该点在左右两侧，则 ymax 为该边最高点 y 值-1（为了防止重复绘制顶点带来的内外侧判断错误）。

b. 根据有序边表生成活化边表：扫描当前扫描线及以下的所有有序边表，将 ymax 大于等于当前 y 值的 Edge 项加入到本扫描线对应的活化边表中，加入时 xi 需要根据当前 y 值和有序边表中的 xi，dx 计算，其余项保持不变。然后根据 xi 对当前扫描线的 Edge 项从小到大排序。

c. 根据活化边表生成各扫描行填充范围：遍历当前扫描行所有 Edge 项，两两扫描，从索引为偶数的项开始，填充到下一个项的 xi 为止。

2 系统框架设计

2.1 实验环境

操作系统：Windows 10
编程语言：C++11
开发环境：Qt Creator，Qt 5.4

2.2 系统架构

用 Qt 向用户提供交互，用 OpenGL 提供底层绘制，考虑到 Qt 与 OpenGL 的兼容性，使用了继承自 QWidget 的子类 QGLWidget 实现 Qt 环境下的 OpenGL 绘制。

QMainWindow 负责与用户交互，并以标签页的形式集成了 QGLWidget；

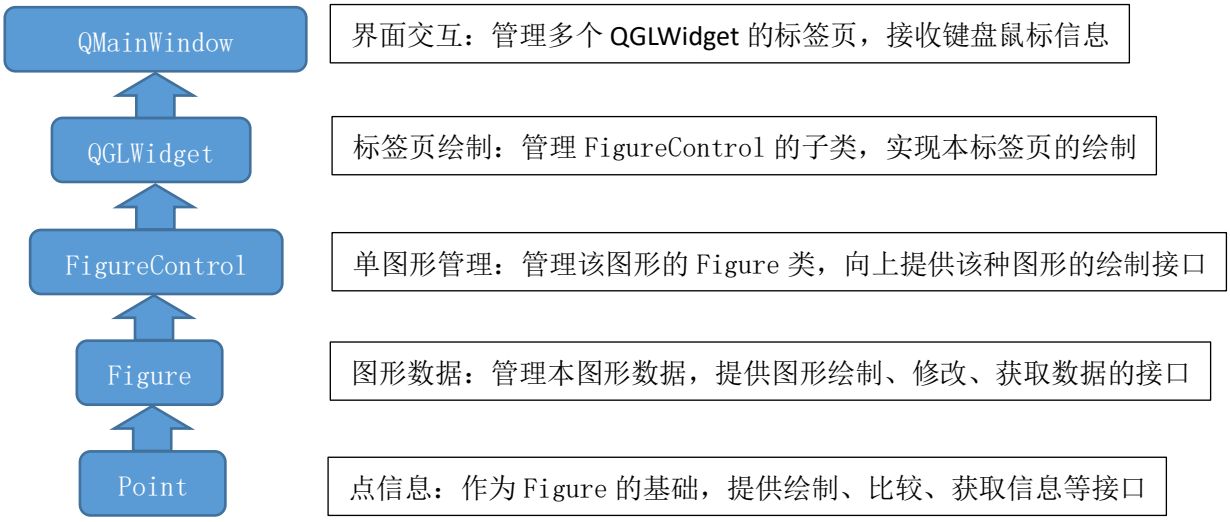
QGLWidget 负责绘制本标签页的内容，并集成了各类 FigureControl，用于将本标签页的交互信号送给不同的图形控制类；

FigureControl 负责该类型图形的交互，有 LineControl、CircleControl、EllipseControl、PolygonControl 等，用于分别提供不同图形的不同交互方式；

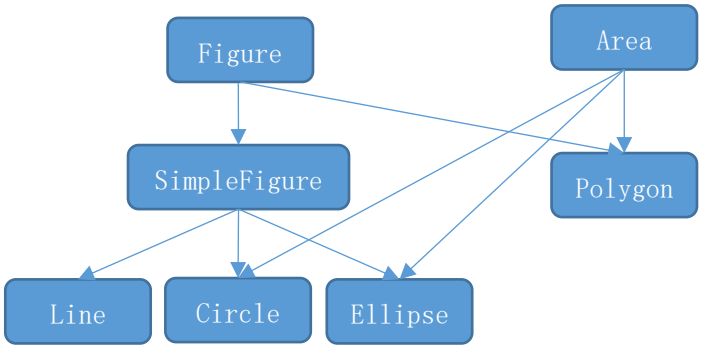
Figure 负责保存该图形的数据信息，包括顶点、圆心、半径以及轮廓点、填充点等，集成了 Point 类，图形绘制通过调用 Point 的 draw 函数完成；

Point 负责实现绘制单个普通点或标记点，并向上提供其他常用功能函数支持。

系统核心架构如下图：



相比于上个月，图形类架构略有改进。改进后的继承关系如下：



其中，Figure 类为一个抽象类，是所有图形的基类；Area 类为所有可填充图形的基类，提供图形填充的接口；SimpleFigure 为直接由点组成的简单几何图形的类；Polygon 类内聚集了 Line，故不是直接由点组成的图形。

3 系统功能介绍及使用说明

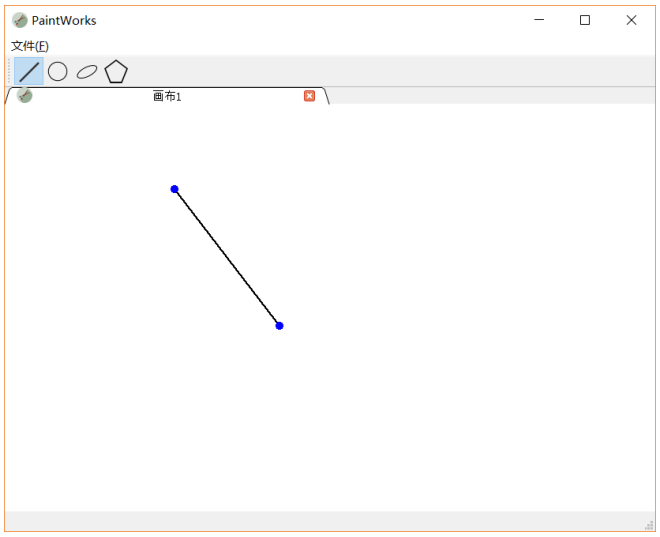
本系统可以绘制 4 种图形：直线、圆、椭圆、多边形，可以实现对圆和多边形的填充。图形相互之间如有覆盖，则按照绘制顺序进行覆盖。绘制过程中会以标记的形式显示当前正在绘制的图形，绘制完成后可以拖动蓝色的标记点对刚完成绘制的图形进行修改。

进入系统之后，“文件”中点选“新建画布”或按快捷键 Ctrl+N 即可开始绘制，点击上方按钮（或按快捷键 Alt+L/C/E/P）可以选择将要绘制的图形。

本报告中所述功能**均已实现**。

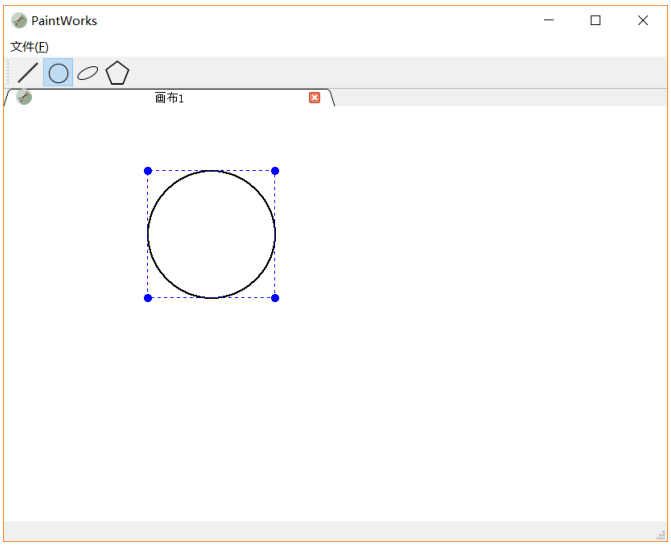
3.1 直线绘制+编辑

直线绘制只需用鼠标点选即可。鼠标左键在直线起始点按下，按住不放，直到移动到直线终点后放开鼠标左键即可完成绘制。在拖动过程中可以看到直线的实时显示。如图：

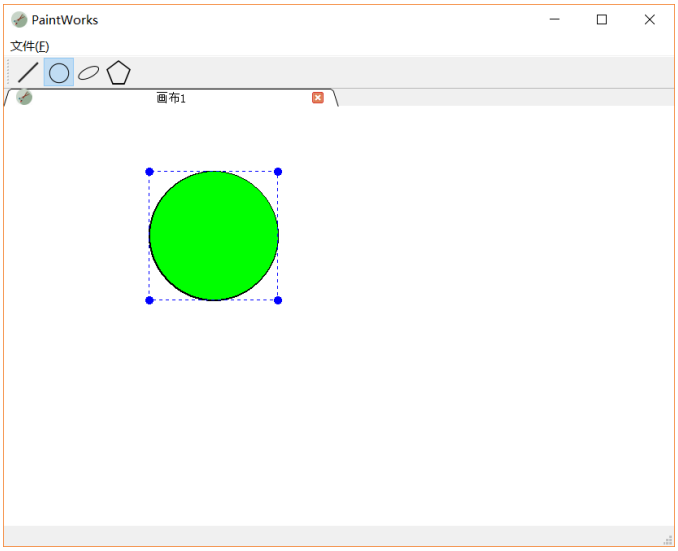


3.2 圆绘制&填充+编辑

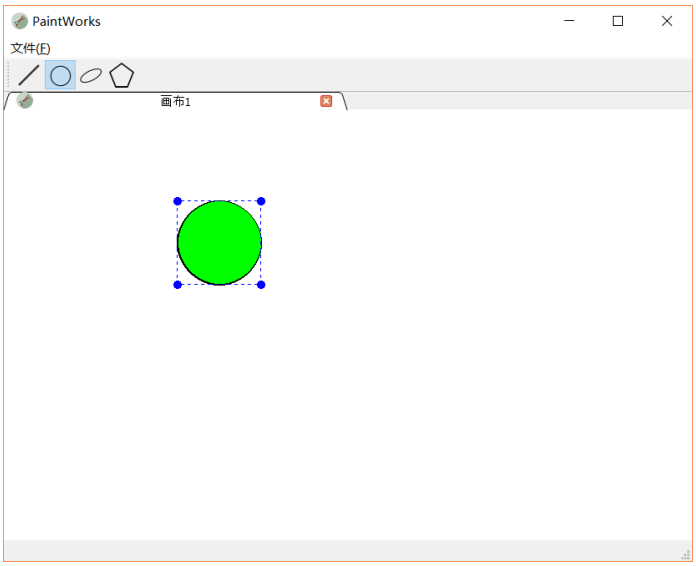
圆的绘制通过鼠标点选圆心+拖动半径完成。鼠标左键点下后拖动即可完成圆的绘制。如图：



按快捷键 F 即可实现填充：

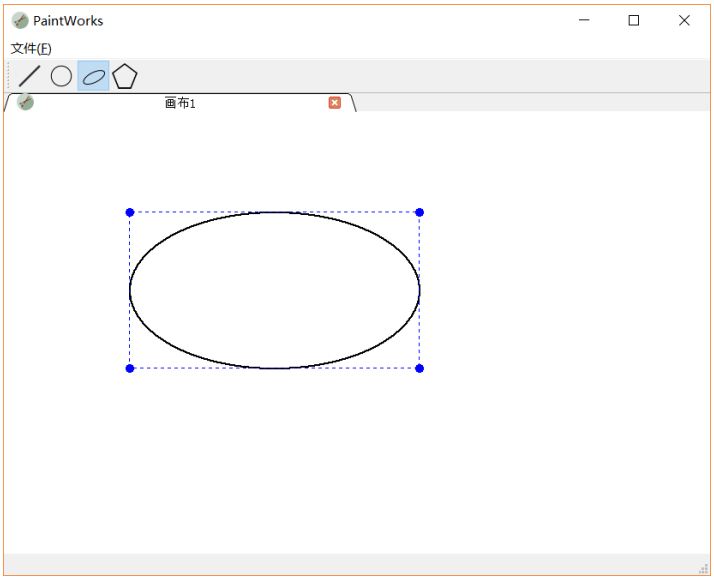


拖动蓝色标记点可以对圆进行编辑，实时更新填充点：



3.3 椭圆绘制+编辑

椭圆绘制也是通过点击+拖动完成，同样完成之后可以拖动蓝色点进行更改，如图：



3.4 多边形绘制&填充+编辑

多边形绘制只需用鼠标点选多个点即可。在起始点位置用鼠标左键点击一下，绘制第一条边，再次点击之后自动开始绘制第二条边，通过同样的方式添加多条边，最终在起始点位置点一下将折线闭合即可完成一个多边形的绘制。效果如下图：

这里考虑到准确点到起始点可能较为困难，做了一些细节调整，点在起始点附近便会自动将终点连到起始点位置。