计算机图形学

第一次实验报告

MiniDraw

PB20000264
韩昊羽

一、实验要求

- 画图小工具，要求有矩形，椭圆，直线，多边形，自由绘画等元素
- 每个图元使用一个类来封装
- 每个图元继承自相同的类 CFigure
- 使用 QT 编程
- 学习类的封装和多态与继承
- 了解鼠标交互操作
- 熟悉面向对象思想
- 拓展：线宽和颜色

二、操作环境

2.1 QT 图形化编程

IDE：Microsoft Visual Studio 2019 community

QT: 5.12.12

2.2 QPainter 和 QVector 库

三、功能介绍

3.1 绘制矩形

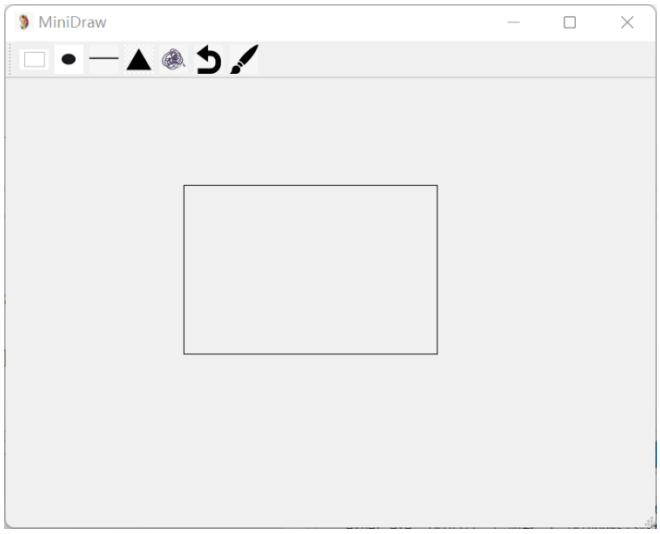


Figure.1 矩形

通过鼠标拖动确定左上角为初始点，右下角为结束点。

3.2 绘制椭圆

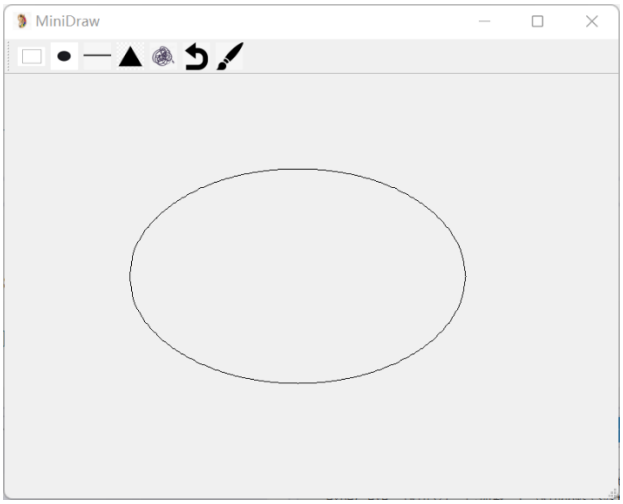


Figure.2 椭圆

通过鼠标拖动确定左上角为椭圆的外接矩形初始点，右下角为椭圆的外接矩形结束点。

3.3 绘制直线

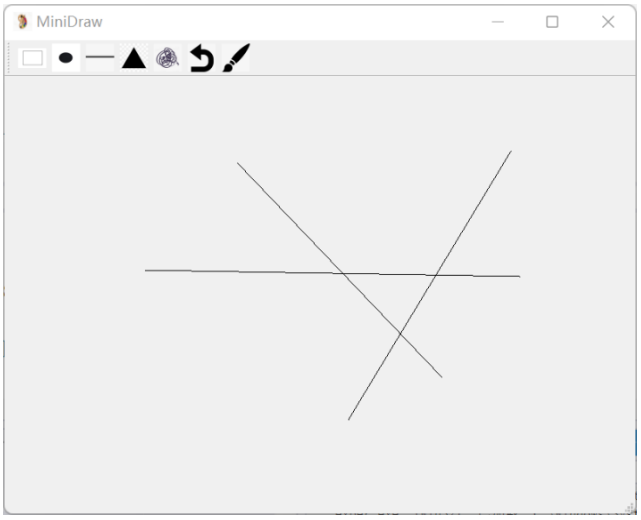


Figure.3 直线

直接确定直线的两个端点。

3.4 绘制多边形

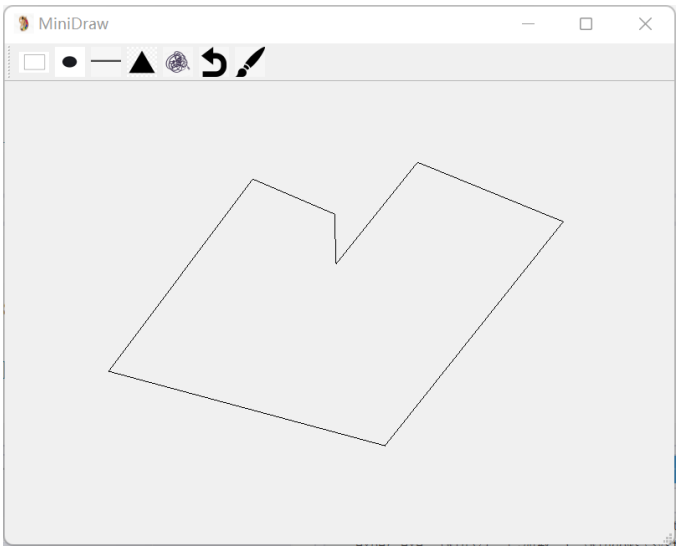


Figure.4 多边形

每一次点击都确定一个多边形的顶点（在确定两个顶点之间追踪鼠标位置），最后点击右键结束绘图。

3.5 绘制自由曲线

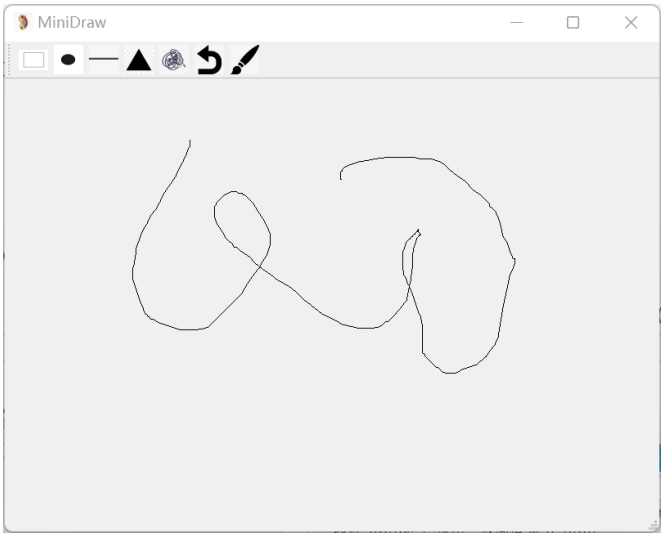


Figure.5 跟随笔迹自由绘图

3.6 撤销

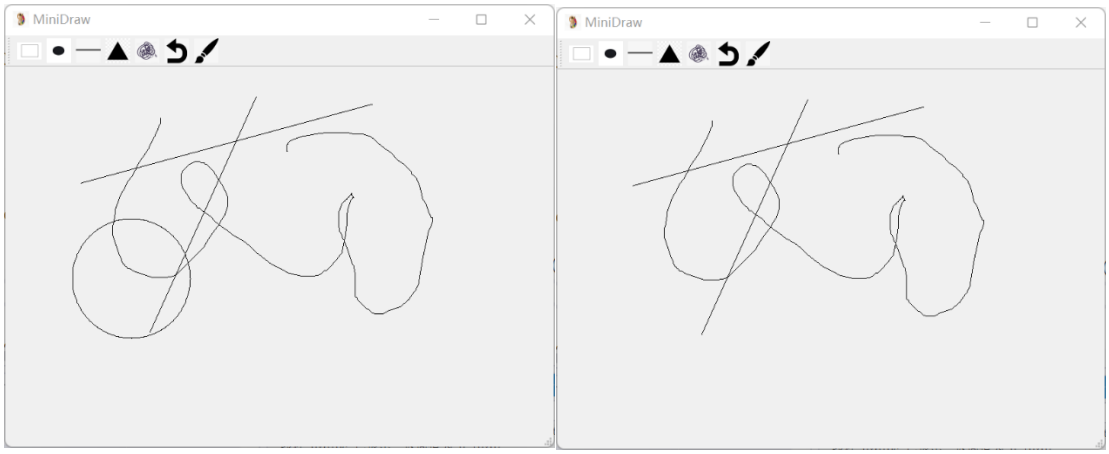


Figure.6 撤销

3.7 设置笔刷

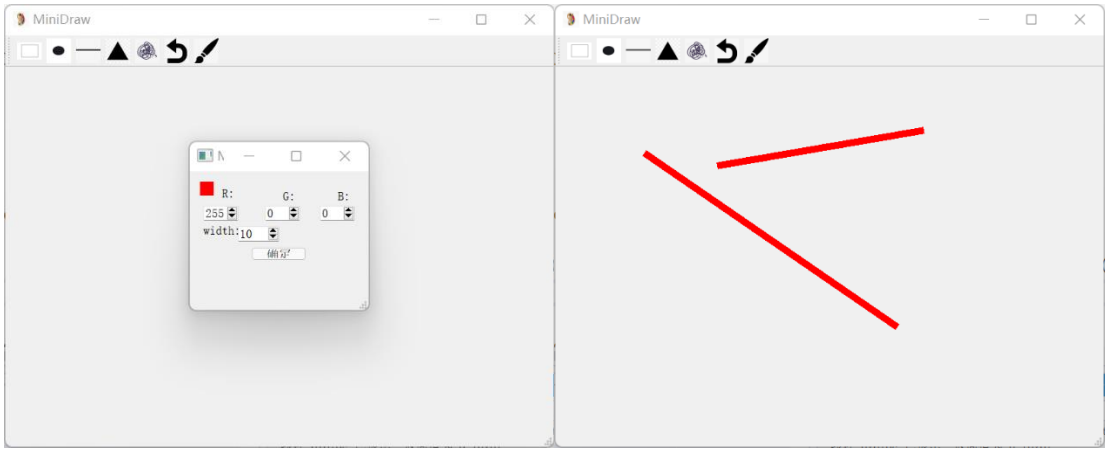


Figure.7 笔刷

可以通过弹出的窗口设置笔的粗细和颜色。

四、架构设计

4.1 文件结构

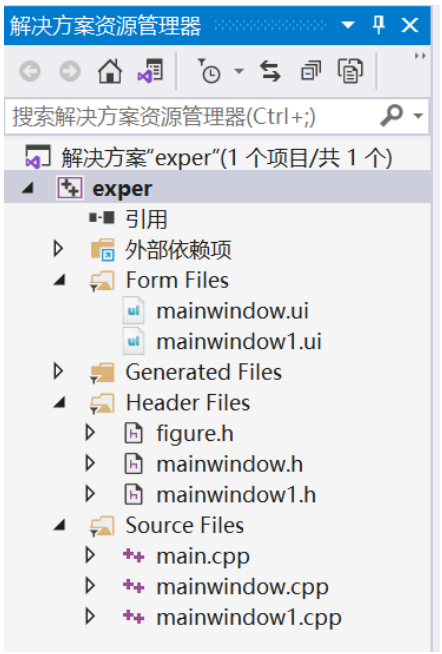


Figure.8 文件结构

分为三种文件(.h ,.ui 和.cpp)，第一个文件 figure.h 主要储存关于图形的操作，包括 Cfigure 和派生类的定义，第二个文件

mainwindow.h 主要储存主窗口以及鼠标事件（函数等在 cpp 文件里实现），第三个文件 mainwindow1.h 主要储存弹出的笔刷窗口。

4.2 类图

CFigure 类：

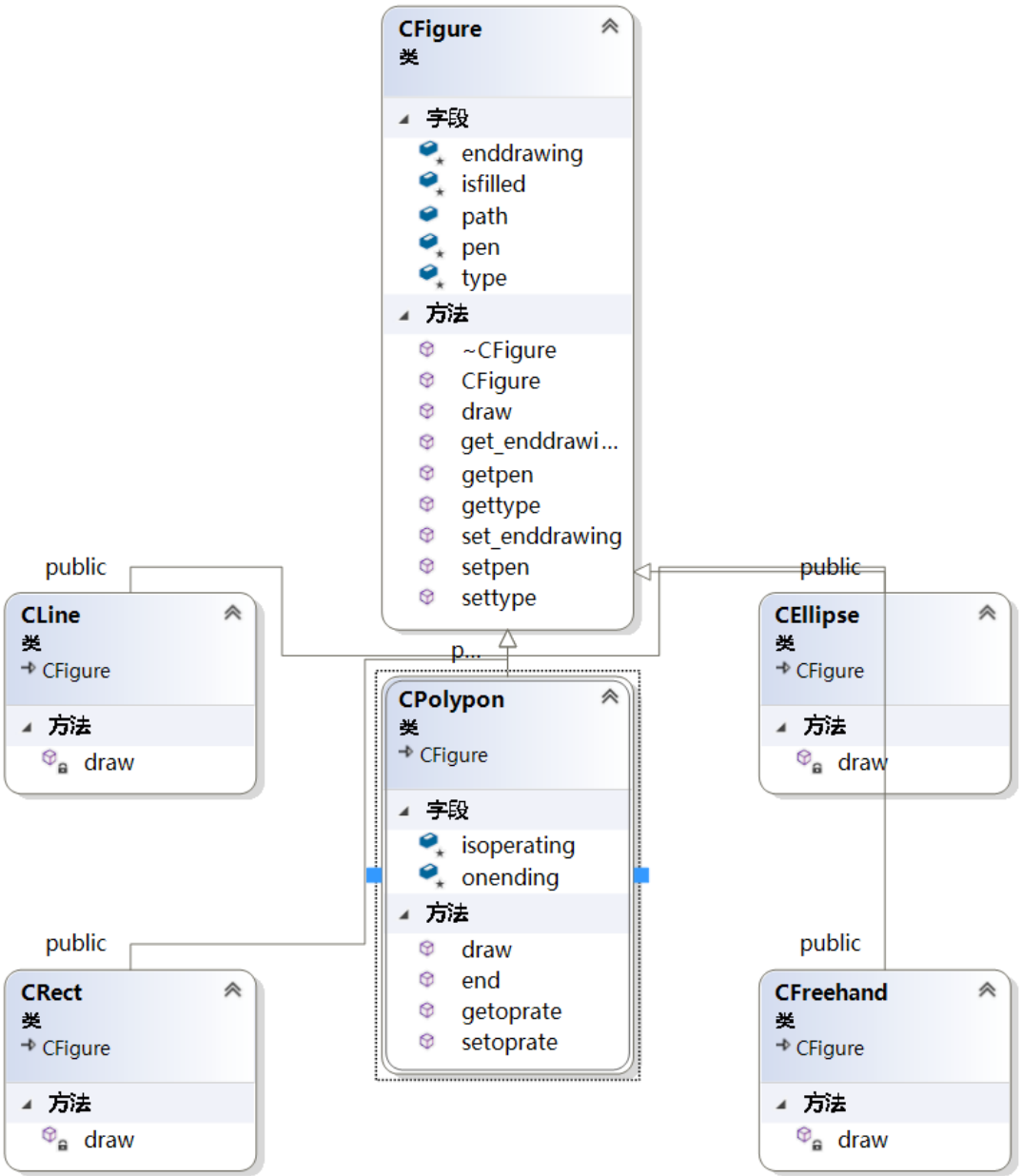


Figure.9 Figure 类图

在基类中主要定义了 enddrawing(是否结束绘画)，isfilled(是否被填充)，path（点的路径），pen(每个图形不同的笔刷)，type（图形的种类）字段，其中只有 path 是公共的，便于绘图的时候访问。Draw 是虚函数，在子类中具体实现，其余都是关于设置和读取的接口函数。其中多边形类因为涉及到什么时候停止绘画，所以多了 isoperate 和 onending 两个变量。

Mainwindow 类：

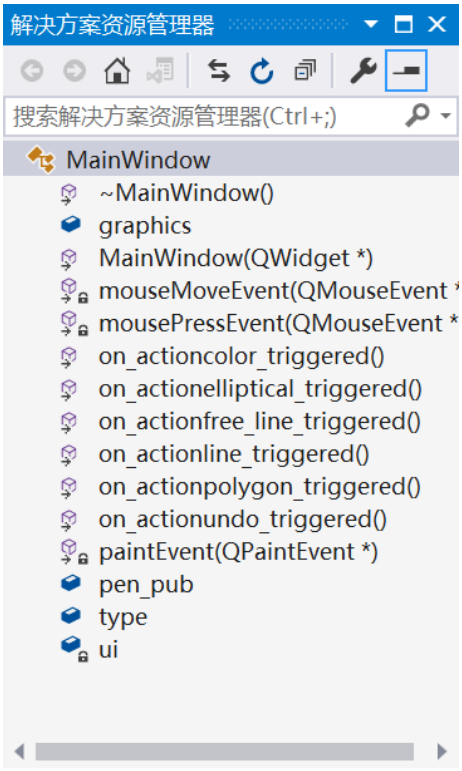


Figure.10 mainwindow 类图

其中关于 mouse 的两个函数是处理鼠标点击事件的函数，后面还有 6 个 trigger 函数是处理 ui 界面点击按钮用的。剩余是和绘图有关的，graphics 主要处理存储的图形，类型是*CFigure 类型（便于访问不同类的不同对象），penpub 和 type 是当前笔刷和图形的类

型。

Mainwindow1 类：

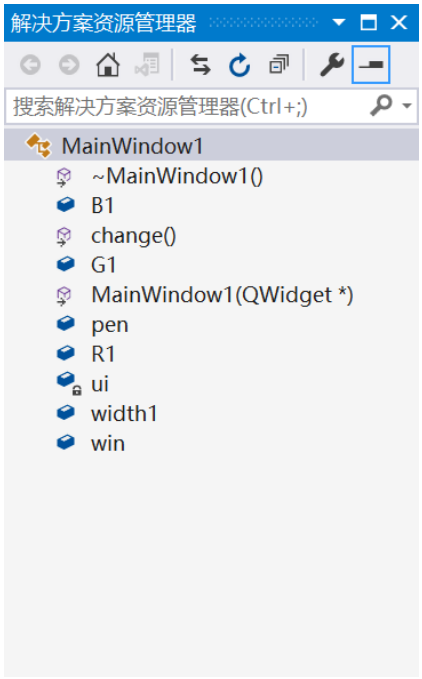


Figure.11 mainwindow1 类图

R1,G1.B1，width1 存储的是当前主界面的笔刷相关值，pen 是调整后的笔刷，win 储存了主界面，便于将参数回传。

五、功能实现

5.1 画图功能

首先主窗体检测鼠标事件，当检测到鼠标按下时，根据当前 mainwindow 内存储的 type 决定创建什么图形，然后跟踪鼠标位置实时更改图像的参数，其中多边形要检测每次是新开一个多边形还是上次多边形的不同顶点。图像的路径数据都存储在 cfigure 类的 path 字段中，主窗体 mainwindow 调用 paintevent 函数将图形绘制到窗体

上并实时更新。

算法部分，除了椭圆外都使用 QT 自带的 QPaint 中 paintline 函数画直线，（freehand 存储的点更加密集），至于椭圆要根据椭圆方程反解出 x,y 的位置，因为计算时间较长使用了在椭圆上采样的方法，同时还要处理椭圆两侧的边界情况。

5.2 画笔功能

通过弹出的窗口，用户可以自定义设置笔刷的粗细和颜色，再传回主窗体，主窗体的 pen_pub 发生变化，在新建图形时就传入新的 pen_pub。

5.3 撤销功能

Graphic 直接弹出，但要注意为空和删除申请的空间。

六、难点难题

6.1 架构问题

由于对 C++和 QT 编程的不熟练，我一开始并没有把 draw 函数放在基类中作为虚函数，而是直接放在 mainwindow 中和鼠标事件一起处理，这样不仅看起来杂乱无章，并且还需要实时访问 cfigure 中各个变量，既耗空间又耗时间，而且有些子类特有的变量还需要强制转化后访问。后来我认识到 draw 本来就应该放在 CFigure 类中，这样不仅可以快速访问内部的变量，还可以将内部的操作隐藏起来，只留

下接口，不会对内部的变量造成干扰，代码也看起来更加整洁。

6.2 Graphics 存储问题

我一开始设想的是在 Graphics（vector）中存储 CFigure 类型，但发现这样没法访问子类中特有的字段，后来通过查阅资料了解到可以存储*CFigure 类型，需要访问特殊字段的时候再强制转化为对应的指针类型，而且需要的存储空间也更小。

6.3 多边形判断绘制中问题

和其他图形不同，多边形在绘制过程中可能松开鼠标，导致如果每点一下添加一个图形的话会创建多个图形，这时就需要判断是否在多边形的绘制过程中。

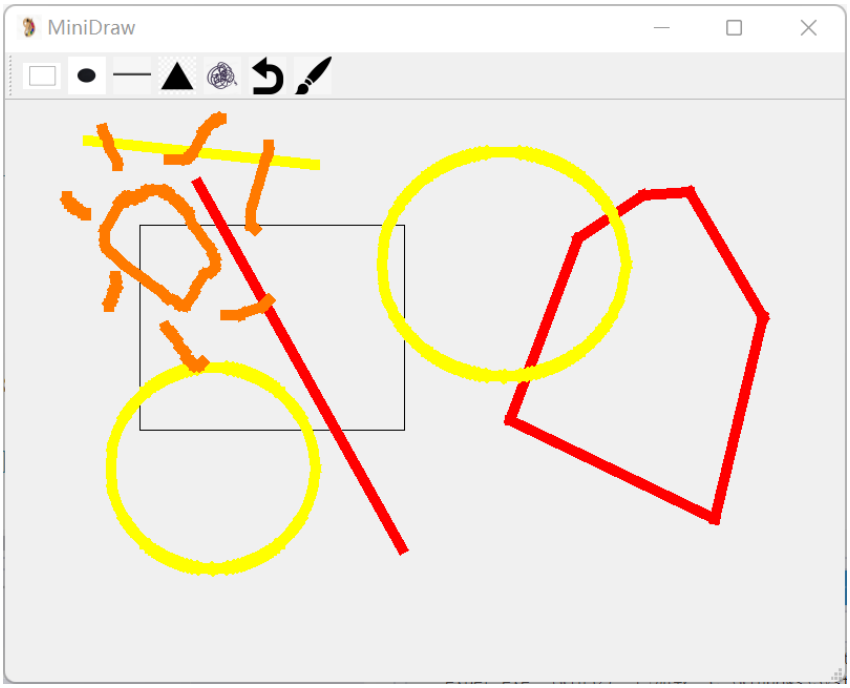
6.4 笔刷数据回传问题

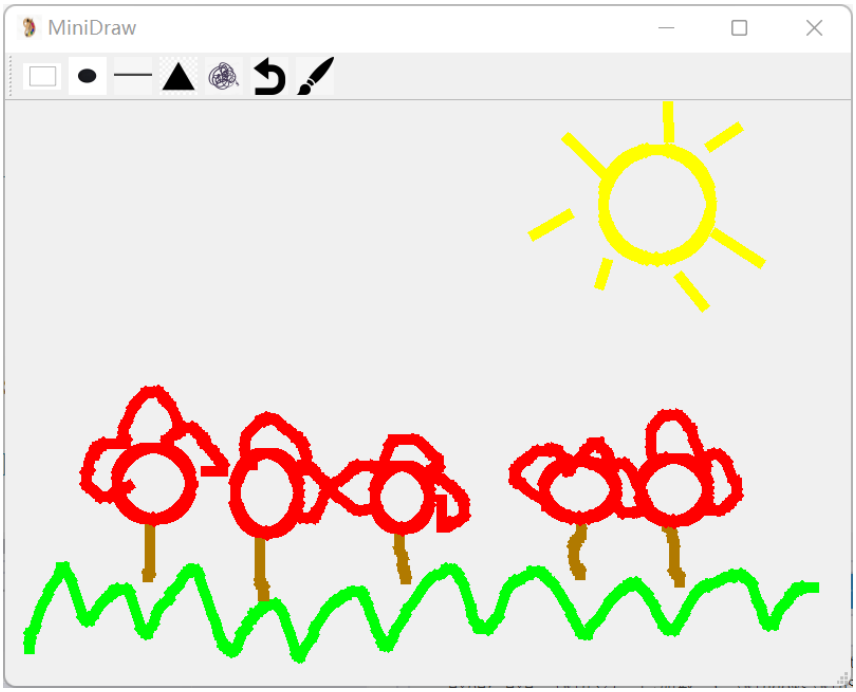
我没有找到两个窗体之间很好的访问函数，于是在创建另一个窗体时传入了主窗体的指针方便回传。

6.5 椭圆绘制问题

椭圆绘制涉及到冗长的数学计算，而且时间相对来说消耗的较长，于是我将椭圆分为均匀等分，通过采样减少数据量加快速度。

七、实验结果





八、问题与展望

8.1 遇到的问题

程序还有很多地方还可以改进，比如不用每一帧都 update，减少

CPU 的工作量，重写事件函数，UI 的观赏性等等，随着学习，我会掌握更多，将程序改进的更好！

8.2 future work

- 图像的填充
- 曲线的绘制
- 效率提高
- 通过拖动点改变曲线
- 更多图形
- 调色盘控件
- 文件的读取和保存