Homework 1

内容来自于本人使用经验和网络，如有错误或者补充欢迎大家直接修改。

.md里的图片需要将库加载到本地才能正常打开。

在文本最后有关于面向对象的补充。

Qt

Qt基础知识

自动内存回收

Qt拥有自己的内存回收机制，可以自动回收内存，实例化的对象不需要手动delete()。

Qt一般会形成对象树，所创建的对象一般在堆区，当析构父类对象时子类对象也会析构，回收时一并回收。

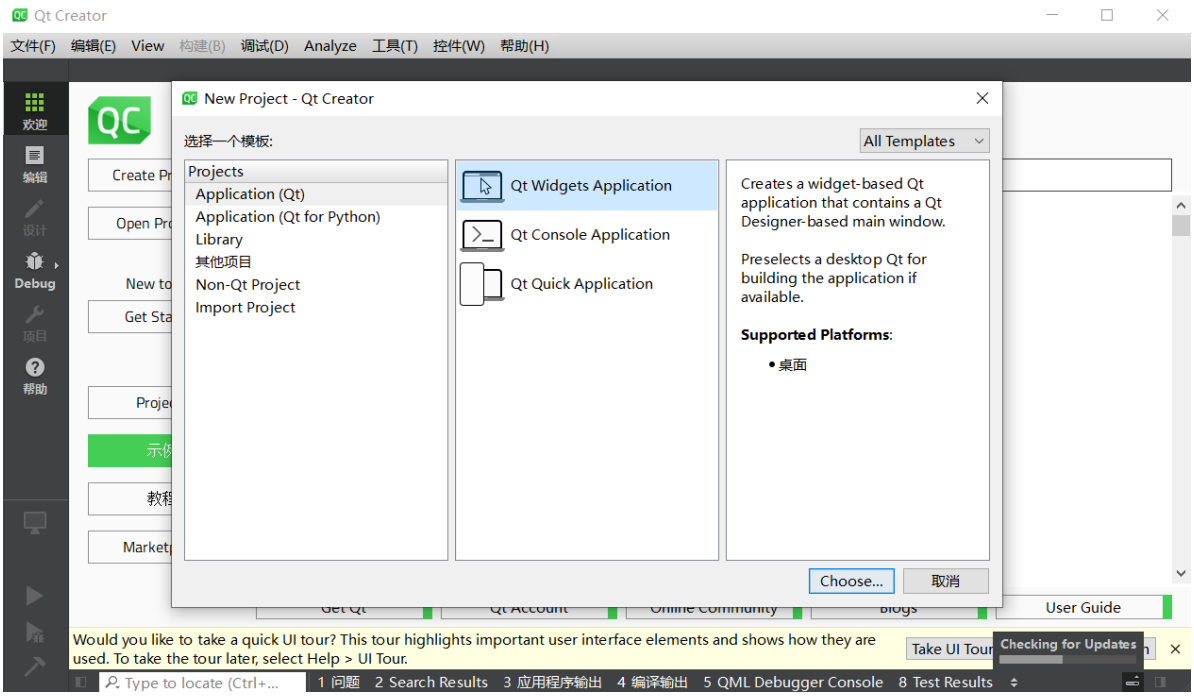
坐标系

- 左上角是坐标(0,0)；
- x以右为正方向；
- y以下为正方向。

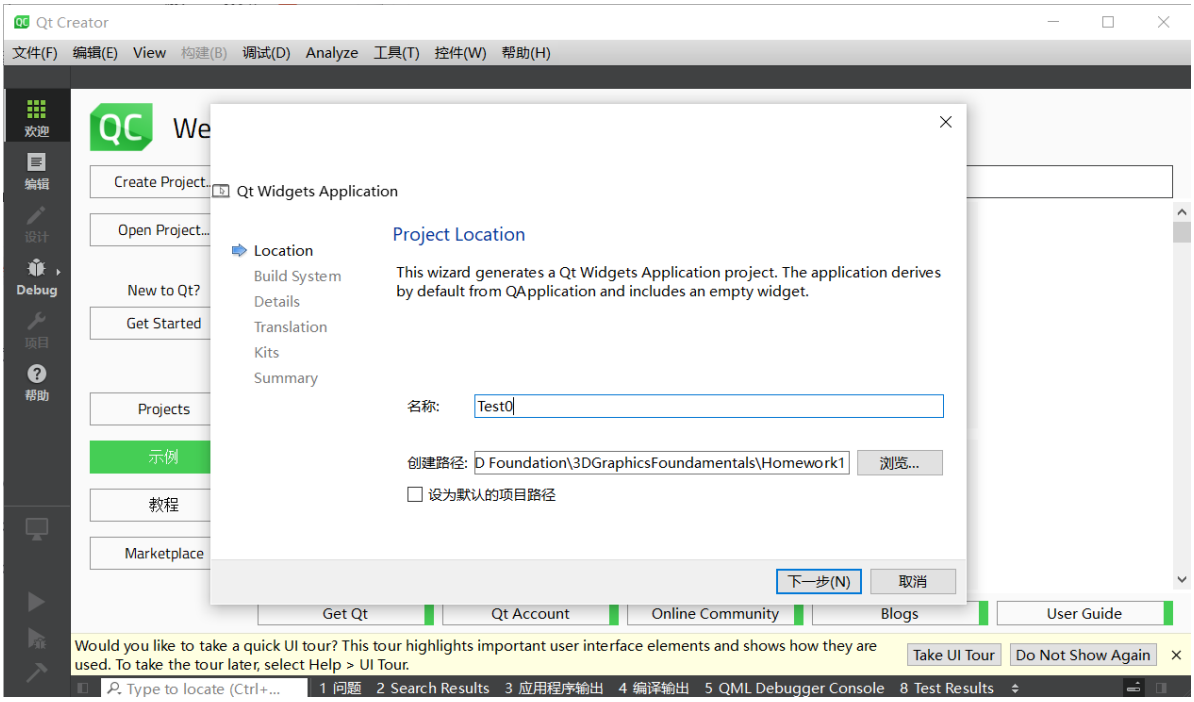
创建项目

只是展示Qt项目是怎样生成的，本次作业不会用到。

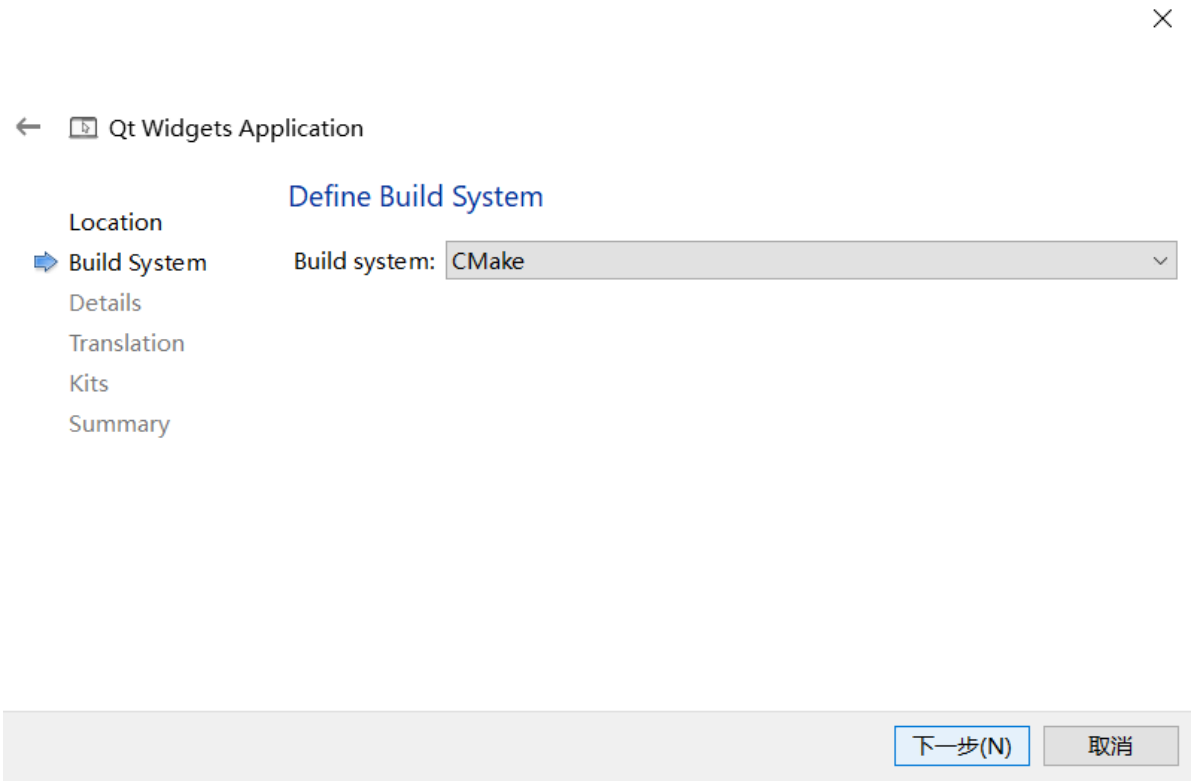
点击Create Project...



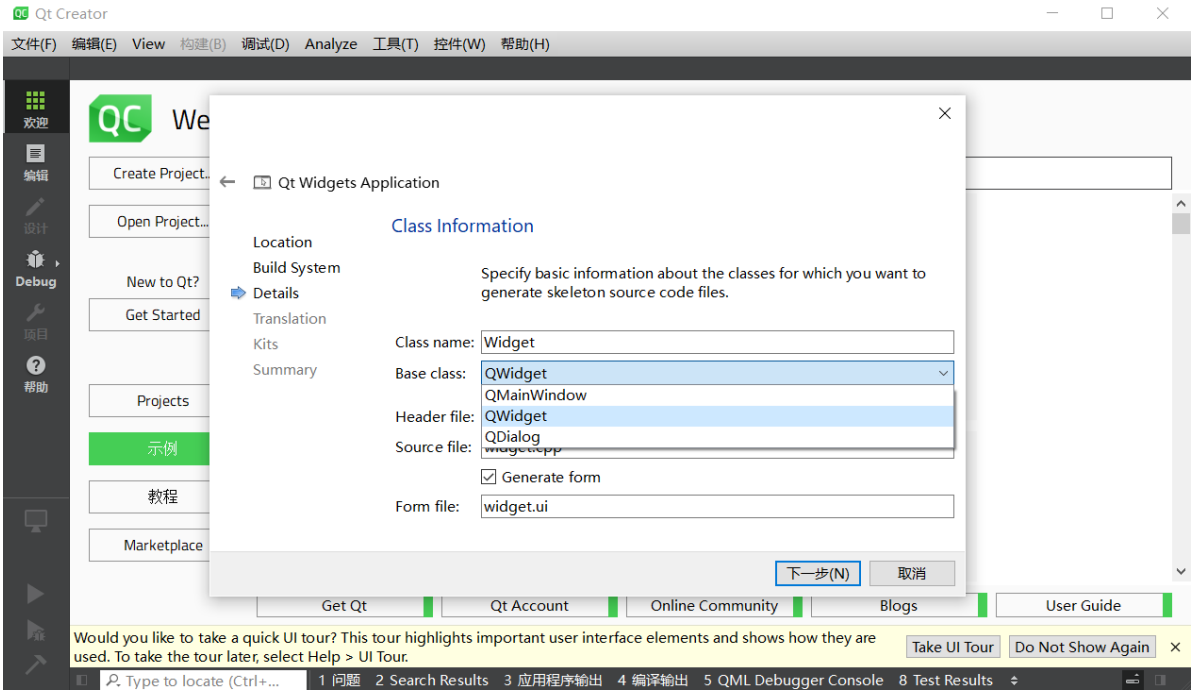
选择Widgets。



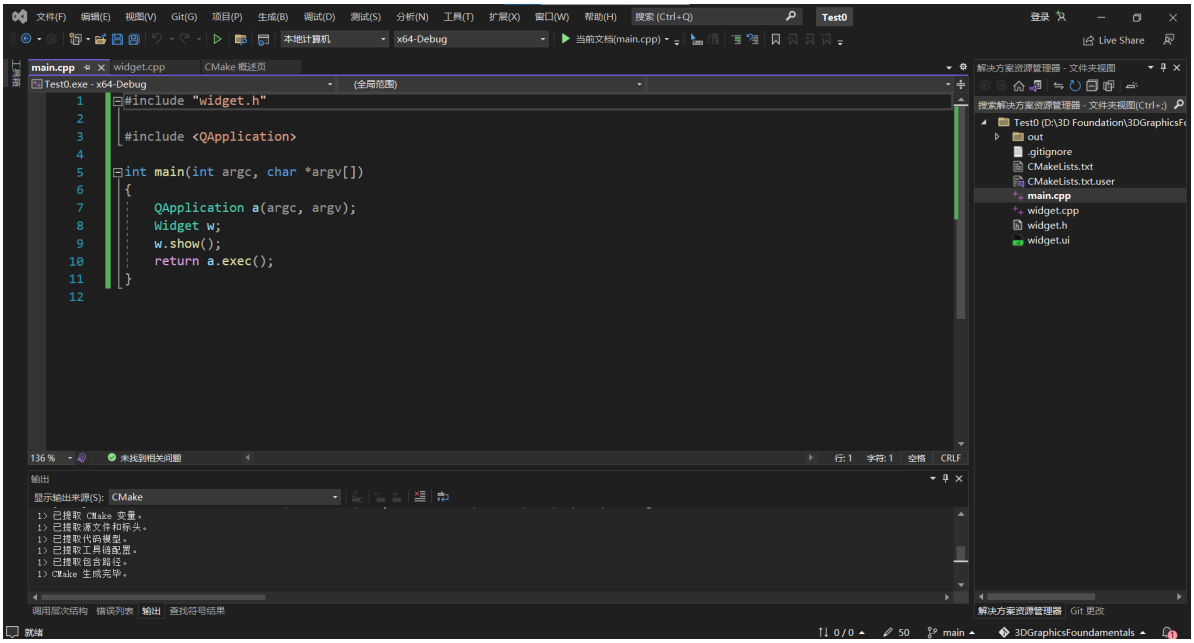
继续。



接下来有三个选项，默认创建的窗口类是`QMainWindow`，可以选择的基类有：`QWidget`、`QMainWindow`、`QDialog`，一个是单一的窗口，一个是主界面，一个是对话框，`MainWindow`比`Widget`多了一些常用的控件，例如菜单栏。



接下来一直下一步即可，最终生成：



1. `main.cpp`：如下

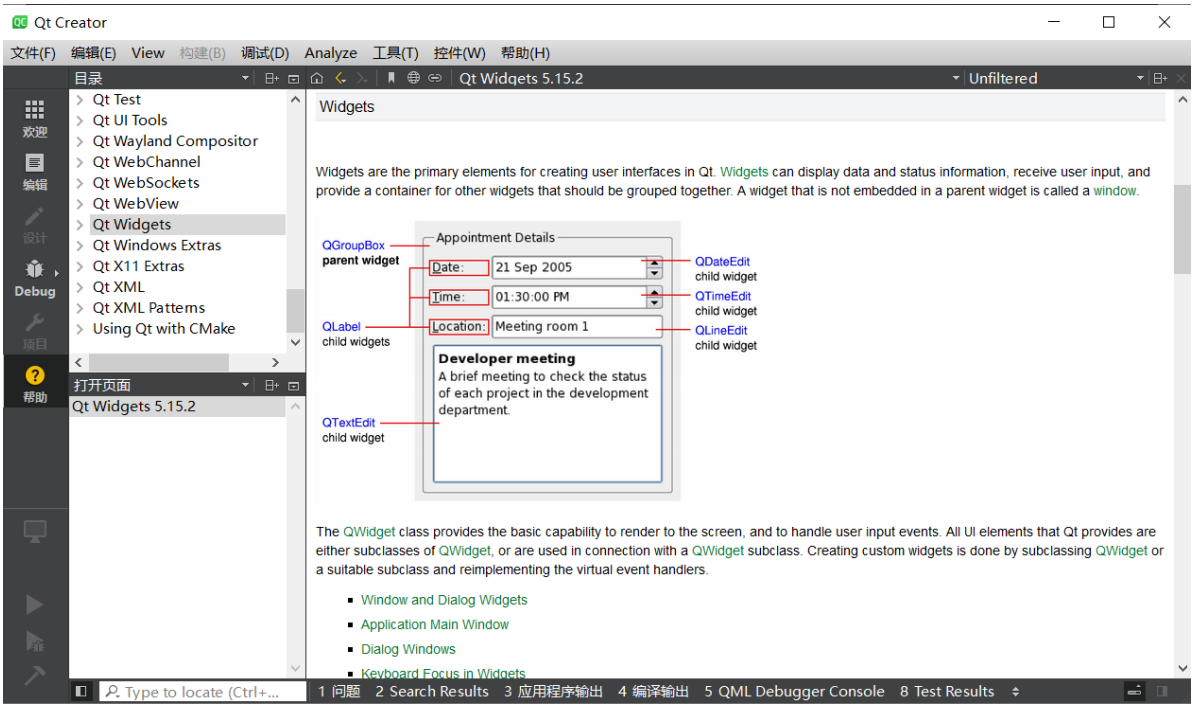
```
QApplication a(argc, argv);        //应用程序对象，有且仅有一个
MainWindow w;                      //实例化窗口对象
w.show();                          //调用show函数 显示窗口

return a.exec();                   //让应用程序对象进入消息循环机制中，代码阻塞到当前行
```

2. `mainwindow.cpp`：这个就是用来书写逻辑代码的主要地方，和`mainwindow.h`是一起的
3. `mainwindow.ui`：ui文件，可视化界面，用于做界面布局。

帮助文档

值得一提的是，Qt自带了help文档，可以查阅自己需要的功能（虽然是英文）。



顺便一提`QtCreator`本身也是可以用来文本编辑的，只不过不太好用。

信号和槽

信号和槽是Qt的一个特色，当一个类运行的时候，调用信号，就会触发相应的槽函数，同时信号函数还可以传递参数给槽函数。

用法

连接函数： connect(参数1, 参数2, 参数3, 参数4)

- 参数1 信号的发送者
- 参数2 发送的信号（函数地址）
- 参数3 信号的接受者
- 参数4 处理的槽函数 （函数的地址）

大家可以点击所给的Project文件夹里的`viewwidget.h`看看哪些是信号哪些是槽。

自定义信号与槽函数

有的时候，只用类自带的信号与槽比较局限，我们还可以用自己定义的信号和槽函数。

自定义信号

自定义的信号要写到头文件下的 signals下，有时候项目没自动生成，自己写上，返回值为void，只需要声明，不需要实现。
可以有参数，同时也可以重载已有的信号。

自定义槽函数

自定义的槽函数，返回值为void，需要声明，也需要实现，可以有参数，可以重载，要写在头文件下的`public slot:`下 或者`public:`或者全局函数。

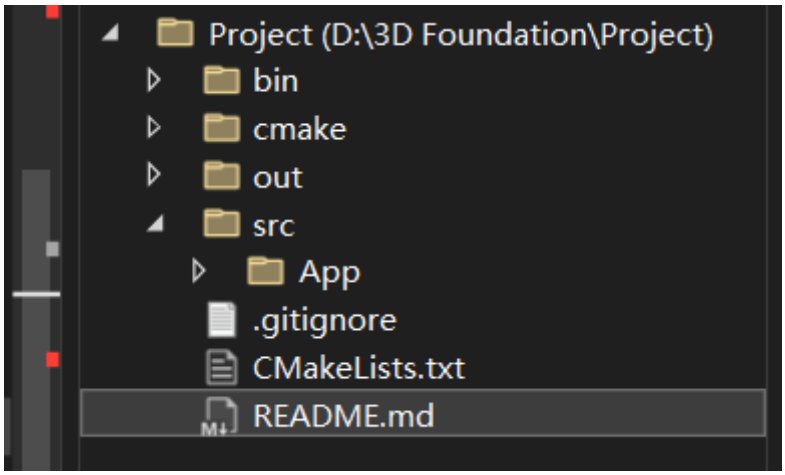
第一次作业

使用示例

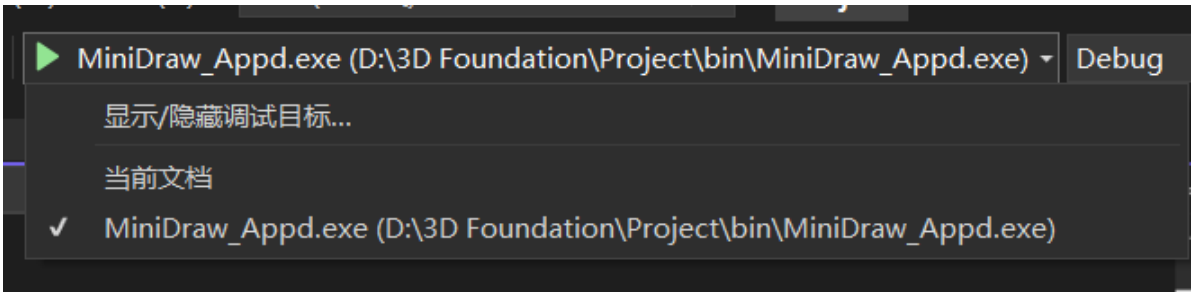
接下来的内容需要结合Project文件进行，举例说明如何用给定的Project文件实现椭圆的绘制。

试运行

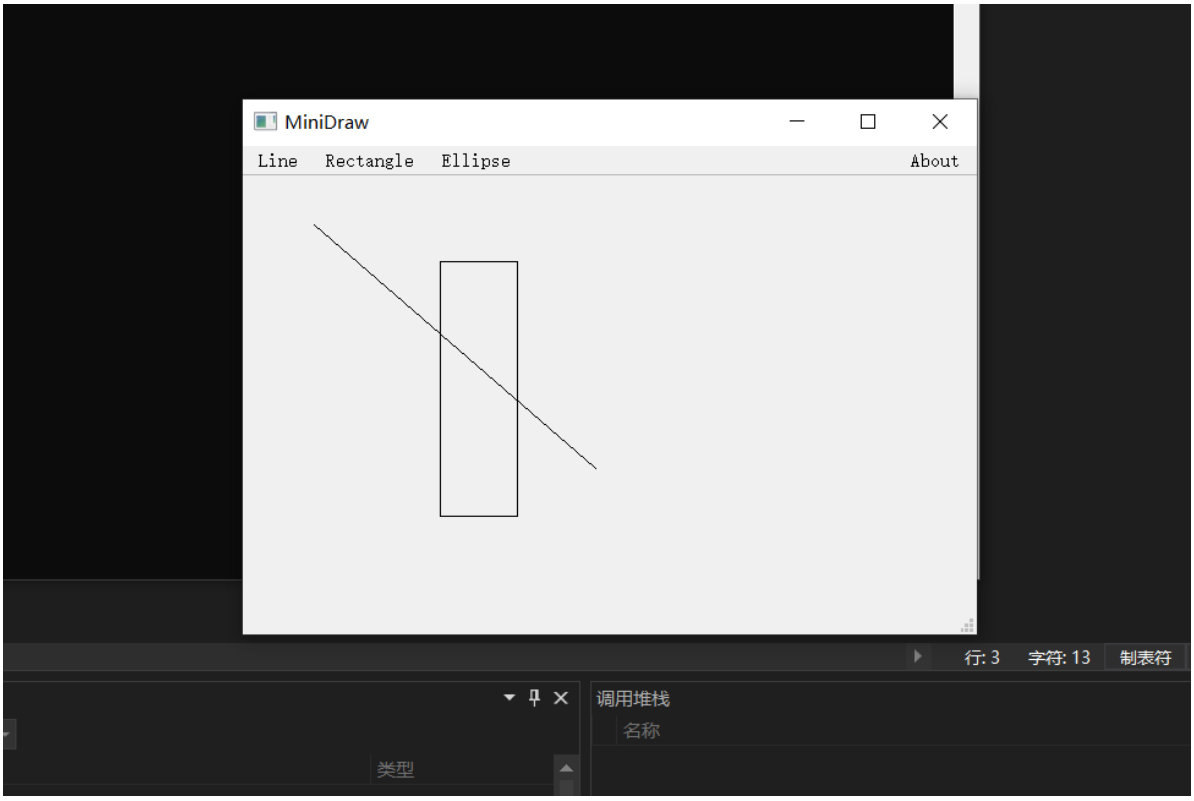
1. 使用已下载的VS打开给定的Project文件夹



2. 右键`CMakeLists`，点击**清除缓存并重新配置**、**生成**和**安装**。
3. 选择启动项为`MiniDraw_Appd.exe`



4. 点击绿色三角形，可以看到运行成功，**但没有`Ellipse`这个按钮**



创建椭圆

1. 右键 App 创建 Elli.h 和 Elli.cpp 两个文件（跳出是否脚本修改点击取消），依次写入：

```
/*这是Elli.h*/

#pragma once                                    //保证同一个文件不被包含多次
#include "Shape.h"                              //引入基类Shape的头文件

class Elli : public Shape {                     //派生类Elli由基类Shape派生
public:
    Elli();                                     //类Elli的构造方法
    ~Elli();                                    //类Elli的析构方法

    void Draw(QPainter& painter);               //类Elli的方法（即属于Elli的函数）
};
```

```
/*这是Elli.cpp*/

#include "Elli.h"

Elli::Elli() { }                                //实现构造方法
Elli::~Elli() { }                               //实现析构方法

void Elli::Draw(QPainter& painter) {            //实现头文件里声明的方法
    painter.drawEllipse(start.x(), start.y(),
        end.x() - start.x(), end.y() - start.y());
}
```

这样我们成功声明并实现了Elli类。

2. 通过上面对Qt的初步了解，我们知道此时应该打开 viewwidget.h 进行槽函数的声明并在 viewwidget.cpp 中实现：

```
#include "Elli.h"                          //记得引入刚刚写好的头文件

public slots:
    void setLine();
    void setRect();
    void setElli();                        //添加槽函数
```

```
void ViewWidget::setElli() {               //实现setElli()方法
    type_ = Shape::kElli;
}

case Shape::kElli:                         //在鼠标左键点击的方法中加入
    shape_ = new Elli();
    break;
```

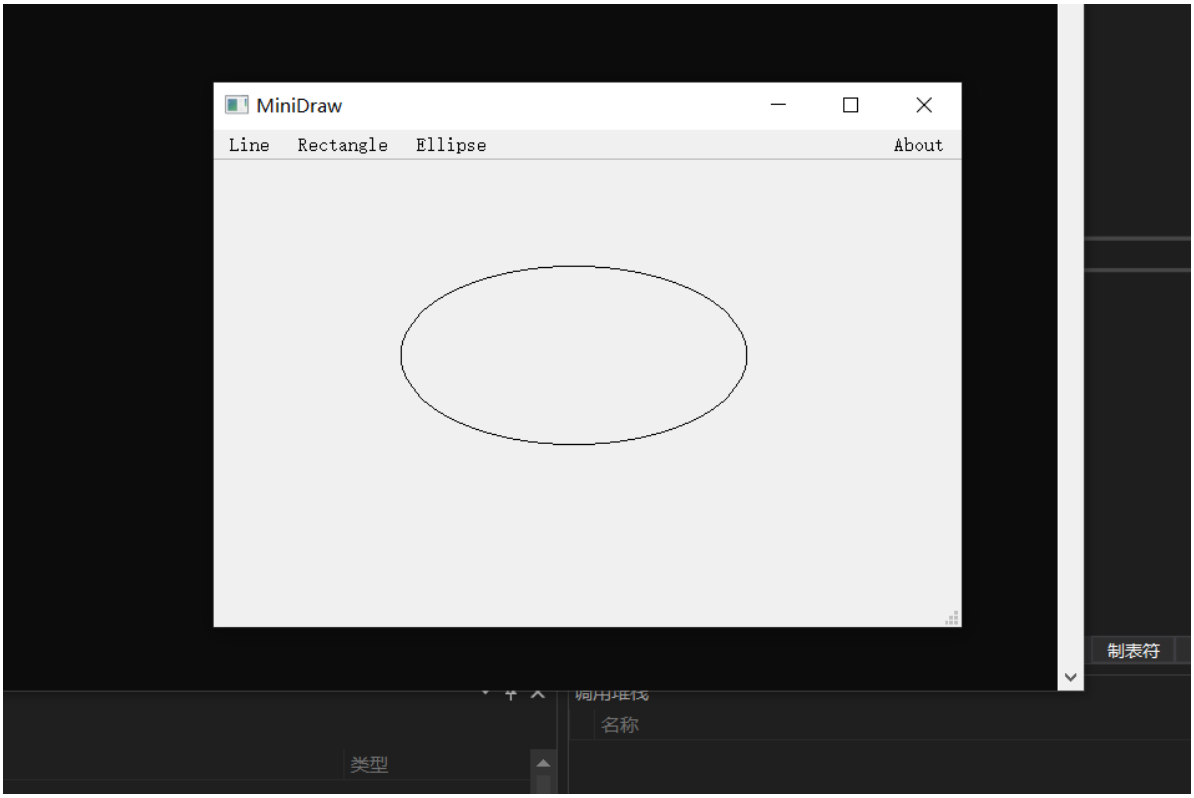
3. 最后，为了让画椭圆这个事件和椭圆这个按钮在窗口显示，来到`minidraw.h`和`minidraw.cpp`。

```
QAction* Action_Elli;                      //声明指针
```

```
//这里语法比较复杂
void MiniDraw::Creat_Action() {
    Action_Elli = new QAction(tr("&Ellipse"), this);
    //tr()是将UTF-8字符转到可以在窗口上显示的字符
    //this是调用了自身，这里可以调用一个QObject类，此处调用自身(MiniDraw类)是用了一次上转
    //这行将刚刚声明的Action_Elli实例化为一个新的QAction类
    connect(Action_Elli, &QAction::triggered, view_widget_,
&ViewWidget::setElli);
    //connect()用于将槽连接
    //这句话翻译一下就是当Action_Elli触发时，在窗口上可视化一个Elli类
}

void MiniDraw::Creat_ToolBar() {
    pToolBar->addAction(Action_Elli);
    //调用QToolBar内addAction库函数
}
```

4. 点击清除缓存并重新配置，再运行。这样就可以进行椭圆操作了。



清除画布

提供清除画布的代码，具体含义自己理解。

```
void ResetAll();            //添加在minidraw.h

Action_Reset = new QAction(tr("&Reset"), this);
connect(Action_Reset, &QAction::triggered, this, &MiniDraw::ResetAll);
                            //添加在minidraw.cpp

void MiniDraw::ResetAll() {
    view_widget_->clearAll();
}                           //添加在minidraw.cpp

void clearAll();            //添加在viewwidget.h

void ViewWidget::clearAll() {
    shape_list_.clear();
}                           //添加在viewwidget.cpp
```

到这里，作业中：

1. 在现有程序基础上编写简易画图程序，实现在Qt窗口上完成绘制直线、矩形、椭圆、多边形和清空画布的功能；
2. 每种图形需用类封装，需要从一个图形类继承。

这两条应该已经可以尝试完成了。

简单的面向对象

我们学过的C语言是面向过程编程，对于一个变量（整形、字符串或是结构体），它的函数和变量都是分开的。

面向对象是更符合人的思维的一种编程思想，我们称一个类为一个对象。比如人，它是一个对象，它有年龄，名字等“属性”，也有吃饭等”方法“（即属于对象的函数）。

```
class People {
    //属性
    int age;
    String name;
    //方法
    void eat() {

    }
}
```

而且大多数情况下，对象有着**A is B**的继承关系，比如`Chinese is People`，所以对象间可以有继承关系。比如Chinese可以继承People的年龄性别等属性，也有比如说省份等自己的属性；继承吃饭的方法，也可以有使用筷子等自己的方法。

```
class Chinese extends People {
    String province;
    void useChopsticks() {

    }
}
```

这就是面向对象的思维方式。