C++实践中期报告

一、项目进展

本项目主要是实现一个扫雷游戏，并在此基础上提供如排行榜，作弊、自定义雷区大小等新型功能。

目前第一轮迭代已经实现的功能是绘制一个固定大小的雷区，并可以左键点击扫雷，如果是雷则游戏结束判断失败；如果是安全方块，则递归到其他附近有雷的方块重复要求判断。可以右键方块进行标记，按照普通方块——旗子——问号——普通方块的顺序进行标记。当所有地雷都被标记为旗子状态时（且没有安全方块被标记），弹窗显示游戏胜利，游戏上方的表情也将从微笑变成耍酷。

总的来说，第一轮迭代难度较大，所耗时间较长，代码部分也较多。在完成第一轮迭代后，我们小组成员立刻开始了第二轮的迭代。在本轮迭代中，我们计划实现一些附加功能——作弊模式、自定义雷区、统计时间等，从而提高游戏的可玩性，让它变得风靡起来。

在上交本文档的时候，我们第二轮迭代已经快要完成。部分预期功能——如作弊模式、自定义雷区已经实现。效果图将放在文章后部以供查看。少部分功能如统计时间还在调试中，但预期再过一天就可以完成第二轮迭代的全部工作，进入第三轮迭代。

二、技术难点及克服

于明辰：

我在本轮中主要负责的工作是在第一轮迭代中写 Comoon 层与 App 层，主要技术难点在于用 Qt Creator 进行 UI 界面的编写、左键点击判断是否为雷——是的话好办，不是的话如何递归寻找到附近的普通方块、右键点击统计旗子与地雷是否一一对应。其他还有一些老师课堂上讲的较为安全的编译方法，如用智能指针等。针对以上三个难点，我依次使用了以下方法来进行克服——

首先是 UI 界面。我在 OOP（面向对象的程序设计）课堂上和队友一起写了一个五子棋的程序。当时没有负责 UI 界面的编写。这次编写扫雷界面，我将那个界面复制了过来，然后按照我们这个程序的逻辑慢慢删改，有不懂的地方就问当初的队友，最后成功改完了。

其次是左键判断是否为雷。这里我采用了通过改变 pos i、j 的方式来找到正常方块，最后就能找到附近没有被做右键点击过的方块。

最后是右键统计旗子数与地雷是否一一对应。如果是的话就导入对应图片显示成功。不是的话就不操作。判断是否一一对应我这里选择统计总雷数，然后与旗子数比较，此外如果有雷子还没被标为旗子（被标为旗子后某个中间变量会变化），就判断不成功，这样可以达成目标。

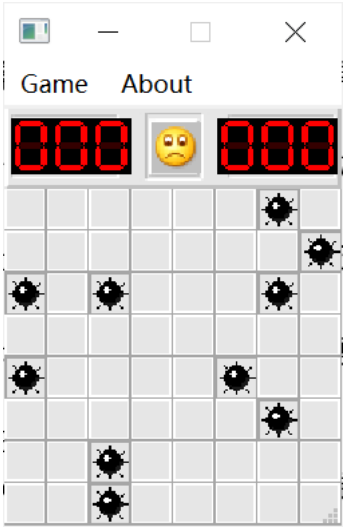
由于我完成的部分是整个程序的基础（雷区、左右键等），所以我的部分的截图被包含在了

其他同学的截图之中，供各位查看。

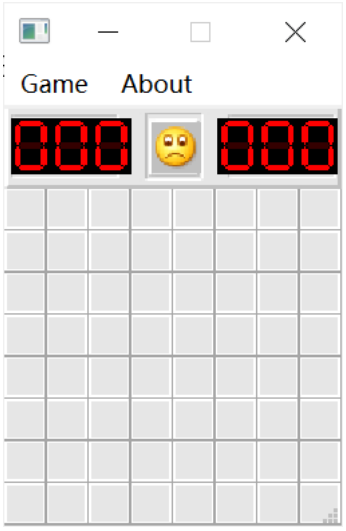
王紫晴：

主要负责自定义雷区，作弊模式以及排行榜功能。在向框架中添加功能时，主要考虑以下几个部分的修改，此处以添加的第一个功能：显示雷区（作弊功能）为例。cheat 功能的使用机制如下：作弊功能可以开启或关闭，单击 cheat 时在这两种状态中进行切换。为了完成这个功能，设置一个变量 is_cheat。运行程序时，初始状态为 cheat 关闭，即变量 is_cheat=0，此时界面上不会出现炸弹的位置。is_cheat=0 时，鼠标左键单击一次 cheat 键后，is_cheat 被置为 1，此时界面上显示全部雷区的位置；反之，is_cheat=1 时左键单击，则 is_cheat 置 0，雷区不再显示。在界面设计上，目前我把作弊功能 cheat 放在 Game 按钮下，作为它的子按钮。

以一局新游戏为例，直接打开 cheat 功能，显示如下：

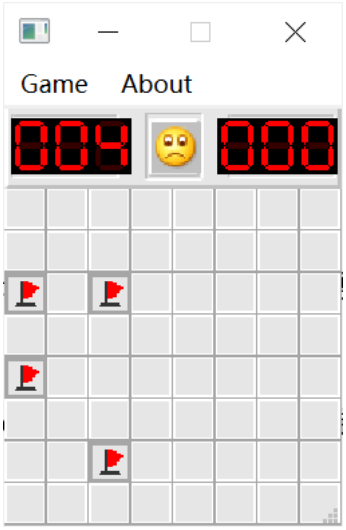


关闭后变为正常显示：

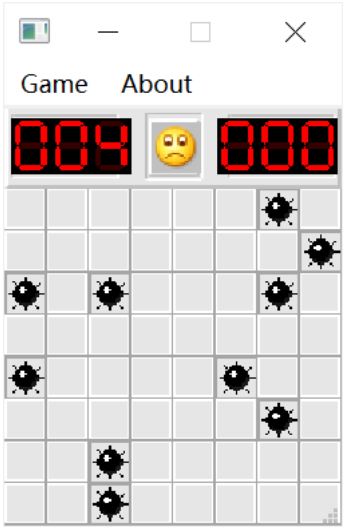


目前，is_cheat=1 时雷区是优先级最高的显示，只有关闭作弊模式时，右键点击才会出现旗

帜或问号。即，关闭 cheat 时设置四面旗子，显示如下：



打开后仍是全显示为雷区：



在整个功能的添加上，主要修改了程序的以下部分：

（1）common->block 层

为 block 增加判断作弊模式的属性 is_cheat，设计相关函数。

```
bool is_cheat;
```

```
bool get_cheat();//7.17 wzq
void set_cheat(bool x);//7.17 wzq
```

```
bool Block::get_cheat(){//7.17 wzq
    return is_cheat;
}

void Block::set_cheat(bool x){
    this->is_cheat = x;
}
```

（2）view->mainwindow 层

添加函数 set_cheat（），实现左键单击一次就改变一次 is_cheat 值的功能。

```
void MainWindow::set_cheat(){//change the cheating model
    if(B->get_cheat()==1){
        B->set_cheat(0);
    }else{
        B->set_cheat(1);
    }
    this->update();
}
```

修改函数 paintEvent()，在 play 中添加一个判断条件 if (B->get_cheat()==0)，即判断作弊模式是否打开，并通过改变界面的绘制来实现功能。

此外，由于它是通过一个界面上的按钮来实现的，在 mainwindow 头文件中，我们需要将 set_cheat()设计成 private slot:

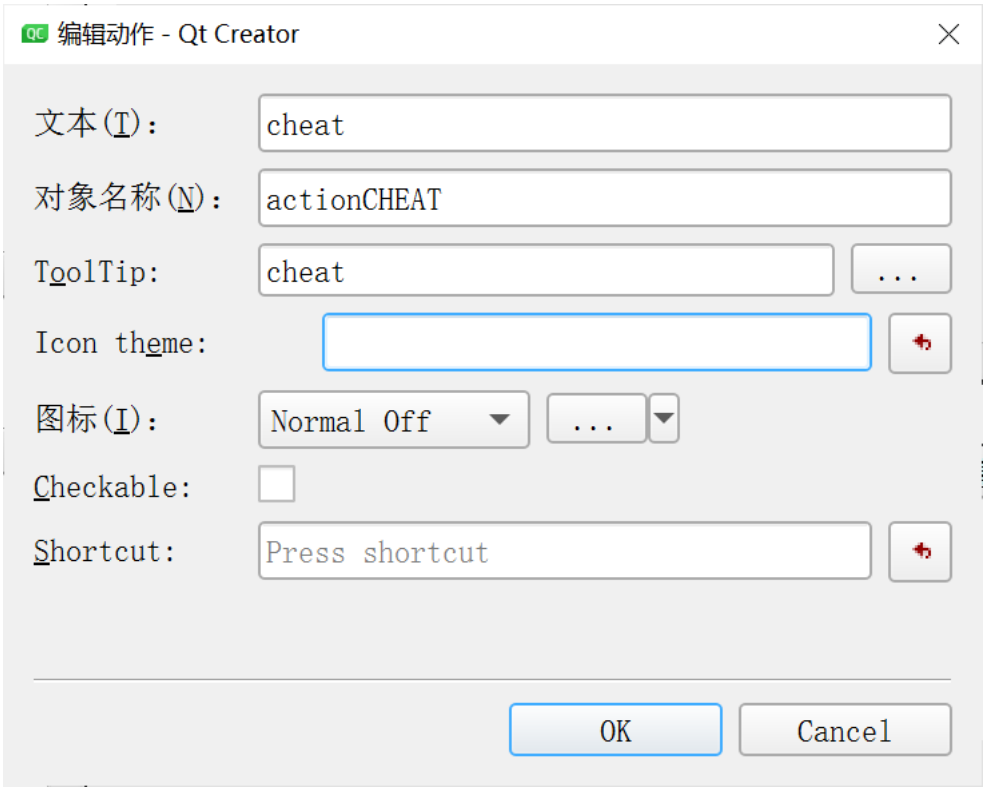
```
private slots:
    void set_custom();
    void set_cheat();
```

并且，我们需要在.cpp 文件中添加以下一行内容，从而完成鼠标操作与程序判断的连接：

```
connect(ui->actioncheat,SIGNAL(triggered(bool)),this,SLOT(set_cheat()));
```

(3)mainwindow.ui

绘制这个按钮，设置大致如下：



在添加这些部分之后，就可以在运行界面上正确显示并实现该功能了。

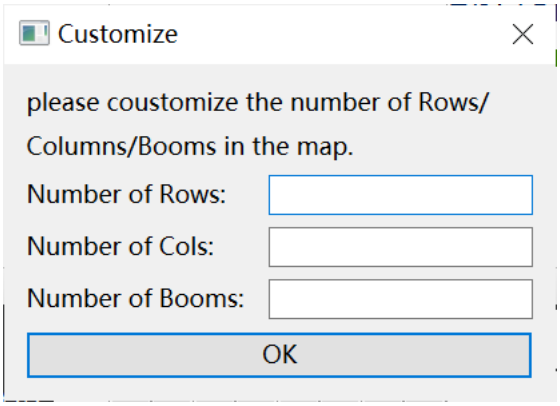
三、协作情况

于明辰：管理 Git，写 Comoon 层与 App 层。
王紫晴：写 View 层。
薛远　：写 ViewModel 层与 Model 层。

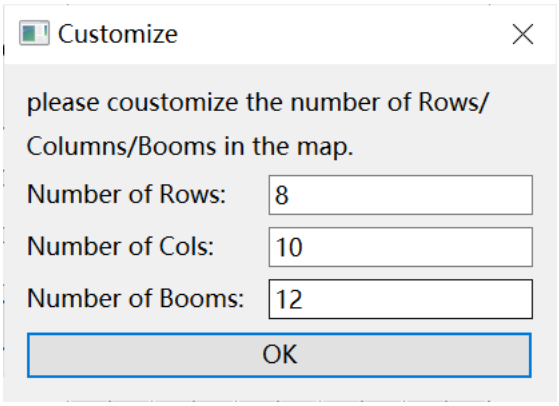
四、部分效果图

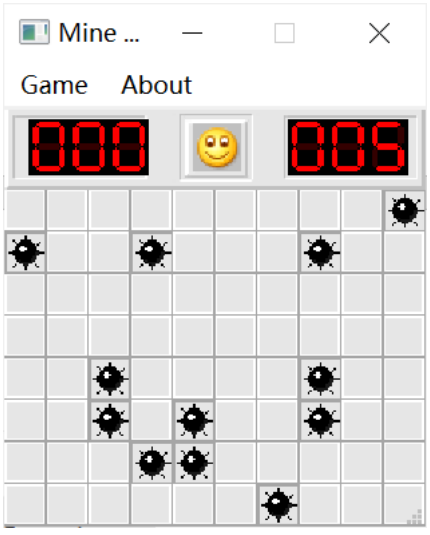
由于部分效果图已经在第二部分技术难点中放出，这部分仅展示一些补充的效果图。

自定义雷区：



填入一组数据作为测试：

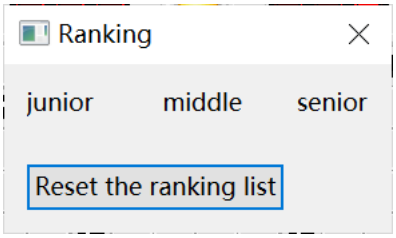




可以看到生成地图中 row=8,col=10,boom=12,符合要求。

排行榜的功能实现已经完成，但由于目前的 timer 是不能正常工作的，需要在三轮迭代中完成对接，改良 timer，从而完成数据写入和排行榜的正常使用。

排行榜初始界面如下：



五、总体心得

在本轮迭代中，时间较为紧迫，一位同学之前还没有接触过 Qt Creator 的使用，因此在编写相关层的时候也较为吃力，好在最后成功完成了第一轮迭代。总的来说，这个程序虽然相比之下较为简单，但麻雀虽小五脏俱全，它很好地锻炼了我们的代码能力。在刚刚学习完 OOP 这样一门 C++的理论课以后，立刻试着与团队一起编写一个较有挑战性的程序，这确实是一种很好的增强 C++代码能力的方式，更提高了我们的自信与兴趣。同时，袁老师上课所讲的内容也使我们受益良多。现在回过头再看上个学期编写的代码，很多都是不太符合规范的。在学习了这几天的内容后，感觉之后的编码之路也可以走的更加流畅了。

感谢老师几天来的辛勤教诲!

六、各人心得

于明辰：

部分心得已在 *二、技术难点及克服* 部分给出。在本轮迭代中，我的任务比较重。由于之前对 Qt Creator 不太熟悉，因此花了一天时间特地熟悉各种比如 UI 界面、.pro、三个层等的编写方法，最后总算懵懵懂懂的了解了一些。结果导致只剩一天半的时间来编写地雷块和左右键的逻辑。。。最后也刚好在老师检查成果的前一个多小时才实现了效果——也算不幸中的万幸。总体下来，对自己分析感觉还是代码能力不够强。之前的 OOP 课期末大程在组内摸鱼，这次遇到了真刀真枪的 C++工程编写，代码能力不强的事实就暴露了。以后还是要加强自己编写代码的能力吧，毕竟学无止境嘛。

王紫晴：

新一轮迭代中，Qt 界面的绘制是比较让人头疼的问题。一方面是界面显示，可以查到的相关函数的介绍并不多，需要通过多次 debug 一点点摸索它的功能，比如 addWidget 和 addLayout 在界面显示上的区别，等等。目前的显示还是不尽如人意，比如对于宽度的控制，界面上出现的效果总是和预期的不完全一致，接下来还需要继续改进。另外一方面是按钮的

使用，熟悉了 slot 槽相关知识之后会好写一点，也在技术难点部分介绍过了。
目前的问题是界面还比较简陋，虽然想要的功能差不多实现了，但是不够美观。后续工作是修改一下 Qt 界面让它更符合用户需求，顺便在过程中测试程序，修改可能出现的 Bug。

薛远：

目前我的工作是做了 C++时间记录。
在于一开始直接调用了 git 上记录程序运行时间的代码，这与从点击开始计时不符。
然后 QTime 库里给的函数大多是返回值或者帮助生成随机数。
所以目前直接做了一个秒表，后续会整合进扫雷的代码里。