

**软件学院大作业任务书**

题 目： 基于Qt实现的经典扫雷游戏

专 业： 软件工程

班 级： 软件工程1910班

姓 名： 王俊洪

学 号： 8002119319

完成人数： 1人

起讫日期： 2020.8.12 — 2020. 8.19

任课教师： 危建国 职称： 讲师

部分管主任： 夏灵林

完成时间： 2020年8月31日前

|  |
| --- |
| **大作业的要求和内容：（包括题目选择范围、技术要求、递交时间、考核方法等）**  一、实验目的  通过设计性、综合性的C++语言程序设计，提高学生编写程序、调试程序的能力，加深C++语言知识的理解和把握，初步掌握面向对象和泛型编程的基本思想以及使用Qt等类库进行软件构建能力。  二、实验内容和要求  要求学生从老师给定的选题或者自定选题（要得到老师的允许），1人1组，根据需求描述（或自己撰写需求描述），独自完成需求分析，程序模块设计以及程序的编写、调试和测试。  要求学生根据需求描述独自完成需求分析，程序模块设计以及程序的编写、调试和测试。综合实训选题的功能较为完善，要完成这些功能需要涉及C++语言程序设计的方方面面，包括应用程序体系结构的确定、类和对象的设计与实现、模板容器以及算法的选用、文件操作、数据库操作、控制台或GUI的输入输出等。大作业必须满足如下要求：   1. 采用**面向对象程序设计，**要运用类的继承、多态等机制； 2. 具备**文件读写或者数据库**功能，程序中的数据要以文件方式保存； 3. 程序中要具备对**STL容器**的运用； 4. 程序中要求具备**异常处理**； 5. 要求具备**菜单功能**，根据用户的输入，执行相应的操作。在**菜单选项**中，必须包含“关于”一项，该菜单项用来显示大作业的基本信息，包括程序名称、版本、完成时间、姓名、班级、学号等信息； 6. 程序要求做到界面友好，对于用户的输入操作要有明确的提示信息，并对输入的数据进行合法性和正确性的检查； 7. 做到工程名、类名、变量名、代码格式等合乎规范，代码中必须对所设计的类的功能或函数功能进行规范的注解。   三、递交时间  在设计完成之后书写设计说明书（实验报告），按规定报告的格式书写，在2020年9月15日前与设计相关文档（包括源代码）一并上交。  四、考核方法  考勤（30%）+平时作业（20%）+设计（50%），其中设计部分比重如下：   1. 日常训练 10% 2. 交流日志 5%   评分依据：是否有真实的交流过程、交流的次数等   1. 需求分析 25%   评分依据：分析是否充分、表述是否明确、功能的实用价值等   1. 文档撰写质量 25%   评分依据：结构完整性、内容充实度、格式符合度、图表规范程度等   1. 代码质量 35%   评分依据：代码复杂度、功能完整性、是否运用了要求的知识点、设计或算法是否有创新等  五、实验报告格式及要求  1、完成需求陈述中所提到的所有功能；  2、要求撰写需求分析、软件设计文档；  3、设计体会和课程建议；  4、设计报告严禁抄袭，否则按不及格论。  最终提交的作业，除提交完整的程序工程代码外，还必须以大作业报告的形式阐述整个程序的实现过程，要求报告内容中必须包括：   1. **技术路线**（要详细说明所使用的主要技术和实现原理并同时**附加个人正面照片**）； 2. **需求分析**（项目介绍、功能需求）； 3. **系统分析与设计**（阐述项目中需解决的关键技术问题，同时要以类关系图、程序流程图、功能模块示意图等辅助系统设计的描述）； 4. **程序设计与实现**（其中内容**不能只是粘贴全部代码**，首先要描述代码工程的文件组织结构，然后以文字的方式阐述代码中每个类的设计意图、主要功能、运用到的关键知识点，主要成员函数的设计意图及功能介绍，对于代码实现中有特色的算法或有创新的地方，可以着重描述（这将视为大作业的加分点）。如果必须给出实现代码才能更好地说明问题时，也必须先有相关的文字叙述，然后才是代码，代码只是作为例证。）； 5. **系统测试**（该部分不能只简单展示几幅程序运行时的截图就完事！要设计几组有代表性的数据来验证程序的各个功能是否达到设计需求。也就是测试是否有不正确或者是遗漏了的功能，在接口上，输入能否正确的接受，能否输出正确的结果；是否有数据结构错误或者外部信息访问错误；性能上是否满足要求；是否有初始化或终止性错误等。建议以表格的形式展示，具体请参考大作业模板的系统测试部分的描述）； 6. **日常训练讨论日志**（以时间顺序记录讨论的时间、地点、讨论的内容，并附加讨论时拍摄的照片）； 7. **个人总结**（该部分必须包含实训项目的工作量、开发过程中遇到的困难以及如果克服困难、个人收获、得到的启示或教训等等，切忌空洞无实际内容或千篇一律的敷衍文字。）；   **参考文献**（该部分给出整个程序从选题、需求分析、系统设计到代码实现过程中所参考的书籍、网上资料等。）。  以压缩文件夹的形式提交，文件夹的命名规则为：学号\_姓名\_班级，如800XXXXXXX\_张三\_软件工程XXX班：  文件夹里面包含软件源代码文件、封包文件、实训报告、数据库文件或者用到的文件、查重报告（可选）。 |
| 教师小结：  成绩：  教师 签名：  系部负责人：  学生 签名： |



**NANCHANG UNIVERSITY**

**程序设计工程实训报告**



**题 目：** 基于Qt实现的经典扫雷游戏

**学 院：** 软件学院

**专 业：** 软件工程

**班 级：** 软件工程1910班

**完成人数：** 1人

**成 员：** 王俊洪

**起讫日期：** 2020.8.12 — 2020.8.19

**任课教师：** 危建国 职称： 讲师

**完成时间：** 2020年8月31日前

**填表日期：** 2020 年 8 月 19

**目 录**

一、 项目技术路线说明 - 1 -

二、 项目需求分析 - 2 -

**2.1** **项目介绍** - 2 -

**2.2 功能需求** - 2 -

三、 系统分析与设计 - 3 -

**3.1** **本程序需解决的关键技术问题** - 3 -

**3.2** **程序流程** - 8 -

**3.3** **功能模块** - 9 -

**3.4** **类的设计** - 11 -

四、 程序设计与实现 - 13 -

**4.1** **工程文件组织结构** - 13 -

**4.2** **程序设计** - 14 -

**4.3** **程序实现** - 15 -

五、 系统测试 - 19 -

**5.1** **测试概要** - 19 -

**5.2** **测试环境与配置** - 19 -

**5.3** **测试方法和工具** - 19 -

**5.4** **测试内容** - 20 -

六、 个人小结 - 25 -

# 项目技术路线说明

本人照片：

本项目是基于Qt实现的一个扫雷的小游戏，该游戏运用了Qt中的窗口类、按钮类、标签类等一些基础的ui控件来实现程序的图形化。

运用了QPropertyAnimation类来实现了基础的一些动画。同时程序利用Qt的消息和事件中的鼠标事件，通过捕获鼠标点击的位置，鼠标点击的按钮来实现扫雷的点击操作。

其中，对于随机地图的生成，用到了qrand()函数，每次随机生成地雷的行和列，如果该行和该列不是地雷的话，则加入到地雷集合中，同时地雷个数加一，知道地雷个数达到预期数量，随机埋雷完毕。

至于扫雷实现，则是运用深度优先搜索不断搜索数字为0的格子周围的格子，递归调用扫雷函数，直到将0附近的数字格子全部扫面完。

最后，通过一个自己设计的QDialog类来完成游戏的成功与否提示，成功还会显示游戏完成所用的时间。

该项目还用到定时器和QString的格式化，定时器用于计时，统计游戏运行时间和剩余的地雷个数，通过QString的格式化，将信息显示到QLabel中，最后在游戏胜利后，时间信息将会用于提示游戏完成时间。

# 项目需求分析

## **项目介绍**

该项目是一个经典的扫雷游戏，该游戏分为了三个不同的模式，简单、一般、困难。

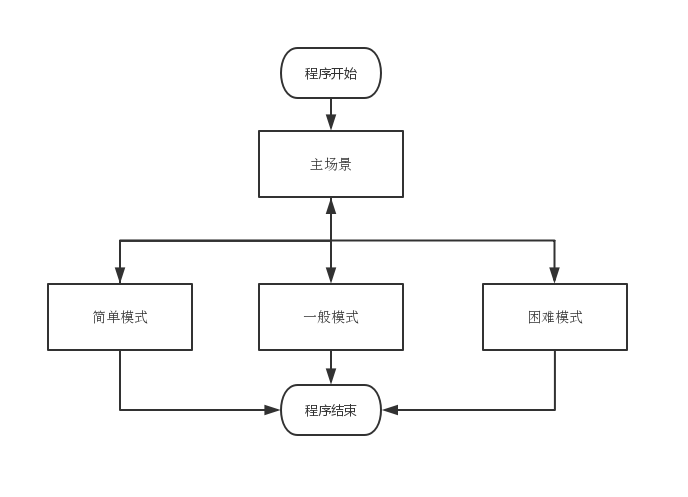
在简单模式下，10\*10的地图中随机埋藏了15颗地雷，一般模式中20\*20的地图中随机埋下了60颗地雷，困难模式则是在20\*40的地图中随机埋下120颗地雷。

地图中格子为空表示该格子周围(与该格子直接接触的格子)没有地雷，格子中的数字则表示该格子周围的地雷个数。玩家正是需要根据格子中的数字来清理地雷，如果玩家清理掉了全部的地雷，则游戏胜利，否则游戏失败。

## **2.2 功能需求**

该项目不需要过于复杂的需求实现，只需要保证一些基本的需求即可。

其中首先需要完成界面之间的转化，转化关系如下图：



其次需要实现左键单击点开格子扫雷，右键单击插入旗子和设置问号，问号可以帮助玩家先标记一个未知格子。

另外游戏结束时能够提示玩家游戏结果。

这个项目需要这些需求就差不多了。

# 系统分析与设计

## **本程序需解决的关键技术问题**

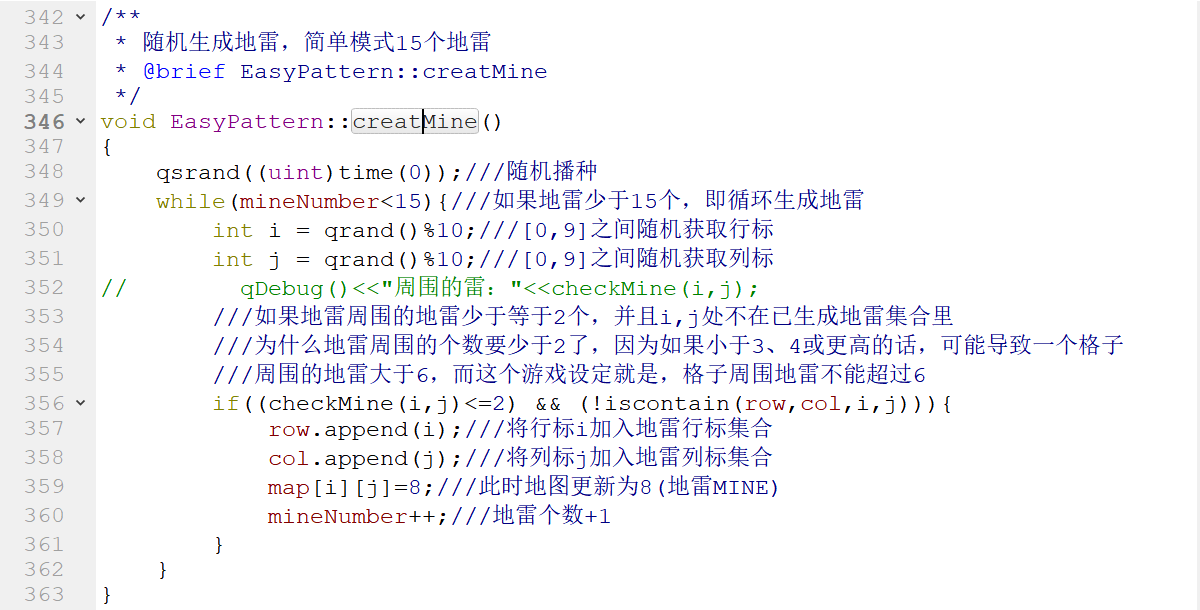
本程序需要解决两个关键的技术问题，一个技术问题是如何随机生成地图，第二个问题是如何完成扫雷的点击实现。

针对第一个问题，随机生成地图，确保每次扫雷的地图都是随机的。于是，看到“随机”二字，自然想到的就是随机函数qrand()，这个函数可以生产一个

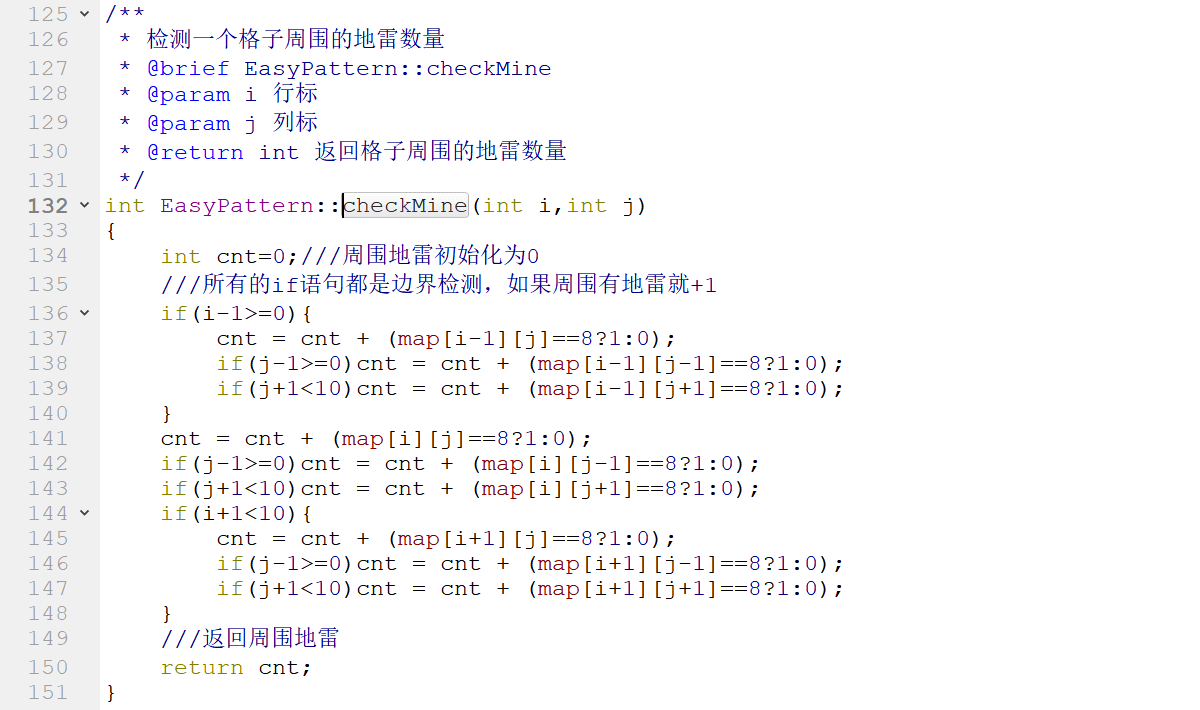
随机数，但是怎么将随机数转化成随机的地图呢？下面是第一个问题的解决方案。

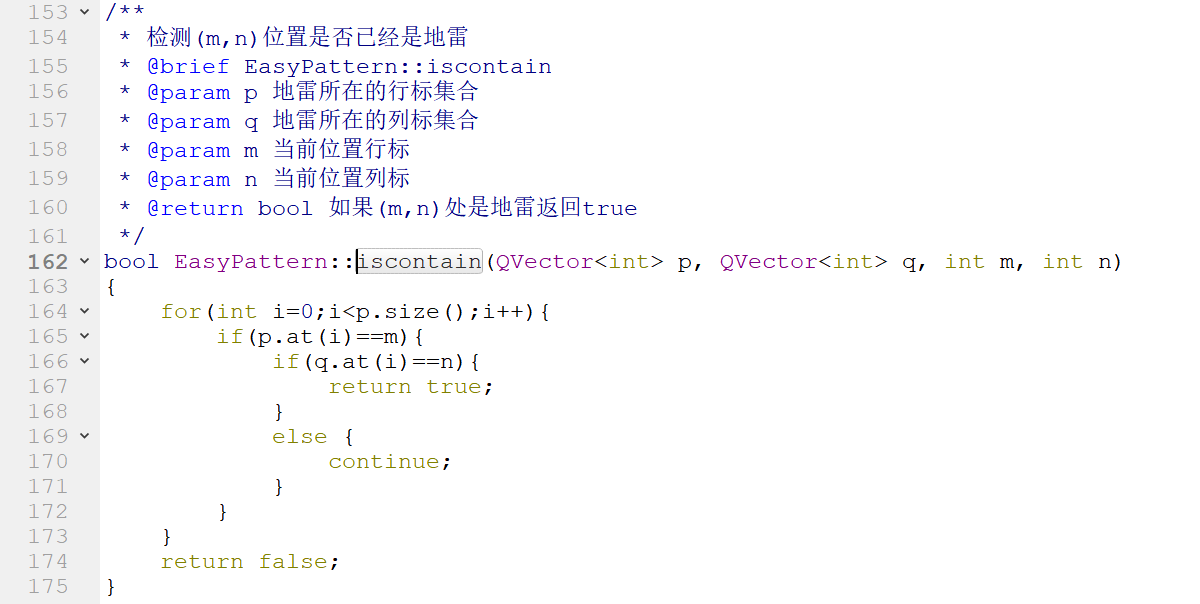
首先，不要考虑直接就随机生成一张包含地雷和数字的地图，因为这样很难处理地雷和数字的关系。所以，本项目采取的是只随机生成地雷，只要地雷确定了位置，那么每一个格子里的数字自然也就确定了，因此，随机生成地图的核心在于随机生成地雷。在地雷随机生成后便可以根据地雷的位置来确定每个格子的数字，这样一张随机的扫雷地图就成功生成了。下面以简单模式为例具体说明。

首先定义两个QVector，用来保存地雷所在的行数和列数，由于地雷所在的行数和列数都是一个数字，便可以运用随机函数来随机生成。每次随机生成地雷所在的行数和列数，并且检测此处是否已经是地雷，如果已经是地雷则跳过，否则就将该处设置为地雷，并将行标和列标分别加入两个QVector中保存起来。以下是随机生成地雷的函数实现的代码。



其中row和col就是储存行标和列标的两个QVector，checkMine()函数是统计这个格子周围的地雷个数，iscontain()函数是判断地雷是否已经存在，这两个函数的实现如下：





到此，基本已经解决随机地图生成的问题了，后面再遍历地图初始化，填上每个位置的数字，然后画出地图即可。

针对第二个问题，如何实现点击完成扫雷操作，首先想到的就是模拟，点击之后查看该处的内容，内容可能有很多情况，下面是第二个问题的解决方案。

首先，地图用的是QLabel来承载，因此需要定义一个二维的指针数组，指向每一个位置的QLabel，其次，需要确定每个格子目前的状态（一共有已点开、未点开、旗子、问号四种状态），这仅仅靠一个地图二维数组是无法实现的，这里可以设计一个结构体，但本项目采取的是在定义一个二维数组用于标记每处的状态。

在左键单击地图某处后，通过鼠标事件发出一个点击信号，在监听点击信号，执行扫雷函数，这里之所以不在鼠标点击事件中直接实现扫雷算法，是因为鼠标点击事件是系统函数，不能手动调用，故不能实现递归调用，因此没有直接在鼠标事件里面实现扫雷算法。监听到点击信号后，进入扫雷函数，执行扫雷算法的实现。具体实现如下：

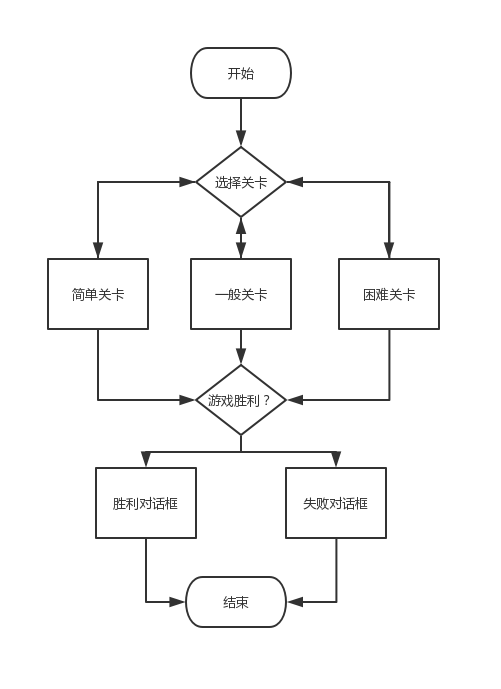


由于注释已经够详细了，这里简要仅做简要说明。

如果鼠标是左键点击某个格子，并且该格子处于为点开状态，则会点开该格子，如果发现点开的是雷，当即游戏结束；如果点开的是数字，则显示该数字即可；但是如果点开的是空白，说明这个格子周围没有地雷，则应该递归点击它周围的格子。这里，你可能会发现，函数第四个参数是一个bool类型，并且参数名为click。这个参数的含义是表示是否是玩家点击的，其中true表示是玩家点开。为什么需要这个参数呢？因为如果是玩家点击到地雷，则会游戏失败，但是，也存在这么一种情况，那就是玩家点击到了空白，系统递归调用sweep函数点击周围的格子，如果这个时候系统点击到了雷，按理说游戏应该不会结束，如果此时游戏结束实在不符逻辑（系统说这个锅我可不背），因此特加这么一个参数用于区分。当然，玩家也可以使用鼠标右键点击未点开的格子，此时格子会在旗子，问号，和初始之间切换。

## **程序流程**

程序流程如下面这幅图所示，这个本就是一个小游戏项目，没有过于复杂的程序流程，下面这幅图定能够说明清楚其中的关系。



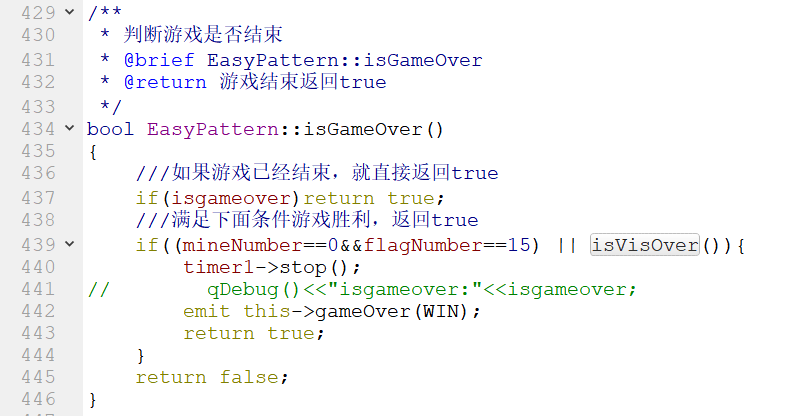
## **功能模块**

整个程序分为三个功能模块，包括随机地图的生成、扫雷的实现和游戏结束的判定，在前面章节中已经提到过地图的生成和扫雷的实现，这里简单叙述一下游戏胜利的判定，与之相关的主要有两个函数，分别为isVisOver()和isGameOver()两个函数。

其中isVisOver()是判断没有点开的是不是全是地雷，或者说不是地雷的是否已经全部点开了，其算法思路就是遍历二维地图数组，如果存在未点开但不是地雷的就返回false，否则返回true，其代码如下：



至于第二个游戏结束函数主要是和判断游戏胜利的，其算法思路就是如果上一个介绍的函数为true或者剩余的地雷为0并且旗子没有多余的（指的是没有插旗的地方不是地雷的旗子），其代码如下：

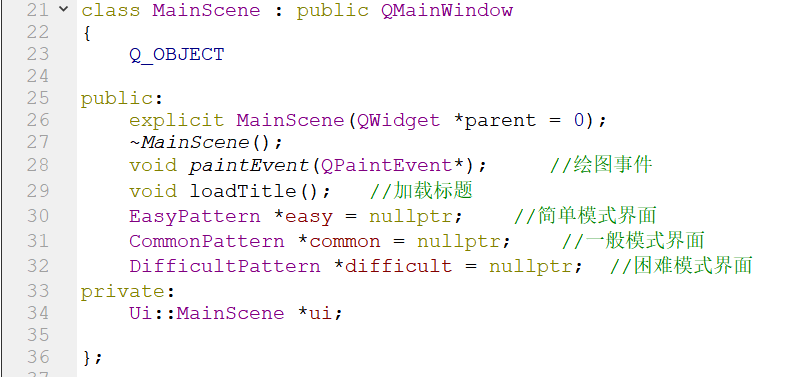


## **类的设计**

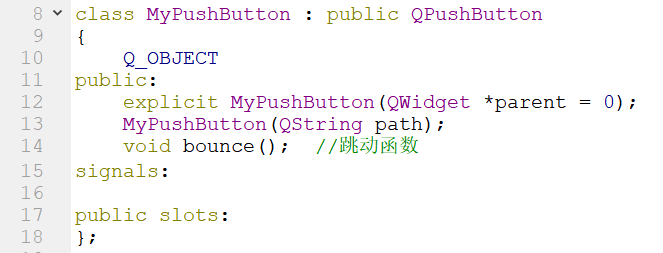
整个项目共包含六个类，下面分别介绍一下各个类

六个类分别为MainScene类、MyPushButton类、GameOverDialog类、EasyPattern类、CommonPattern类和DifficultPattern类

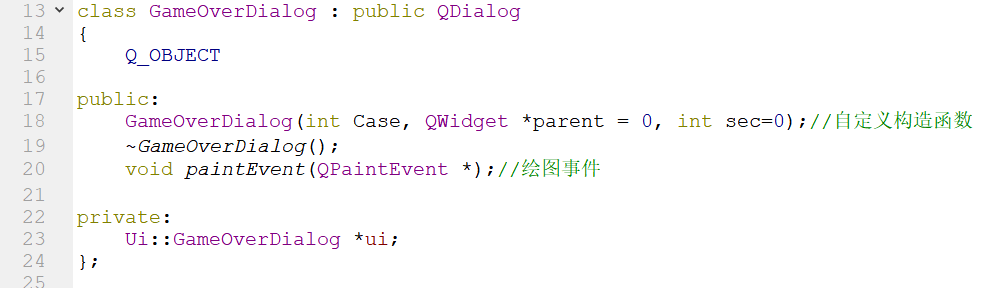
MainScene类表示主场景类，即游戏一运行的界面，该类中的成员如下图：



MyPushButton类是自定义按钮类，程序中的按钮都是该类的对象，该类的成员如下图：



GameOverDialog类是游戏结束后弹出的对话框类，该类会根据游戏的结果（成功还是失败）弹出不同的对话框，该类的成员如下：



EasyPattern类、CommonPattern类和DifficultPattern类是三个不同的游戏难度的界面类，每个难度的界面不同，因此设计到了三个不同的类中，其中的成员基本都相似，这里以EasyPattern类为例示范：



这里几个类就介绍完了，不做过多说明，因为注释写得很详细，如果想关注其实现可以在源代码中查看（[./源代码/MineSweeper/MineSweeper.pro](源代码/MineSweeper/MineSweeper.pro" \o "点击此处打开源代码)）

# 程序设计与实现

## **工程文件组织结构**

本项目包括1个工程文件、6个头文件、7个源文件、5个界面文件、1个资源文件以及一个其他文件。

工程文件是MineSweeper.pro，这是整个工程项目文件，里面是项目的一些配置。

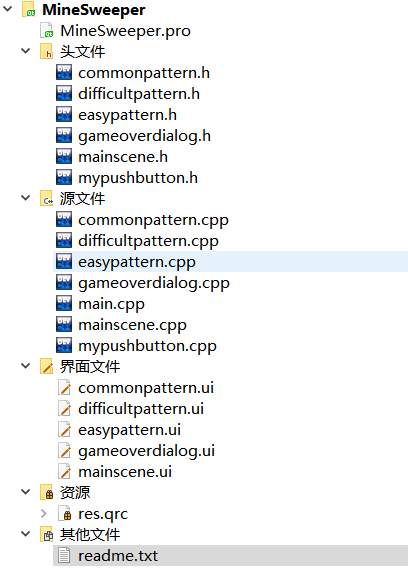
头文件包括mainscene.h、mypushbutton.h、gameoverdialog.h、easypattern.h、commonpattern.h、difficultpattern.h六个，这些分别是对应类的头文件，实现一个类对应一个头文件。

源文件包括mainscene.cpp、mypushbutton.cpp、gameoverdialog.cpp、easypattern.cpp、commonpattern.cpp、difficultpattern.cpp以及main.cpp，除了main.cpp，其余的都是对应内的源文件，都是有对应头文件的，其类的定义在头文件，类的成员函数的实现在源文件。

界面文件包括mainscene.ui、gameoverdialog.ui、easypattern.ui、commonpattern.ui、difficultpattern.ui五个，界面文件里面是界面的设计文件。

资源文件包括res.qrc，资源文件便是项目的图片等资源的二进制文件。

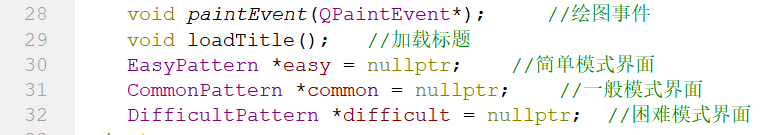
最后有一个其他文件readme.txt，这个文件主要是说明程序中一些常量定义的含义。



## **程序设计**

前文提到过，程序包含6个类。

其中MainScene类的设计就是为了显示主场景，下面对几个重要成员做介绍



paintEvent函数，几乎每个界面类都有此函数，此函数是一个重载函数，主要用于绘制背景，以后其他类中的该函数将不做介绍了。

loadTitle函数是通过QPropertyAnimation动画类来加载标题的一个函数，这个函数主要是用于动画弹出标题“经典扫雷”。

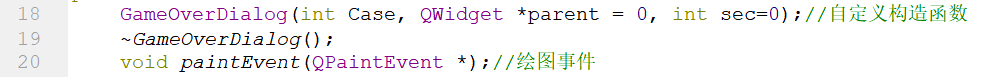
另外，还包括三个成员变量，分别对应三种模式的窗口，初始值都是空指针。

MyPushButton类的设计是为了实现自定义按钮，其中的成员如下：



其中构造函数是为了初始化按钮，以传入的path为路径的图片作为按钮背景，而bounce()函数是为了实现点击按钮，按钮跳动。

GameOverDialog类的设计是为了实现游戏结束根据游戏结果弹出不同的对话框，其中的成员如下：



构造函数是根据传入的参数不同来初始化不同的对话框。

其余的三个类是分别实现了不同的游戏场景：简单、一般和困难，由于每个场景的窗口参数不同，因此分别设计了三个类。

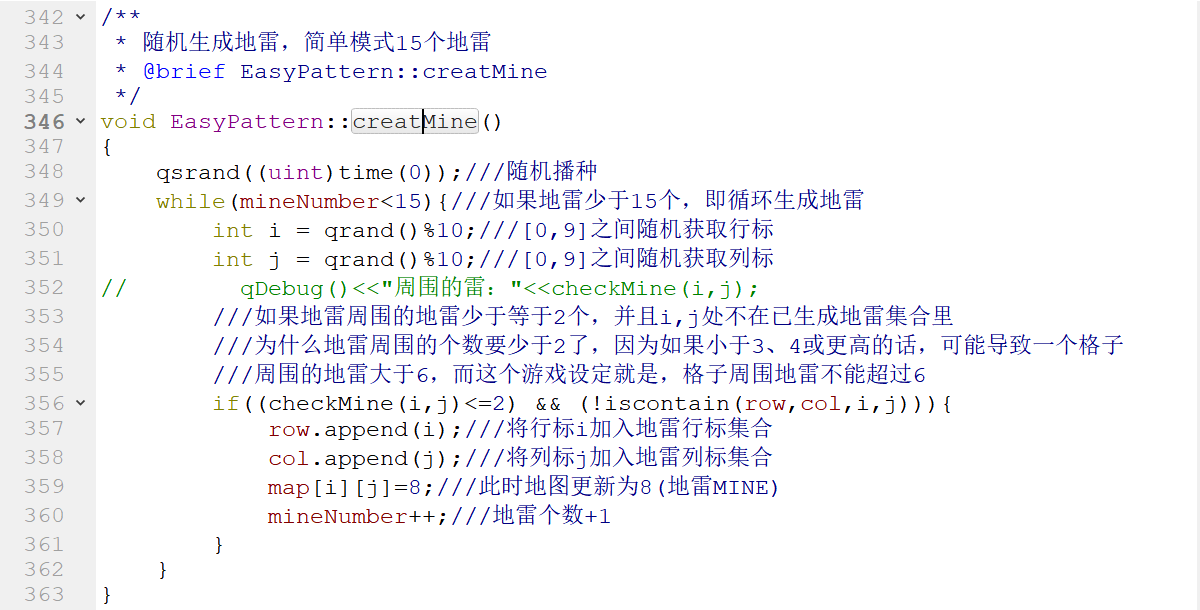
## **程序实现**

程序的实现比较简单，这里着重介绍两大亮点：随机地雷的生成和扫雷的实现。

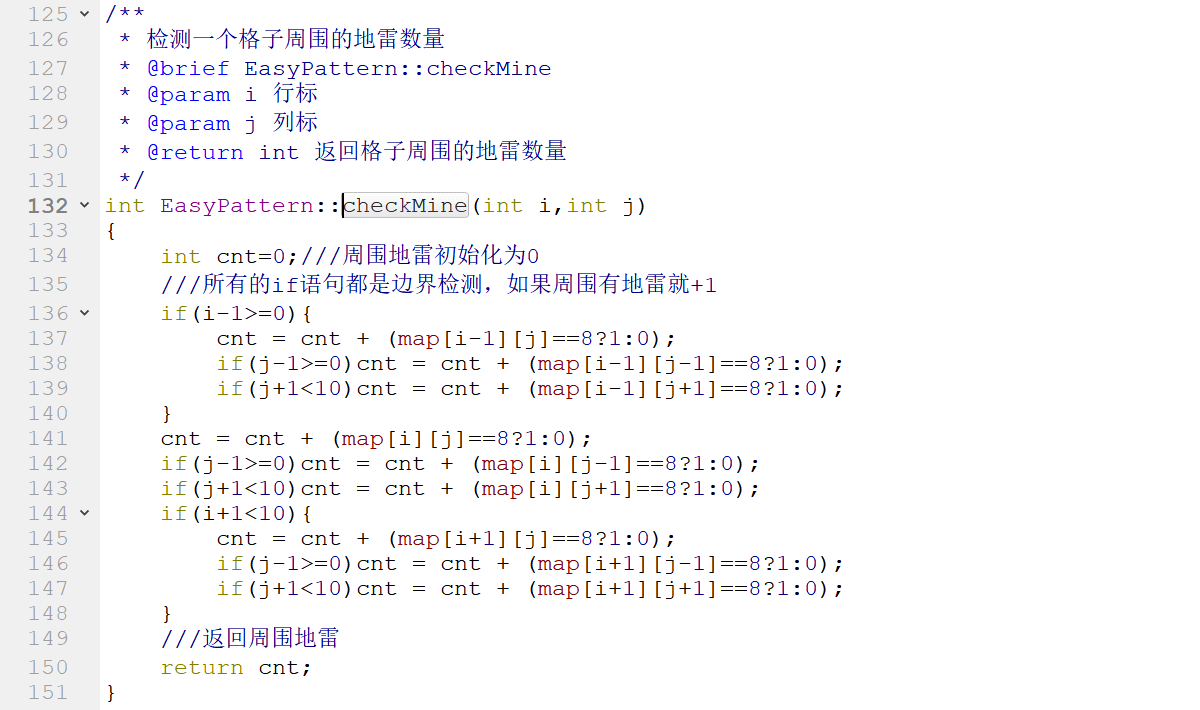
随机生成地图，确保每次扫雷的地图都是随机的。看到“随机”二字，自然想到的就是随机函数qrand()，这个函数可以生产一个随机数，但是怎么将随机数转化成随机的地图呢？下面是第一个问题的解决方案。

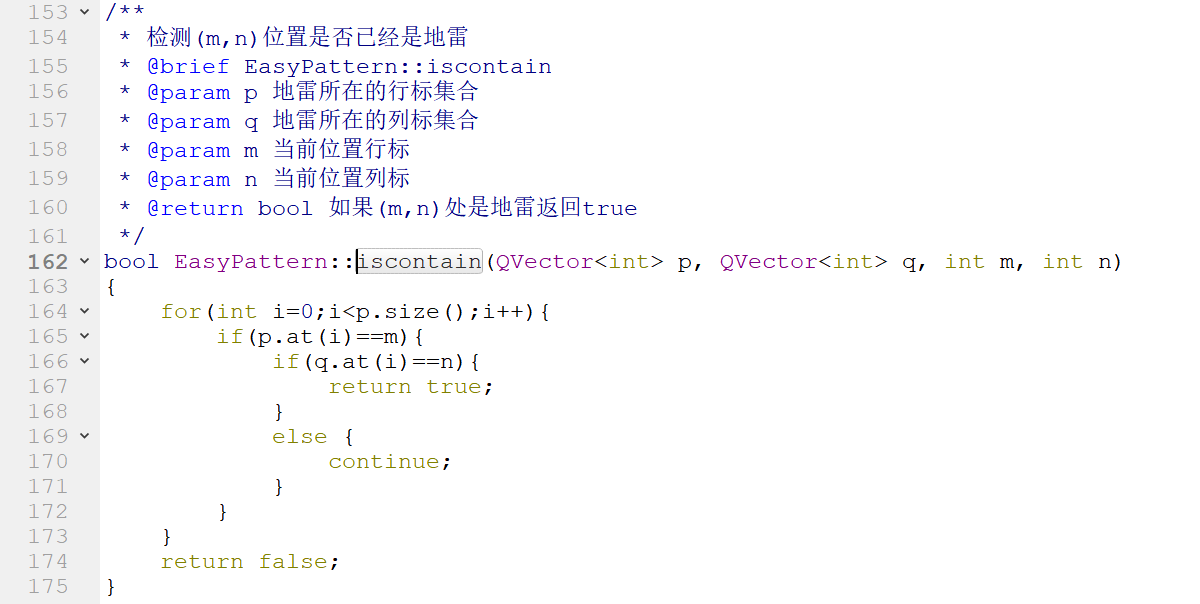
首先，不要考虑直接就随机生成一张包含地雷和数字的地图，因为这样很难处理地雷和数字的关系。所以，本项目采取的是只随机生成地雷，只要地雷确定了位置，那么每一个格子里的数字自然也就确定了，因此，随机生成地图的核心在于随机生成地雷。在地雷随机生成后便可以根据地雷的位置来确定每个格子的数字，这样一张随机的扫雷地图就成功生成了。下面以简单模式为例具体说明。

首先定义两个QVector，用来保存地雷所在的行数和列数，由于地雷所在的行数和列数都是一个数字，便可以运用随机函数来随机生成。每次随机生成地雷所在的行数和列数，并且检测此处是否已经是地雷，如果已经是地雷则跳过，否则就将该处设置为地雷，并将行标和列标分别加入两个QVector中保存起来。以下是随机生成地雷的函数实现的代码。



其中row和col就是储存行标和列标的两个QVector，checkMine()函数是统计这个格子周围的地雷个数，iscontain()函数是判断地雷是否已经存在，这两个函数的实现如下：





到此，基本已经解决随机地图生成的问题了，后面再遍历地图初始化，填上每个位置的数字，然后画出地图即可。

第二个亮点就是递归实现扫雷，运用到了最基础的深度搜索算法。具体介绍如下。

首先，地图用的是QLabel来承载，因此需要定义一个二维的指针数组，指向每一个位置的QLabel，其次，需要确定每个格子目前的状态（一共有已点开、未点开、旗子、问号四种状态），这仅仅靠一个地图二维数组是无法实现的，这里可以设计一个结构体，但本项目采取的是在定义一个二维数组用于标记每处的状态。

在左键单击地图某处后，通过鼠标事件发出一个点击信号，在监听点击信号，执行扫雷函数，这里之所以不在鼠标点击事件中直接实现扫雷算法，是因为鼠标点击事件是系统函数，不能手动调用，故不能实现递归调用，因此没有直接在鼠标事件里面实现扫雷算法。监听到点击信号后，进入扫雷函数，执行扫雷算法的实现。具体实现如下：



由于注释已经够详细了，这里简要仅做简要说明。

如果鼠标是左键点击某个格子，并且该格子处于为点开状态，则会点开该格子，如果发现点开的是雷，当即游戏结束；如果点开的是数字，则显示该数字即可；但是如果点开的是空白，说明这个格子周围没有地雷，则应该递归点击它周围的格子。这里，你可能会发现，函数第四个参数是一个bool类型，并且参数名为click。这个参数的含义是表示是否是玩家点击的，其中true表示是玩家点开。为什么需要这个参数呢？因为如果是玩家点击到地雷，则会游戏失败，但是，也存在这么一种情况，那就是玩家点击到了空白，系统递归调用sweep函数点击周围的格子，如果这个时候系统点击到了雷，按理说游戏应该不会结束，如果此时游戏结束实在不符逻辑（系统说这个锅我可不背），因此特加这么一个参数用于区分。当然，玩家也可以使用鼠标右键点击未点开的格子，此时格子会在旗子，问号，和初始之间切换。

这个函数最大的难点就是递归的实现，其中还要区分是否是人为点击，以及递归的边界也就是递归出口的设计。

# 系统测试

## **测试概要**

由于程序过于简单，因此测试这方面也比较简单，但是由于扫雷比较费时，因此，测试时打印生成的地图来根据地图进行测试。

测试目的，检测程序是否还存在某些浅显的bug，从而去解决这些bug。

测试点：测试左键点击是否可以展开、测试点击已经展开的是否变化、测试点击到地雷会发生什么、测试点击空白时的递归扫雷函数是否能正确执行、测试右键格子的变化、测试游戏结束窗口是否正确弹出等等。

本次测试发现的问题不会出现在提交的版本中，在提交的版本中已经修复了本次是测试发现的bug。

## **测试环境与配置**

测试环境是Windows10环境下，测试环境Qt。

## **测试方法和工具**

测试方法是尝试运行，根据测试点来寻找是否出现bug，测试工具为Qt Creator5.11版本。

## **测试内容**

1. 随机地图的生成时间测试

时间的测试通过ctime库中的clock()函数来统计的，具体测试时间代码见源文件。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试编号 | 预期结果 | 实际结果 | 是否合格 | 备注 |
| 01 | <1s | 15ms | PASS | 简单模式 |
| 02 | <1s | 14ms | PASS |
| 03 | <1s | 55ms | PASS | 一般模式 |
| 04 | <1s | 46ms | PASS |
| 05 | <1s | 78ms | PASS | 困难模式 |
| 06 | <1s | 72ms | PASS |

1. 左键单击测试

单击测试包括单击未点开的，单击已点开的，单击开为空白格，单击开为地雷，单击开为数字这几项测试，都通过才算通过

1)单击未点开的

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试编号 | 预期结果 | 实际结果 | 是否合格 | 备注 |
| 01 | 点开 | 点开 | PASS | 简单模式 |
| 02 | 点开 | 点开 | PASS |
| 03 | 点开 | 点开 | PASS | 一般模式 |
| 04 | 点开 | 点开 | PASS |
| 05 | 点开 | 点开 | PASS | 困难模式 |
| 06 | 点开 | 点开 | PASS |

2)单击已点开的

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试编号 | 预期结果 | 实际结果 | 是否合格 | 备注 |
| 01 | 未变化 | 未变化 | PASS | 简单模式 |
| 02 | 未变化 | 未变化 | PASS |
| 03 | 未变化 | 未变化 | PASS | 一般模式 |
| 04 | 未变化 | 未变化 | PASS |
| 05 | 未变化 | 未变化 | PASS | 困难模式 |
| 06 | 未变化 | 未变化 | PASS |

1. 单击开为空白格

预期结果为：递归点开周围格子，直到遇到数字格结束，且不点开地雷格，下面用A表示这种效果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试编号 | 预期结果 | 实际结果 | 是否合格 | 备注 |
| 01 | A | A | PASS | 简单模式 |
| 02 | A | A | PASS |
| 03 | A | A | PASS | 一般模式 |
| 04 | A | A | PASS |
| 05 | A | A | PASS | 困难模式 |
| 06 | A | A | PASS |

4)单击开为地雷

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试编号 | 预期结果 | 实际结果 | 是否合格 | 备注 |
| 01 | 游戏失败 | 游戏失败 | PASS | 简单模式 |
| 02 | 游戏失败 | 游戏失败 | PASS |
| 03 | 游戏失败 | 游戏失败 | PASS | 一般模式 |
| 04 | 游戏失败 | 游戏失败 | PASS |
| 05 | 游戏失败 | 游戏失败 | PASS | 困难模式 |
| 06 | 游戏失败 | 游戏失败 | PASS |

5)单击开为数字

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试编号 | 预期结果 | 实际结果 | 是否合格 | 备注 |
| 01 | 展开该数字 | 展开该数字 | PASS | 简单模式 |
| 02 | 展开该数字 | 展开该数字 | PASS |
| 03 | 展开该数字 | 展开该数字 | PASS | 一般模式 |
| 04 | 展开该数字 | 展开该数字 | PASS |
| 05 | 展开该数字 | 展开该数字 | PASS | 困难模式 |
| 06 | 展开该数字 | 展开该数字 | PASS |

测试发现，左键几乎不存在bug

1. 右键单击测试

右键单击测试包括右键单击能否出现旗子，右键再次单击能否出现问号，

右键再次单击是否回到最初，在非最初条件下左键单击是否能够点开，右键单击已经点开过的能否改变。

1)右键单击能否出现旗子

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试编号 | 预期结果 | 实际结果 | 是否合格 | 备注 |
| 01 | 出现旗子 | 出现旗子 | PASS | 简单模式 |
| 02 | 出现旗子 | 出现旗子 | PASS |
| 03 | 出现旗子 | 出现旗子 | PASS | 一般模式 |
| 04 | 出现旗子 | 出现旗子 | PASS |
| 05 | 出现旗子 | 出现旗子 | PASS | 困难模式 |
| 06 | 出现旗子 | 出现旗子 | PASS |

2)右键再次单击能否出现问号

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试编号 | 预期结果 | 实际结果 | 是否合格 | 备注 |
| 01 | 出现问号 | 出现问号 | PASS | 简单模式 |
| 02 | 出现问号 | 出现问号 | PASS |
| 03 | 出现问号 | 出现问号 | PASS | 一般模式 |
| 04 | 出现问号 | 出现问号 | PASS |
| 05 | 出现问号 | 出现问号 | PASS | 困难模式 |
| 06 | 出现问号 | 出现问号 | PASS |

3)右键再次单击是否复原

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试编号 | 预期结果 | 实际结果 | 是否合格 | 备注 |
| 01 | 复原 | 复原 | PASS | 简单模式 |
| 02 | 复原 | 复原 | PASS |
| 03 | 复原 | 复原 | PASS | 一般模式 |
| 04 | 复原 | 复原 | PASS |
| 05 | 复原 | 复原 | PASS | 困难模式 |
| 06 | 复原 | 复原 | PASS |

4)在非最初条件下左键单击是否能够点开

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试编号 | 预期结果 | 实际结果 | 是否合格 | 备注 |
| 01 | 不能点开 | 不能点开 | PASS | 简单模式 |
| 02 | 不能点开 | 不能点开 | PASS |
| 03 | 不能点开 | 不能点开 | PASS | 一般模式 |
| 04 | 不能点开 | 不能点开 | PASS |
| 05 | 不能点开 | 不能点开 | PASS | 困难模式 |
| 06 | 不能点开 | 不能点开 | PASS |

5)右键单击已经点开过的能否改变

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试编号 | 预期结果 | 实际结果 | 是否合格 | 备注 |
| 01 | 不会改变 | 会改变 | ERROR | 简单模式 |
| 02 | 不会改变 | 会改变 | ERROR |
| 03 | 不会改变 | 会改变 | ERROR | 一般模式 |
| 04 | 不会改变 | 会改变 | ERROR |
| 05 | 不会改变 | 会改变 | ERROR | 困难模式 |
| 06 | 不会改变 | 会改变 | ERROR |

此次测试发现，程序存在这么一个问题：在点开的格子上右键单击会出现旗子，这违背了游戏规则。

但是，在提交版本中这个bug已经解决掉了，不用担心。

4、游戏结束界面的弹出测试

此次测试只包括两个内容：游戏胜利和游戏失败的对话框能否顺利弹出

1)游戏胜利

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试编号 | 预期结果 | 实际结果 | 是否合格 | 备注 |
| 01 | 弹出胜利对话框 | 弹出胜利对话框 | PASS | 简单模式 |
| 02 | 弹出胜利对话框 | 弹出胜利对话框 | PASS |
| 03 | 弹出胜利对话框 | 弹出胜利对话框 | PASS | 一般模式 |
| 04 | 弹出胜利对话框 | 弹出胜利对话框 | PASS |
| 05 | 弹出胜利对话框 | 弹出胜利对话框 | PASS | 困难模式 |
| 06 | 弹出胜利对话框 | 弹出胜利对话框 | PASS |

2)游戏失败

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试编号 | 预期结果 | 实际结果 | 是否合格 | 备注 |
| 01 | 弹出失败对话框 | 弹出失败对话框 | PASS | 简单模式 |
| 02 | 弹出失败对话框 | 弹出失败对话框 | PASS |
| 03 | 弹出失败对话框 | 弹出失败对话框 | PASS | 一般模式 |
| 04 | 弹出失败对话框 | 弹出失败对话框 | PASS |
| 05 | 弹出失败对话框 | 弹出失败对话框 | PASS | 困难模式 |
| 06 | 弹出失败对话框 | 弹出失败对话框 | PASS |

3)游戏结束后还可不可以点击开

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试编号 | 预期结果 | 实际结果 | 是否合格 | 备注 |
| 01 | 不能 | 不能 | PASS | 简单模式 |
| 02 | 不能 | 不能 | PASS |
| 03 | 不能 | 不能 | PASS | 一般模式 |
| 04 | 不能 | 不能 | PASS |
| 05 | 不能 | 不能 | PASS | 困难模式 |
| 06 | 不能 | 不能 | PASS |

经过上面的测试，发现游戏存在一个bug，但是该bug在提交的版本中已经是解决了的。

另外，测试时还发现一个不算bug的bug，那就是只有当旗子插到地雷格子上才能更新剩余地雷，而旗子又是可以无限插的，因此不建议玩家通过这个方式来完成游戏，这个设计，只是为了提示玩家，而不是引导玩家。

# 个人小结

本项目虽不是什么大工程，但是对于初学Qt的我来说，还是要花上一点时间的，我几乎花了一个周来完成这个项目。下面来谈谈我在开发中遇到的一些困难。

遇到的第一个困难就是，对地图数组没有初始化，造成生成的地图数组始终不是预期的，这个问题上可是花费了我大把时间，至于后来我是怎么找到这个问题的，那便是我将代码复制到C++的编译环境下，只改语法，不该思路，用单步调试才发现的，调试初期我还没意识到那个初值是随机的，我一直在怀疑是我的程序在哪步更改了它的内容，直到当时看Java时突然看到定义的变量一定要赋初值，才解决了我一个疑惑，虽然这个错误很小，当时在初学者手中绝对经常犯，并且影响可能还不小，因此，建议以后在程序设计中一定要注意对新的变量赋予初值，特别是指针变量的时候，更要注意这个问题。

遇到的第二个问题便是代码复制问题，由于我的简单、一般、困难三个类几乎相同，于是我便采取了代码直接复制，在局部更改的方法。本来这可以大大节省我的时间，但是却因为一些疏忽，将重要的代码没有做修改，从而造成了数组越界，结果反而浪费了我很多时间。最后我不得不一个一个比较，最终发现原来是数组越界造成的。因此，在平时的开发中，即是代码比较相似，也不建议采取复制粘贴再修改的方法，相似的代码可以抽象成函数或者方法。

第三个问题上面已经说了，那就是数组越界，这个问题也是不起眼，但是却致命，甚至能引起异常造成程序异常退出。所以平时为了防止数组越界，我们可以采取一些边界检测手法，或者自己多加注意，尽量不要造成数组越界。

开发中，还遇到过很多问题，但是比较困扰我的就这三个问题，因此，我们应该在平时写代码时便养成良好的代码习惯，养成良好的代码风格，这样不仅能够少出错，即是出错了也方便寻找错误点。

本次的项目开发，我还是学到了很多东西，包括知识点和一些开发技巧等等，希望以后的学习中能够不断探索。