寒假训练项目——扫雷报告

班级：自 36 姓名：刘柏 学号：2013011548 日期：2014.2.14

一、基本资料

1、 项目名称：扫雷
2、 语言：C/C++
3、 应用程序框架：Qt 5.2.0
4、 开发环境：Qt Creator

二、完成情况

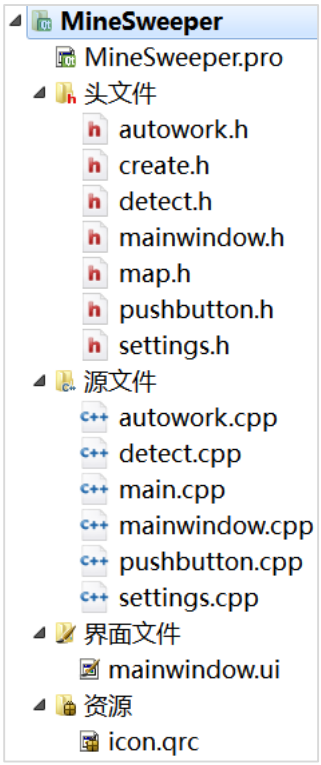
1、 已实现通过鼠标点击进行操作。
2、 已实现基本流程和功能（菜单栏、标记小旗、标记问号、计时器、剩余旗数、重新开始、自定义雷区大小和雷数、帮助指南等）。
3、 已实现扩散算法（翻开空方块时，自动翻开周围的非雷方块）。
4、 已实现把部分函数写入库并通过库调用来使用。

三、不足之处

1、 未实现“扫雷英雄榜”。
2、 玩家点开第一个方块时仍有可能触雷。
3、 由于Qt5对中文支持性不足，“游戏玩法”中许多汉字无法正常显示，故采用了图片格式。
4、 代码过长，扩散算法效率不高。

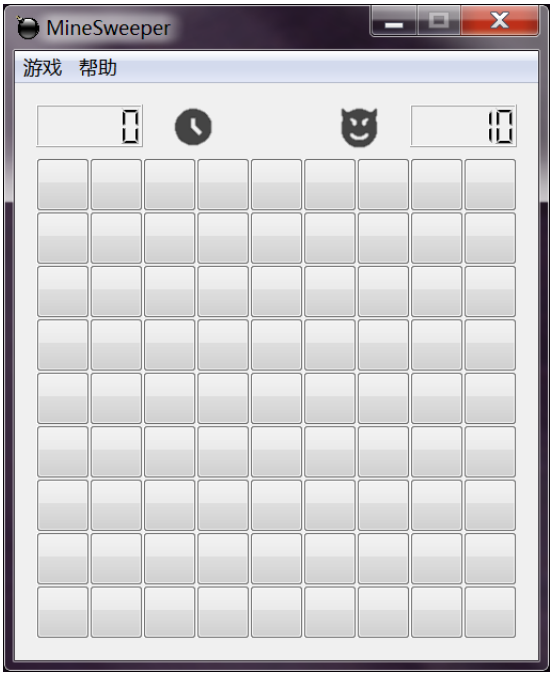
四、相关截图

1、 目录

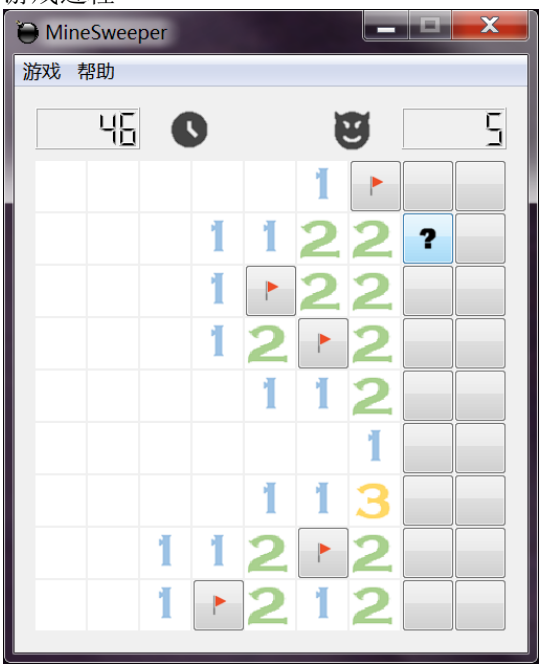


2、 进入游戏

3、游戏初始界面



4、游戏过程



5、玩家获胜

6、玩家失败

五、头文件代码

```
/*!
 * autowork.h
 * Autowork 的声明。
 * 本类用于自动翻开空方块周围的非雷方块。
 * 思路：
 * 转变思维方式，从“自动翻开空方块周围的非雷方块”转换为
 * “扫描所有方块，当某一非雷且尚未翻开的方块周围有空方块时，
 * 该方块自动翻开”
```

```
 */

#ifndef AUTOWORK_H
#define AUTOWORK_H

#include <QObject>
#include "pushbutton.h"

class Autowork:public QObject
{
    Q_OBJECT

public:
    Autowork(); //构造函数
    ~Autowork();  //析构函数
    void Acquire(int r, int c, int rt, int ct, int **f, int **p,
PushButton ***b);  //用于初始化相关数据
    int Judge(int r, int c);  //用于判断一个方块周围是否有空方块

public slots:
    void Scan();  //用于扫描所有方块，当某一非雷且尚未翻开的方块周围有已翻开的空方
块时，该方块自动翻开

private:
    int row, column;  //分别表示翻开的方块所在的行、列
    int RowTotal, ColumnTotal;  //分别表示雷区总行数、雷区总列数
    int **flag;  //表示整个雷阵方块被翻开的情况
    int **pointer;  //表示地雷分布图
    PushButton ***button;  //表示代表方块的按钮组
};

#endif // AUTOWORK_H

////////////////////
/*!
 * create.h
 * 此头文件包含创建按钮组、雷区分布标签组、
 * 雷阵被翻开的情况组的函数。
 */

#ifndef CREATE_H
#define CREATE_H

#include "pushbutton.h"

#include <QLabel>

//该函数动态生成了 row*column 的按钮组
PushButton ***ButtonCreate(int row, int column)
{
    PushButton ***area;
    int i;

    area = (PushButton***)malloc(row*sizeof(PushButton**));
    for (i = 0; i<row; i++)
```

```
        *(area+i) = (PushButton**)malloc(column*sizeof(PushButton*));

    return area;
}

//该函数动态生成了 row*column 的标签组
QLabel ***LabelCreate(int row, int column)
{
    QLabel ***background;
    int i;

    background = (QLabel***)malloc(row*sizeof(QLabel**));
    for (i = 0; i<row; i++)
        *(background+i) = (QLabel**)malloc(column*sizeof(QLabel*));

    return background;
}

//该函数动态生成了 row*column 的二维组并对其初始化
//0 代表此方块尚未被翻开
int **FlagCreate(int row, int column)
{
    int **flag;
    int i, j;

    flag = (int**)malloc(row*sizeof(int*));
    for (i = 0; i<row; i++)
        *(flag+i) = (int*)malloc(column*sizeof(int));

    for (i = 0; i<row; i++)
        for (j = 0; j<column; j++)
            flag[i][j] = 0;

    return flag;
}

#endif // CREATE_H

////////////////////
/*!
 * detect.h
 * Detect 类的声明。
 * 本类用于鼠标左键点击方块时作出相应的回应，
 * 包括自动翻开空方块周围的方块、判断玩家输赢。
 */

#ifndef DETECT_H
#define DETECT_H

#include "pushbutton.h"

#include <QObject>

class Detect:public QObject
{
    Q_OBJECT
```

```
public:
    Detect();  //构造函数
    ~Detect();  //析构函数
    void Acquire(int r, int c, int rt, int ct, int **f, int **p, int w,
PushButton ***b);  //用于初始化相关数据
    int Check();  //用于判断当前玩家是否已赢得本局游戏
    void ResultDialog(int res);  //用于显示表示输赢的"结果"对话框

public slots:
    void React();  //用于在翻开一个方块时，根据其代表的地雷分布情况选择相应的措施
    void Restart();  //用于重启游戏

signals:
    void BottonHide();  //当翻开空方块时，发出此信号，促使周围非雷方块自动翻开
    void TimerControl(int w);  //当翻开第一个方块时，发出此信号使计时器开始工
作；当玩家赢或输时，发出此信号使计时器停止工作

private:
    int row, column;  //分别表示翻开的方块所在的行、列
    int RowTotal, ColumnTotal;  //分别表示雷区总行数、雷区总列数
    int **flag;  //表示整个雷阵方块被翻开的情况
    int **pointer;  //表示地雷分布图
    int WhetherStart;  //表示游戏是否已经开始（0 为未开始，1 为已开始）
    PushButton ***button;  //表示代表方块的按钮组
};

#endif // DETECT_H

////////////////////
/*!
 * mainwindow.h
 * MainWindow 类的声明。
 * 本类用于创建游戏界面框架，包括菜单栏、计时器、
 * 剩余旗数显示。
 */

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QLCDNumber>
#include <QTimer>

namespace Ui
{
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
```

```
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();  //析构函数
    void Acquire(int r, int c, int m);  //用于初始化相关数据

public slots:
    void TimeManage(int w);  //用于根据开始、结束信号而控制计时器
    void TimeShow();  //用于显示计时
    void FlagNum(int f);  //用于显示剩余的旗数
    void NewShow();  //用于重启游戏
    void HelpShow();  //用于显示"游戏玩法"
    void InfoShow();  //用于显示"关于扫雷"

private:
    Ui::MainWindow *ui;
    QAction *setAction;

    QLCDNumber *time_count, *flag_left;  //分别代表显示计时、剩余旗数的 LCD
    QTimer *time_counter;  //计时器
    int row, column, mines;  //分别代表雷区行数、雷区列数、雷数
    int start, end;  //分别代表计时开始时的时间、计时长度
    int JudgeStart;  //代表对是否已经开启计时器的判断（0 为未开启，1 为已开启）
};

#endif // MAINWINDOW_H

////////////////////
/*!
 * map.h
 * 此头文件用于创建记录雷阵信息的二维数组：
 * -1 代表雷，0~8 代表这一方块周围的雷数。
 */


#ifndef MAP_H
#define MAP_H

#include <malloc.h>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int **MapCreate(int r, int c, int m);   //创建记录雷阵信息的二维数组
int **MineCreate(int r, int c, int m);   //随机创建 r 行，c 列，m 个雷的原始雷阵数组（有雷区为-1，无雷区为 0）
int MineCount(int **pointer, int r, int c);    //统计位于第 r 行第 c 列的方块周围的雷数

/*!
 *思路：
 *通过调用 MineCreate、MineCount 来完成记录
 *雷阵信息的二维数组的创建。
 */
int **MapCreate(int r, int c, int m)
```

```
{
    int i, j;
    int **map, **mine;

    map = (int**)malloc(r*sizeof(int*));
    for (i = 0; i<r; i++)
        *(map+i) = (int*)malloc(c*sizeof(int)); //创建记录雷阵信息的二维数组

    mine = MineCreate(r, c, m);   //创建原始雷阵数组

    for (i = 0; i<r; i++)
        for (j = 0; j<c; j++)
            map[i][j] = MineCount(mine, i+1, j+1);  //统计每个方块周围的雷数并
赋值给记录雷阵信息的二维数组

    return map;
}

/*!
 * 思路:
 * 先建立长度为 r*c 的一维数组，均初始化为 0。
 * 再通过时间随机函数，将其中 m 个数字标为-1，代表被选中的
 * 地雷埋藏点。
 * 再采用取整、取余的方法将长为 r*c 的一维数组化为 r 行 c 列的
 * 二维数组。
 */
int **MineCreate(int r, int c, int m)
{
    int **pointer;
    int *temp;
    int i, j, t;

    srand(time(NULL));

    temp = (int*)malloc(r*c*sizeof(int));

    for (i = 0; i<r*c; i++)
        temp[i] = 0;              //先建立一维数组，方便初始随机化

    for (i = 0; i<m; i++)
    {
        t = rand()%(r*c);
        if (temp[t ] == -1)
            i--;
        else
            temp[t] = -1;         //将尚未被标成雷的区域赋值-1，表示在这里放置雷
    }

    pointer = (int**)malloc((r+2)*sizeof(int*)); // +2 是因为采用加边法，将
row*column 阵变成了
    for (i = 0; i<r+2; i++)                      //(row+2)*(column+2)阵，避
免对边界条件下
        *(pointer+i) = (int*)malloc((c+2)*sizeof(int));   //某个方格周围雷数
的繁琐讨论
```

```
    for (i = 0; i<r+2; i++)
        for (j = 0; j<c+2; j++)
            pointer[i][j] = 0;

    for (i = 0; i<r*c; i++)
        pointer[i/c+1][i%c+1] = temp[i];

    return pointer;
}

/*!
 * 思路：
 * 先判断值，若为-1，代表此处埋雷，直接返回-1，
 * 终止函数。
 * 若不为-1，代表此处无雷，开始计算周围雷数，返回
 * 周围雷数。
 */
int MineCount(int **pointer, int r, int c)
{
    int num = 0;
    if (pointer[r][c] == -1)
        return -1;
    else
    {
        if (pointer[r-1][c-1] == -1)
            num++;
        if (pointer[r-1][c] == -1)
            num++;
        if (pointer[r-1][c+1] == -1)
            num++;
        if (pointer[r][c-1] == -1)
            num++;
        if (pointer[r][c+1] == -1)
            num++;
        if (pointer[r+1][c-1] == -1)
            num++;
        if (pointer[r+1][c] == -1)
            num++;
        if (pointer[r+1][c+1] == -1)
            num++;
        return num;
    }
}

#endif // MAP_H

////////////////////
/*!
 * pushbutton.h
 * PushButton 类的声明。
 * 这是一个继承了 QPushbutton 的类。
 * 用于创造出新型按钮，以实现扫雷游戏中左键点击方块
 * 能够将其翻开，右键点击一次方块插上小旗、点击两次
 * 变成问号、点击三次恢复原有按钮的目的。
```

```
 */

#ifndef PUSHBUTTON_H
#define PUSHBUTTON_H

#include <QPushButton>

class PushButton : public QPushButton
{
    Q_OBJECT
public:
    explicit PushButton(QWidget *parent = 0);
    void Acquire(int t);  //用于初始化 times

protected:
    void mousePressEvent(QMouseEvent *e);  //用于重写 mousePressEvent 函数

signals:
    void LeftClicked();  //左键点击时，发出此信号
    void FlagChange(int f);  //右键点击使方块插上小旗或显示问号时，发送此信号，从
而使相应的显示装置改变显示数字

private:
    int times;  //方块上的标记（0 代表无标记，1 代表小旗，2 代表问号）
};

#endif // PUSHBUTTON_H

////////////////////
/*!
 * settings.h
 * Settings 类的声明。
 * 这个类用于在开始游戏前弹出“设置”对话框，提醒玩家
 * 设置游戏中雷区的长、宽以及雷数。
 */


#ifndef SETTINGS_H
#define SETTINGS_H

#include <QObject>

class Settings:public QObject
{
    Q_OBJECT

public:
    Settings();  //构造函数
    ~Settings();  //析构函数
    void Acquire(int r, int c, int m);  //用于初始化相应数据
    void SettingsShow();  //用于显示“设置”对话框
    int RowReturn();  //用于将 row 值传到外部
    int ColumnReturn();  //用于将 column 值传到外部
    int MinesReturn();  //用于将 column 值传到外部
```

```
public slots:
    void RowSetting(int r);  //用于设定 row 值
    void ColumnSetting(int c);  //用于设定 row 值
    void MinesSetting(int m);  //用于设定 row 值

private:
    int row, column, mines;  //分别代表雷区行数、雷区列数、雷数
};

#endif // SETTINGS_H

////////////////////
```

六、源文件代码

```
/*!
 * autowork.cpp
 * Autowork 类的实现。
 */

#include "pushbutton.h"
#include "autowork.h"

#include <QLabel>
#include <malloc.h>

Autowork::Autowork()
{

}

Autowork::~Autowork()
{

}

void Autowork::Acquire(int r, int c, int rt, int ct, int **f, int **p,
PushButton ***b)
{
    row = r;
    column = c;
    RowTotal = rt;
    ColumnTotal = ct;
    flag = f;
    pointer = p;
    button = b;
}

/*!
 * 思路:
 * 若对方块的所处位置分类讨论，太过繁琐（因为要考虑到边、角）。
 * 故新建立(RowTotal+2)*(ColumnTotal+2)阵，将原有方阵周围
 * 加上一圈空白的边，从而避开繁琐的分类讨论。
 */
```

```
int Autowork::Judge(int r, int c)
{
    int **tempmap;
    int **tempflag;
    int i, j;

    tempmap = (int**)malloc((RowTotal+2)*sizeof(int*)); // +2 是因为采用加
边法，将 row*column 阵变成了
    for (i = 0; i<RowTotal+2; i++)
//(RowTotal+2)*(ColumnTotal+2)阵，避免对边界条件下
        *(tempmap+i) = (int*)malloc((ColumnTotal+2)*sizeof(int));   //某个
方格周围雷数的繁琐讨论
    for (i = 0; i<RowTotal+2; i++)
        for (j = 0; j<ColumnTotal+2; j++)
            tempmap[i][j] = -1;
    for (i = 0; i<RowTotal; i++)
        for (j = 0; j<ColumnTotal; j++)
            tempmap[i+1][j+1] = pointer[i][j];

    tempflag = (int**)malloc((RowTotal+2)*sizeof(int*)); // +2 是因为采用
加边法，将 row*column 阵变成了
    for (i = 0; i<RowTotal+2; i++)
//(RowTotal+2)*(ColumnTotal+2)阵，避免对边界条件下
        *(tempflag+i) = (int*)malloc((ColumnTotal+2)*sizeof(int));   //某
个方格周围雷数的繁琐讨论
    for (i = 0; i<RowTotal+2; i++)
        for (j = 0; j<ColumnTotal+2; j++)
            tempflag[i][j] = 0;
    for (i = 0; i<RowTotal; i++)
        for (j = 0; j<ColumnTotal; j++)
            tempflag[i+1][j+1] = flag[i][j];

    if (tempmap[r][c] == 0&&tempflag[r][c] == 1)
        return 1;
    if (tempmap[r][c+1] == 0&&tempflag[r][c+1] == 1)
        return 1;
    if (tempmap[r][c+2] == 0&&tempflag[r][c+2] == 1)
        return 1;
    if (tempmap[r+1][c] == 0&&tempflag[r+1][c] == 1)
        return 1;
    if (tempmap[r+1][c+2] == 0&&tempflag[r+1][c+2] == 1)
        return 1;
    if (tempmap[r+2][c] == 0&&tempflag[r+2][c] == 1)
        return 1;
    if (tempmap[r+2][c+1] == 0&&tempflag[r+2][c+1] == 1)
        return 1;
    if (tempmap[r+2][c+2] == 0&&tempflag[r+2][c+2] == 1)
        return 1;

    return 0;
}

/*!
 * 思路：
 * 不断地扫描所有方块，遇到周围有空方块的非雷且尚未翻开的方块时，
```

```
 * 翻开该方块。直到所有应该被翻开的方块均已被翻开。
 */
void Autowork::Scan()
{
    int i, j;
    int whether = 0;
    while (1)
    {
        for (i = 0; i<RowTotal; i++)
            for (j = 0; j<ColumnTotal; j++)
                if (Judge(i, j) == 1&&pointer[i][j] != -1&&flag[i][j] ==
0)
                {
                    flag[i][j] = 1;
                    button[i][j]->hide();
                    whether = 1;
                }
        if (whether == 0)
            break;
        whether = 0;  //用以保障所有该被自动翻开的的方块均已自动翻开
    }
}

////////////////////
/*!
 * detect.h
 * Detect 类的实现。
 */

#include "pushbutton.h"
#include "detect.h"

#include <QDialog>
#include <QLabel>
#include <QApplication>
#include <QProcess>

Detect::Detect()
{

}

Detect::~Detect()
{

}

void Detect::Acquire(int r, int c, int rt, int ct, int **f, int **p,
int w, PushButton ***b)
{
    row = r;
    column = c;
    RowTotal = rt;
    ColumnTotal = ct;
    flag = f;
    pointer = p;
```

```
    WhetherStart = w;
    button = b;
}

int Detect::Check()
{
    int whether = 1;
    int i, j;

    for (i = 0; i<RowTotal; i++)
        for (j = 0; j<ColumnTotal; j++)
            if (pointer[i][j] != -1&&flag[i][j] == 0)
                whether = 0;

    return whether;
}

/*!
 * 思路:
 * 创建对话框，拥有一样的按钮，但是分输、赢两种情况
 * 来设置文本框内容。
 */
void Detect::ResultDialog(int res)
{
    int i, j;

    QDialog *result = new QDialog;
    QLabel *tell;
    QPushButton *restart, *quit;

    result->setFixedSize(320, 120);
    result->setWindowFlags(Qt::CustomizeWindowHint);  //隐藏对话框的边框

    for (i = 0; i<RowTotal; i++)
        for (j = 0; j<ColumnTotal; j++)
            {
                flag[i][j] = 1;
                button[i][j]->hide();
            }

    tell = new QLabel(result);
    if (res == 1)
        tell->setText(tr("恭喜，您赢了!"));
    else
        tell->setText(tr("抱歉，您输了!"));
    tell->setGeometry(100, 20, 120, 20);

    restart = new QPushButton(result);
    restart->setGeometry(70, 50, 85, 50);
    restart->setText(tr("重玩"));

    quit = new QPushButton(result);
    quit->setGeometry(165, 50, 85, 50);
    quit->setText(tr("结束"));
```

```
    QObject::connect(restart, SIGNAL(clicked()), this, SLOT(Restart()));
    QObject::connect(quit,&QPushButton::clicked,&QApplication::quit);

    tell->show();
    result->exec();
}

/*!
 * 思路：
 * 当该函数第一次被触发时，发送使计时器开始计时的信号。
 * 触到-1，代表触雷，玩家输，发送使计时器停止计时的信号，
 * 游戏结束，弹出“结果”对话框。
 * 触到 0，代表触到空方块，发送自动翻开方块的信号。
 * Check()为 1 时，玩家赢，发送使计时器停止计时的信号，
 * 游戏结束，弹出“结果”对话框。
 */
void Detect::React()
{
    flag[row][column] = 1;
    if (WhetherStart == 0)
    {
        emit TimerControl(0);
        WhetherStart = 1;
    }
    if (pointer[row][column] == -1)
    {
       emit TimerControl(1);
       ResultDialog(0);
    }
    if (pointer[row][column] == 0)
       emit BottonHide();
    if (Check() == 1)
    {
       emit TimerControl(1);
       ResultDialog(1);
    }
}

void Detect::Restart()
{
    QProcess *p = new QProcess(this);
    QString str = QApplication::applicationFilePath();
    p->startDetached(str, QStringList());
    QApplication::quit();
}

////////////////////
/*!
 * main.cpp
 * 思路：
 * 先通过“设置”对话框，将雷区初始化为 row 行，
 * column 列，mines 个雷。
 * 然后通过标签将雷阵信息的二维数组用图标表示，形成背景图。
 * 最后用按钮组覆盖背景图，并通过连接信号和槽使按钮组形成
 * 对鼠标左键单击、右键单击的反馈机制。
```

```
 */

#include "mainwindow.h"
#include "settings.h"
#include "pushbutton.h"
#include "detect.h"
#include "autowork.h"

#include "create.h"
#include "map.h"

#include <QApplication>
#include <QObject>
#include <QDesktopWidget>

int main(int argc, char *argv[])
{
    int row, column, mines;
    int i, j;
    int **mine_map;
    int **mine_flag;

    QApplication a(argc, argv);
    Settings set;
    MainWindow w;
    PushButton ***mine_area;
    QLabel ***mine_background;
    Detect unit[20][36];
    Autowork item[20][36];

    //!将雷区初始化
    set.Acquire(9, 9, 10);  //将 set 初始化
    set.SettingsShow();  //弹出"设置"窗口
    row = set.RowReturn();  //为 row 赋值
    column = set.ColumnReturn();  //为 column 赋值
    if (set.MinesReturn()>row*column)
        mines = row*column;  //为 mines 赋值
    else  //当返回的 mines 值不大于方块数时才用返回值
        mines = set.MinesReturn();

    w.Acquire(row, column, mines);  //将窗口初始化
    w.setFixedSize(40+50*column, 120+50*row);  //设置窗口尺寸

    mine_map = MapCreate(row, column, mines);
    mine_flag = FlagCreate(row, column);
    mine_area = ButtonCreate(row, column);
    mine_background = LabelCreate(row, column);

    //!形成雷阵信息背景图
    for (i = 0; i<row; i++)
           for (j = 0; j<column; j++)
           {
               mine_background[i][j] = new QLabel(&w);
               mine_background[i][j]->setGeometry(20+50*j, 100+50*i, 50,
50);
```

```
            if (mine_map[i][j] == -1)
            {
                QPixmap pix_mine(":/icon/mine");
                mine_background[i][j]->setPixmap(pix_mine);
            }
            if (mine_map[i][j] == 0)
            {
                QPixmap pix_num0(":/icon/num0");
                mine_background[i][j]->setPixmap(pix_num0);
            }
            if (mine_map[i][j] == 1)
            {
                QPixmap pix_num1(":/icon/num1");
                mine_background[i][j]->setPixmap(pix_num1);
            }
            if (mine_map[i][j] == 2)
            {
               QPixmap pix_num2(":/icon/num2");
               mine_background[i][j]->setPixmap(pix_num2);
            }
            if (mine_map[i][j] == 3)
            {
               QPixmap pix_num3(":/icon/num3");
               mine_background[i][j]->setPixmap(pix_num3);
            }
            if (mine_map[i][j] == 4)
            {
               QPixmap pix_num4(":/icon/num4");
               mine_background[i][j]->setPixmap(pix_num4);
            }
            if (mine_map[i][j] == 5)
            {
               QPixmap pix_num5(":/icon/num5");
               mine_background[i][j]->setPixmap(pix_num5);
            }
            if (mine_map[i][j] == 6)
            {
               QPixmap pix_num6(":/icon/num6");
               mine_background[i][j]->setPixmap(pix_num6);
            }
            if (mine_map[i][j] == 7)
            {
               QPixmap pix_num7(":/icon/num7");
               mine_background[i][j]->setPixmap(pix_num7);
            }
            if (mine_map[i][j] == 8)
            {
                QPixmap pix_num8(":/icon/num8");
                mine_background[i][j]->setPixmap(pix_num8);
            }
            mine_background[i][j]->show();
        }

//!创建按钮组并连接相应的信号和槽
for (i = 0; i<row; i++)
    for (j = 0; j<column; j++)
    {
```

```
            mine_area[i][j] = new PushButton(&w);
            mine_area[i][j]->Acquire(0);
            mine_area[i][j]->setGeometry(20+50*j, 100+50*i, 50, 50);

            unit[i][j].Acquire(i, j, row, column, mine_flag, mine_map, 0,
mine_area);
            item[i][j].Acquire(i, j, row, column, mine_flag, mine_map,
mine_area);

            QObject::connect(mine_area[i][j], SIGNAL(LeftClicked()),
&unit[i][j], SLOT(React()));
            QObject::connect(mine_area[i][j], SIGNAL(FlagChange(int)),
&w, SLOT(FlagNum(int)));
            QObject::connect(&unit[i][j], SIGNAL(BottonHide()),
&item[i][j], SLOT(Scan()));
            QObject::connect(&unit[i][j], SIGNAL(TimerControl(int)), &w,
SLOT(TimeManage(int)));
        }

    w.move((QApplication::desktop()->width()-w.width())/2,
(QApplication::desktop()->height()-w.height())/2);
    w.show();

    return a.exec();
}

////////////////////
/*!
 * mainwindow.cpp
 * MainWindow 类的实现。
 */

#include "mainwindow.h"
#include "ui_mainwindow.h"

#include <QAction>
#include <QMenuBar>
#include <QObject>
#include <QDialog>
#include <QLabel>
#include <QTime>
#include <QProcess>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    //ui->setupUi(this);  //因本程序尚未用到 UI 界面，故将其隐藏；但不删除，以备之
后拓展程序之需要

    //!为菜单栏添加动作
    QAction *new_action = new QAction(QIcon(":/icon/new"), tr("重玩"),
this);
    QObject::connect(new_action, SIGNAL(triggered()), this,
SLOT(NewShow()));
```

```
    QAction *settings_action = new QAction(QIcon(":/icon/settings"),
tr("设置"), this);
    QObject::connect(settings_action, SIGNAL(triggered()), this,
SLOT(NewShow()));

    QAction *help_action = new QAction(QIcon(":/icon/help"), tr("游戏玩法
"), this);
    QObject::connect(help_action, SIGNAL(triggered()), this ,
SLOT(HelpShow()));

    QAction *info_action = new QAction(QIcon(":/icon/info"), tr("关于扫雷
"), this);
    QObject::connect(info_action, SIGNAL(triggered()), this,
SLOT(InfoShow()));

    QMenu *Game = menuBar()->addMenu(tr("游戏"));
    Game->addAction(new_action);
    Game->addAction(settings_action);

    QMenu *Help = menuBar()->addMenu(tr("帮助"));
    Help->addAction(help_action);
    Help->addAction(info_action);
}

MainWindow::~MainWindow()
{
    delete ui;
}

/*!
 * 思路:
 * 当该类接收到数据后，根据数据创建 LCD 显示窗。
 */
void MainWindow::Acquire(int r, int c, int m)
{
    row = r;
    column = c;
    mines = m;

    JudgeStart = 0;
    time_counter = new QTimer(this);
    connect(time_counter, SIGNAL(timeout()), this, SLOT(TimeShow()));
//实现每隔一段时间触发一次 TimeShow()

    time_count = new QLCDNumber(this);
    time_count->setGeometry(20, 50, 100, 40);
    time_count->setSegmentStyle(QLCDNumber::Filled);
    time_count->display(0);
    time_count->show();

    QLabel *time_icon = new QLabel(this);
    time_icon->setGeometry(145, 45, 50, 50);
    QPixmap pix_time(":/icon/time");
    time_icon->setPixmap(pix_time);
```

```
    flag_left = new QLCDNumber(this);
    flag_left->setSegmentStyle(QLCDNumber::Filled);
    flag_left->show();

    QLabel *flag_icon = new QLabel(this);
    flag_icon->setGeometry(50*column-150, 45, 50, 50);
    QPixmap pix_flag(":/icon/flagleft");
    flag_icon->setPixmap(pix_flag);

    flag_left->setGeometry(50*column-80, 50, 100, 40);
    flag_left->display(mines);
}

/*!
 * 思路:
 * 开启计时器时，记录当前时间。
 * 结束计时器时，断开计时器的信号和槽的连接，从而停止计时。
 */
void MainWindow::TimeManage(int w)
{
    if (w == 0&&JudgeStart == 0)
    {
        JudgeStart = 1;
        time_counter->start(1000);
        QTime TimeStart = QTime::currentTime();
        start = TimeStart.minute()*60+TimeStart.second();
    }
    else if (w == 1)
        time_counter->disconnect(this);
}

/*!
 * 思路:
 * 每隔一秒这个函数就会被触发一次，显示计时长度。
 */
void MainWindow::TimeShow()
{
    QTime TimeEnd = QTime::currentTime();
    end = TimeEnd.minute()*60+TimeEnd.second()-start;
    time_count->display(end);
}

/*!
 * 思路:
 * f 为-1，代表插上了一个小旗，使剩余小旗的数量 mines 减 1。
 * f 为 1，代表有一个小旗被替换成了问号，剩余小旗的数量增加了一个。
 */
void MainWindow::FlagNum(int f)
{
    mines = mines+f;
    flag_left->display(mines);
}

void MainWindow::NewShow()
{
```

```
    QProcess *p = new QProcess(this);
    QString str = QApplication::applicationFilePath();
    p->startDetached(str, QStringList());
    close();
}

void MainWindow::HelpShow()
{
    QDialog *help = new QDialog;
    QLabel *help_text = new QLabel;

    help->setWindowTitle(tr("游戏玩法"));
    help->setWindowIcon(QIcon(":/icon/help"));
    help->setFixedSize(762, 627);

    help_text = new QLabel(help);
    QPixmap pix_help_text(":/icon/helptext");
    help_text->setPixmap(pix_help_text);

    help_text->show();
    help->exec();
}

void MainWindow::InfoShow()
{
    QDialog *info = new QDialog;
    QLabel *info_text = new QLabel;

    info->setWindowTitle(tr("关于扫雷"));
    info->setWindowIcon(QIcon(":/icon/info"));
    info->setFixedSize(480, 135);

    info_text = new QLabel(info);
    QPixmap pix_info_text(":/icon/infotext");
    info_text->setPixmap(pix_info_text);

    info_text->show();
    info->exec();
}

////////////////////
/*!
 * pushbutton.cpp
 * PushButton 类的实现。
 */

#include "pushbutton.h"

#include <QtGui>

PushButton::PushButton(QWidget *parent) :
    QPushButton(parent)
{

}
```

```
void PushButton::Acquire(int t)
{
    times = t;
}

/*!
 * 思路:
 * 重写 mousePressEvent 函数。
 * 单击鼠标左键时发出左键被击的信号，并隐藏相应的按钮。
 * 单击鼠标右键时，利用 times 判断按钮究竟该显示什么图标。
 */
void PushButton::mousePressEvent(QMouseEvent *e)
{
    if(e->button() == Qt::LeftButton&&times != 1)
    {
        emit LeftClicked();
        this->hide();
    }
    if(e->button() == Qt::RightButton)
    {
        if (times == 0)
        {
            times = 1;
            this->setIcon(QIcon(":/icon/flag"));
            emit FlagChange(-1);  //发送信号，告知剩余小旗数应该减 1
        }
        else if (times == 1)
        {
            times = 2;
            this->setIcon(QIcon(":/icon/unsure"));
            emit FlagChange(1);  //发送信号，告知剩余小旗数应该加 1
        }
        else if (times == 2)
        {
            times = 0;
            this->setIcon(QIcon(":/icon/clear"));  //实际上是一张透明图片
        }
    }

    QPushButton::mousePressEvent(e);
}

////////////////////
#include "settings.h"

#include <QDialog>
#include <QLabel>
#include <QSpinBox>
#include <QPushButton>
#include <QGridLayout>

Settings::Settings()
{

}
```

```
Settings::~Settings()
{

}

void Settings::Acquire(int r, int c, int m)
{
    row = r;
    column = c;
    mines = m;
}

/*!
 * 思路:
 * 创建对话框、标签、SpinBox、确认按钮，并将其放置在
 * 网格布局管理器。
 */
void Settings::SettingsShow()
{
    QDialog *settings;
    QLabel *label_row, *label_column, *label_mines;
    QSpinBox *settings_row, *settings_column, *settings_mines;
    QPushButton *confirm;
    QGridLayout *layout;

    settings = new QDialog;
    settings->setWindowTitle(tr("设置"));
    settings->setWindowIcon(QIcon(":/icon/settings"));

    label_row = new QLabel(settings);
    label_row->setText(tr("行数(9-20)"));

    label_column = new QLabel(settings);
    label_column->setText(tr("列数(9-36)"));

    label_mines = new QLabel(settings);
    label_mines->setText(tr("雷数"));

    settings_row = new QSpinBox(settings);
    settings_row->setRange(9, 20);
    settings_row->setValue(9);
    QObject::connect(settings_row, SIGNAL(valueChanged(int)), this,
SLOT(RowSetting(int)));

    settings_column = new QSpinBox(settings);
    settings_column->setRange(9, 36);
    settings_column->setValue(9);
    QObject::connect(settings_column, SIGNAL(valueChanged(int)), this,
SLOT(ColumnSetting(int)));

    settings_mines = new QSpinBox(settings);
    settings_mines->setRange(1, 720);
    settings_mines->setValue(10);
    QObject::connect(settings_mines, SIGNAL(valueChanged(int)), this,
SLOT(MinesSetting(int)));
```

```
    confirm = new QPushButton(settings);
    confirm->setText(tr("确定"));
    QObject::connect(confirm, SIGNAL(clicked()), settings,
SLOT(close()));

    layout = new QGridLayout;
    layout->addWidget(label_row, 0, 0);
    layout->addWidget(label_column, 1, 0);
    layout->addWidget(label_mines, 2, 0);
    layout->addWidget(settings_row, 0, 1);
    layout->addWidget(settings_column, 1, 1);
    layout->addWidget(settings_mines, 2, 1);
    layout->addWidget(confirm, 3, 0, 1, 2);

    settings->setLayout(layout);

    settings->exec();
}

int Settings::RowReturn()
{
    return row;
}

int Settings::ColumnReturn()
{
    return column;
}

int Settings::MinesReturn()
{
    return mines;
}

void Settings::RowSetting(int r)
{
    row = r;
}

void Settings::ColumnSetting(int c)
{
    column = c;
}

void Settings::MinesSetting(int m)
{
    mines = m;
}

////////////////////
```