# “五子棋程序”实验报告

# 实验一：

## 基于easyx的五子棋程序

### 【实验目的】

1. 参考www.easx.cn上教程，实现一个五子棋程序。

2. 棋盘可以自己画，对弈程序请自行在网上找相关代码。

3.在实验楼上提交报告，代码群里由班发统一提交。

### 【实验内容】

**1. 安装easyx:下载地址**

安装界面


**2. 代码片段**

- 各类头文件及自定义函数

```
#include "stdio.h"
#include "graphics.h"
#include "conio.h"
#include "windows.h"

void init_system();//初始化系统
void close_system();//关闭系统
```

```
void init_globales();//初始化参数

void play_chess();//开始五子棋游戏
void gamestart();//开始下棋
int  count_xyk_by_message(MOUSEMSG *m, int *x, int *y, int *a, int *b);//计算鼠标所在区域状态
int  judge(int a, int b, int c, int chess[][15]);//判断胜负条件
void regret(int x2, int y2, int chess[15][15], int chessbox[][2]);//悔棋

void draw_chessmen(int x, int y, int num, int a, int b, int chess[][15]);//绘制黑白棋子
void draw_chessboard();//画棋盘、菜单、状态框，画标题
void text();//绘制标题
void draw_menu();//绘制菜单
void over_black();//绘制黑棋胜字样
void over_white();//绘制白棋胜字样
void position();//绘制状态栏及内容
void text_black();//绘制黑棋下字样
void text_white();//绘制白棋下字样
void draw_regret(int m, int n, int x2, int y2);//所有悔棋情况
void draw_re(int x2, int y2);//绘制悔棋棋子
```

- 绘制棋盘

```
void draw_chessboard()
{
    int i;
//  loadimage(NULL, "E:\\个人\\PICTURE\\123.jpg");
    setbkcolor(BLACK);
    cleardevice();/*用背景色清空屏幕*/
    setlinestyle(PS_SOLID, 1);
    setlinecolor(WHITE);
    rectangle(m_x_box, m_y_box, m_x_box1, m_y_box1);
    setfillcolor(RGB(172, 81, 24));
    setfillstyle(BS_SOLID);
    floodfill((m_x0 + m_x01) / 2, (m_y0 + m_y01) / 2, WHITE);

    setlinestyle(PS_SOLID, 2);
    setlinecolor(BLACK);
    rectangle(m_x0 - 4, m_y0 - 4, m_x01 + 4, m_y01 + 4);

    setlinestyle(PS_SOLID, 1);
    setlinecolor(BLACK);
    rectangle(m_x0, m_y0, m_x01, m_y01);

    for (i = m_x0 + m_wh; i < m_x01; i += m_wh)//绘制交叉线
    {
        line(i, m_y0, i, m_y01);
    }
    for (i = m_y0 + m_wh; i < m_y01; i += m_wh)
    {
        line(m_x0, i, m_x01, i);
    }

    setlinecolor(BLACK);
    setfillcolor(BLACK);/*绘制画笔颜色以及填充颜色*/

    fillcircle(m_x0 + m_wh * 3, m_y0 + m_wh * 3, 3);//绘制星位
    fillcircle(m_x0 + m_wh * 11, m_y0 + m_wh * 3, 3);
    fillcircle(m_x0 + m_wh * 3, m_y0 + m_wh * 11, 3);
```

```
    fillcircle(m_x0 + m_wh * 11, m_y0 + m_wh * 11, 3);
    fillcircle(m_x0 + m_wh * 7, m_y0 + m_wh * 7, 3);

    draw_menu();/*绘制菜单*/
    position();/*绘制状态框*/
    text();/*绘制标题*/
}
```

- 绘制棋子

```
void draw_chessmen(int x, int y, int num, int a, int b, int chess[][15])
{
    if (num % 2 == 0)
    {
        setlinecolor(BLACK);
        setfillcolor(BLACK);
        fillcircle(x, y, 15);
        chess[a][b] = NUM_BLACK;
    }

    if (num % 2 != 0)
    {
        setlinecolor(WHITE);
        setfillcolor(WHITE);
        fillcircle(x, y, 15);
        chess[a][b] = NUM_WHITE;
    }
}
```

- 判断胜负

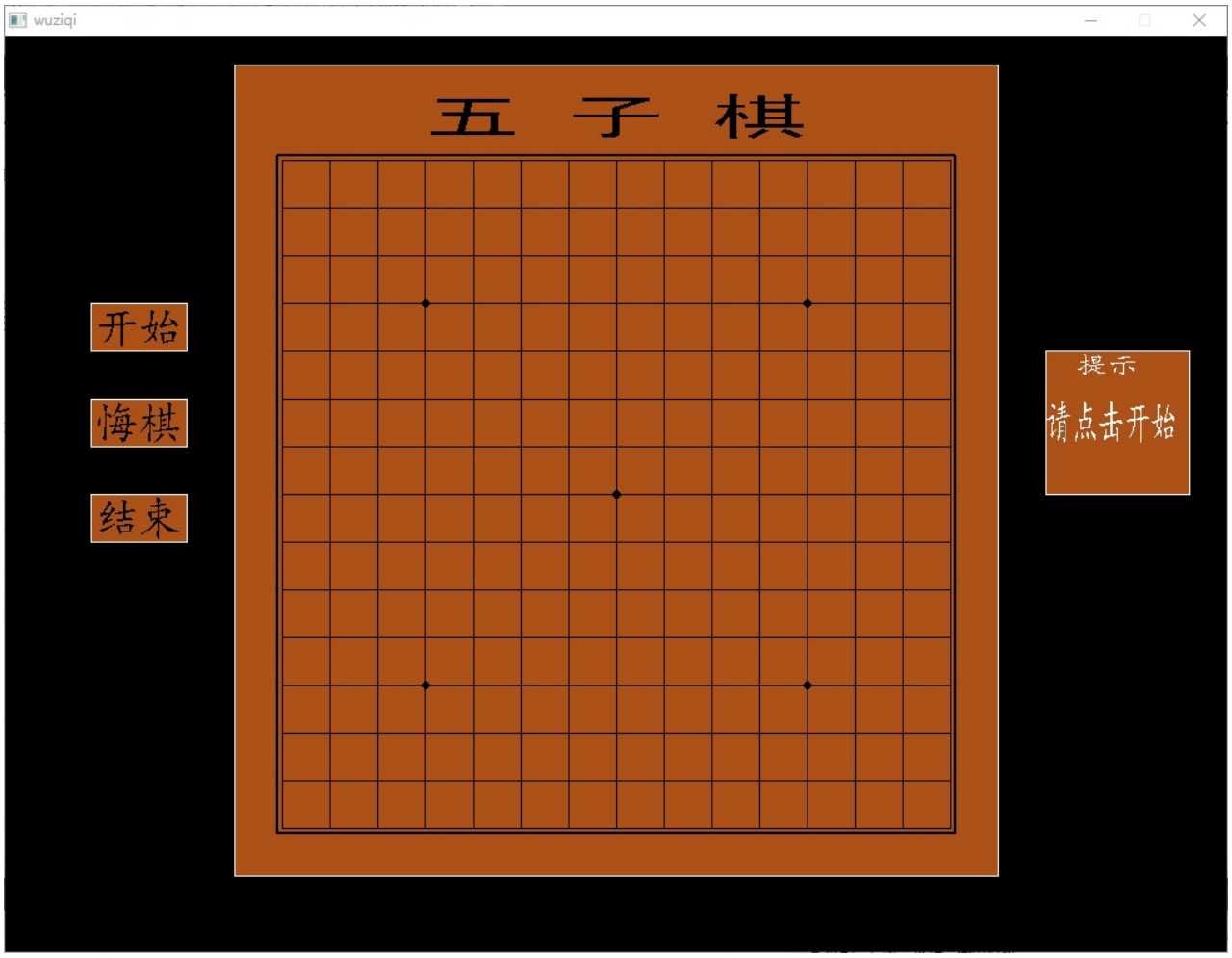
```
int  judge(int a, int b, int c, int chess[][15])
{
    int i, j, n1, n2; //i 表示行，j表示列
    while (1)
    {
        n1 = 0;
        n2 = 0;
        /*水平向左统计同种颜色棋子个数*/
        for (i = a, j = b; j >= 0; j--)
        {
            if (chess[i][j] == c)
                n1++;
            else
                break;
        }
        /*水平向右统计同种颜色棋子个数*/
        for (i = a, j = b; j <= m_row1; j++)
        {
            if (chess[i][j] == c)
                n2++;
            else
                break;
        }
        if (n1 + n2 - 1 >= 5)
```

```
        {
            return(1);
            break;
        }
        /*垂直向上统计同种颜色棋子个数*/
        n1 = 0;
        n2 = 0;
        for (i = a, j = b; i >= 0; i--)
        {
            if (chess[i][j] == c)
                n1++;
            else
                break;
        }
        /*垂直向下统计同种颜色棋子个数*/
        for (i = a, j = b; i <= m_row1; i++)
        {
            if (chess[i][j] == c)
                n2++;
            else
                break;
        }
        if (n1 + n2 - 1 >= 5)
        {
            return(1);
            break;
        }
        /*向左上方统计同种颜色棋子个数*/
        n1 = 0;
        n2 = 0;
        for (i = a, j = b; i >= 0, j >= 0; i--, j--)
        {
            if (chess[i][j] == c)
                n1++;
            else
                break;
        }
        /*向右下方统计同种颜色棋子个数*/
        for (i = a, j = b; i <= m_row1, j <= m_row1; i++, j++)
        {
            if (chess[i][j] == c)
                n2++;
            else
                break;
        }
        if (n1 + n2 - 1 >= 5)
        {
            return(1);
            break;
        }
        /*向右上方统计同种颜色棋子个数*/
        n1 = 0;
        n2 = 0;
        for (i = a, j = b; i >= 0, j <= m_row1; i--, j++)
        {
            if (chess[i][j] == c)
                n1++;
            else
                break;
        }
```

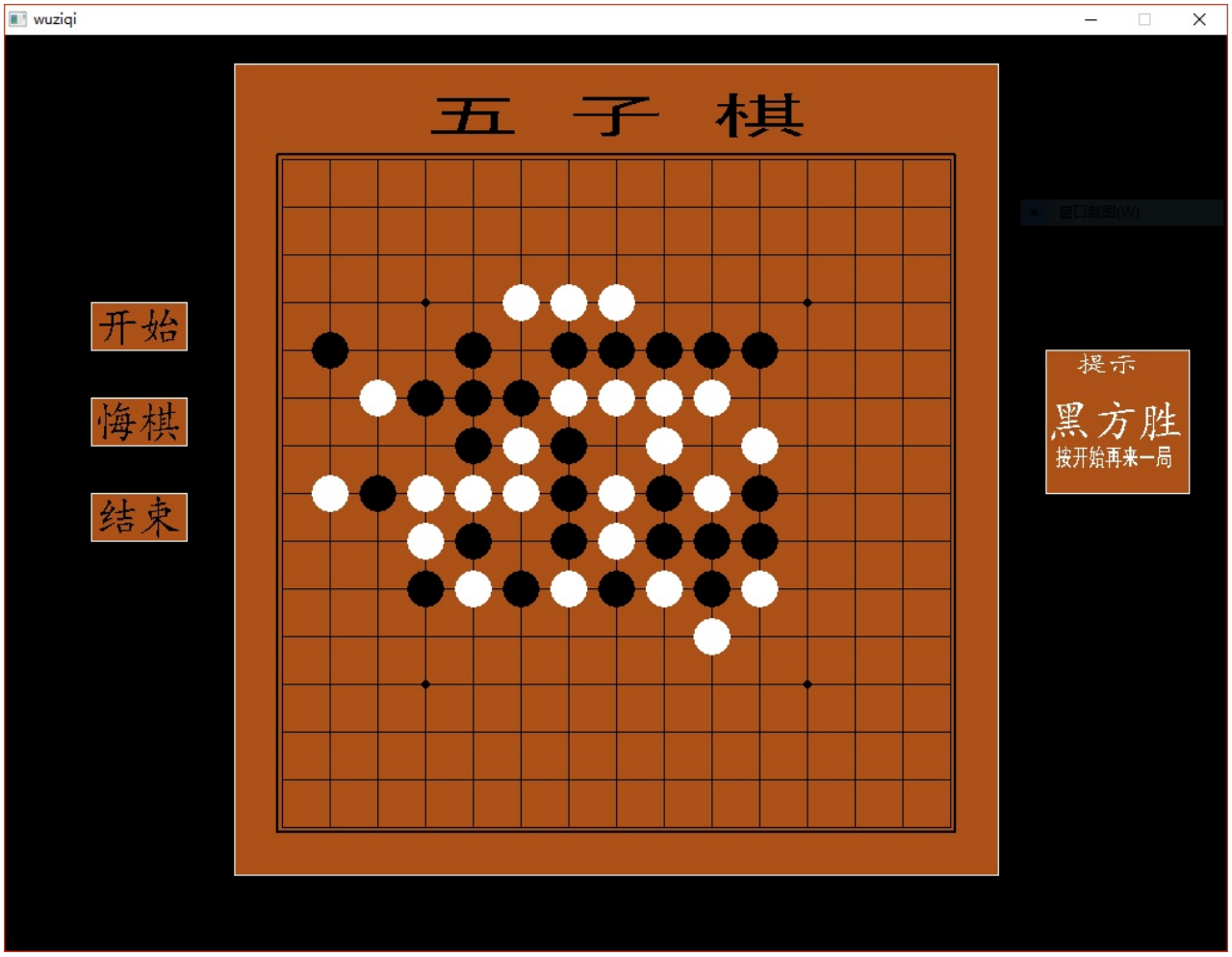
```
        /*向左下方统计同种颜色棋子个数*/
        for (i = a, j = b; i <= m_row1, j >= 0; i++, j--)
        {
            if (chess[i][j] == c)
                n2++;
            else
                break;
        }

        if (n1 + n2 - 1 >= 5)
        {
            return(1);
            break;
        }
        return(0);
        break;
    }
}
```

**3.测试结果**

- 开始界面测试




- 输赢界面测试




**4.参考博客及源码：**

https://blog.csdn.net/qq_41728697/article/details/80990917

# 实验二：

## 基于Qt的五子棋程序

### 【实验目的】

i. 借用Qt开发平台编写能在Linux系统中运行的五子棋程序。
ii. 在ubuntu下安装Qt。

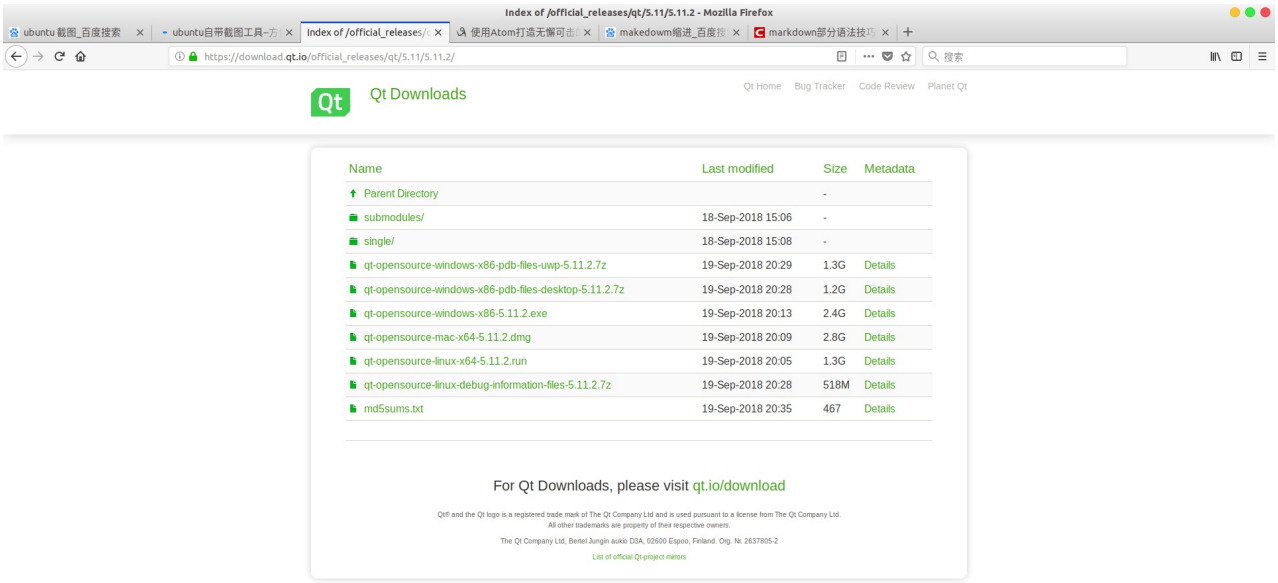
iii. 测试五子棋程序。

## 【实验内容】

**1. 安装Qt:**

安装地址

打开网址显示如下界面：

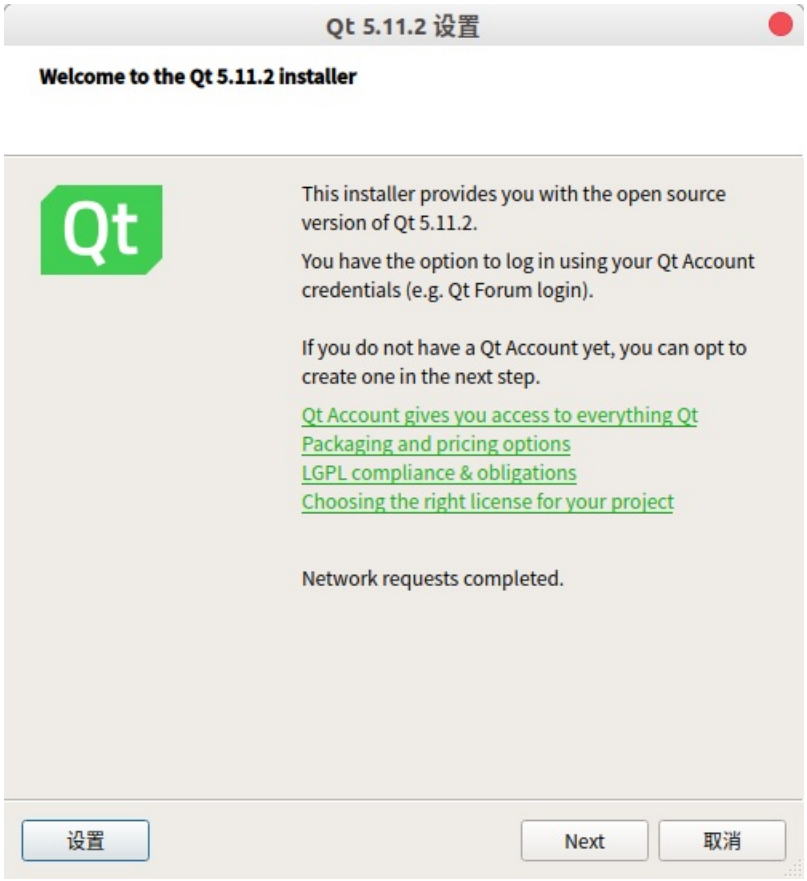



Linux系统选择那个.run安装文件。(我这里选择qt-opensource-linux-x64-5.11.2.run)

下载完成后cd到qt-opensource-linux-x64-5.11.2.run所在的目录下.

在命令端口中输入安装命令：sudo ./qt-opensource-linux-x64-5.11.2.run

弹出如下界面：

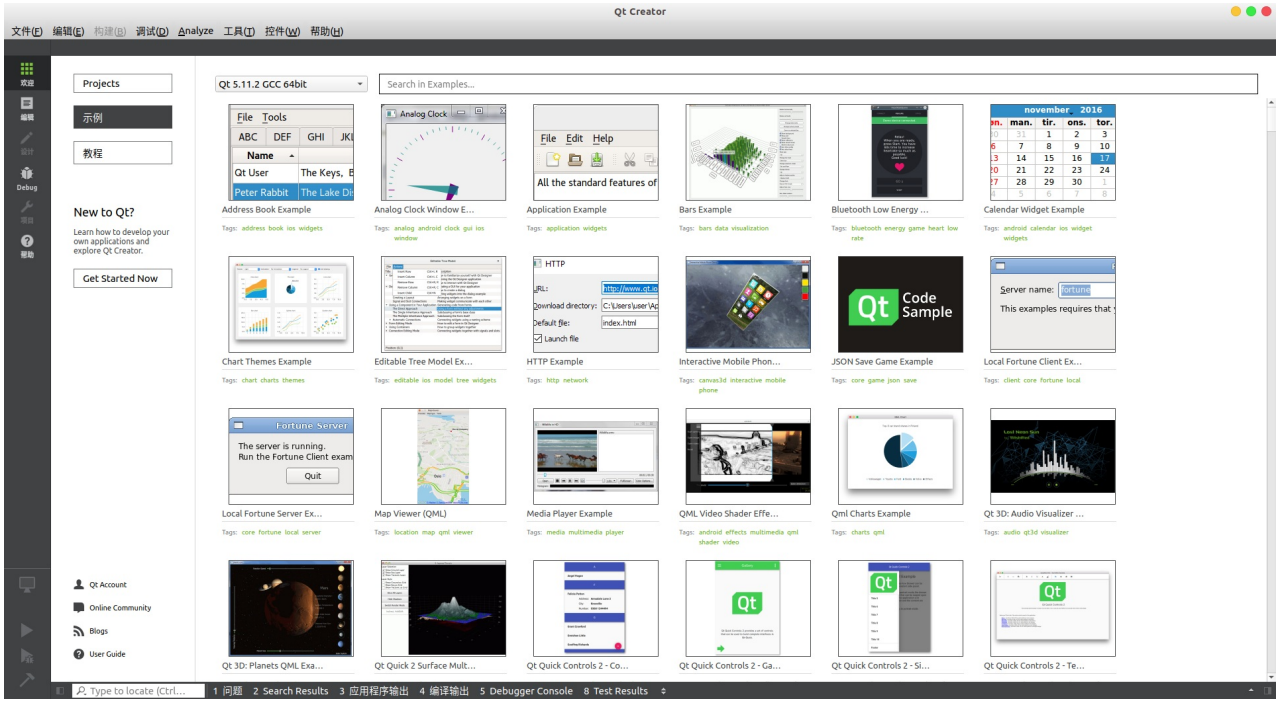



按照提示一步步安装完成。

安装 libgl1-mesa-dev

在命令端口中输入命令：sudo apt-get install libgl1-mesa-dev 进行安装。

打开Qt Creator如图：




**2. 代码片段**

- 棋子类Item.h：

  包含一个QPoint圆心坐标，

  和一个bool变量，

  代表是黑方还是白方

```
#pragma once
#include "qvector.h"

class Item
{
public:
    Item(void);
    Item(QPoint pt,bool bBlack);
    ~Item(void);

    bool operator==(const Item &t1)const
    {
        return (m_pt == t1.m_pt) && (m_bBlack == t1.m_bBlack);
    }

    QPoint m_pt;
    bool m_bBlack;
};
```

- MainWindow主窗口类

  QVector m_items 保存所有棋子。

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "Item.h"
#include "qmap.h"

namespace Ui {
class MainWindow;
}

#define CHESS_ROWS      15
#define CHESS_COLUMES   15
#define RECT_WIDTH      50
#define RECT_HEIGHT     50

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

protected:
    void paintEvent(QPaintEvent *);
    void mousePressEvent(QMouseEvent *);
private:
    void DrawChessboard();
    void DrawItems();
    void DrawItemWithMouse();

    void DrawChessAtPoint(QPainter& painter,QPoint& pt);

    int CountNearItem(Item item,QPoint ptDirection);    //统计某个方向(共8个方向)上的相连个数，用QPoint表示统计方向，如(1,1)表示右下方，(-1,0）表示向左
    //bool FindItem(Item item);
private:
    Ui::MainWindow *ui;

    QVector<Item> m_items;
    bool m_bIsBlackTurn;    //当前该黑棋下
};

#endif // MAINWINDOW_H
```

- 判断是否五子连的算法：（统计某个棋子item在某个方向ptDirection上，相邻的同色棋子数目）

  方向用QPoint来表示，是取8个方向上1个单位坐标的点，例如：向上（0,1）、向右上（1,1）、向右（1,0）等

```
int MainWindow::CountNearItem(Item item,QPoint ptDirection)
{
    int nCount = 0;
    item.m_pt += ptDirection;

    while (m_items.contains(item))
```

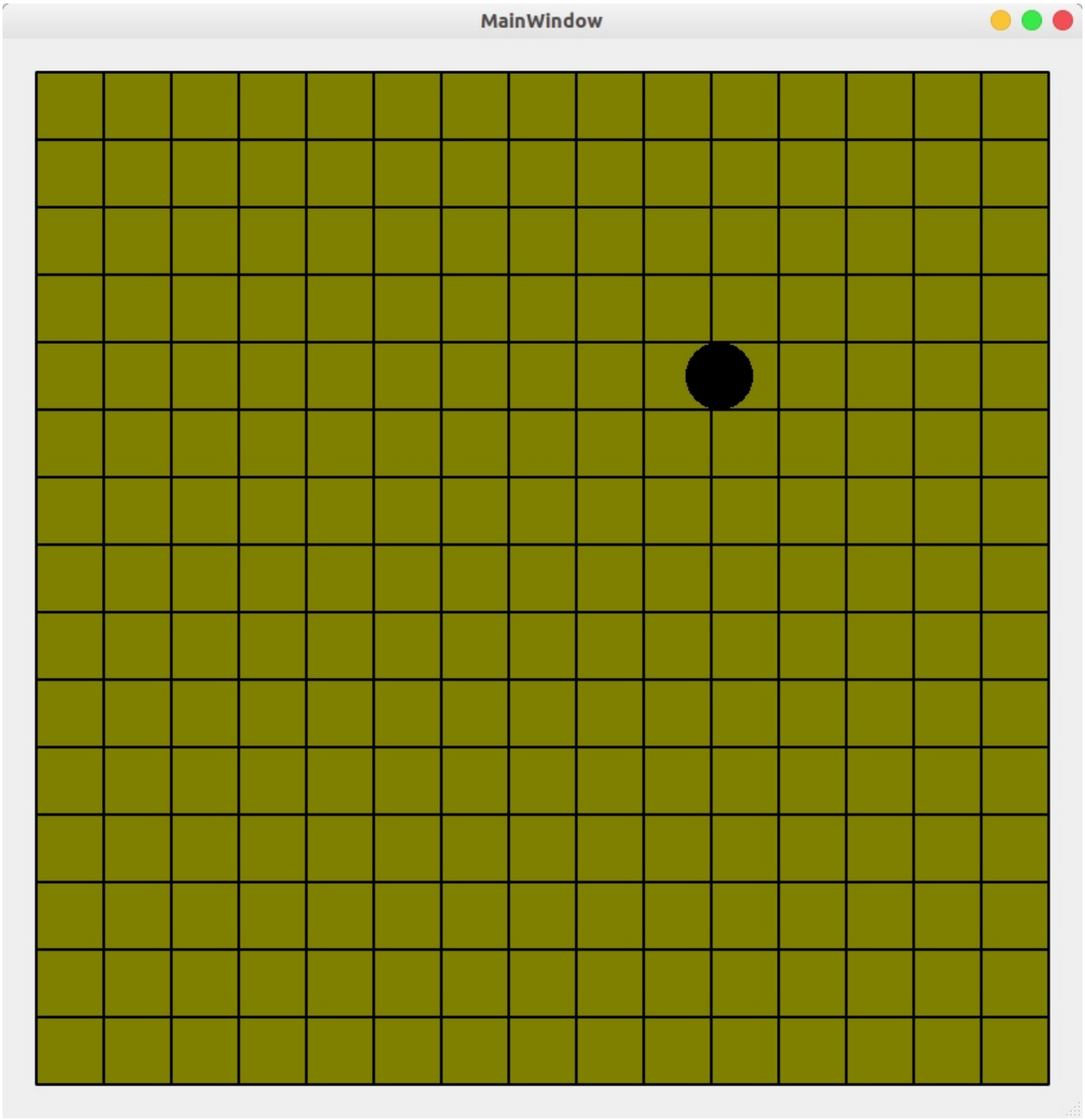
```
    {
        nCount++;
        item.m_pt += ptDirection;
    }
    return nCount;
}
```

- 鼠标点击消息：

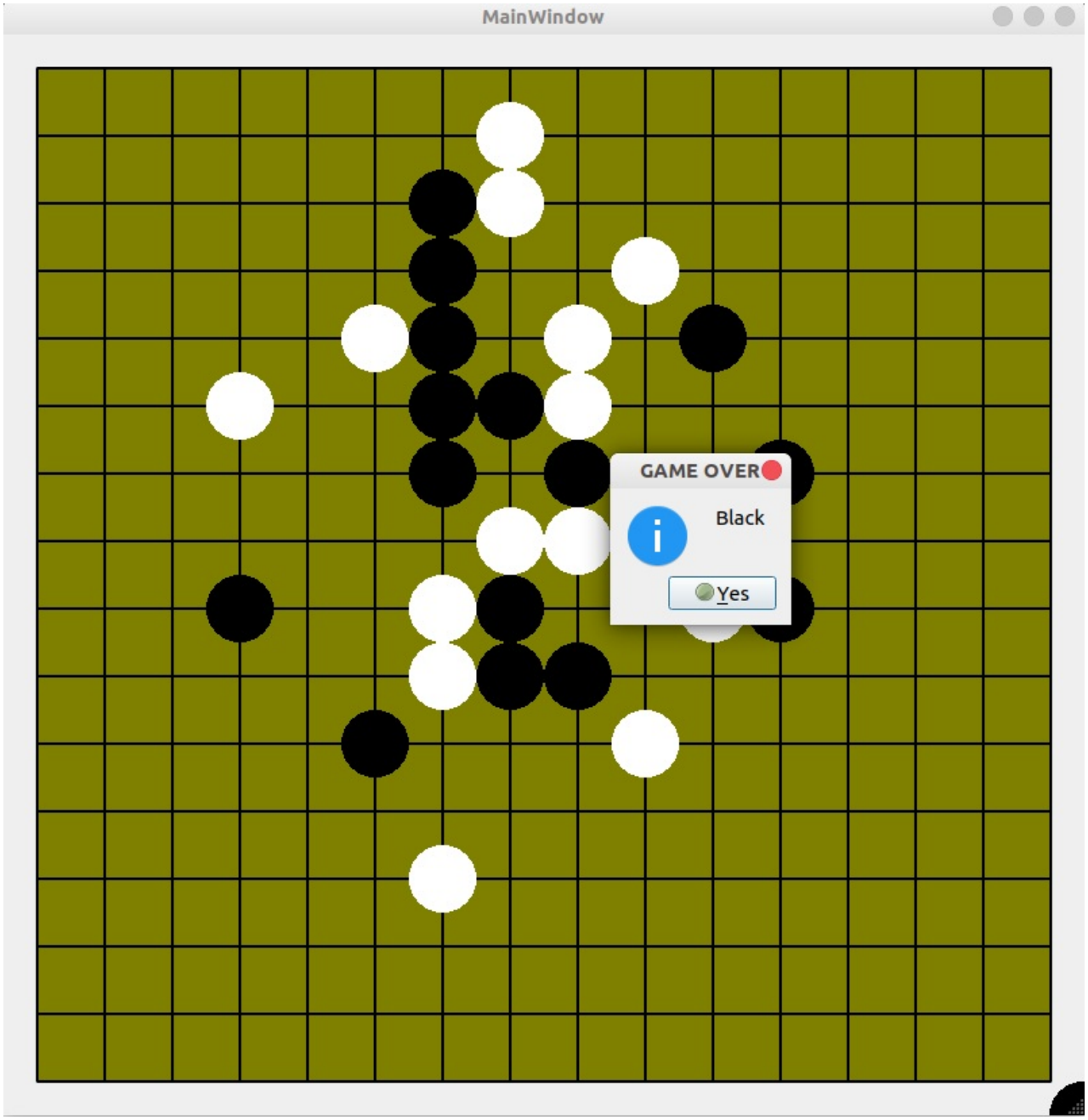
```
 void MainWindow::mousePressEvent(QMouseEvent * e)
{
    //求鼠标点击处的棋子点pt
    QPoint pt;
    pt.setX( (e->pos().x() ) / RECT_WIDTH);
    pt.setY( (e->pos().y() ) / RECT_HEIGHT);

    //如果已存在棋子，就什么也不做
    for (int i = 0; i<m_items.size(); i++)
    {
        Item item = m_items[i];
        if (item.m_pt == pt)
        {
            //已有棋子
            return;
        }
    }
    //不存在棋子，就下一个
    Item item(pt,m_bIsBlackTurn);
    m_items.append(item);

    //统计4个方向是否五子连
    int nLeft =         CountNearItem(item,QPoint(-1,0));
    int nLeftUp =       CountNearItem(item,QPoint(-1,-1));
    int nUp =           CountNearItem(item,QPoint(0,-1));
    int nRightUp =      CountNearItem(item,QPoint(1,-1));
    int nRight =        CountNearItem(item,QPoint(1,0));
    int nRightDown =    CountNearItem(item,QPoint(1,1));
    int nDown =         CountNearItem(item,QPoint(0,1));
    int nLeftDown =     CountNearItem(item,QPoint(-1,1));
    if ( (nLeft + nRight) >= 4 ||
         (nLeftUp + nRightDown) >= 4 ||
         (nUp + nDown) >= 4 ||
         (nRightUp + nLeftDown) >= 4 )
    {
        QString str = m_bIsBlackTurn?"Black":"White";
        QMessageBox::information(NULL,  "GAME OVER",str, QMessageBox::Yes , QMessageBox::Yes);
        m_items.clear();
        //NewGame();
        return;
    }
    //该另一方下棋了
    m_bIsBlackTurn = !m_bIsBlackTurn;
}
```

#### 3.测试结果

- 开始界面测试




- 输赢界面测试




**4.参考博客及源码：**

https://blog.csdn.net/dpsying/article/details/53770148

# 心得总结：

MATLAB老师布置C++作业我起初是很不理解的，而且我并没有学过C++。后来我在网上找来easyx库编写的五子棋程序，在vc++6.0上运行成功后，觉得easyx库并不够强大，不仅只能在vc和vs上使用，而且不能跨平台使用。于是我就在网上寻找怎样在Linux系统上解决图形化编程。网上的答案有两个一种是Qt，另一种是gtk。而gtk依赖程序可能比较繁琐，且到现在我也没运行成功过gtk的五子棋程序。

Qt不仅可以跨品台使用而且可以在嵌入式设备上使用，解决嵌入式图形编程，对于自动化专业，嵌入式这个

词很敏感。这次的实验程序可能都是直接拷贝别人的成果。一方面自己在C++领域现在可能真的很弱，一方面也想多联系一下让自己自学了半年的Linux。

最后我还想说一下写实验报告的过程。因为第一次用实验楼，我发现这里的平台与linux关系很大，让我感觉很亲切。开始写报告的时候我才知道实验报告要用markdown语法，然而在实验楼的环境里我插入图片一直失败，我便在网上找markdown编辑器，下载了markdownpad2用来将图片转为链接解决了图片插入诗派的问题。刚开始我在marxi.co上编写，后来找到atom插件的方法，我便转到atom编辑器上编辑。又查了一些markdown语法最终写完了这篇报告。感觉收获很多，觉得大学生还是要多自学点东西，不要天天打游戏的好。