用 Qt 设计自己的方块游戏

这个教程介绍方块游戏类的使用。教程共分三个部分。第一部分是简介，让你用最简单的办法实现自己的方块游戏。第二部分是功能展示，这里通过一个例子对这个类的所有功能进行了演示。第三部分是游戏分析，通过对接口函数的展示和对整个游戏设计流程的分析，帮助你更好的理解源码。

需要说明的是：我们下面使用的编程环境是基于 Qt 4.6的 Qt Creator 1.3.0 Windows 版本。这个类包含的四个文件 myitem.cpp， myitem.h， gamearea.cpp， gamearea.h 会和本教程打包在一起，你也可以到网上下载。

下载地址可以到我的博客 http://hi.baidu.com/yafeilinux 进行查看。

基于这个类我已经写了一个范例游戏：俄罗斯方块劳拉版

可以到 csdn下载：http://download.csdn.net/source/1866707

或到 qt论坛下载：http://www.qtcn.org/bbs/read.php?tid=24169

第一部分：简介

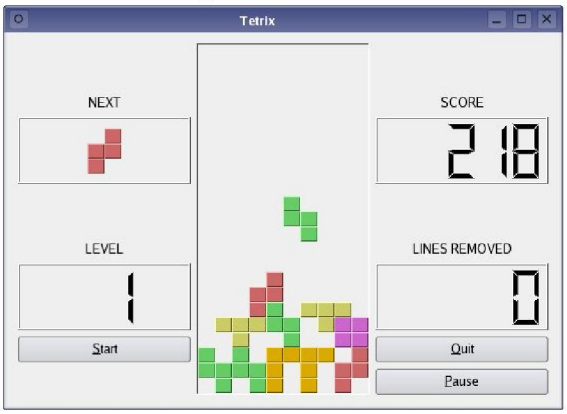
其实在 Qt Creator 中已经有了俄罗斯方块的例子，大家可以在帮助中搜索 Tetrix 进行查看。其内容如下：

Tetrix Example

iles:

- widgets/tetrix/tetrixboard.cpp
- widgets/tetrix/tetrixboard.h
- widgets/tetrix/tetrixpiece.cpp
- widgets/tetrix/tetrixpiece.h
- widgets/tetrix/tetrixwindow.cpp
- widgets/tetrix/tetrixwindow.h
- widgets/tetrix/main.cpp
- widgets/tetrix/tetrix.pro

he Tetrix example is a Qt version of the classic Tetrix game.



但是对于初学者，这个例子并不是那么容易就能看懂。所以我结合这个例子和网上的一些资料，写了一个比较简单的方块游戏类。希望能帮助初学者更好的理解这个例子和写出自己的方块游戏。

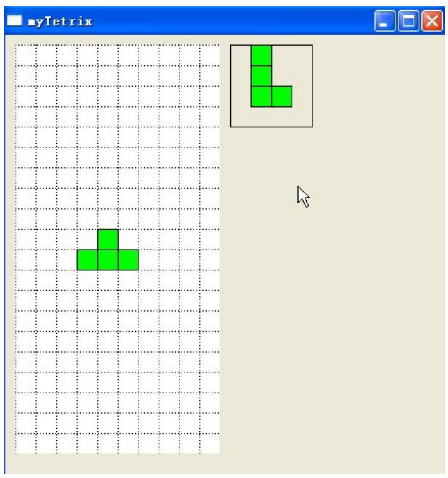
我这里已经对所有的功能函数进行了整理，最简单的，你只需要定义一个对象就能让这个游戏运行起来。

写最简单的游戏

1. 新建 Empty Qt4 Project，我们这里命名为 **myTetrix** 。
2. 向工程里添加新的普通文本文件，命名为 main.cpp 。
3. 将 myitem.cpp， myitem.h， gamearea.cpp， gamearea.h 四个文件复制到工程文件夹下。
4. 将上面四个文件加入到工程中。
5. 将 main.cpp 的内容更改如下：

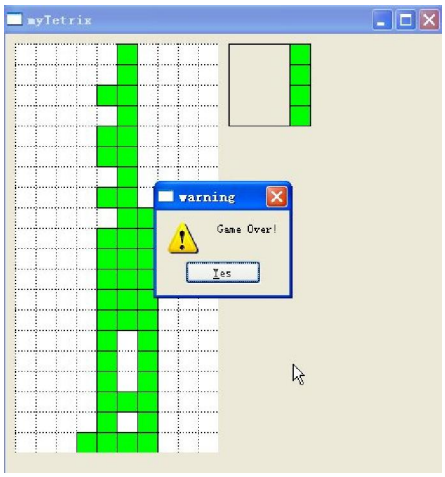
```
#include <QtGui/QApplication>

#include "gamearea.cpp"

int main(int argc,char* argv[])

{

    QApplication app(argc,argv);

    GameArea box(500);

    box.show();

    return app.exec();

}
```

6. 然后运行程序。效果如下图。



当游戏结束时会弹出提示框，确定后游戏将重新开始。

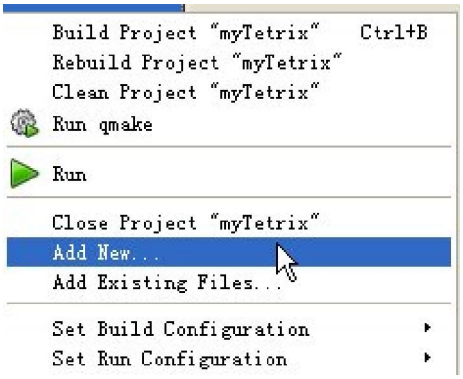
如下图所示。



7．可以看到，用这个类建立方块游戏是这么的简单。我们只是在主函数里新建了一个该类的对象，其中的参数为方块下移的速度，单位是毫秒，上面的 500 即 0.5 秒。

提示：

（如何向工程中添加文件）



在工程文件夹上点右键，弹出的菜单中 Add New 表示添加新文件，Add Existing Files 表示添加工程文件夹中已有的文件。

第二部分：功能展示

要想实现更强大的功能，我们就需要应用控制窗体，而让这个游戏区域只作为它的一个部件。为了更好的控制游戏，我们也需要自己建立定时器，而不再应用该类自带的定时器了。

核心功能：

（一）建立工程。

1．首先建立工程 Qt4 Gui Application，这里命名为 Tetris，选用 QWidget 作为 Base class 。

2．然后将 myitem.cpp， myitem.h ，gamearea.cpp， gamearea.h 四个文件复制到工程文件夹下并添加到工程中。

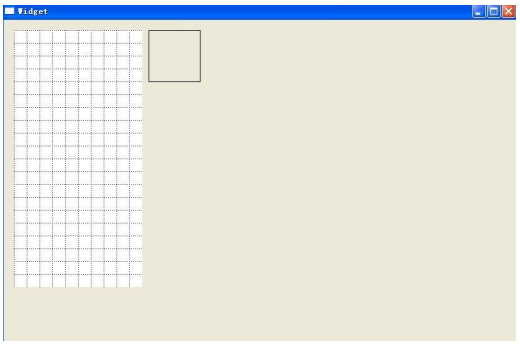
3．在 widget.h 中添加`#include "gamearea.h"`的头文件包含。并在下面的 private 中声明一个游戏类对象 `GameArea *gameArea;`

4．在 widget.cpp 的构造函数里添加语句。

```
Widget::Widget(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Widget)
{
    ui->setupUi(this);
    this->resize(800,500);
    this->gameArea = new GameArea(this);
}
```

这里重新设定了主窗口大小，并在主窗口上新建了一个游戏区域对象。

5．这时运行程序效果如下。



可以看到，因为使用了另一个构造函数，没有使用该类自带的定时器，所以只是显示了游戏区域，游戏并没有运行。

（二）添加定时器和开始按钮，让游戏可以运行。

1．在 widget.h 里的 private 中添加定时器对象和分数变量的声明。

```
QTimer *timer;
int score;
```

在 public 中添加显示分数函数的声明。

```
void doScore(int);
```

添加槽函数的声明。

```
private slots:
void timer_upDate();
```

2．在 widget.cpp 文件中的构造函数里添加下面的语句：

```
this->timer = new QTimer(this);
connect(this->timer,SIGNAL(timeout()),this,SLOT(timer_upDate()));
score =0;
```

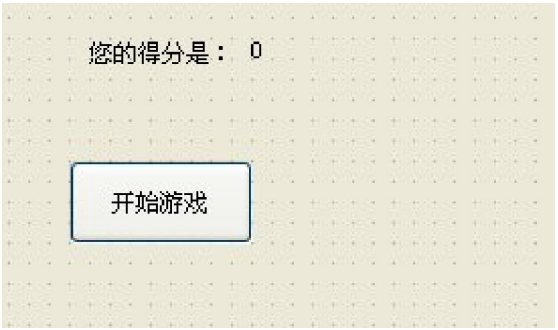
定义了定时器并进行了信号和槽函数的关联，初始化分数为 0；

3．然后在下面定义两个函数。

```
void Widget::timer_upDate() //定时器溢出处理
{
    this->gameArea->moveOneStep(); //先移动一步，这时并没有显示出来
    if(this->gameArea->isMoveEnd()) //如果无法移动，到底了或结束了
    {
        if(this->gameArea->isGame_Over()) //如果是结束了
        {
```

```
                this->timer->stop(); //停止计时
                QMessageBox::warning(this,tr("warning"),tr("Game
    Over!"),QMessageBox::Yes);
                //弹出对话框
                this->score =0; //清空分数
                this->gameArea->init_Game(); //重新开始游戏
                this->gameArea->gameStart();
                this->timer->start(500);
            }
            else  //如果是移动到底了
            {
                this->gameArea->nextItem(); //出现下一个图形
                int num = this->gameArea->getFullRowNum(); //获得已满的行数
                this->doScore(num); //显示分数
                this->gameArea->gameStart(); //继续游戏
            }
        }
        else //如果没有到底
        {
            this->gameArea->do_MoveNext(); //显示方块下移一步后的界面
        }
}

void Widget::doScore(int num)  //显示分数
{
        score += num*100;
        this->ui->label_2->setText(tr("%1").arg(score));
}
```

4．在设计器中向主窗口上添加两个标签 label 和 label_2，其中 label 写上“ 你的分数是：”，label_2 写上“ 0 ”；然后再添加一个开始按钮。添加完后效果如下。

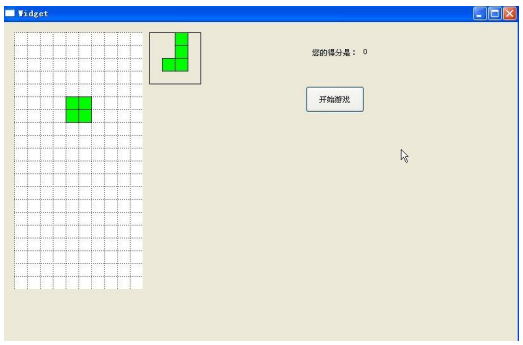


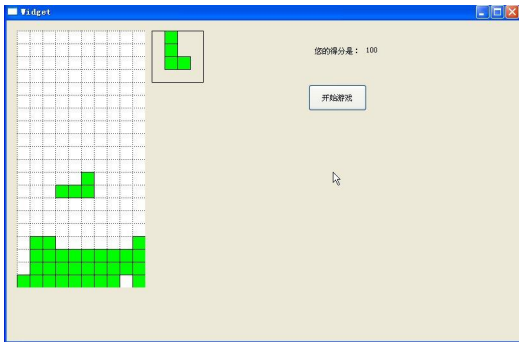
5．然后右击“ 开始游戏 ”按钮，选择其单击事件的槽函数。更改如下。

```
    void Widget::on_pushButton_clicked() //开始按钮
{
        this->gameArea->init_Game(); //第一次进入游戏时进行的初始化
        this->gameArea->gameStart();  //开始游戏
```

```
    this->timer->start(500);  //开启定时器
    this->gameArea->setFocus();  //让游戏区域获得焦点，这样才能响应键盘
}
```

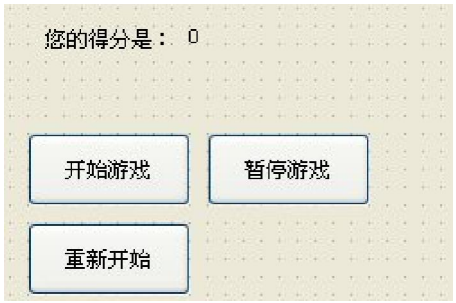
6．现在游戏已经可以正常进行了。运行效果如下。

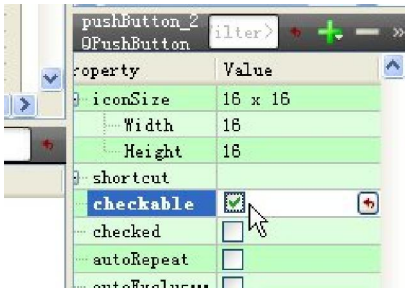




（三）添加暂停和重新开始按钮，完成基本的控制功能。

1．在主窗口上添加“ 暂停游戏 ”和“ 重新开始 ”两个按钮。在“ 暂停游戏 ”按钮的属性中将 checkable 选中。如下图所示。





2．分别进入两个按钮的单击事件槽函数。修改如下。

```
void Widget::on_pushButton_2_clicked() //暂停按钮
```

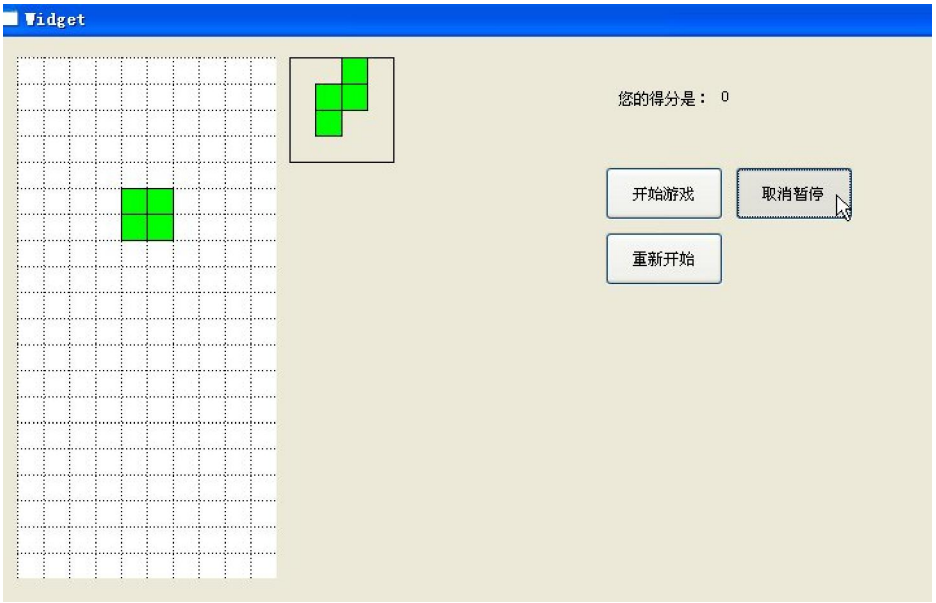
```
{
        if(this->ui->pushButton_2->isChecked())
        {
            this->timer->stop();
            this->ui->pushButton_2->setText(tr("取消暂停"));
        }
        else
        {
            this->timer->start(500);
            this->ui->pushButton_2->setText(tr("暂停游戏"));
    this->gameArea->setFocus();
        }
}

void Widget::on_pushButton_3_clicked() //重新开始
{
        this->timer->stop();
        this->on_pushButton_clicked();
}
```

3．在 main.cpp 中添加语句，让程序中可以使用中文。

添加`#include <QTextCodec>`的头文件包含。

在 main()函数里添加 `QTextCodec::setCodecForTr(QTextCodec::codecForLocale());`语句。

4．程序运行效果如下。



高级功能的应用

（一）改变颜色和给方块添加图片。

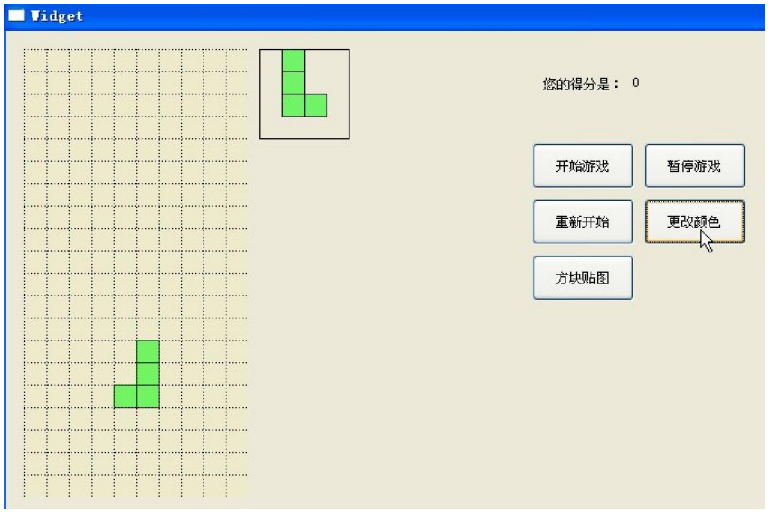
1．添加“ 更改颜色 ”按钮和“ 方块贴图 ”按钮。如下图。

2．更改其单击事件槽函数。如下。

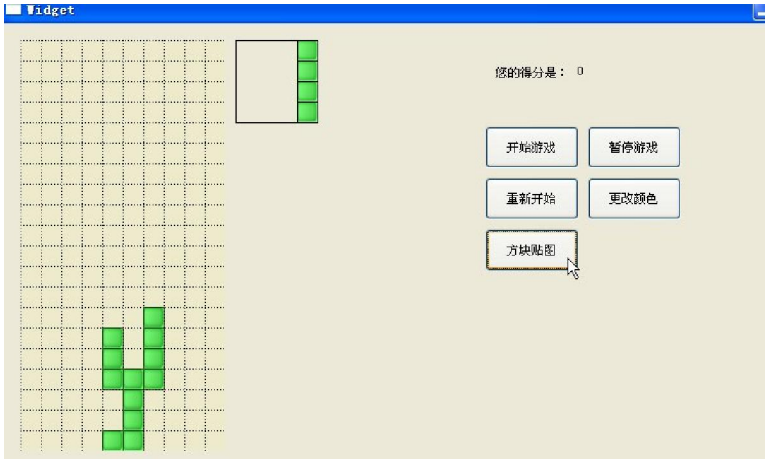
```
void Widget::on_pushButton_4_clicked()  //改变颜色
{
    this->gameArea->setGameAreaColor(QColor(255,255,0,qrand()%255));
    //更改游戏区域背景颜色
    this->gameArea->setBoxBrushColor(QColor(0,255,0,qrand()%255));
    //更改小方块背景颜色
    this->gameArea->setBoxPenColor(QColor(0,0,0,qrand()%255));
    //更改小方块边框颜色
    this->gameArea->draw_gameArea();
    //更新游戏区域
    this->gameArea->setFocus();
}

void Widget::on_pushButton_5_clicked() //方块贴图
{
    this->gameArea->set_draw_box_picture(true);
    //确认使用方块背景图片
    this->gameArea->setBoxPicture("box.gif");
    //添加方块背景图片
    this->gameArea->draw_gameArea();
    //更新显示区域
    this->gameArea->setFocus();
}
```

3．运行效果如下。

点击“ 改变背景 ”按钮后，游戏区域背景，方块的填充颜色和边框颜色都改变了。



点击“ 方块贴图 ”按钮。注意，只有方块颜色的透明度不是 255 时，才能看见贴图。所以，如果开始游戏后直接按“ 方块贴图 ”按钮，是不能显示出背景图片的，我们需要先改变颜色。



（二）是否显示背景网格和下一个要出现的方块。

1．添加“ 网格显示 ”按钮和“ 方块提示 ”按钮。并将它们属性中的 checkable 选中。界面如下。

2．修改它们的单击事件槽函数。

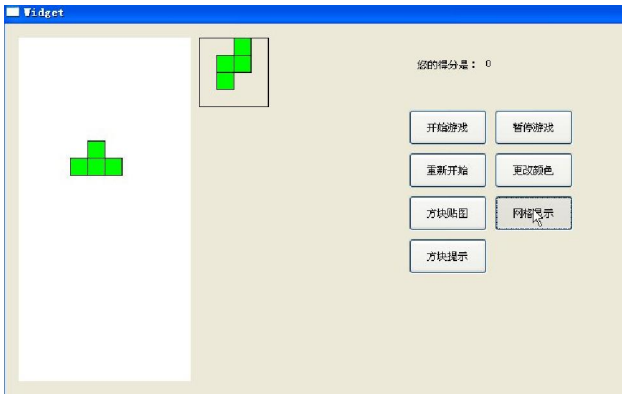
```
void Widget::on_pushButton_6_clicked()  //网格显示
{
    if(this->ui->pushButton_6->isChecked())
    {
```

```
        this->gameArea->setDrawGrid(false);
    }
    else
    {
        this->gameArea->setDrawGrid(true);
    }
     this->gameArea->draw_gameArea();
     this->gameArea->setFocus();
}
```

void Widget::on_pushButton_7_clicked() //方块提示

```
{
    if(this->ui->pushButton_7->isChecked())
    {
        this->gameArea->setDrawNextItem(false);
    }
    else
    {
        this->gameArea->setDrawNextItem(true);
    }
     this->gameArea->draw_gameArea();
     this->gameArea->setFocus();
}
```

3．运行效果如下。



（三）添加方块移动的声音。

1．添加“ 打开声音 ”按钮，并将其属性中的 checkable 选中。

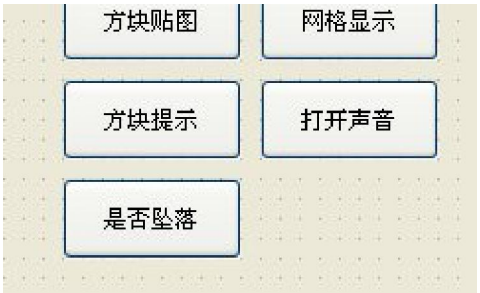
2．修改其单击事件槽函数。

```
void Widget::on_pushButton_8_clicked()    //声音开关
{
    if(this->ui->pushButton_8->isChecked())
    {
        this->gameArea->setPlaySound_itemChange("changeItem.wav",true);
        this->gameArea->setPlaySound_moveDown("moveDown.wav",true);
        this->gameArea->setPlaySound_moveLeft("moveLeft.wav",true);
        this->gameArea->setPlaySound_moveRight("moveLeft.wav",true);
        this->ui->pushButton_8->setText(tr("关闭声音"));
    }
    else
    {
        this->gameArea->setPlaySound(false); //关闭音乐
        this->ui->pushButton_8->setText(tr("打开声音"));
    }
    this->gameArea->setFocus();
}
```

3．我们把需要的声音文件放到工程文件夹下的 debug 文件夹下。注意：只能是 wav 格式的。然后运行程序，测试一下效果。

（四）添加向下按键移动步数设置。

1．添加“ 是否坠落 ”按钮，并将其属性中的 checkable 选中。



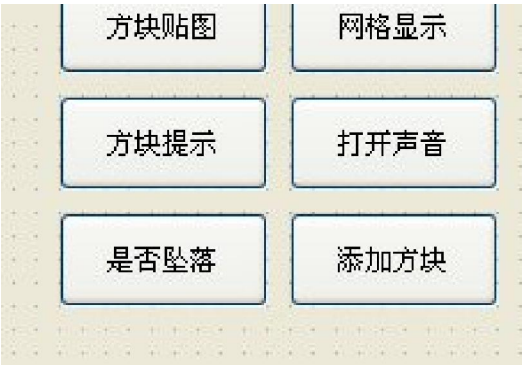
2．更改其单击事件槽函数。

```
void Widget::on_pushButton_9_clicked() //是否坠落
{
    if(this->ui->pushButton_9->isChecked())
    {
        this->gameArea->setKey_Down_Move_oneStep(true);
        //按一下向下方向键，下移一步
    }
    else
    {
        this->gameArea->setKey_Down_Move_oneStep(false);
        //按一下向下方向键，移动到底
    }
    this->gameArea->setFocus();
}
```

3．运行程序，测试一下效果。

（五）自己添加方块。

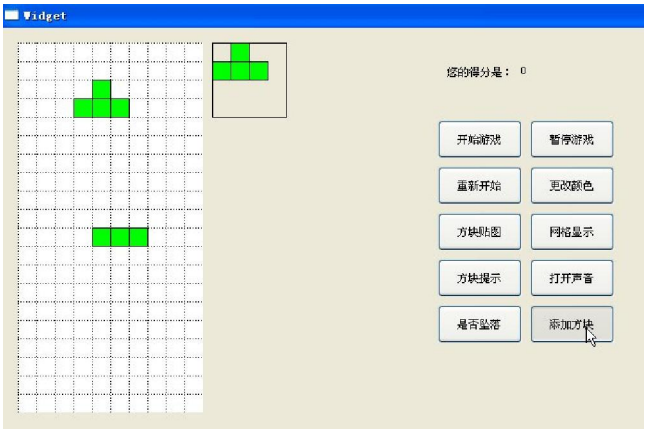
1．添加“添加方块”按钮。



2．修改其单击事件槽函数。

```
void Widget::on_pushButton_10_clicked() //添加方块
{
    this->gameArea->init_Game();
    //清空游戏区域
    this->gameArea->setbox(10,4);
    this->gameArea->setbox(10,5);
    this->gameArea->setbox(10,6);
    //在第 10 行第 4，5，6 列添加三个方块
    this->gameArea->gameStart();
    //重新开始游戏
    this->gameArea->draw_gameArea();
    this->gameArea->setFocus();
}
```

3．运行程序，效果如下。



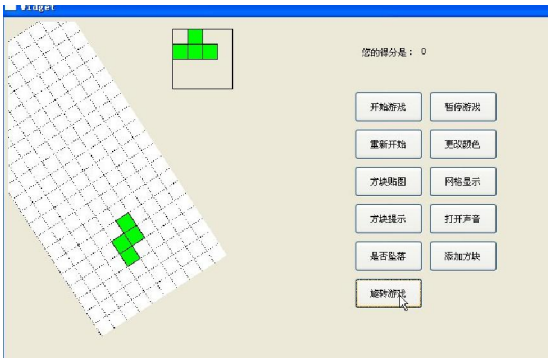
（六）设置旋转游戏区。

1．添加“ 旋转游戏 ”按钮。

2．修改其单击事件槽函数。

```
void Widget::on_pushButton_11_clicked() //旋转游戏
{
        this->gameArea->setRotate(true);
         //开启旋转
        this->gameArea->setGameAreaPixOrigin(100,200);
         //设置游戏区域新的坐标原点
        this->gameArea->setGameAreaPix(-100,-200);
           //设置游戏区域的位置
        this->gameArea->setRotateAngle(qrand()%360);
           //旋转度数
        this->gameArea->draw_gameArea();
         this->gameArea->setFocus();
}
```

3．运行程序，效果如下。



第三部分：游戏分析

(一)可被外部调用的功能函数的原型。

```
GameArea(QWidget *parent = 0); //不带定时器的构造函数
GameArea(int speed,QWidget *parent = 0); //带定时器的构造函数

//以下是核心功能控制函数
void init_gameArea(int X,int Y,int frame_width,int frame_height,int width,int height,int
boxStep,int start_x,int start_y);
void moveOneStep();
bool isMoveEnd();
bool isGame_Over();
void init_Game();
void gameStart();
void nextItem();
int getFullRowNum();
void do_MoveNext();
void draw_gameArea();

//以下是设置颜色函数
void setGameAreaColor(QColor color=Qt::white);
void setBoxBrushColor(QColor color=Qt::green);
void setBoxPenColor(QColor color=Qt::black);
void set_draw_box_picture(bool Bool=false);
void setBoxPicture(QString fileName);

//以下是设置声音函数
void setPlaySound_moveLeft(QString fileName,bool Bool=false);
void setPlaySound_moveRight(QString fileName,bool Bool=false);
void setPlaySound_moveDown(QString fileName,bool Bool=false);
void setPlaySound_itemChange(QString fileName,bool Bool=false);
void setPlaySound(bool Bool=false);

//以下是设置游戏区域旋转函数
void setRotate(bool Bool=false);
void setGameAreaPixOrigin(int x,int y);
void setGameAreaPix(int x,int y);
void setRotateAngle(int angle);

//以下是其他功能函数
void setKey_Down_Move_oneStep(bool Bool = false);
void setDrawGrid(bool Bool = true);
void setDrawNextItem(bool Bool =true);
void setbox(int row,int col);
```

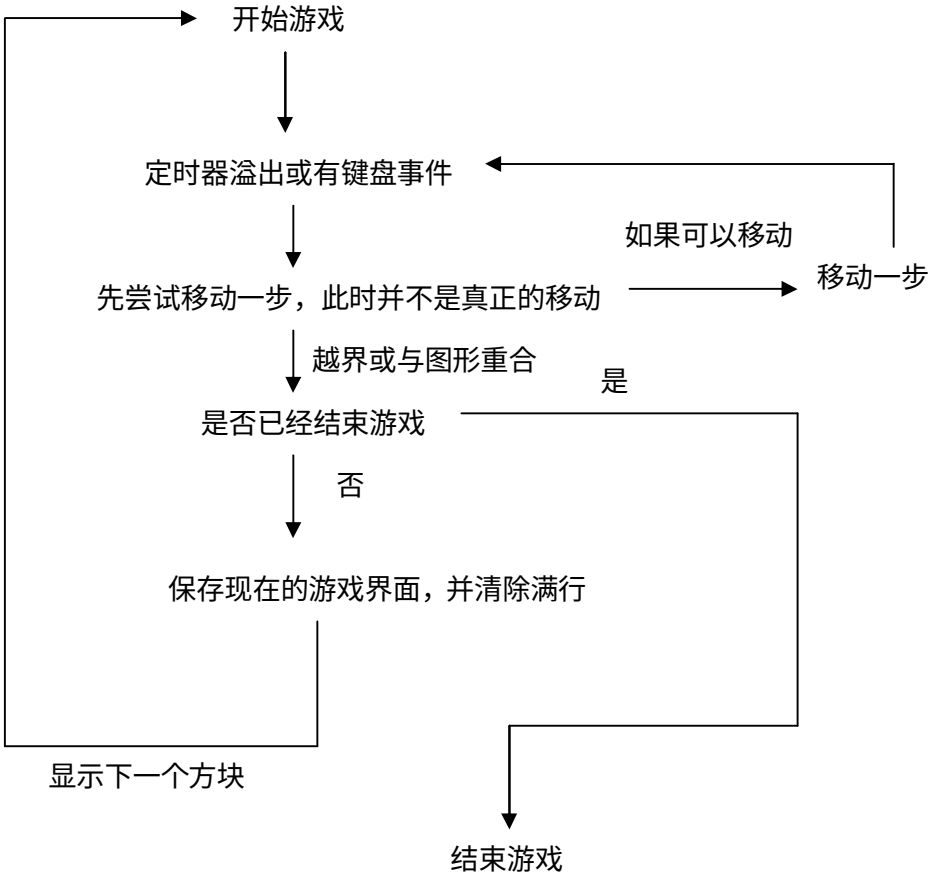
（二）游戏流程分析。

这里共有四个文件 myitem.cpp ，myitem.h， gamearea.cpp， gamearea.h

其中 myitem.cpp， myitem.h 是方块类的定义文件，它用来提供随机产生的方块。

gamearea.cpp， gamearea.h 是游戏区域类的定义文件，它实现了游戏的所有功能。

为了帮助大家更好的理解源码，我们这里描述一下游戏的实现流程。



难点解析：

1. **游戏实现的核心方法是什么？**

   这里是利用了数组对整个游戏区域进行存储。

2. **游戏是怎么实现显示现在和已有的图形的？**

   这里利用了两个数组，方块每移动一步，都对整个数组进行一次备份。

3. **游戏是怎么判断方块已经重合的？**

   这里是先尝试让方块移动一步，判断其是否与其他图形重合，如果不重合就移动。如果重合了，就进行其他操作。

4. **游戏是怎么进行消行的？**

   这里是让已满的行和它上面的所有的行均等于其上面的一行。

到这里整篇教程就结束了。这个方块游戏类还有很多功能不太完善，为了使代码更容易理解，也舍去了一些功能，例如让不同方块显示不同的颜色等。大家可以参考 Qt 自带的例程进行完善，也很希望大家对这个类的源码进行修改和完善。

最后，如果你喜欢我的写作风格，并且初学 Qt，可以在我的空间查看 Qt Creator 系列教程，希望能对你的入门有所帮助。