运行程序，将UI文件的内容显示到界面上。构造TetrixWindow（调用TetrixWindow的构造函数），构造一个Controller类型的对象，用于控制游戏中的两个Board，调用Controller的构造函数时，获取当前的时间用于生成一个随机数种子，用于随机生成一个形状。初始化了变量moveDownOK=true，表示形状在mainBoard上可以自由下落。passShape=0表示nextBoard上的形状还没有给到mainBoard上。Shape=NULL表示放上去之前还没有形状。将各个控件的信号与自定义的槽连接起来实现交互。

重写了2个事件，一个是接受用户的按钮，用户按下上下左右时触发，控制形状的旋转、加速下落、左移和右移。一个是绘图事件，在超时的时候触发，先得到每一个小方格在窗口上的的像素位置和逻辑颜色，然后再用逻辑颜色填充界面上的方格，并用画笔画出有颜色的小方格的边界，达到形状显示在界面上的效果。

点击开始按钮，调用startGame槽函数。开始计时并设置超时时间间隔，初始化状态isStarted=true，初始化等级level=0。然后控制器调用mainPlaceAShape和nextPlaceAShape函数分别在mainBoard和nextBoard上放置形状。

在mainPlaceAShape函数中，先将shape指针从上一个回合中指向的那个形状移开，让其指向当前回合中的形状，然后调用placeAShape函数放置形状。

在nextPlaceAShape函数中，先生成一个随机形状，然后把它传递到passShape中。调用placeAShape函数放置形状。

在placeAShape函数中，接受参数有放置在哪个Board上，放置的位置在哪，放置的Shape是什么（在游戏中，每个不同的Shape使用宏定义用整数去标识）。对于放置在mainBoard上，放置的位置是水平的中心和竖直的开始，放置的是由nextBoard中出递过来的形状。对于放置在nextBoard上，放置的位置是水平的中心和竖直的中心，放置的形状由随机数生成。根据不同的形状，调用Board的getStartLocation函数获得放置的位置。根据这个位置调用Board的函数getAGrid获得那一行那一列的一个方块即为形状中编号为1的方块。根据第一个方块和在哪个Board上，调用形状的构造函数构造一个Shape类型的对象。placeAShape函数返回的是一个Shape类型的指针对象。

在getStartLocation函数中，传入的参数是形状和位置，这个位置决定了形状是放置在哪个Board上，计算出并返回开始放置的位置，几行几列。这个开始位置是形状中编号为1的方块所在的位置。

在Shape的构造函数中，传入的参数是编号为1的方块的和Board，根据第1个方块的位置，求出其余3个方块的位置然后再把颜色渲染到Board的位置上。

调用了mainPlaceAShape函数之后，就会设置moveDownOK为true，则在下一次超时时，既可以下落。

放置好方块之后，方块处于自由下落的状态。调用控制器的moveDown函数，函数中调用isMoveable函数判断是否可以往下移，如果可以，再调用move函数实现向下移动。

在isMoveabel函数中，传入了布尔值参数isMoveDown当被moveDown调用时为真，当形状下落到底部或碰到其他方块时，将控制器中的moveDownOK赋值为false表示不能再向下移动，当前回合结束。

向左向右移动同理。

下落过程中，可以对形状进行旋转。调用控制器的rotate函数，先调用setBoardColor函数将Board上原来位置的颜色清空，然后调用Shape对象的rotate函数，最后在旋转到的位置上渲染上原来的颜色。

Shape的rotate函数中，调用了computeRotateCenter函数用于计算出旋转中心（即编号为2的方块的位置）。在Shape中，数据成员whichPosition记录了旋转的次数，rotatePositions数组记录了形状处于不同方位时各个方块的位置，使用了查找表的方法。Shape的rotate函数中，还调用了isValidPosition函数来判断是否可以旋转到那个位置，如果可以，则把旋转到的位置的Board上的方块对应到Shape的grids数组中。

computeRotateCenter函数在每一个子类中都有重写，用rotatePositions数组记录了形状处于4个不同的方位时编号为1、3、4的方块相对于编号为2的方块的位置。

当一个回合结束时，当前方块不再下落，moveDownOK=false，此时可能会出现消除行的情况，用控制器的numRowsClear函数判断是否需要消除行，如果大于0则需要。

在numRowsClear函数中，调用了traverseRows函数，并传递一个计数作为参数。返回一个回合中消除的行数。

在traverseRows函数中，从下往上依次调用Board的isFull函数判断每一行是否是满的。如果找到满的行，则计数加一。

消除行后，相应的要加上分数。调用控制器的computeScore函数计算这一回合中得到的分数。

调用控制器的clearRows函数，此函数又调用traverseRows函数，传递的参数为NULL，因此这时不作计数，而是每找到一行满的则调用Board的clearRow函数将这一行清空。

调用控制器的rowsMoveDown函数，消除了满的行后，将上面的行整体往下移。在函数中调用Board的rowCutPaste函数实现将某一行的内容复制粘贴到另一行。

rowCutPaste函数接受2个参数，实现将第n行各个方块上的颜色给到第m行上对应位置的方块，并将第n行上各个方块的颜色清空。

在每一个回合开始时，在放置方块上去之前，先调用控制器的canBePlace函数判断是否输掉游戏，如果输了则调用gameOver函数停止计时器。

游戏开始

在面板上放置方块

方块在面板上自由下落

控制方块的移动和旋转

判断是否需要消除行

消除行时行的整体的向下移动

判断游戏是否结束