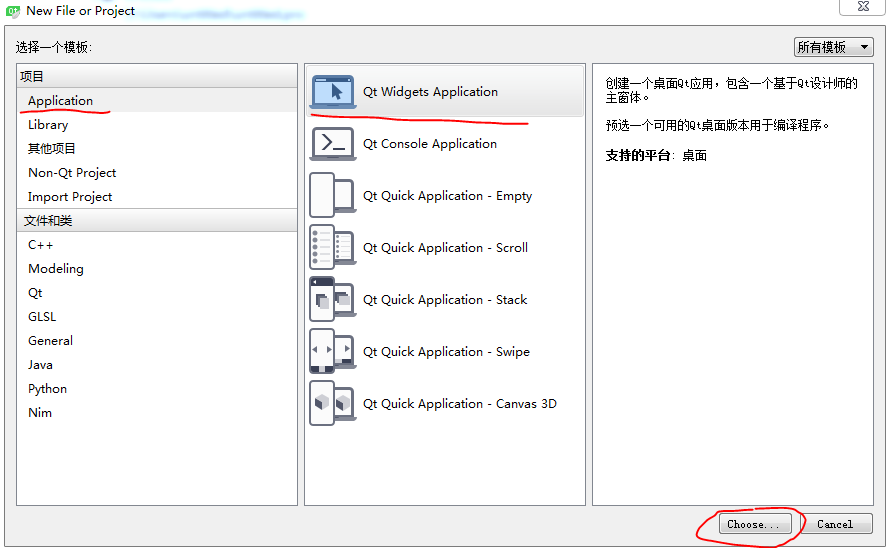
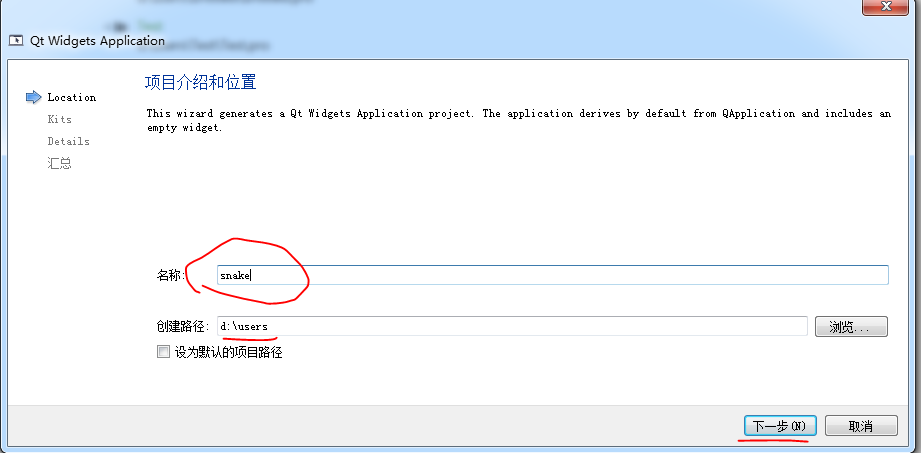
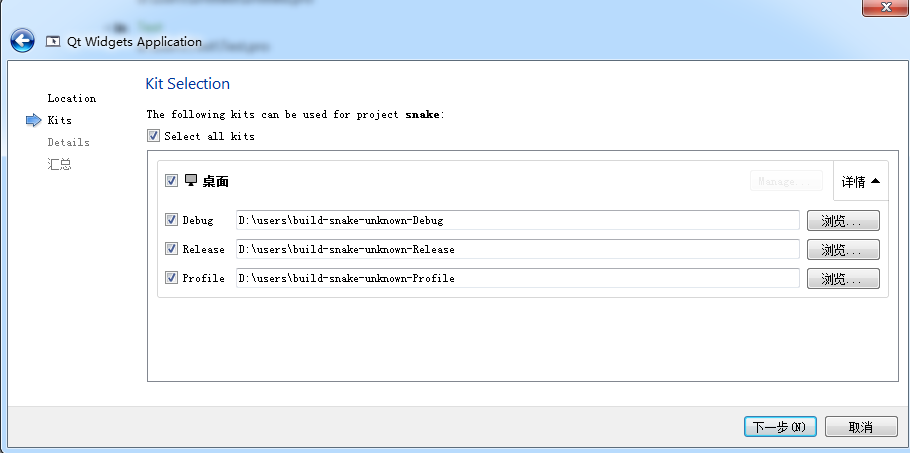
**一步一步做，做一步调试一步！**

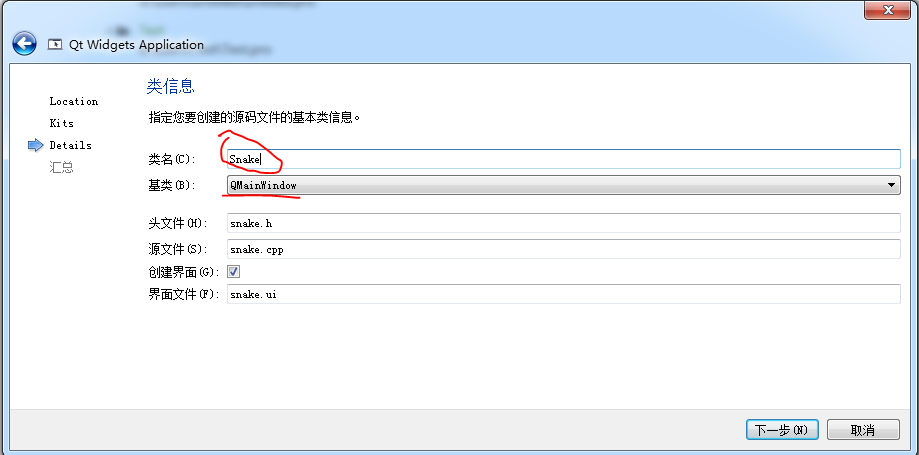
**新建一个窗口项目**

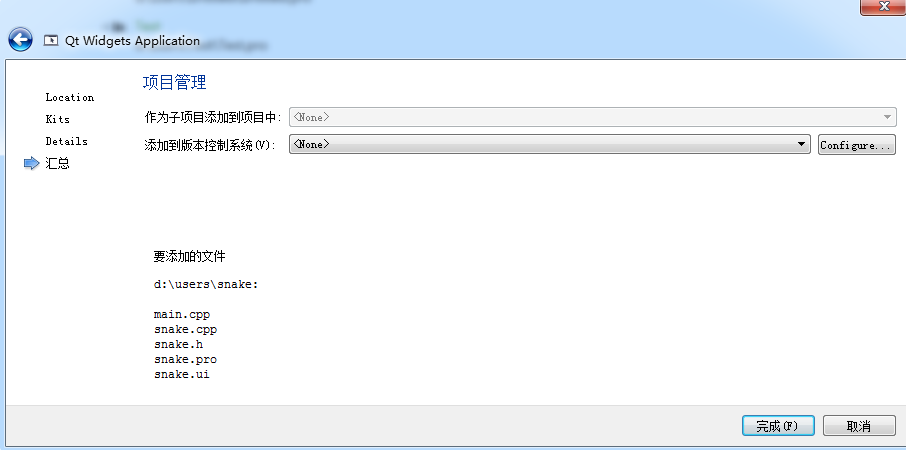




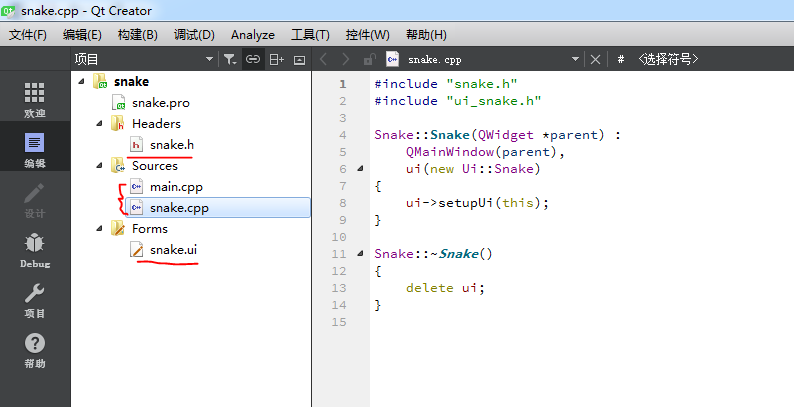


给自己的类起个名字，基类可以选择QmainWindow，

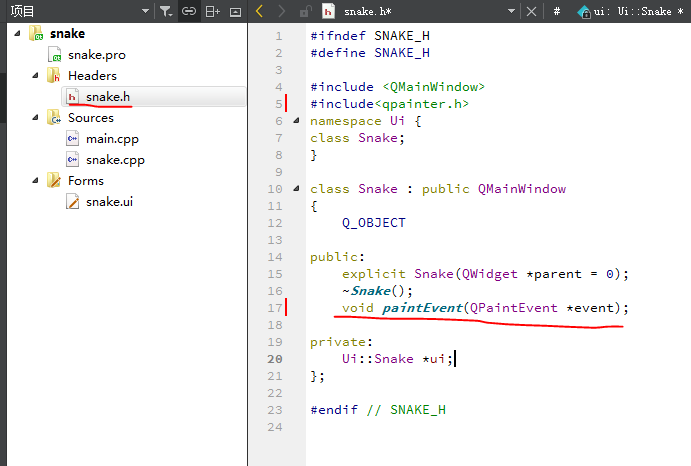




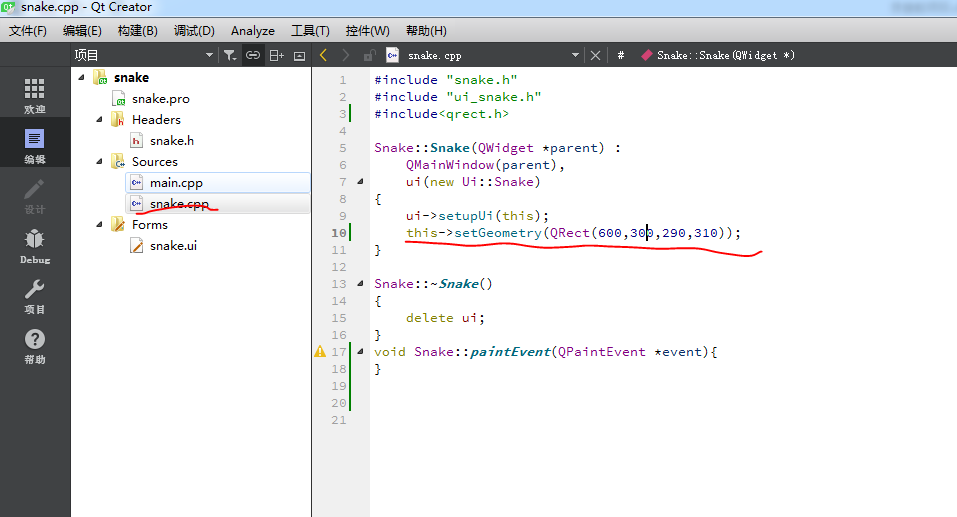
Main是程序的入口，snake.cpp是自己应用程序的主程序，snake.ui是界面，可以双击进入界面设计器。头文件和主程序里都已经有了部分代码，这时是可以运行的，只不过窗体里没有任何内容。



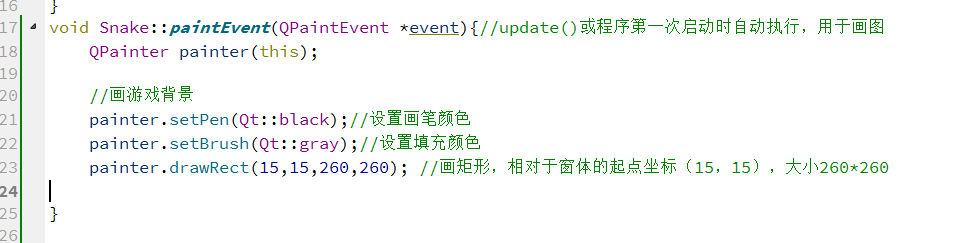
**第一步：画游戏界面**。要用到paintEvent函数，paintEvent(QPaintEvent\*)函数是QWidget类中的虚函数，用于ui的绘制，会在多种情况下被其他函数自动调用，比如程序第一次启动，或update()时（界面刷新），注意要包括头文件。



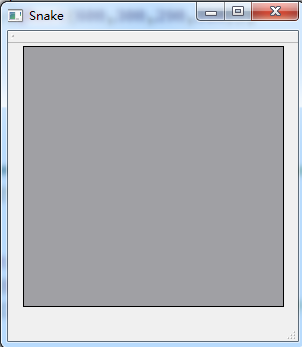
首先我要在构造函数里写明我这个窗体显示在什么位置，如规定窗体显示在以坐标（600,300）为起始点，大小为290\*310的矩形（QRect）范围内



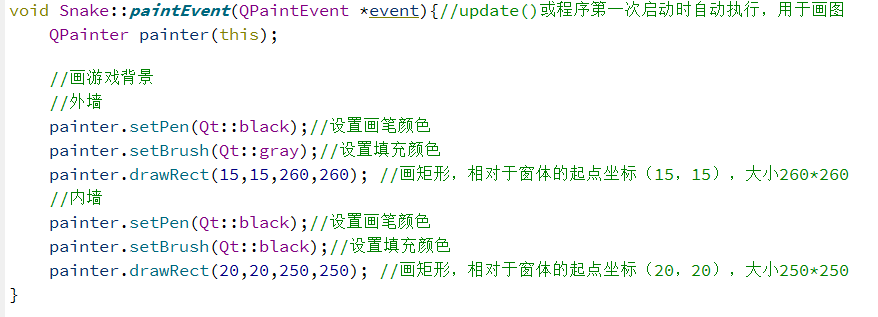
先画一个外围矩形（墙的外围）



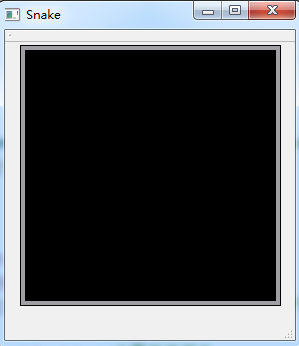
运行效果：黑色边框，灰色填充



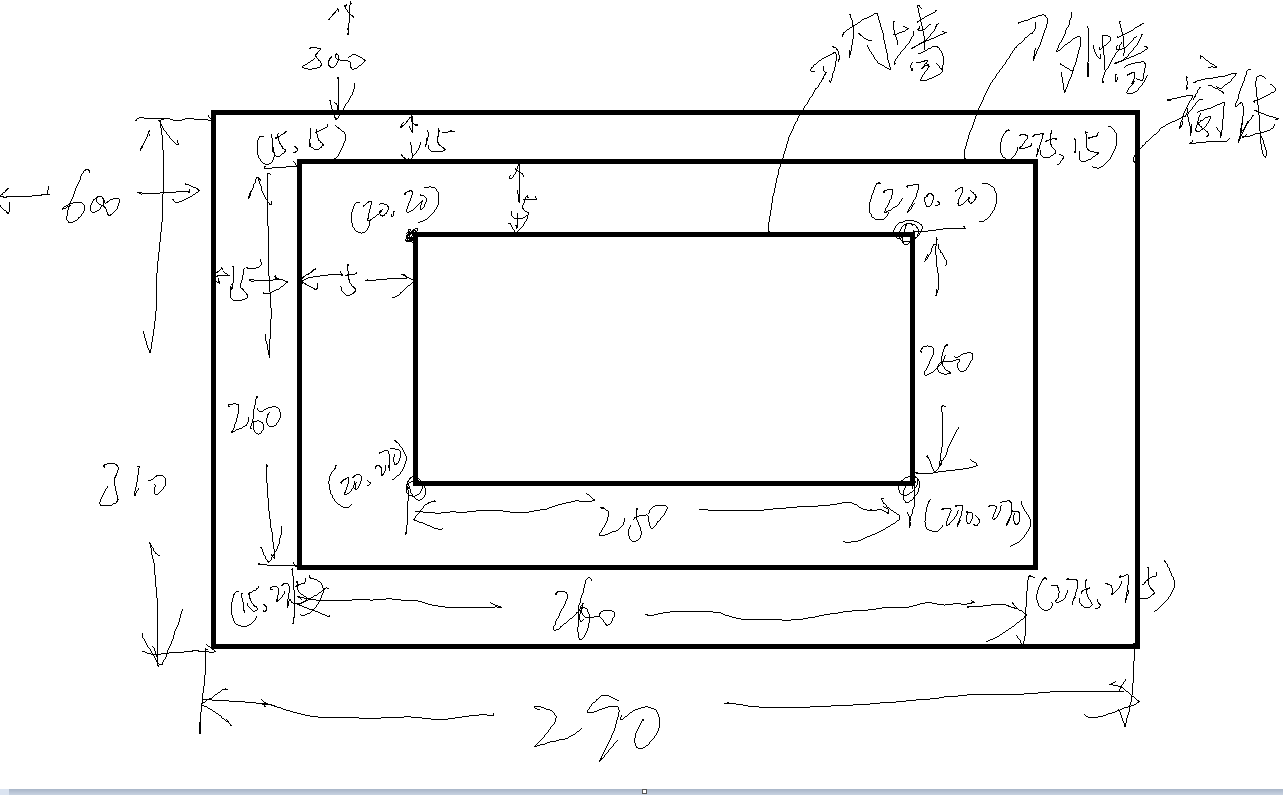
再画内层矩形（墙的内围）



运行效果：里层黑色边框，黑色填充



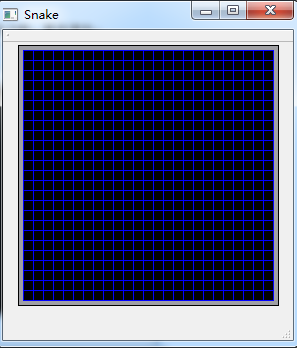
此时窗体及内部组件的分布如下，可以算出各个点的坐标值，这个值很重要！



为了看起来方便，我要在界面分出10\*10的小格子，drawLine函数有4个参数，前两个是起始横纵坐标，后两个是终点横纵坐标

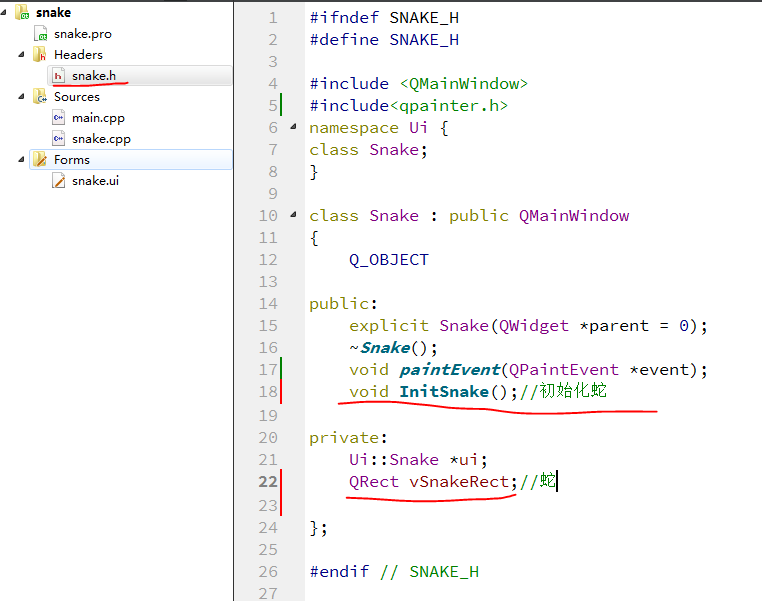


运行效果：



**第二步，画蛇。**从简单到复杂，可以先画一个小方块儿代表蛇，让这个小方块能移动。

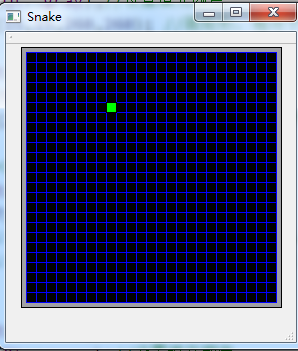
蛇是由n个小方块组成的，可以把蛇的初始化放在自定义的函数中进行：



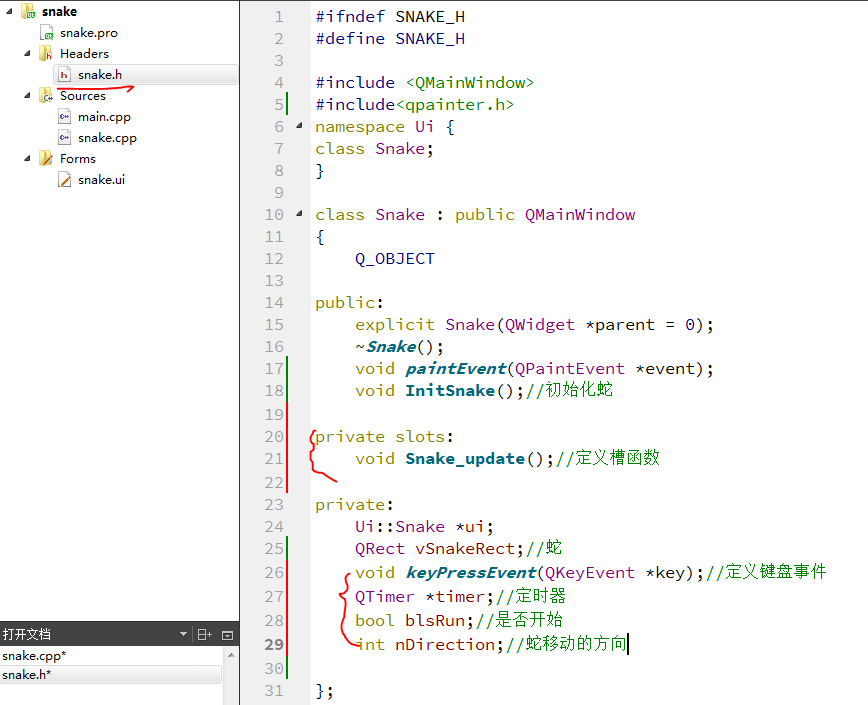
在paintEvent函数中调用初始化蛇函数InitSnake，并将蛇画在界面上，注意：本例中只画了一个小方块代表蛇。



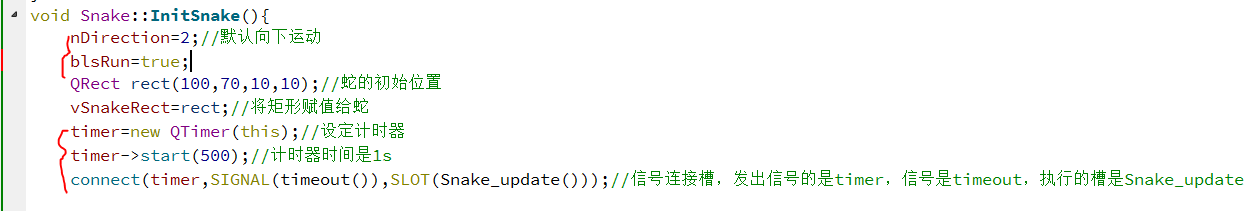
运行效果：



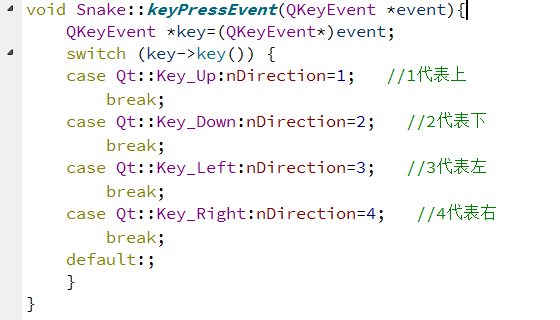
**第三步，让小方块动起来。**让蛇动是靠界面的刷新实现的，蛇动就是小方块的坐标更新，所以再添加一个自定义函数Snake\_update()，来进行坐标的更新，要想让蛇的位置不断更新，就要求每隔一段时间执行一次Snake\_update()函数，这个时间就是计时器事件Qtimer，时间到了就执行Snake\_update()函数，这个就是信号和槽，时间到了是信号，执行的函数Snake\_update()就是槽，所以要将这个函数定义为槽。Snake\_update()执行后，要刷新界面，以让蛇在界面上被重新画出，这里就用到update()函数，update函数执行后，会调用paintEvent（）函数进行画图。同时要用键盘来操作蛇的运动方向，还涉及到了键盘事件，为了记录蛇移动的方向，定义了一个整型变量nDirection。



计时器事件，记得设置默认移动方向和blsRun的值，确保Init\_Snake函数在paintEvent中只执行一次：



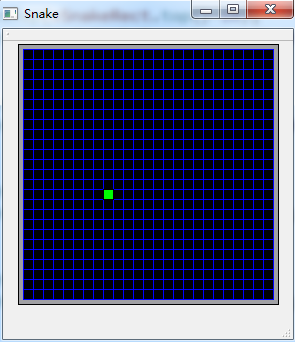
键盘事件：



蛇（小方块移动），坐标变换：

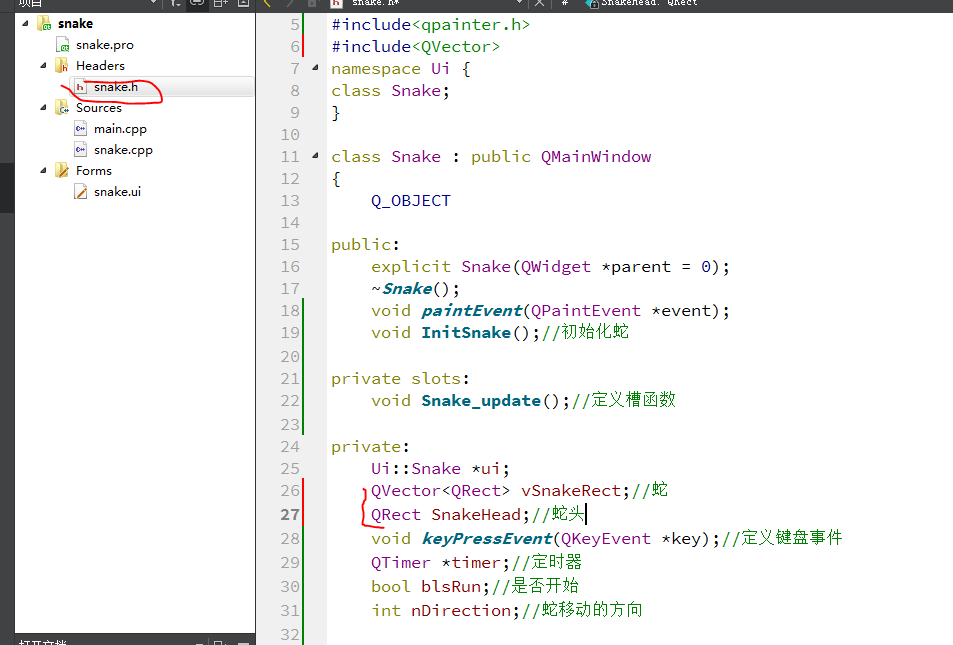


运行效果（小方块的位置已经发生了变化，且按方向键可以改变方向）：

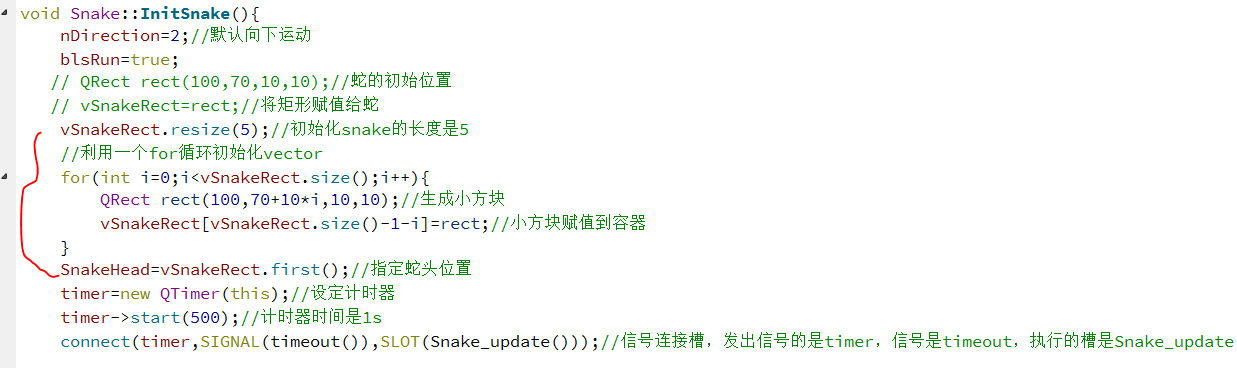


**第四步，小方块变蛇**。可以先画蛇，不移动，我会把移动的代码先注释掉。

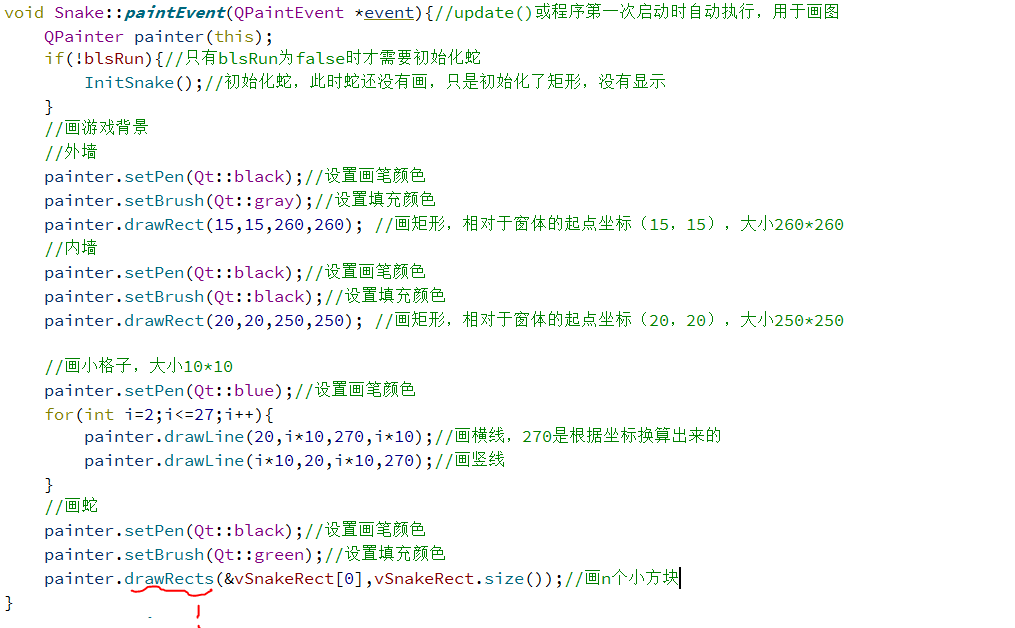
蛇由n个小方块组成，可以用数组，也可以用链表，QT里为我们提供了vector容器，用起来更方便，之前我们定义的vSnakeRect，只是一个小方块，现在我把他转变为Qvect的容器。并定义一个小方块是蛇头。



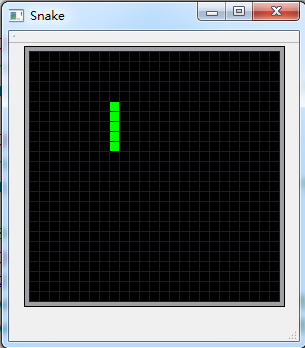
这样我初始化蛇的时候就不能只用一个小方块了，要用n个，我设定蛇身初始化长度为5个小方块。



经过上面的改写之后，我们要把这个容器里的小方块画到界面中。

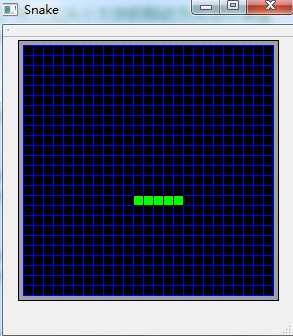


运行效果：

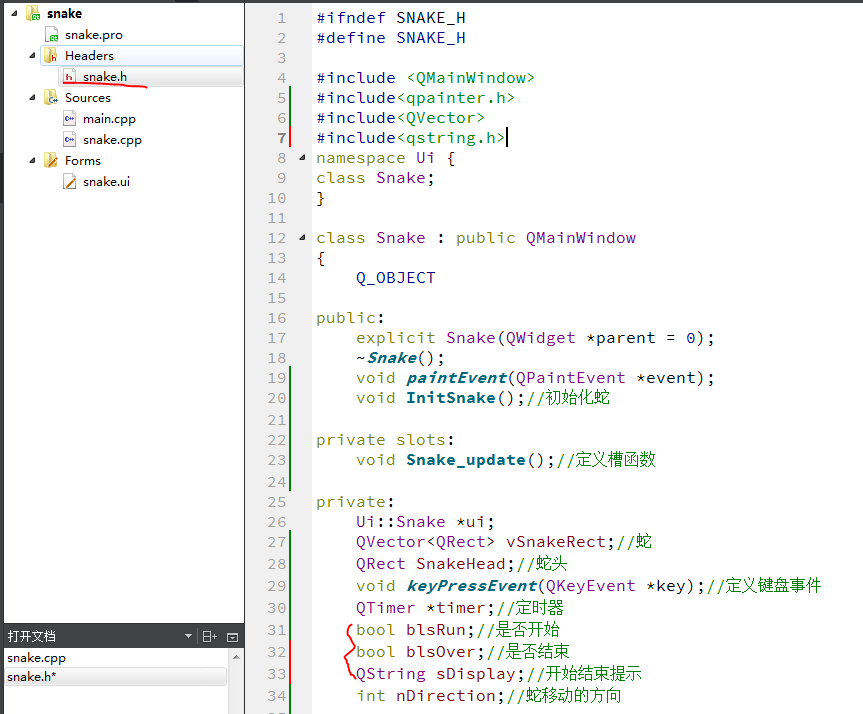


**第五步：让蛇动起来。**蛇已经有了，下面我们要让他移动，同样是在update里进行坐标变换。

运行效果（动态的）



蛇能移动了，但我们发现，当蛇撞到墙的时候并没有什么影响，所以我们要加一个撞到墙的判断，让游戏结束，为了让游戏效果更好，我们在游戏开始前显示“游戏开始”，在游戏结束后显示“游戏结束”





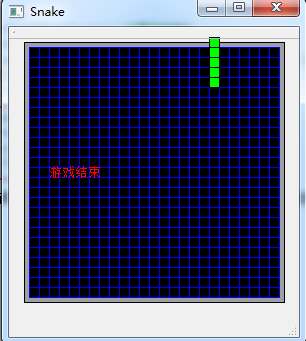
显示游戏开始，记得在Snake\_update里把sDisplay设为空，否则将会一直显示：



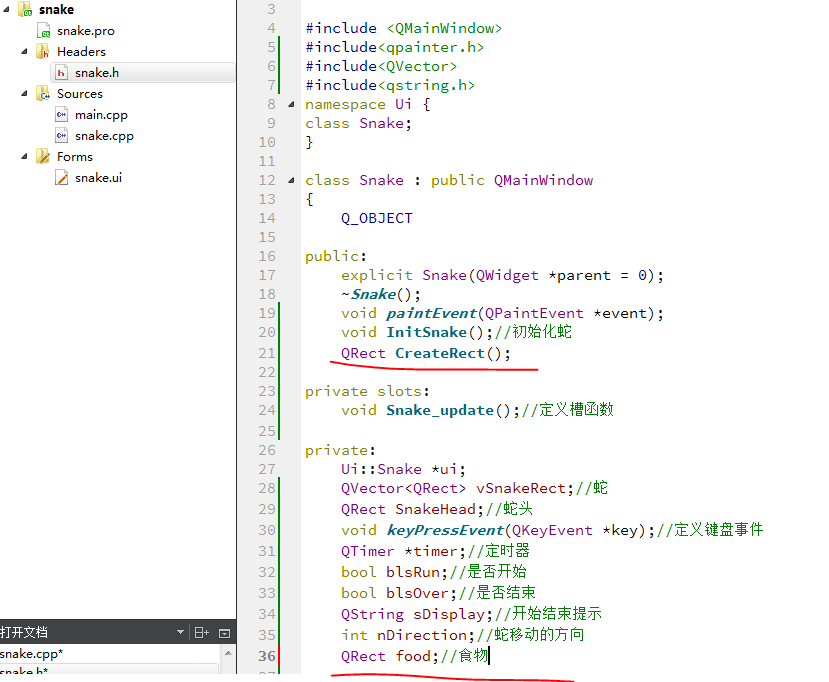
撞墙之后游戏结束，在Snake\_update里进行判断，还是坐标的判断：

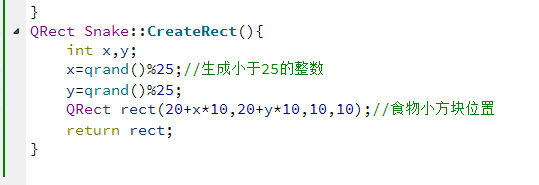


运行效果：

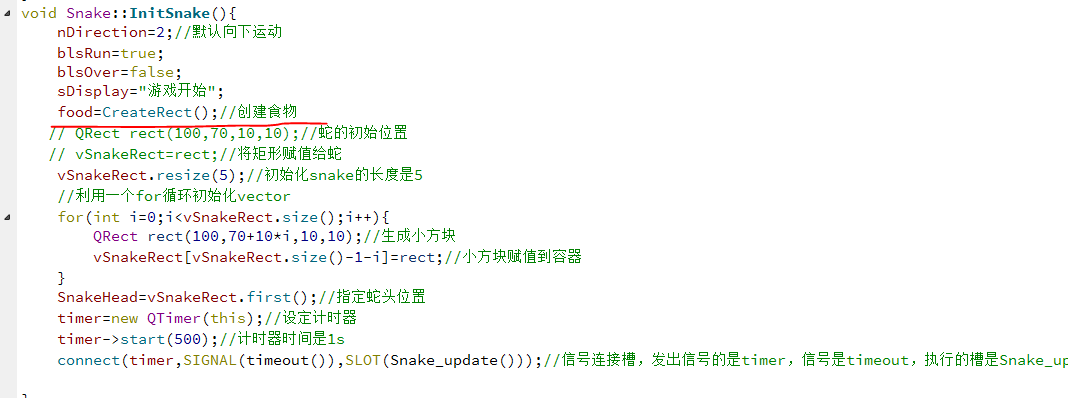


**第六步，蛇吃食物。**蛇有了，下面我们要显示食物，食物是随机产生的一个小方块，这里就不判断小方块产生的时候和蛇身重合的情况了。





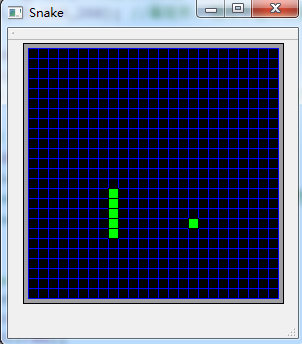
调用生成食物的函数



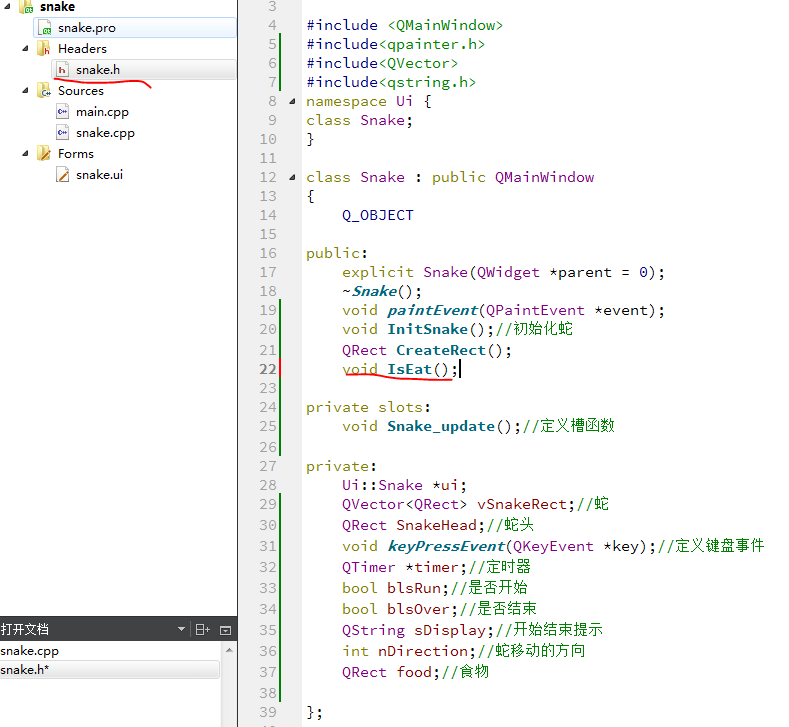
把食物画在页面上：

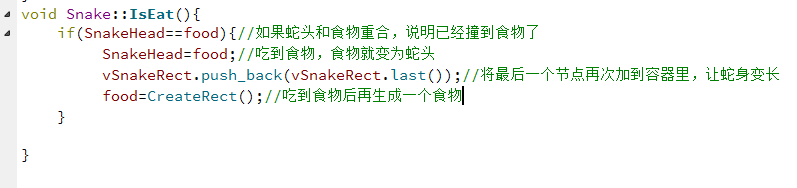


运行效果：



蛇吃食物会变长，吃到食物需要判断蛇头和食物是否重合，我们做一个IsEat函数来判断蛇是否能吃到食物，吃到食物之后变长，只需将最后一个节点再添加到vector容器即可，以达到变长的效果。

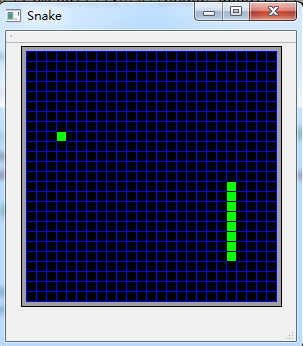




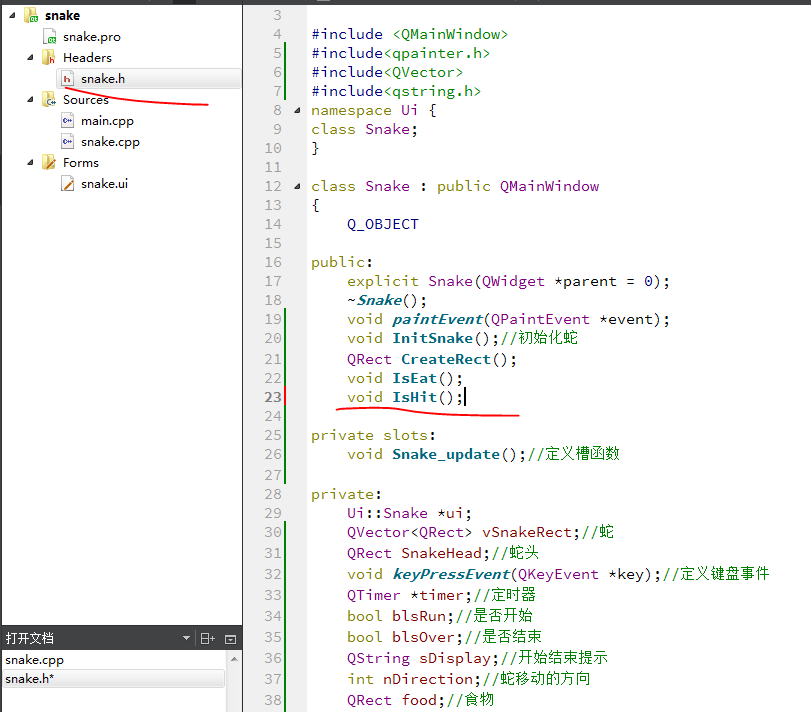
在snake\_update里调用IsEat函数

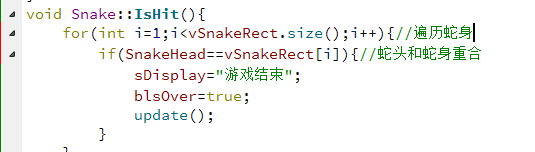


运行效果：



在游戏中，如果蛇撞到自己也会导致游戏结束，需要我们做一个函数来判断蛇是否撞到了自己，这个很简单，只需判断蛇头和蛇身是否重合即可。

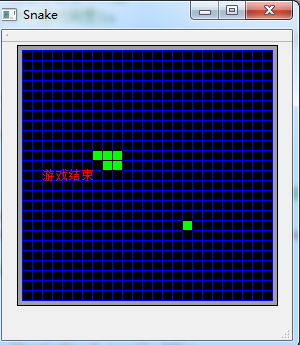




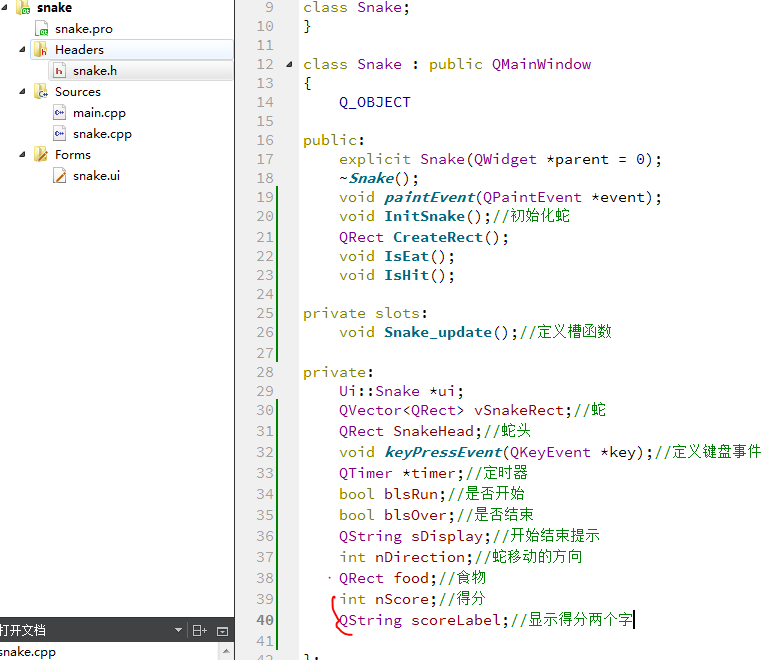
在snake\_update()中调用



运行效果：



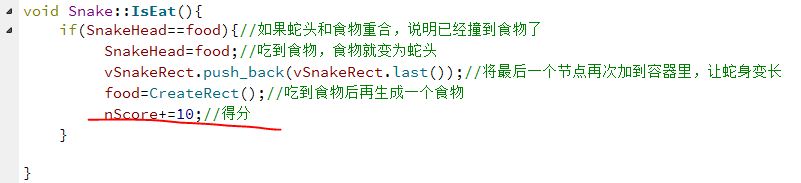
如果想统计的分信息很简单，只需设置一个整型变量，每次吃到食物之后递增就可以了。



在Init\_Snake里初始化两个变量



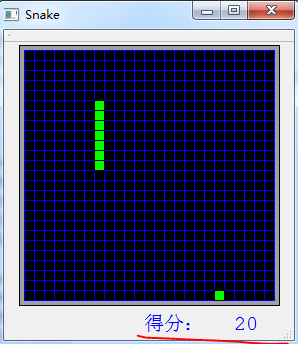
吃到食物，得分增加：



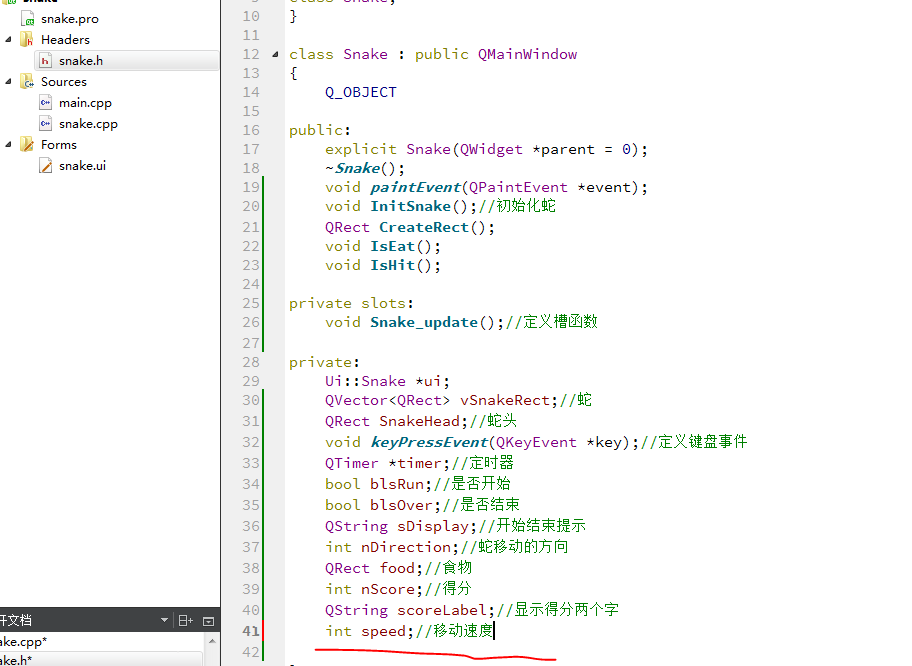
将得分画在界面上：



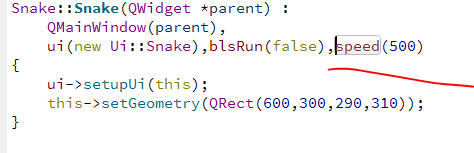
运行效果：

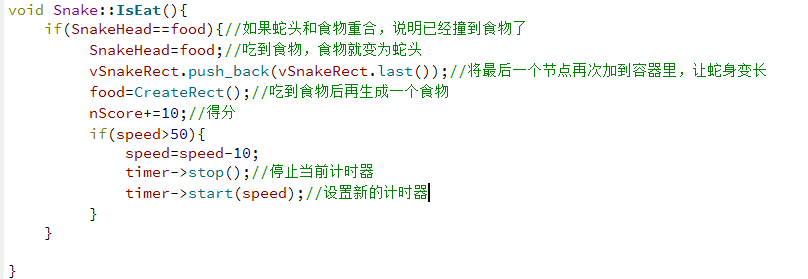


我们还可以让蛇吃到食物后移动速度加快，timer->start(500);//计时器时间是0.5s，只需把这里的500换成变量即可，然后在蛇吃到食物后，把这个变量减小



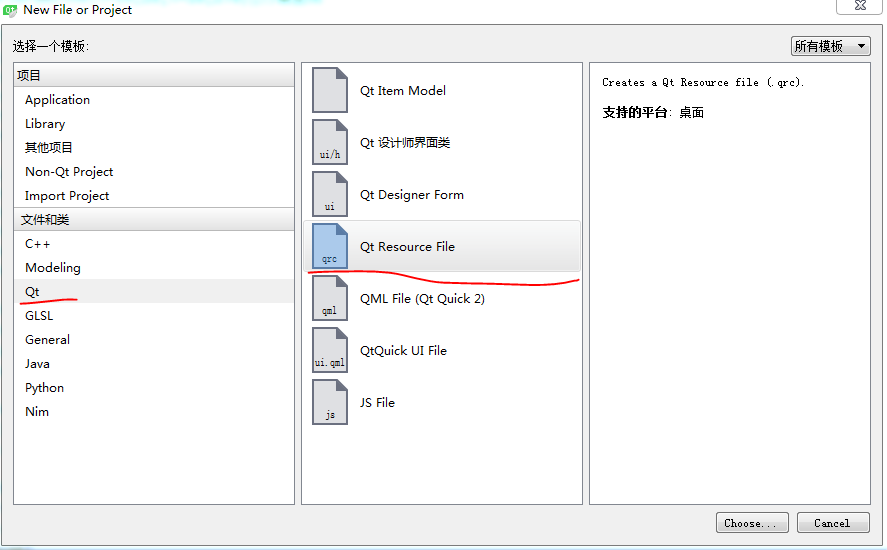
初始化speed



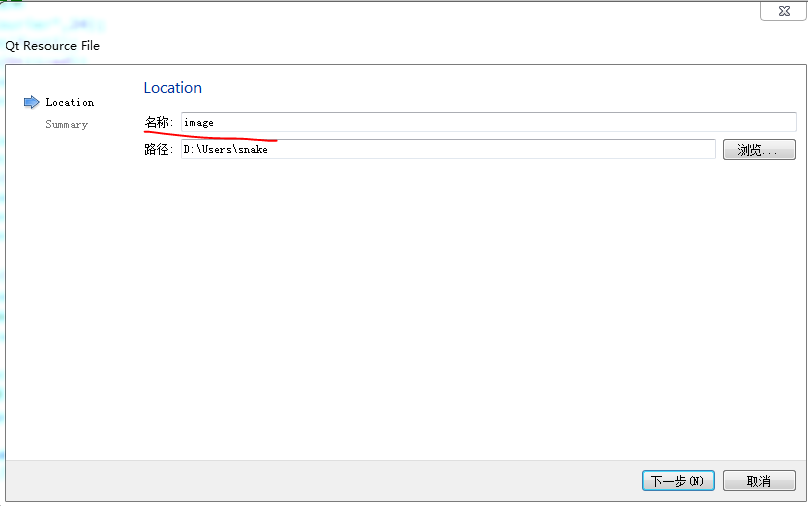


**最后美化一下**，把背景色和食物换成图片（蛇换成图片比较复杂，因为蛇头分方向，这里不做演示）

* **第一步，添加Qt资源文件。往项目中添加新文件，选择Qt分类中的Qt Resource File，文件名称改为“myImage”，其他选项默认即可。**
* **第二步，添加资源。建立好资源文件后会默认进入资源管理界面，就是新建的myImage.qrc文件中。现在先到项目文件夹myMainWindow中新建一个名为“images”的文件夹，并在其中放入两张图标图片，比如这里放入了一个“new.png”和一个“open.png”图片。（注意：Qt的资源系统要求资源文件必须放在与qrc文件同级或子级目录下，如果放在其他地方，当添加资源时会提示将文件复制到有效的位置。）**
* **然后回到Qt Creator中，在资源管理界面，按下“添加”按钮，选择“添加前缀”，然后将属性栏中的前缀改为“/myImage”，再点击“添加”按钮，选择“添加文件”，在弹出的对话框中进入到前面新建的images文件夹中，选中那两张图片，点击打开即可。这时myImage.qrc文件中就出现了添加的图片的列表。最后按下Ctrl+S快捷键，保存对文件的修改（注意：这一点很重要，如果没有保存，在下面使用图片时将看不到图片）。**

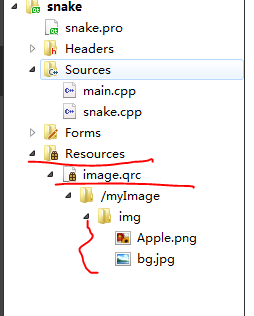


输入一个名字



**在编写代码时使用new.png图片，那么就可以将其路径指定为“:/myImage/images/new.png”， 前缀“/myImage”是添加资源时手动设置的。**

**有了资源文件如下图：**



这时可以在代码中引用了；



运行效果

