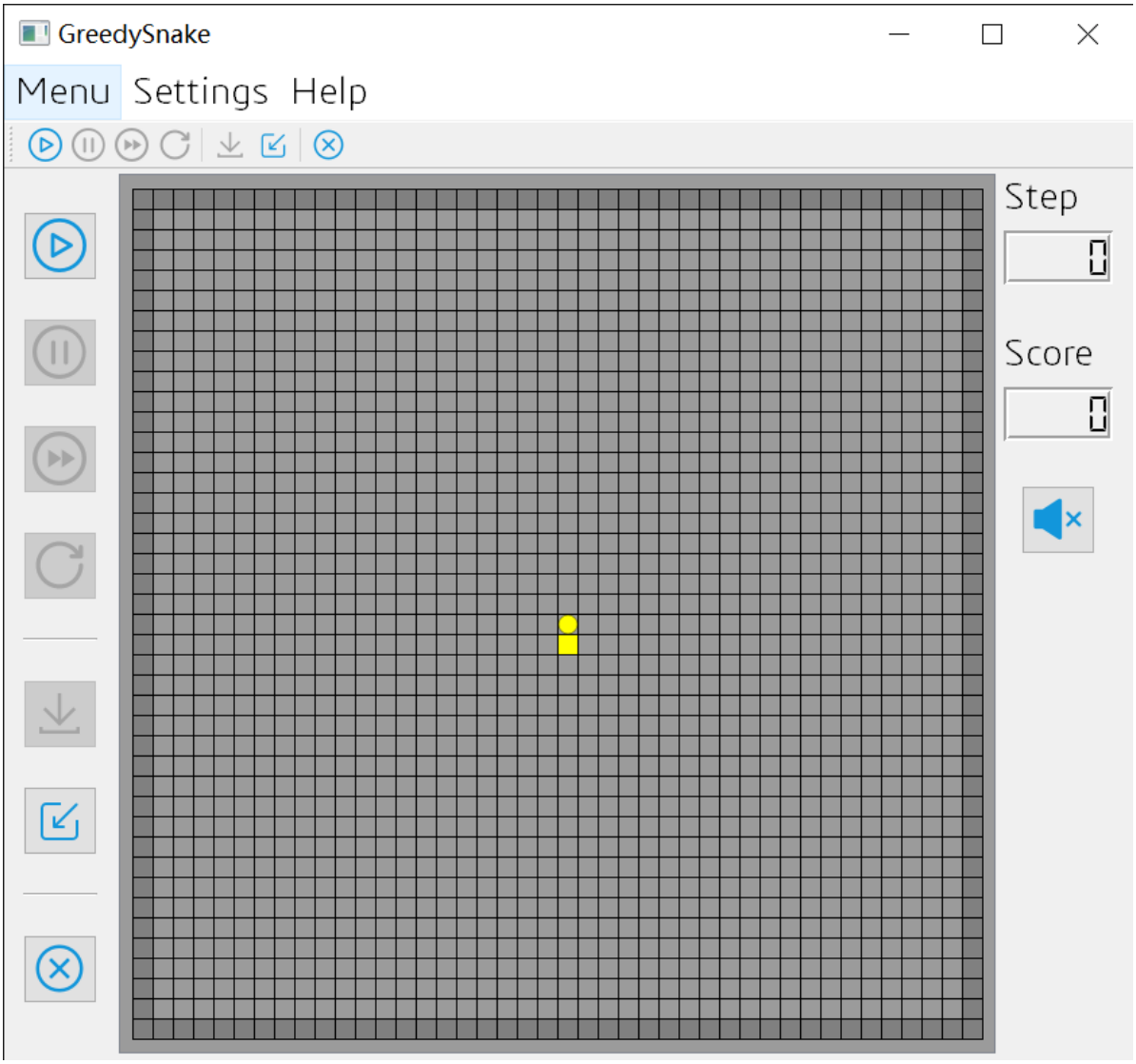
GreedySnake设计文档

2019011336 谢芷钰

1.功能介绍

1.1 基本功能

完成了大作业要求的基本图形界面功能。

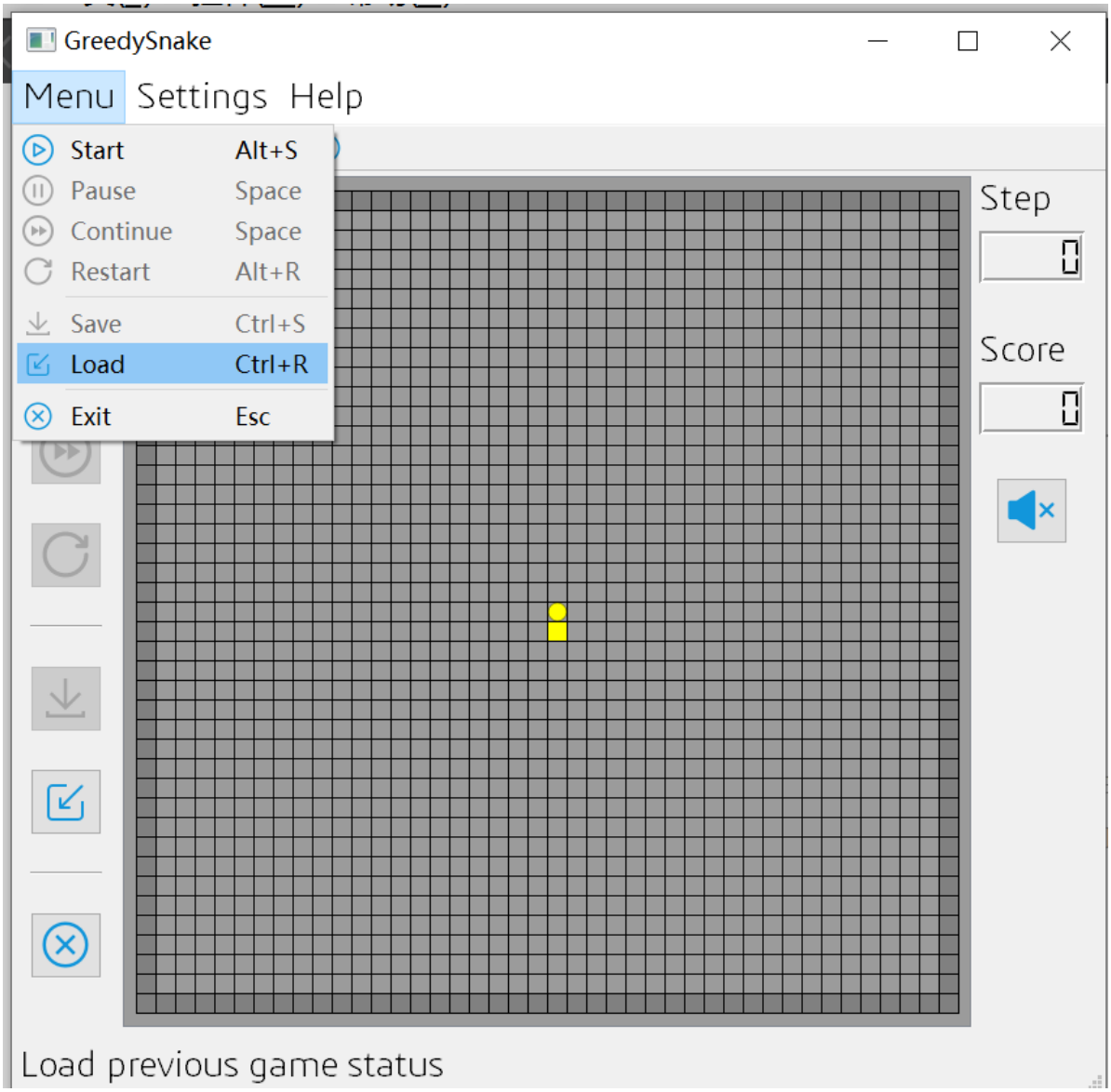


菜单栏、工具栏、游戏界面的按钮分别实现了**开始游戏、暂停游戏、继续游戏、重新开始、退出游戏、保存游戏、载入游戏**这7个功能；其中工具栏和游戏界面的按钮以**图标**形式显示。

7个功能也另外设置了快捷键，用户通过菜单栏/工具栏/游戏界面/快捷键方式使用均可，且效果完全相同。

当功能处于**不可用状态**时，界面按钮图标变灰，且所有方式均无法使用该功能。

界面底部是**StatusBar**，当鼠标移动到7个功能在菜单栏/工具栏/游戏界面的相应按钮位置时，StatusBar将显示对应功能的详细解释以提示用户。

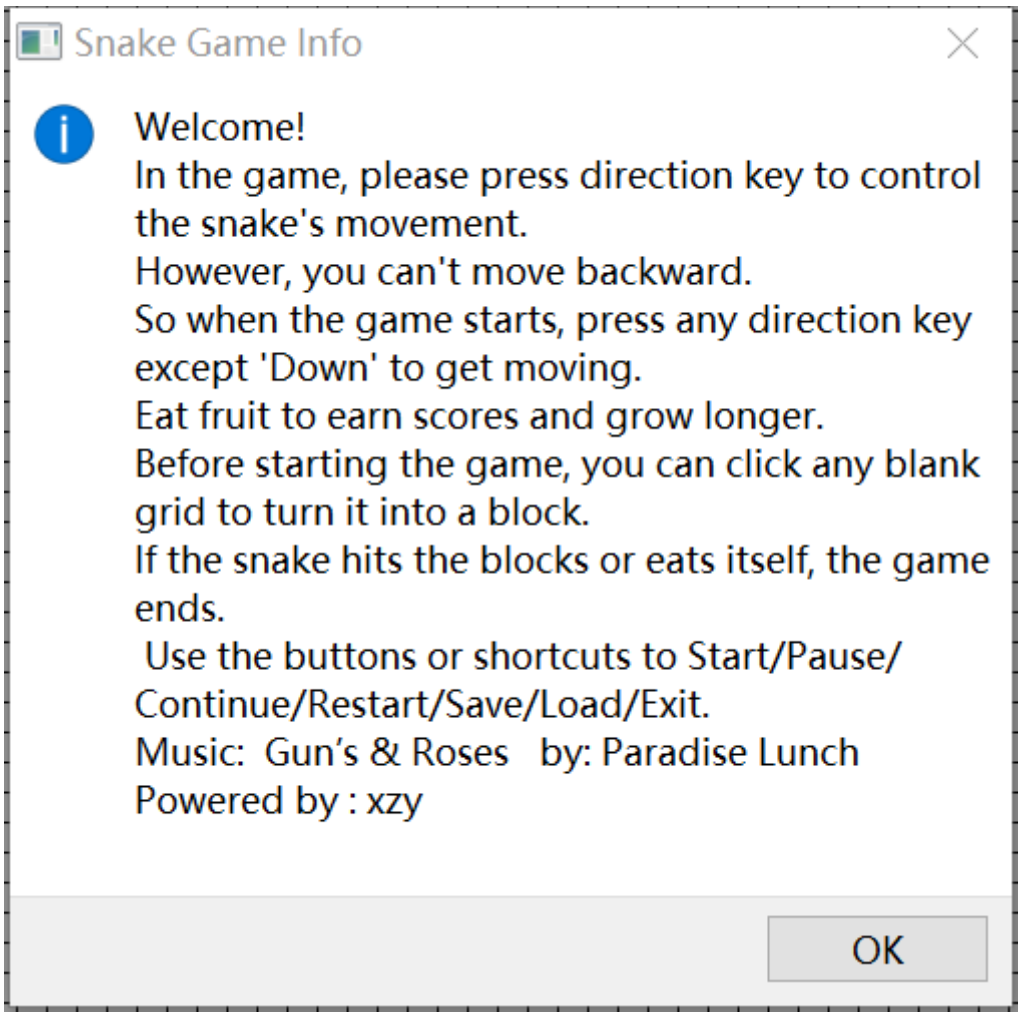


界面右侧有**Step**和**Score**计数器，分别记录从游戏开始以来贪吃蛇移动的步数、以及贪吃蛇吃的果实数。

1.2 使用帮助

在菜单栏中的Help一项中点击Info，弹窗出现游戏的使用帮助。

游戏规则可见弹窗内容，与大作业规定相同。

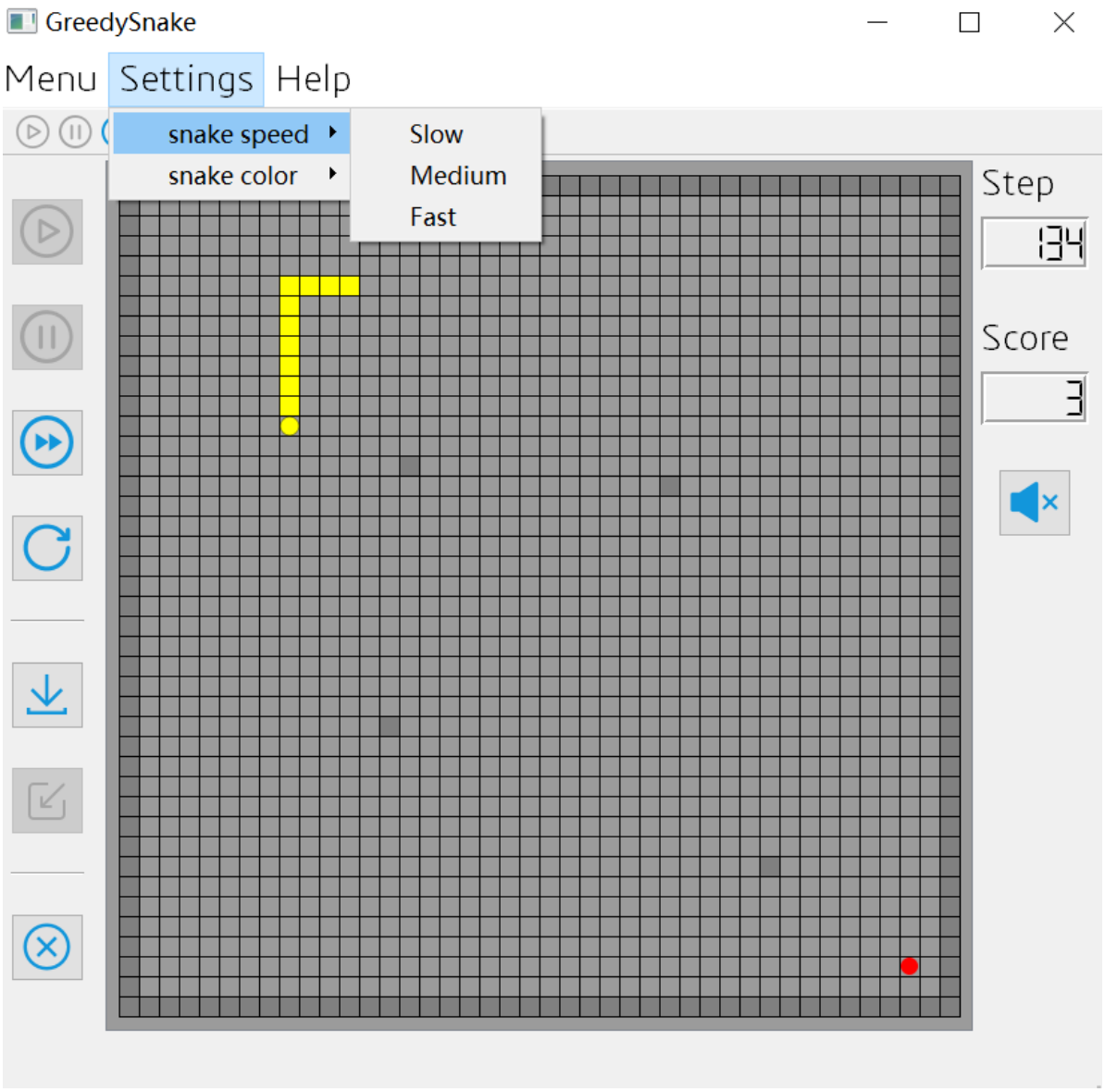


1.3 游戏设置

本功能位于菜单栏中的Settings项。

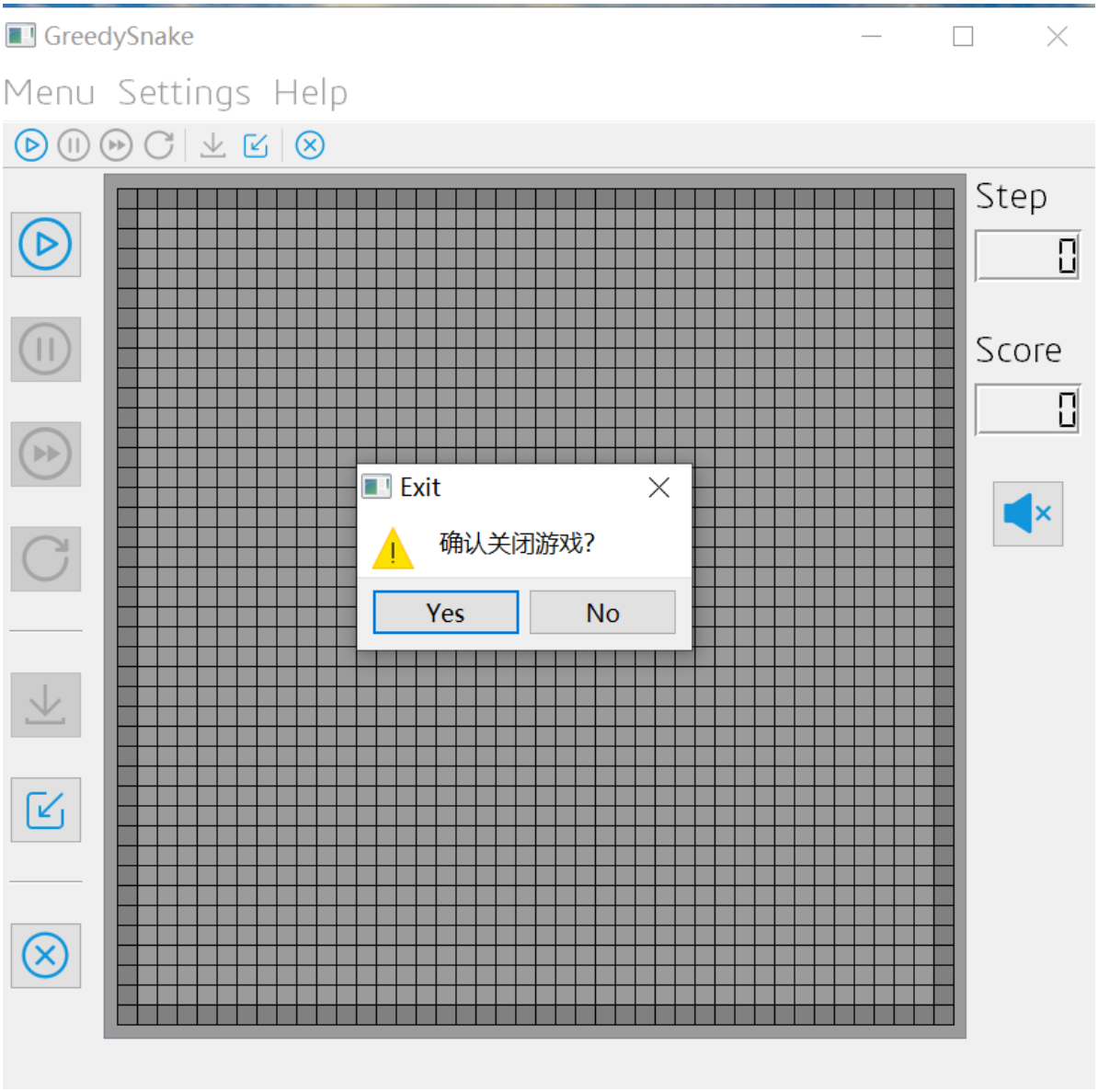
用户可以随时选择snake_speed（有Slow, Medium, Fast三个选项）来更改贪吃蛇的速度；选择snake_color（有Yellow，Green两个选项）实时更改贪吃蛇身体颜色。重启游戏并不会更改速度与颜色设置。

打开游戏时，速度默认为Medium，颜色默认为Yellow。



1.4 关闭提醒

关闭游戏时，将跳出“确认关闭游戏”的弹窗。如果选择“Yes”，则窗口关闭；如果选择“No”，则弹窗关闭，保持原来游戏状态。



1.5 背景音乐

点击界面右侧设有喇叭图标的按钮可以播放或静音背景音乐。进入游戏时默认处于静音状态。

2. 设计思路

2.1 UI界面

使用Qt Designer设计。

UI界面上方是菜单栏和工具栏，主体部分从左到右分为三块（使用水平布局）：左边是7个垂直排列的按钮，中间是GraphicsView显示游戏主界面，右边是步数和得分的计数器以及背景音乐按钮。

2.2 MainWindow：

继承`QMainWindow`，主要功能是处理与UI界面、用户交互相关的内容（主要包括UI界面最开始的初始化、处理菜单工具栏和功能按键的点击、Settings设置、展示更新的游戏得分等）。

游戏主体部分主要运用`QGraphicsView`、`QGraphicsScene`和`QGraphicsItem`三类编写。

2.3 view：

是UI界面的子部件（`QGraphicsView`类），用于展示游戏场景（scene），实现scene的图像化。

2.4 scene：

是MainWindow的私有成员（`QGraphicsScene`类），场景可添加或删除各个物品（包括格子、障碍物、水果、蛇），并通过循环调用`advance()`函数来实现场景更新（scene的`advance()`函数将调用在场景内所有item的`advance()`函数以实现这一功能）。

2.5 game：

MainWindow的私有成员，程序中唯一的游戏管理器（`Game`类，继承QObject），用于控制游戏的开始、停止、继续、结束，游戏过程中负责开始或暂停循环计时器、记录本局游戏的步数和分数，并将更新传达到MainWindow更新UI界面；内部存有指向`scene`的指针，决定scene中物体的添加和删除。

2.6 Items：

本游戏实现了QGraphicsItem类的四个子类Grid, Block, Fruit, Snake，由`game`负责创建和删除。

QGraphicsItem类有两个纯虚函数`paint`和`boudingRect`，其中`paint`需要实现这个物品的图像，`boundingRect()`需要返回能够包围住物品图像的矩形。这两个函数为QGraphicsView提供了必要的信息。除了这两个函数，四个子类分别实现了一些独有的功能。

- **Grid**：格子，共有40*40个，每个格子都作为一个类对象。
- **Block**：（潜在）障碍物，共有42*42个（包括四周的围墙），每个障碍物作为一个类对象。

`game`会控制不同游戏时段grid和block是否enable（只有当游戏处于未开始状态时它们是有效的）。

每局游戏准备开始（未开始状态）时，`game`会清空`scene`中的所有物品，然后构造40*40的Grid和42*42的Block并将它们加入到`scene`内。其中围墙处的block处于可见状态，其他位置的block不可见。用户可以点击任意位置（除了蛇所在的两格和围墙）的grid，程序通过调用grid的`mousePressEvent`函数，将对应位置的block设为visible。block覆盖在grid之上，故如果再次点击该位置，block被选中，它的`mousePressEvnet`函数会将自己隐藏起来。

- **Fruit**：果实。（并没有什么值得说明的地方）
- **Snake**：

  蛇。蛇的头是圆形，身子是矩形，设计尽管简陋，却可以完（简）全（陋）避免载入游戏存档的时候找不到蛇头的情况。

  类内存有私有成员`direction`记录蛇的方向、`QList<QPoint>`记录蛇身体的位置、`willgrow`整型保存接下来蛇将变长几格。

  另外，Snake类内存有指向`game`游戏控制器的指针，在Snake类的`advance`函数中，将处理以下事件：

  1、如果游戏刚刚开始，还没有按下第一个有效方向键，蛇不动，直接返回；

  2、游戏步数增加，调用game中相应接口处理并记录，game中接口再发射信号，ui界面计数器更新；

  3、更新蛇的位置信息，检查蛇是否撞到障碍物或自己，如果撞到了，调用game中`gameOver`接口处理，接口停止循环计时器、再发出游戏结束信号，MainWindow中槽函数更新游戏状态、更新按钮的Enabled状态。

  4、检查蛇是否吃到果实，如果吃到了，首先更新`willgrow`变量，然后调用game中`fruitEaten`接口处理，game中的接口函数发射信号更新ui界面分数、并随机一个新的果实更新到scene中。

3. 流程概括

- 用户通过menuBar/toolBar/PushButtons/shortCuts激发某个功能action后，MainWindow处理与用户**交互**相关部分内容（例如文件对话框、更新可用或不可用功能），然后调用game中相应接口处理与**游戏**相关的内容（例如游戏文件的具体输入`inputData`输出`outputData`，游戏的初始化`GameInit`、开始`GameStarts`、暂停`GamePause`、继续`GameContinue`、结束`GameOver`等都有相应函数处理）。
- game中的接口将对scene里应该有哪些item、这些item的状态是什么进行**管理**，并做好**游戏状态**的记录（必要时通知mainWindow进行相关处理和更新）、游戏循环更新**计时器**的管理、一些**初始化**工作，处理`KeyEvent`并更新snake的方向。
- game中向下存有通往各个item的指针，item在必要时（如蛇撞墙、吃了果实）也会通知game处理；game中向上存有通往scene的指针，能够管理场景中的item及状态；game也会与mainwindow进行沟通，用于两个计时器的更新、通知游戏终止。