贪吃蛇程序设计文档

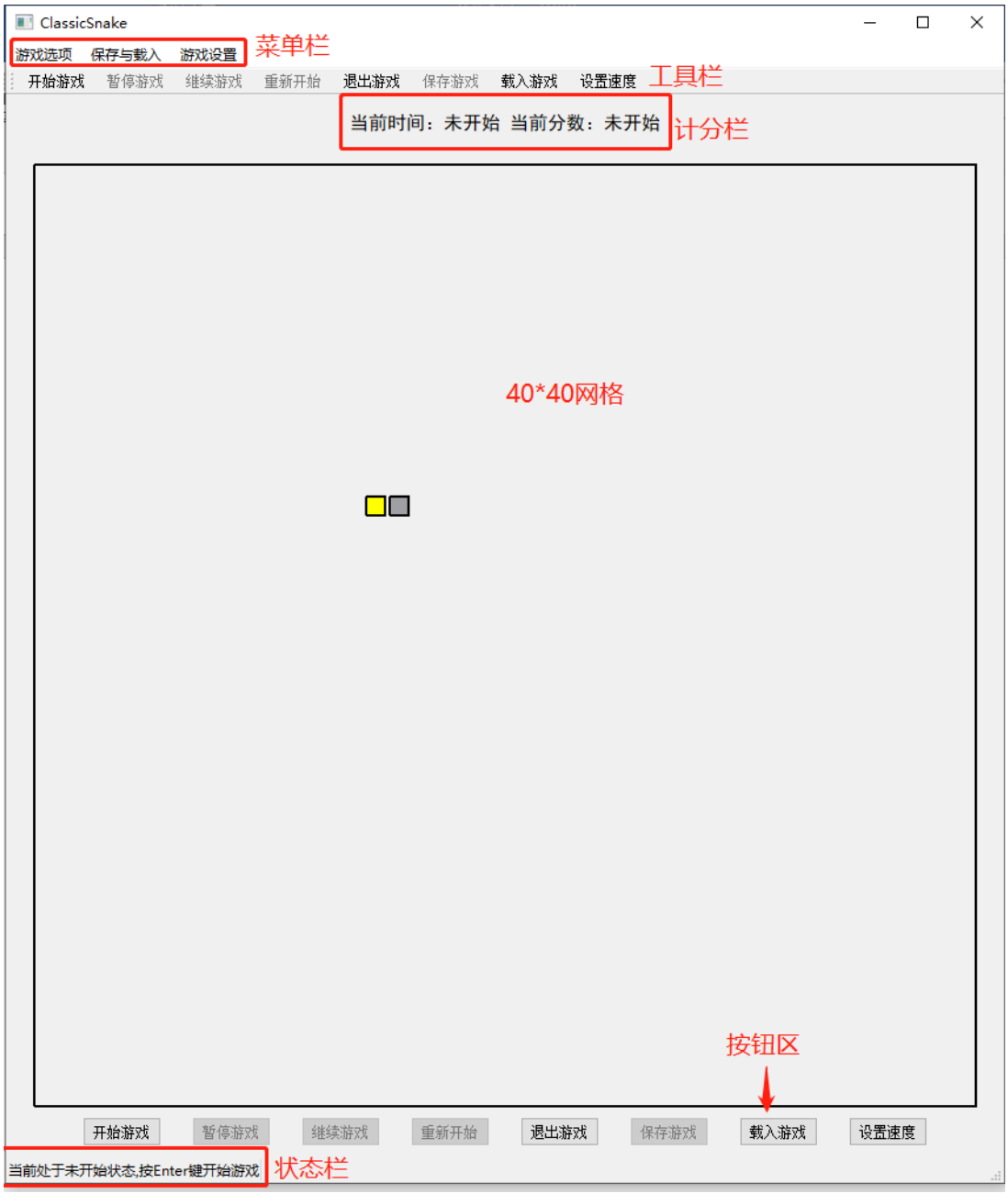
1. 总体结构介绍

1.1 程序简介

本程序是实现贪吃蛇游戏的程序。本程序基于 Qt 编写，基于 QMainWindow 类，实现了贪吃蛇的基本功能。在程序中，玩家可以操控贪吃蛇在 40*40 的网格上下左右移动。玩家可以自行设置障碍物，果实由系统随机生成。玩家可以操控贪吃蛇躲避障碍物，不能超出边界，并尽量吃到果实。玩家控制的贪吃蛇离开 40*40 的网格或者咬到自身或者撞到障碍物则视为游戏失败，游戏失败程序会弹出窗口提示玩家，程序进入终止状态。本游戏没有胜利条件，存活的条件下，吃的果实越多，玩家控制的贪吃蛇越长，操控难度越高，分数也越高。玩家在游戏中可以随时暂停和保存当前的游戏状态，下次开始游戏时可以读取先前存储的游戏状态。另外玩家还可以自行设置时间流逝的速度(0.01 倍-10.00 倍速)，时间流逝得越快，游戏难度越高。游戏界面如下：

1.2 程序界面及功能介绍

程序主界面如下：



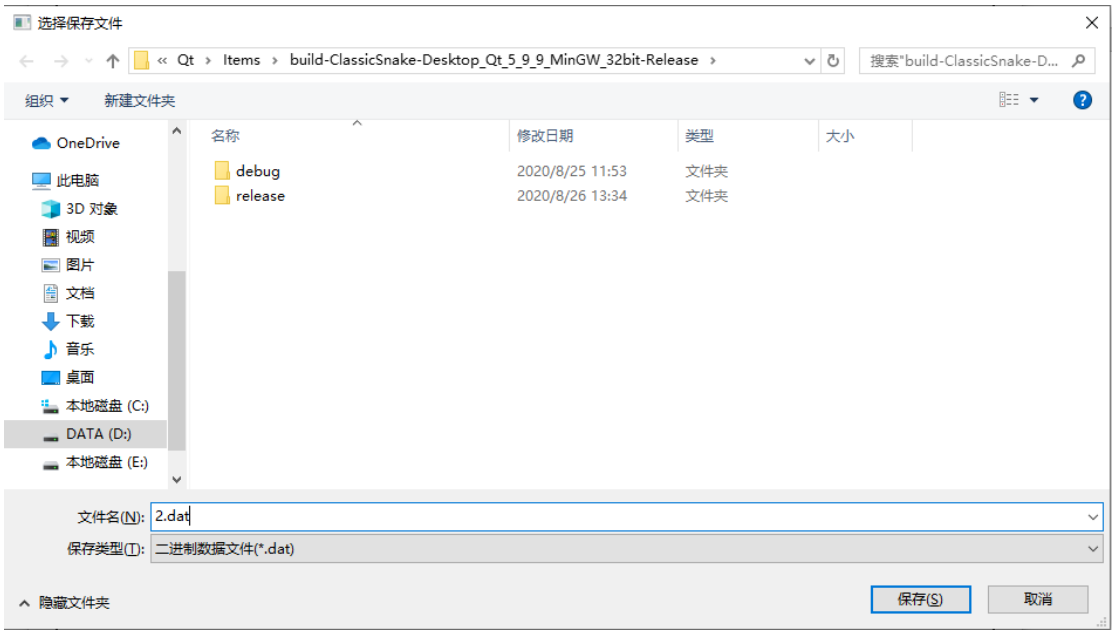
程序界面由六个部分组成：菜单栏，工具栏，计分栏，40*40 网格，按钮区，状态栏。程序界面设计中加入了布局管理和弹簧，因此改变窗口大小时，程序能自动调整布局，确保整体布局合理美观，游戏界面居中。

1.2.1 菜单栏

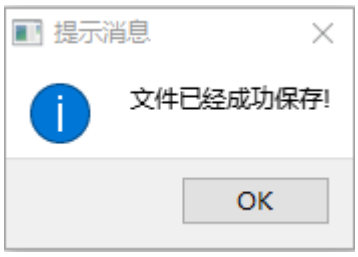
菜单栏共有三个菜单：游戏选项，保存与载入，游戏设置。

游戏选项菜单下有五个菜单项：开始游戏，暂停游戏，继续游戏，重新开始，退出游戏。开始游戏选项当程序处于未开始状态时激活，玩家点击开始游戏，程序即进入到游戏状态，游戏运行画面不断更新，此时玩家可以操纵贪吃蛇。开始游戏的快捷键为 Enter 键；暂停游戏选项当程序处于游戏状态时激活。玩家点击暂停游戏，程序进入暂停状态，此时画面停止更新，玩家不可操纵贪吃蛇，但是可以保存游戏。暂停游戏的快捷键为 P 键；继续游戏选项当程序处于暂停状态时激活。玩家点击继续游戏，程序又恢复到游戏状态，此时玩家可以操控贪吃蛇。继续游戏的快捷键为 Space 键；重新开始选项当程序处于终止状态或暂停状态时激活。玩家点击重新开始，未保存的游戏进度丢失，40*40 的网格恢复到初始模式，程序进入未开始状态。重新开始的快捷键为 Backspace 键；在程序运行的任一时刻都可以点击退出游戏，点击该选项后，游戏退出，未保存的进度丢失。退出游戏的快捷键为 Alt+X 键。

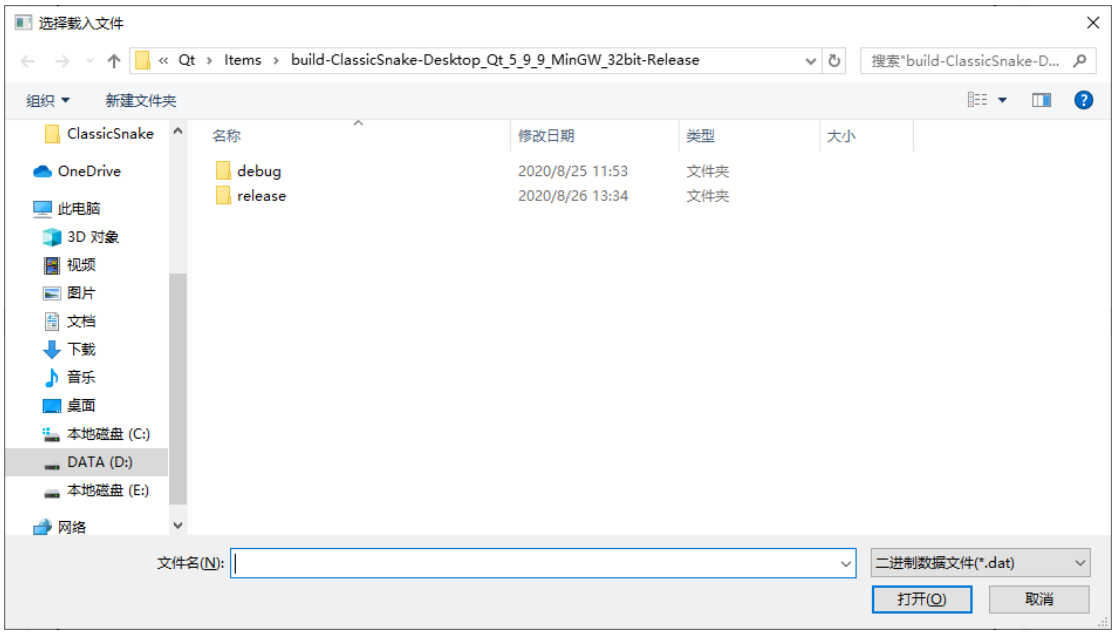
保存与载入菜单下有两个菜单项：保存游戏、载入游戏。保存游戏选项在程序处于暂停状态时激活，玩家点击保存游戏可以将当前游戏的进度保存到自己指定的二进制文件中（如下图）。



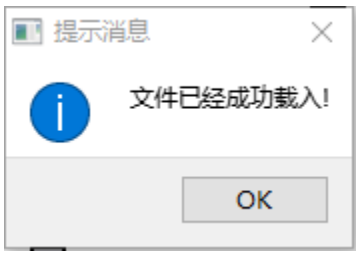
保存游戏的快捷键为 Ctrl+S。保存成功后，程序会弹出提示信息（如下图）。



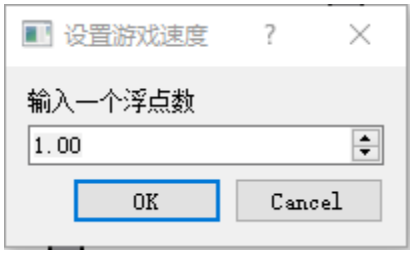
载入游戏选项在程序处于未开始状态时激活，玩家点击载入游戏可以将以往的游戏进度导入到当前程序中（如下图）。



载入游戏的快捷键为 Ctrl+L。载入成功后，游戏进入暂停状态，程序会弹出提示信息（如下图）。



游戏设置菜单下只有一个菜单项：设置速度。玩家可以在程序运行的任一时刻点击此菜单项，点击此菜单项后，程序会弹出一个设置速度的窗口（如下图）。



玩家可以输入一个浮点速，并按下“OK”即可完成速度的设置。如输入 2.00，再按下“OK”，则游戏的时间流逝速度设置为默认速度的 2 倍。设置速度的快捷键是 Alt+S。

1.2.2 工具栏

工具栏上列举了游戏的所有功能，每一个工具项对应了菜单栏中的同名菜单项。点击工具栏的工具项和点击菜单栏中的同名项是完全等效的。把它们列举在工具栏是为了使程序界面更加完整以及方便玩家进行界面交互。

1.2.3 计分栏

计分栏由一个 QLabel 实现。计分栏在程序处于未开始状态时会显示两个“未开始”，在程序处于其他状态时则会显示当前进度累计流逝的游戏时间（也就是贪吃蛇前进的格子数）以及累计的分数（每吃一个果实得 5 分），如下图。

1.2.4 40*40 网格

40*40 网格是游戏的区域，也是所有游戏元素可能出现的位置。每个网格的大小相等，都为 20*20（pixels^2）的正方形，且各个网格无缝相接构成一个点阵。特别地，网格在游戏界面中是不可见的，玩家仅仅能看到 40*40 网格点阵的边界，用黑色实线表示，玩家在控制贪吃蛇移动时不可超出这个边界。

下面是游戏中出现的各个元素的介绍：

上图是玩家控制的贪吃蛇，黄色黑边的正方形为贪吃蛇的头部，灰色黑边的正方形为贪吃蛇的一节身子。贪吃蛇的头部和每节身子实际各占一个方格的位置，但在显示上要比一个方格的大小略小一些。贪吃蛇的初始长度为 2，即一个头部和一节身子。贪吃蛇只能前进不能后退，玩家通过↑↓←→控制贪吃蛇的前进方向，但贪吃蛇不可以原地掉头（也就是贪吃蛇不能往当前行进方向的相反方向变向）。贪吃蛇碰到障碍或者离开游戏区域或者头部触碰到身体的任何一节则游戏结束。贪吃蛇的头部接触到果实后，接下来三个单位时间，尾部不动，头部延长三个格子。

上图是游戏区域随机生成的果实。果实占一个方格的位置，但在显示上要比一个方格的大小更小一些。同一时间游戏区域内只能有一个果实。每当一个果实被吃掉后，会随机在空白的网格上产生另一个果实（即果实不会与障碍和贪吃蛇重合）。

上图是障碍。障碍占一个方格的位置，在显示上与一个方格的大小一致。玩家可以在未开始状态下用鼠标左键点击空白的网格设置障碍，若点击位置已有障碍，则会取消原有的障碍。

1.2.5 按钮区

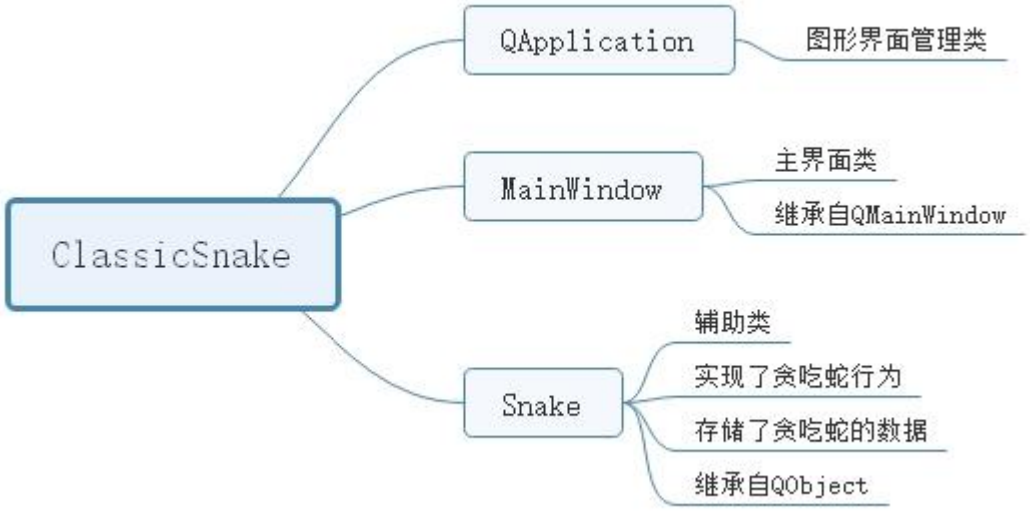
按钮区由若干个 QToolButton 组成。按钮区的按钮包括了游戏的所有功能，每一个按钮对应了菜单栏中的同名菜单项。点击按钮区的按钮和点击菜单栏中的同名项是完全等效的。

把它们列举在按钮区是为了使程序界面更加完整以及方便玩家进行界面交互。

1.2.6 状态栏

状态栏用于提示用户程序当前所处的状态，以及进入其他状态的快捷键。

1.3 程序结构及设计思路介绍



本程序大致有三个主要的类，QApplication 作为图形界面的管理类，其实例对象创建于主函数中并发挥作用。本程序使用了 Qt 内置的 QApplication，并没有对其进行修改。MainWindow 是程序的主界面类，实现了图形界面的主要交互功能，其实例对象创建于主函数中并调用了 show 成员函数用于界面显示。Snake 类是定义贪吃蛇行为和状态的类。本程序根据面向对象的思想，把贪吃蛇这一行为比较复杂的对象独立出来成为一个类。Snake 类的实例对象是 MainWindow 类的一个成员变量。

注意到菜单项、工具栏和按钮区对应项的功能是完全一致的。因此没有必要为它们各自设置槽函数，比较明智的方法是用 Action 抽象各个项的行为。因此本程序为每个项都设置了 Action，并编写其触发槽函数，接着将 Action 绑定到各个区域对应的项中。例如对于按

钮区的有一个 QToolButton，我们为它绑定开始游戏的 Action，这样它就实现了开始游戏按钮的功能。

本程序实现的难点在于界面更新与状态管理，这也是本程序的创新之处。

界面更新。本程序在 MainWindow 类中定义了两个 QTimer*定时器：updateScreenTimer 和 updateSystemTimer。它们分别负责定时更新画面状态和系统组件状态，定时周期默认分别为 120ms 和 100ms。

状态标识。出于程序可读性的考虑，本程序使用定义在 MainWindow 类中的枚举类型 Status 来表示不同的程序状态，其中 Ready 表示未开始状态，Gaming 表示游戏状态，Stop 表示暂停状态，End 表示游戏结束。

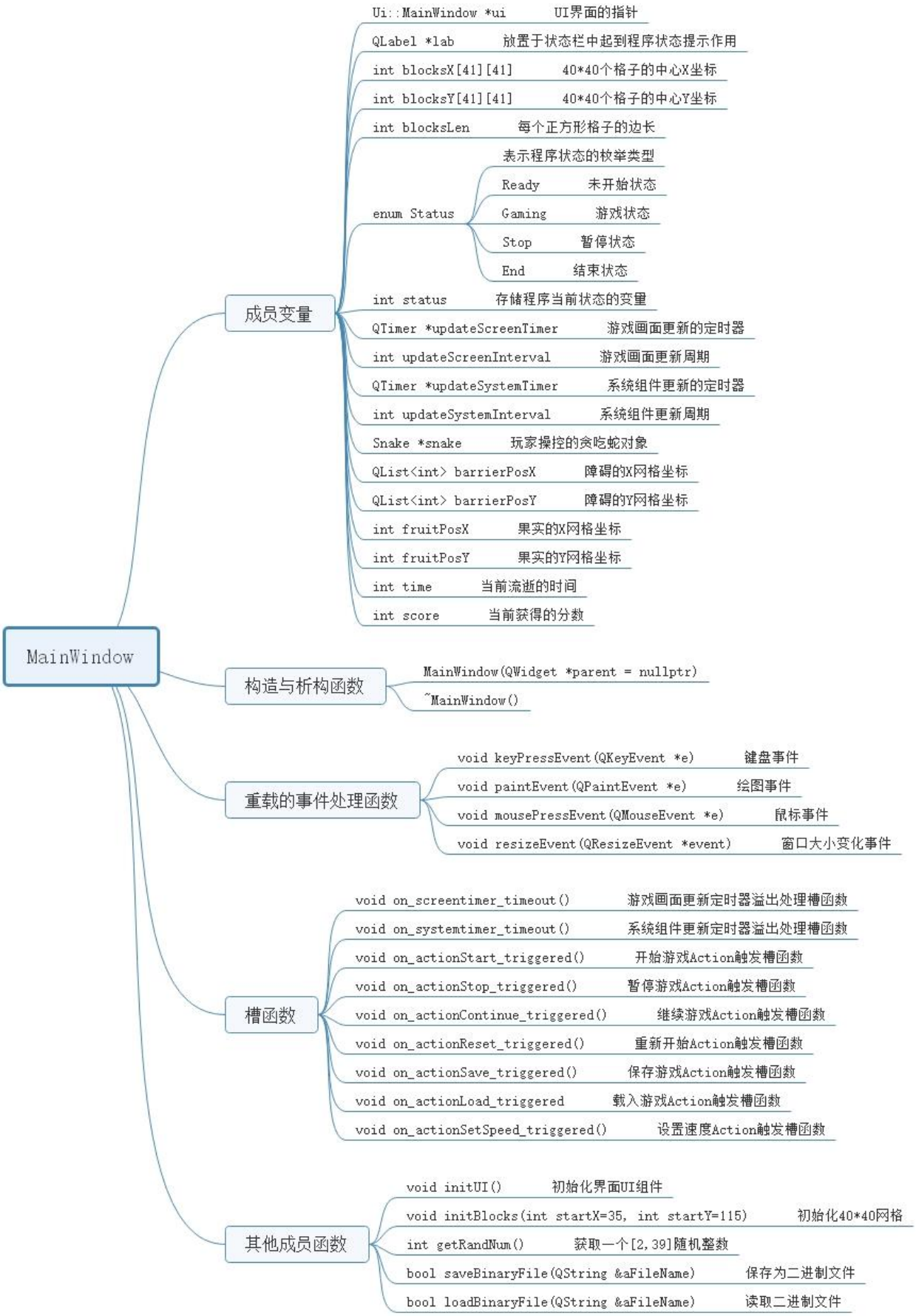
状态管理。状态管理分为两个部分，一个是当前状态的权限管理和转移管理。当前状态的权限管理的实现方式主要通过枚举类型判断实现，在定时器处理函数和一些事件处理函数中都会根据当前状态的枚举值进行不同的处理，这就确保了某一状态不会获得其他状态的权限。至于状态转移管理，状态转移主要有两种实现方式，用户触发菜单项（本质是 QAction），以及达成失败条件。对于前者，由于在每个状态都有严格的权限管理，因此不会出现错误的状态转移。例如在游戏状态时开始游戏菜单项是 disable 的，用户根本无法点击，因此就不会触发错误的状态转移。此外，本程序在每个菜单项对应的 QAction 的槽函数都设定了转移到某个状态的预处理操作，如对于暂停游戏菜单项，其槽函数实现了将当前的状态设置为 Stop，并且让 updateScreenTimer 停止计时，这样就进入到了暂停状态。对于后者，程序在结算失败条件的位置设置了转移到结束状态的预处理操作。总之，对于状态管理，每个状态的进入操作以及退出操作都是确定的，每个状态的权限也是确定的，因此状态之间转移的正确性也是确定。这里的设计思路借助了有限自动机的有关思想。

2. 主要接口介绍

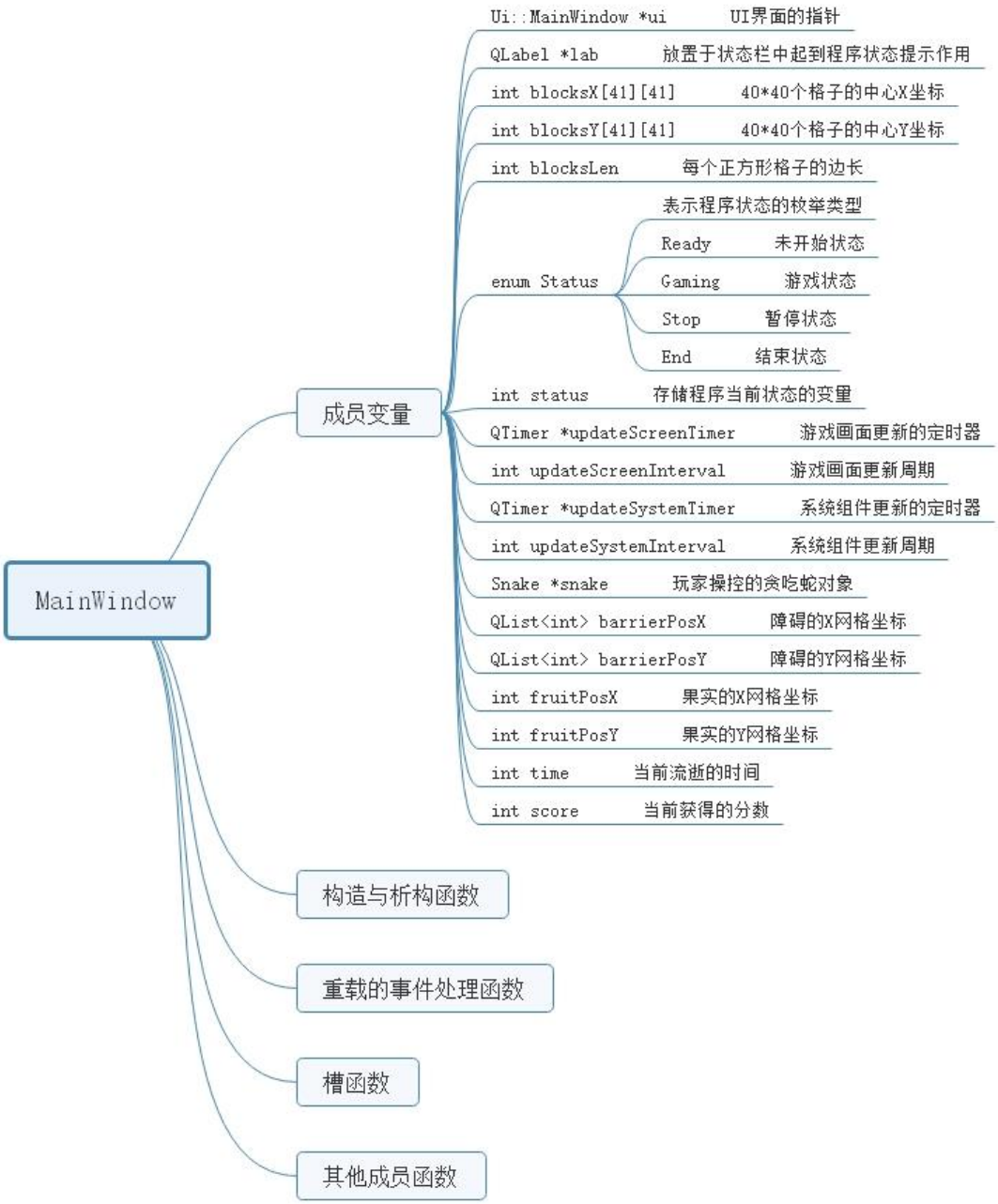
2.1 MainWindow 类介绍

2.1.1 整体结构

MainWindow 类是程序主界面的实现类，也是整个图形界面显示与交互的核心实现类。该类继承自 QMainWindow，具备 QMainWindow 的一切基本接口。在此基础上，MainWindow 还额外定义了一些成员变量与成员函数，便于更好地与用户进行交互，处理用户的输入以及保存游戏过程中产生的数据。以下是该类的整体结构图，结构图中只标明了 MainWindow 类相比 QMainWindow 新增加的定义与重载的函数。


MainWindow
成员变量
Ui::MainWindow *ui UI界面的指针
QLabel *lab 放置于状态栏中起到程序状态提示作用
int blocksX[41][41] 40*40个格子的中心X坐标
int blocksY[41][41] 40*40个格子的中心Y坐标
int blocksLen 每个正方形格子的边长
enum Status 表示程序状态的枚举类型
Ready 未开始状态
Gaming 游戏状态
Stop 暂停状态
End 结束状态
int status 存储程序当前状态的变量
QTimer *updateScreenTimer 游戏画面更新的定时器
int updateScreenInterval 游戏画面更新周期
QTimer *updateSystemTimer 系统组件更新的定时器
int updateSystemInterval 系统组件更新周期
Snake *snake 玩家操控的贪吃蛇对象
QList<int> barrierPosX 障碍的X网格坐标
QList<int> barrierPosY 障碍的Y网格坐标
int fruitPosX 果实的X网格坐标
int fruitPosY 果实的Y网格坐标
int time 当前流逝的时间
int score 当前获得的分数
构造与析构函数
MainWindow(QWidget *parent = nullptr)
~MainWindow()
重载的事件处理函数
void keyPressEvent(QKeyEvent *e) 键盘事件
void paintEvent(QPaintEvent *e) 绘图事件
void mousePressEvent(QMouseEvent *e) 鼠标事件
void resizeEvent(QResizeEvent *event) 窗口大小变化事件
槽函数
void on_screentimer_timeout() 游戏画面更新定时器溢出处理槽函数
void on_systemtimer_timeout() 系统组件更新定时器溢出处理槽函数
void on_actionStart_triggered() 开始游戏Action触发槽函数
void on_actionStop_triggered() 暂停游戏Action触发槽函数
void on_actionContinue_triggered() 继续游戏Action触发槽函数
void on_actionReset_triggered() 重新开始Action触发槽函数
void on_actionSave_triggered() 保存游戏Action触发槽函数
void on_actionLoad_triggered 载入游戏Action触发槽函数
void on_actionSetSpeed_triggered() 设置速度Action触发槽函数
其他成员函数
void initUI() 初始化界面UI组件
void initBlocks(int startX=35, int startY=115) 初始化40*40网格
int getRandNum() 获取一个[2,39]随机整数
bool saveBinaryFile(QString &aFileName) 保存为二进制文件
bool loadBinaryFile(QString &aFileName) 读取二进制文件

2.1.2 **成员变量介绍**



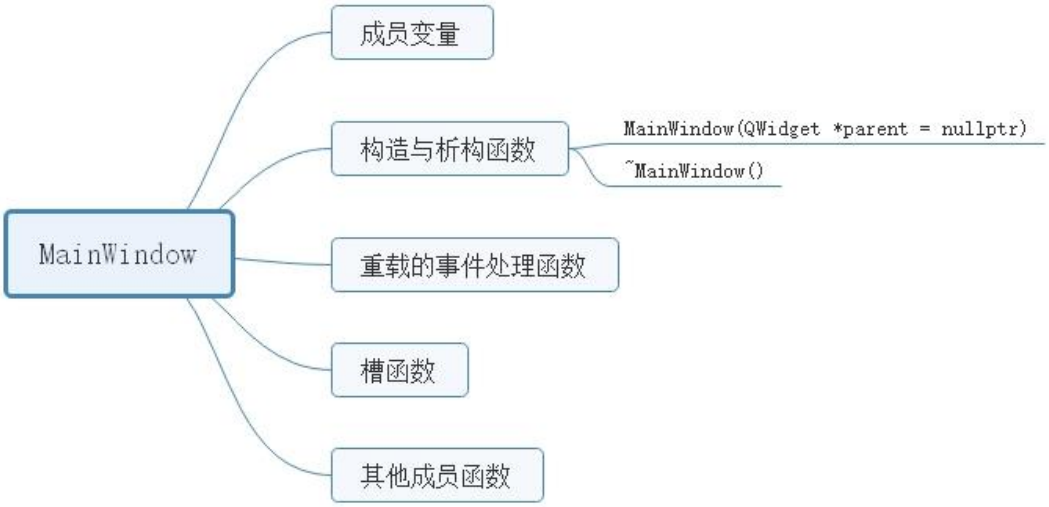
上图比较详细地介绍了各个成员变量设置的意图。下面对成员变量的设置细节做一些补充，其余不多做赘述。

成员变量全部为私有的，这是为了保证类的良好封装性。并且在程序中确实不存在外界访问 MainWindow 类的成员变量的情况。

blocksX 和 blocksY 是两个 41*41 的二维数组，用于存储各个网格在界面中的像素坐标。这里开了 41*41 而非 40*40，是为了使每个格子的下标从 1 开始，符合一般的习惯。此外，每个格子的位置可以由它在二维数组中的下标唯一确定。又由于游戏中的所有元素仅会出现在某个格子的位置，因此我们就没有必要记录元素的像素坐标，只需记录它在哪个网格就可以了。结构图中的“网格坐标”指的就是这样的坐标。例如一个果实的网格 X 坐标为 15，网格 Y 坐标为 20，这样这个果实的像素坐标就是(blocksX[15], blocksY[20])，我们在绘制的时候只需要根据这个像素坐标绘制即可。

两个 QTimer 类的指针 updateScreenTimer 和 updateSystemTimer 分别用于更新游戏的画面和系统各个组件的状态。由于游戏画面仅在程序处于游戏状态时才会更新，因此前一个定时器只会在游戏状态时才会触发，其余状态都会停止。而系统的各个组件在程序处于任一状态时都需要更新，特别地，我们需要根据程序当前所处的状态来更新组件的状态，因此后一个定时器必须在程序运行的每一时刻都要工作。这是二者最大的区别。

2.1.3 构造与析构函数介绍



MainWindow(QWidget *parent = nullptr)

返回值：none

参　数：QWidget *类型的父指针。

功　能：构造 MainWindow 类主界面并且初始化各个成员变量。

实　现：通过初始化列表初始化各个成员变量（初始化为默认值）。特别地，将 blocksLen 初始化为 20（每个格子边长为 20 像素），将 status 初始化为 Ready（游戏进入即为未开始状态），将 updateScreenInterval 初始化为 120（画面更新的默认周期为 120ms），将 updateSystemInterval 初始化为 100（系统更新的默认周期为 100ms），将 fruitPosX 和 fruitPosY 都初始化为-1（果实网格坐标为(-1,-1)表明还没有果实）。接着调用 initUI()初始化 UI，再调用 initBlocks()初始化 40*40 网格。最后初始化两个定时器，将两个定时器的 timeout()信号与对应的槽函数（on_screentimer_timeout()以及 on_systemtimer_timeout()）关联。特别地，此时 updateSystemTimer 出于开始状态，而 updateScreenTimer 处于停止状态（因为初始为未开始状态，画面不需要更新）。

~MainWindow()

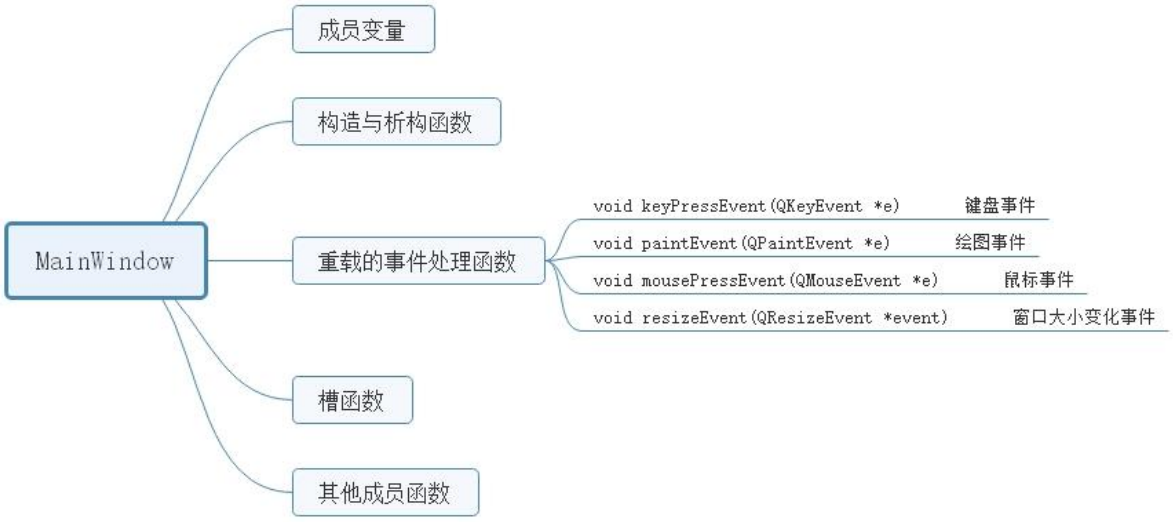
返回值：none

参　数：none

功　能：析构 MainWindow 类主界面。

实　现：删除 ui 指针指向的内存。

2.1.4 事件处理函数介绍



void keyPressEvent(QKeyEvent *e)

返回值：none

参　数：QKeyEvent *类型的事件指针。

功　能：处理用户的键盘输入事件，也就是控制贪吃蛇方向的输入。

实　现：该函数只有在 Gaming 状态下才会对键盘输入事件进行处理。首先获取 e 指针的 key()返回值（也就是获取用户按下了哪个键）。处理↑↓←→键：通过 snake 指针的 getNextHead()函数获取按照用户指定的方向前进一个格子后的蛇头坐标，再通过 snake 指针的 getSecondPos()函数获取当前蛇的第二节身子的位置（头部的下一节）。如果二者为同一个位置（说明蛇掉头了），那么就不作任何处理直接返回；否则通过 snake 指针的 setDirection ()函数设置蛇的移动方向为用户的指定方向。

void paintEvent(QPaintEvent *e)

返回值：none

参　数：QPaintEvent *类型的事件指针。

功　能：完成画面的绘制。

实　现：依次绘制障碍、贪吃蛇和果实和网格的外边界。如果果实坐标为(-1, -1)，说明果实不存在，那么就不绘制。障碍为纯黑色，大小与一个格子相同；果实为纯红色，边长比格子小 10 个像素；蛇头为黑边黄心，蛇身为黑边灰心，二者的边长均比一个格子小 4 个像素。

void mousePressEvent(QMouseEvent *e)

返回值：none

参　数：QMouseEvent *类型的事件指针。

功　能：处理用户的鼠标输入事件，也就是绘制障碍的输入。

实　现：该函数只有在 Ready 状态下才会对键盘输入事件进行处理。获取 e 指针的 button()返回值（也就是获取用户按下了哪个键）。处理鼠标左键：首先找出鼠标点击位置对应的网格序号(X, Y)。如果鼠标点击在网格边界外，那么直接返回。接着寻找(X, Y)位置是否已经有障碍，如果已经存在，则删除该障碍，否则在(X, Y)设置新障碍。最后调用 update()更新画面。

void resizeEvent(QResizeEvent *e)

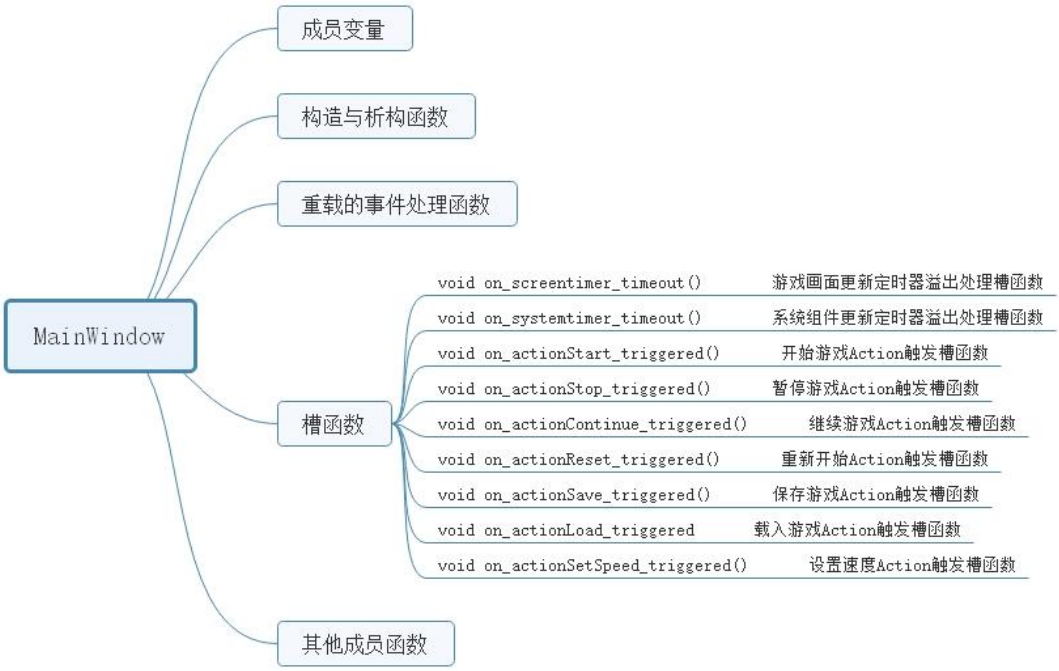
返回值：none

参　数：QResizeEvent *类型的事件指针。

功　能：对窗口大小改变事件做出响应，确保网格始终水平居中。

实　现：获取窗口当前的宽度，调用 initBlocks()更新各个格子的坐标，使网格整体居中。调用 update()更新画面。

2.1.5 槽函数介绍



void on_screentimer_timeout()

返回值：none

参　数：none

功　能：更新游戏画面相关的各个参数。

实　现：首先通过 snake 指针调用 getNextHead(snake->getDirection())获取蛇按照当前行进方向移动一步后的头坐标。判断这个坐标是否离开了网格边界或与障碍坐标重合或与贪吃蛇头部以外的部分重合。如果是，则将状态 status 置为 End、停止 updateScreenTimer 并弹出游戏结束的提示框，否则通过 snake 指针调用 move()让蛇向前移动一步，并让 time 自增 1，更新 lab 计分栏。如果移动后的蛇头部与果实坐标重合，那么通过 snake 指针调用 setStretch()设置蛇的拉伸余量为 3（也就是接下来可以伸长 3 个格子），并且 score 自增 5，更新计分栏。接着判断当前 40*40 网格上是否存在果实，如果不存在，则调用 getRandNum()生成随

机的且不与当前的贪吃蛇和障碍重合的果实坐标。最后调用 update()更新画面。

void on_ systemtimer_timeout()

返回值：none

参　数：none

功　能：更新各个 Action 的状态。

实　现：枚举 status 状态，根据状态更新各个 Action 的状态。例如当 status 为 Ready 时，将暂停游戏、继续游戏、重新开始、保存游戏对应的 Action 设置为 disable，将开始游戏、退出游戏、载入游戏对应的 Action 设置为 enable。

void on_actionStart_triggered()

返回值：none

参　数：none

功　能：实现开始游戏对应的 Action 的功能。

实　现：将 status 设置为 Gaming，开启 updateScreenTimer。

void on_actionStop_triggered()

返回值：none

参　数：none

功　能：实现暂停游戏对应的 Action 的功能。

实　现：将 status 设置为 Stop，关闭 updateScreenTimer。

void on_actionContinue_triggered()

返回值：none

参　数：none

功　能：实现继续游戏对应的 Action 的功能。

实　现：将 status 设置为 Gaming，开启 updateScreenTimer。

void on_actionReset_triggered()

返回值：none

参　数：none

功　能：实现重新开始对应的 Action 的功能。

实　现：将 status 设置为 Ready，关闭 updateScreenTimer。将各个变量恢复初始状态，删除 snake 指针指向的内容，并且重新分配的 Snake 实例对象，重置状态栏，最后调用 update()更新画面。

void on_actionSave_triggered()

返回值：none

参　数：none

功　能：实现保存游戏对应的 Action 的功能。

实　现：首先调用 QFileDialog::getSaveFileName()打开对话框让用户选择一个文件，接着调用 saveBinaryFile()将数据存入该文件中，如果保存成功则弹出对话框提示用户保存成功。

void on_actionLoad_triggered()

返回值：none

参　数：none

功　能：实现载入游戏对应的 Action 的功能。

实　现：首先调用 QFileDialog::getOpenFileName()打开对话框让用户选择一个文件，接着调用 loadBinaryFile()将该文件的数据读入内存中，如果载入成功则弹出对话框提示用户载入成功。接着将 status 设置为 Stop，关闭 updateScreenTimer，更新状态栏 lab，进入暂停状态，最后调用 update()更新画面。

void on_actionSetSpeed_triggered()

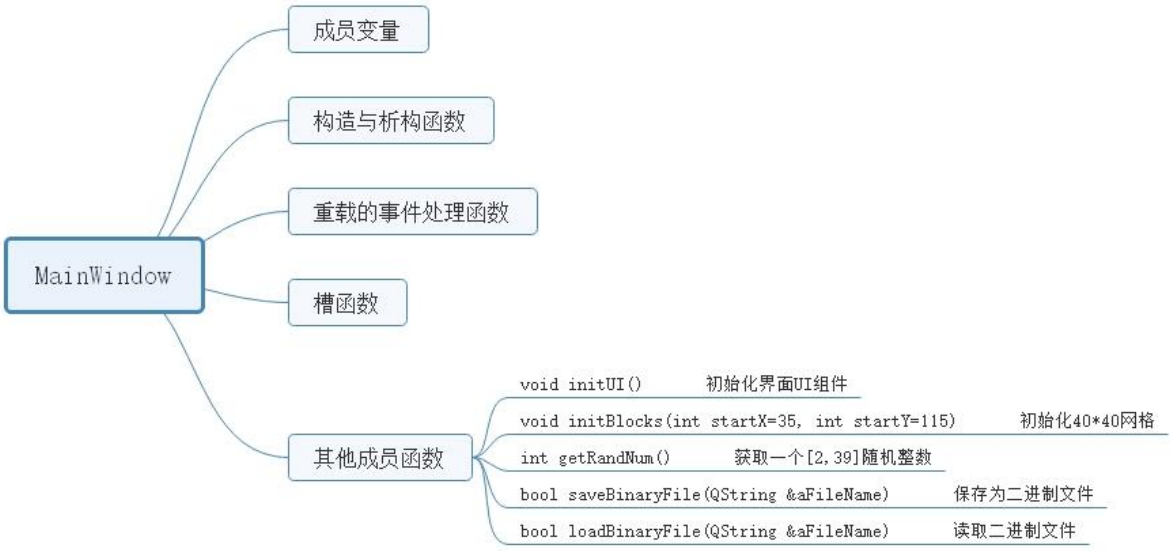
返回值：none

参　数：none

功　能：实现设置速度对应的 Action 的功能。

实　现：首先调用 QInputDialog::getDouble()打开对话框让用户输入一个浮点数。如果用户点击了确定，那么就将 updateScreenTimer 的定时周期设置为默认周期的 1/inputValue（setInterval(updateScreenInterval/ inputValue)，inputValue 为用户输入的浮点数）。用户输入的数值越大，速度越快，周期越短。浮点数的范围为[0.01, 10.00]。

2.1.6 其他成员函数介绍



void initUI ()

返回值：none

参　数：none

功　能：初始化主界面的 UI 组件。

实　现：首先调用每个 toolButton 的成员函数 setDefaultAction，为每个按钮绑定对应的 Action。接着为 centralWidget 设置强焦点模式，确保其能捕获按键信号，最后为 lab 设置文本，将 lab 添加到 statusbar 中。

void initBlocks (int startX, int startY)

返回值：none

参　数：int 类型变量 startX 和 startY。

功　能：初始化 40*40 网格中每个格子的中心像素坐标。

实　现：以 startX 和 startY 为左上角第一个格子的中心 X 像素坐标和 Y 像素坐标（即 blocksX[1][1]=startX, blocksY[1][1]=startY），以 blocksLen 为每个格子的边长，计算出每个格子的中心像素坐标并存入 blocksX 和 blocksY 中。

int getRandNum()

返回值：一个 int 类型变量

参　数：none。

功　能：获取一个[2, 39]的随机整数。

实　现：利用 qrand()模 40 得一个 40 以内的随机数，如果该随机数为 0，那么自增 2，如果为 1，那么自增 1。这样做是为了使随机数处于[2, 39]范围内，从而使得果实不会出现在网格边界上或网格边界外，降低吃果实的风险。

bool saveBinaryFile(QString &aFileName)

返回值：一个 bool 型变量

参　数：一个 QString &型变量。

功　能：将内存中的数据存入 aFileName 中。

实　现：首先以只写方式打开文件，若打开失败则返回 false。接着利用 QDataStream 依次将当前时间、分数、果实位置、障碍位置、蛇的状态存入文件中，最后关闭文件，返回 true。

bool loadBinaryFile(QString &aFileName)

返回值：一个 bool 型变量

参　数：一个 QString &型变量。

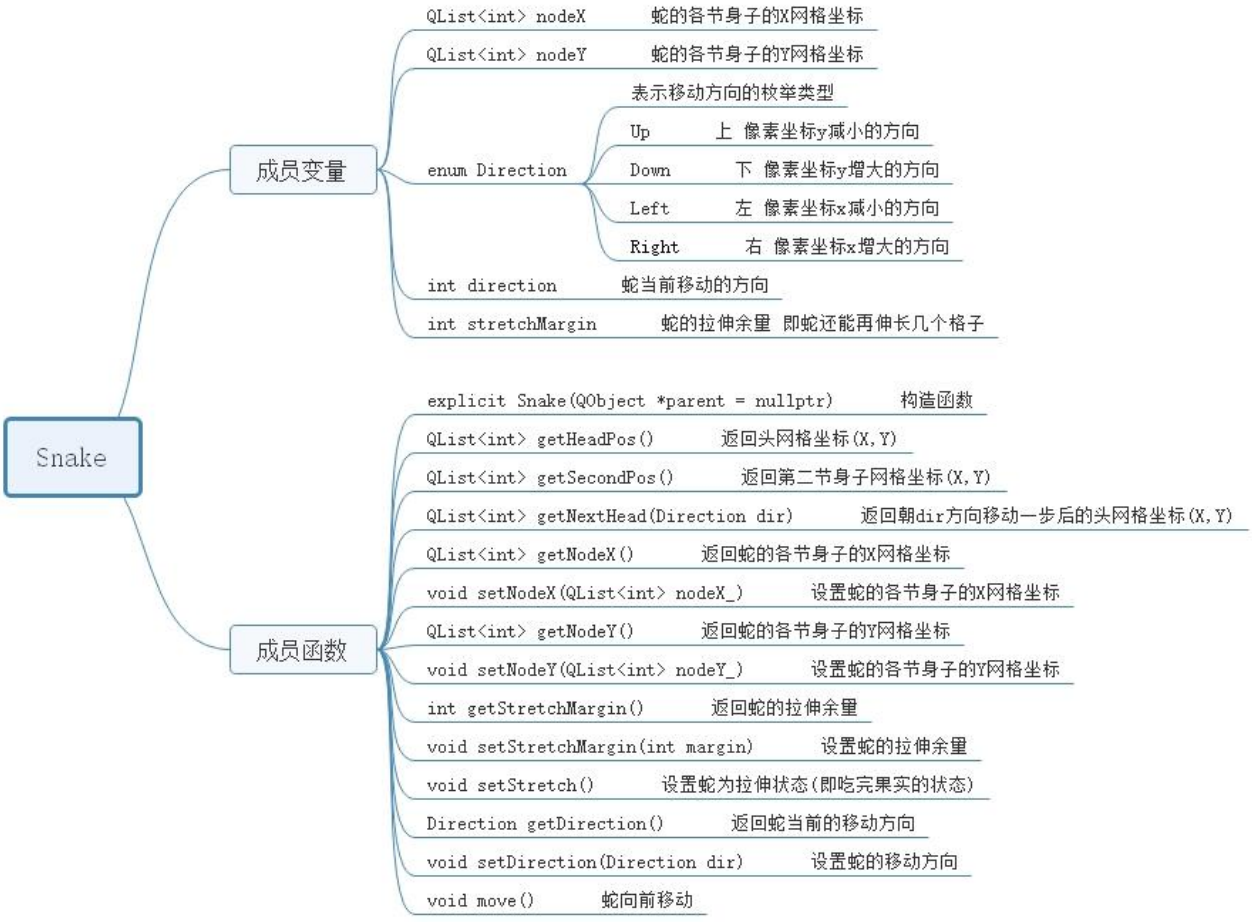
功　能：将 aFileName 中的数据读入内存中。

实　现：首先以只读方式打开文件，若打开失败则返回 false。接着利用 QDataStream 依次从文件中读入当前时间、分数、果实位置、障碍位置、蛇的状态，接着将相应的变量设置为从文件中读入的对应值，最后关闭文件，返回 true。

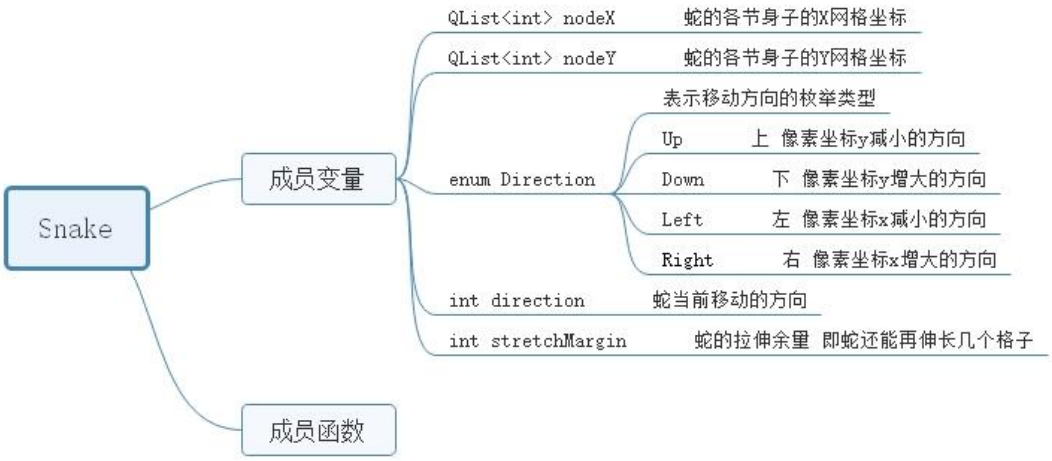
2.2　Snake 类介绍

2.2.1 整体结构

在游戏中，我们需要正确的绘制卡牌需要知道卡牌的号码、花色以及。该类继承自 QObject，且定义域了 Q_Object 宏。以下是该类的整体结构图。



2.2.2 **成员变量**



上图比较详细地介绍了各个成员变量设置的意图。下面对成员变量的设置细节做一些补充，其余不多做赘述。

出于类封装性的考虑，四个成员变量全部为私有，但该类定义了外界访问的接口函数，因此外界仍然可以进行访问和一定程度地修改这些变量。特别地，枚举类型 Direction 定义在 public 区域，因为对于 setDirection()函数，外界调用时需要传入一个 Direction 型变量，因此 Direction 需对外界可见。

变量 direction 存储了贪吃蛇当前的移动方向，而 stretchMargin 则存储了贪吃蛇还能再伸长几个格子。当一个贪吃蛇吃了果实后，接下来的三步它就能各伸长一个格子，换言之，每吃一个果实，蛇就获得了三个格子的伸长余量，并且需要在接下来的三步中用完它。因此我们有必要记录贪吃蛇当前还剩几个格子可以伸长，这就是 stretchMargin。

2.2.3 **成员函数**

explicit Snake(QObject *parent = nullptr)

返回值：none

参　数：QObject *类型的父指针。

功　能：构造 Snake 类实例对象，并且初始化各个参数。

实　现：设置贪吃蛇的初始状态：将 direction 初始化为 Left，也就是让贪吃蛇初始向左移动；将 stretchMargin 初始化为 0，也就是让贪吃蛇初始无法伸长；将 nodeX 初始化为{15, 16}，将 nodeY 初始化为{15, 15}，也就是蛇头的初始网格坐标为(15, 15)，只有一节除蛇头以外的蛇身，且该蛇身的初始网格坐标为(16, 15)。

QList<int> getHeadPos()

返回值：一个 QList<int>列表。

参　数：none

功　能：获取头的网格坐标(X, Y)。

实　现：取 nodeX 的第一项和 nodeY 的第一项组成 QList<int>列表返回。

QList<int> getSecondPos()

返回值：一个 QList<int>列表。

参　数：none

功　能：获取第二节身子的网格坐标(X, Y)。

实　现：取 nodeX 的第二项和 nodeY 的第二项组成 QList<int>列表返回。

QList<int> getNextHead(Direction dir)

返回值：一个 QList<int>列表。

参　数：一个 Direction 型变量 dir，表示一个方向

功　能：获取贪吃蛇按照 dir 方向移动一步后的头网格坐标(X, Y)。

实　现：根据 dir 的值计算未来的头坐标（以下皆指网格坐标），例如若 dir 为 Up，即蛇接下来会向上移动，那么未来的头坐标就是(现在头 X 坐标, 现在头 X 坐标 - 1)，也就是(nodeX[0], nodeY[0] - 1)。最后返回这个未来的头坐标(X, Y)。

QList<int> getNodeX()

返回值：一个 QList<int>列表。

参　数：none

功　能：获取 nodeX。

实　现：直接返回成员变量 nodeX。

QList<int> getNodeY()

返回值：一个 QList<int>列表。

参　数：none

功　能：获取 nodeY。

实　现：直接返回成员变量 nodeY。

void setNodeX(QList<int> nodeX_)

返回值：none

参　数：一个 QList<int>列表。

功　能：设置 nodeX 为给定的值。

实　现：直接将成员变量 nodeX 赋值为 nodeX_。

void setNodeY(QList<int> nodeY_)

返回值：none

参　数：一个 QList<int>列表。

功　能：设置 nodeY 为给定的值。

实　现：直接将成员变量 nodeY 赋值为 nodeY_。

int getStretchMargin()

返回值：一个 int 类型变量。

参　数：none

功　能：获取拉伸余量 stretchMargin。

实　现：直接返回成员变量 stretchMargin。

void setStretchMargin(int margin)

返回值：　none

参　数：一个 int 类型变量。

功　能：设置拉伸余量 stretchMargin。

实　现：直接将成员变量 stretchMargin 赋值为 margin。

void setStretch ()

返回值：none

参　数：none

功　能：该函数往往在贪吃蛇吃到果实后直接调用，用于设置贪吃蛇的拉伸余量为吃到果实后的状态（也就是 3），该函数比上一个函数更方便一些。上一个函数更适用于 debug，可以将拉伸余量设置为任意值。

实　现：直接将成员变量 stretchMargin 赋值为 3。

Direction getDirection()

返回值：一个 Direction 型变量。

参　数：none

功　能：获取贪吃蛇当前的移动方向。

实　现：直接返回成员变量 direction（需要强制类型转换为 Direction）。

void setDirection(Direction dir)

返回值：none

参　数：一个 Direction 型变量。

功　能：设置贪吃蛇的移动方向。

实　现：直接将成员变量 direction 赋值为 dir。

Void move()

返回值：none

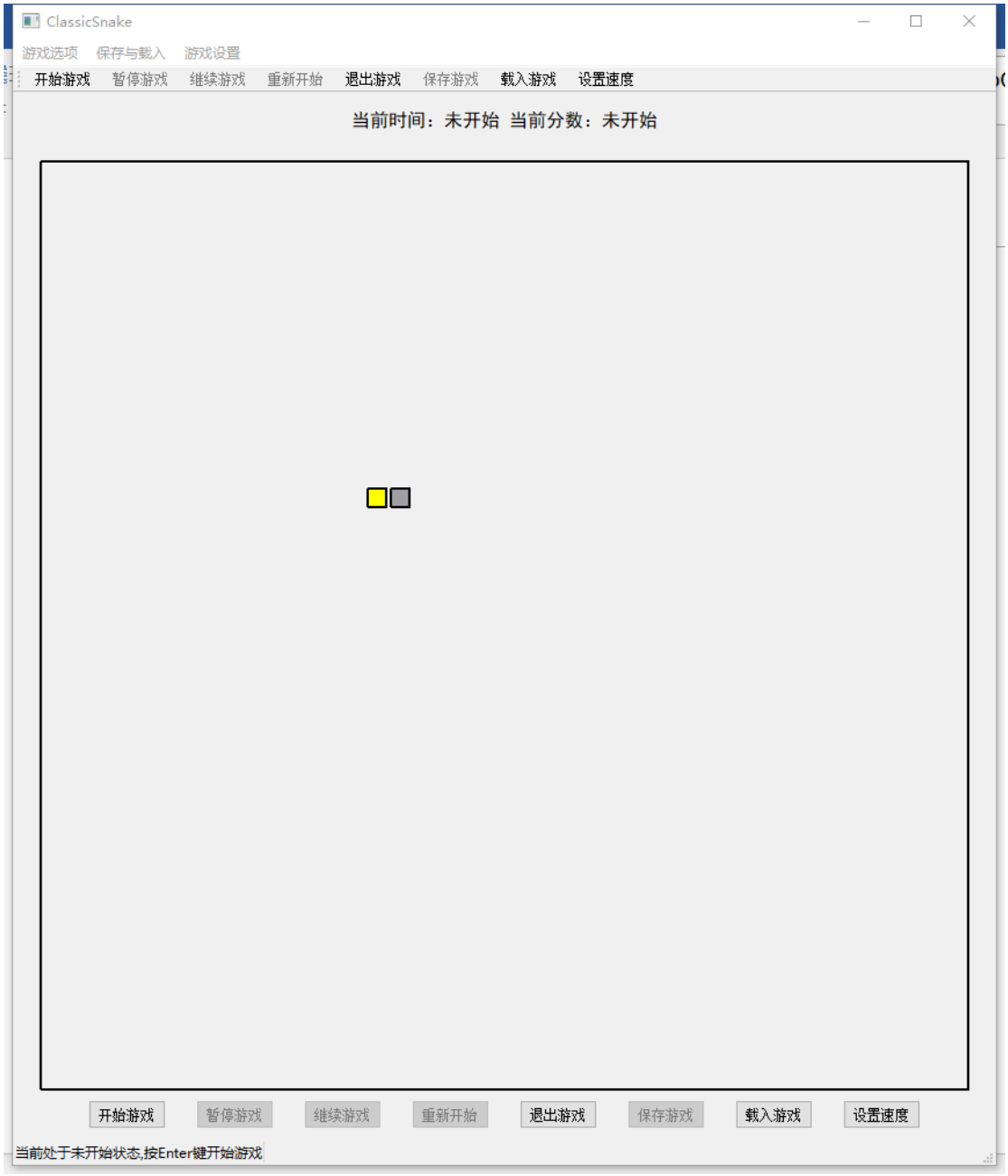
参　数：none

功　能：使贪吃蛇向前移动一步。

实　现：首先根据成员变量 direction 计算以当前行进方向移动一步后的头坐标，实现方法与 getNextHead()类似。接着是一个分支：若当前拉伸余量 stretchMargin 不为 0，那么直接在 nodeX 和 nodeY 开头加入下一步的头坐标，等效于尾部不变，头部向前延长一格，并且让拉伸余量 stretchMargin 自减 1;若当前拉伸余量 stretchMargin 为 0,则在 nodeX 和 nodeY 开头加入下一步的头坐标并分别去掉最后一个位置，等效于蛇向前移动了一格。

3. 游戏实况演示

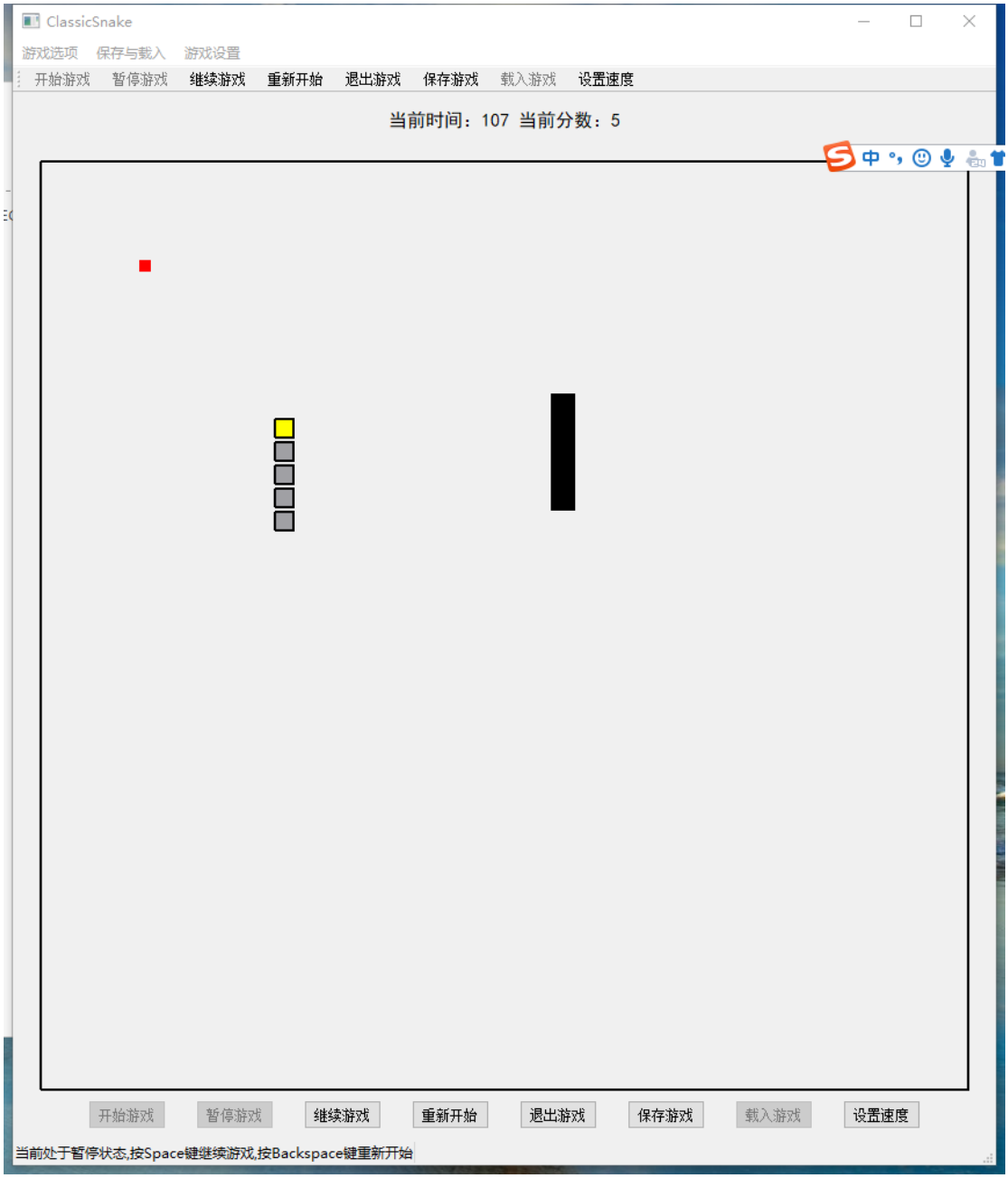
游戏未开始


ClassicSnake
游戏选项
保存与载入
游戏设置
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
设置速度
当前时间：未开始 当前分数：未开始
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
设置速度
当前处于未开始状态,按Enter键开始游戏

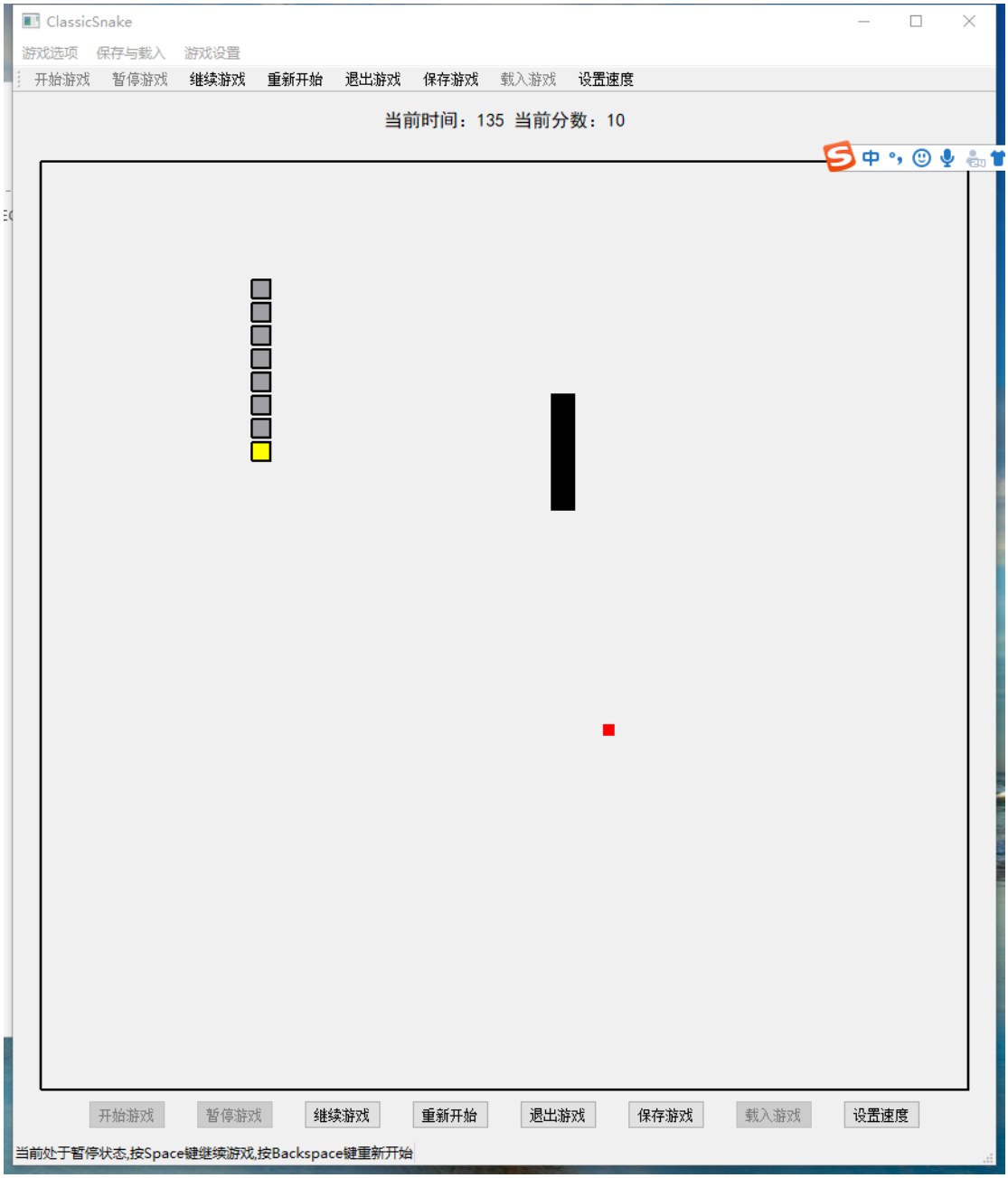
设置障碍


ClassicSnake
游戏选项
保存与载入
游戏设置
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
设置速度
当前时间：未开始 当前分数：未开始
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
设置速度
当前处于未开始状态,按Enter键开始游戏

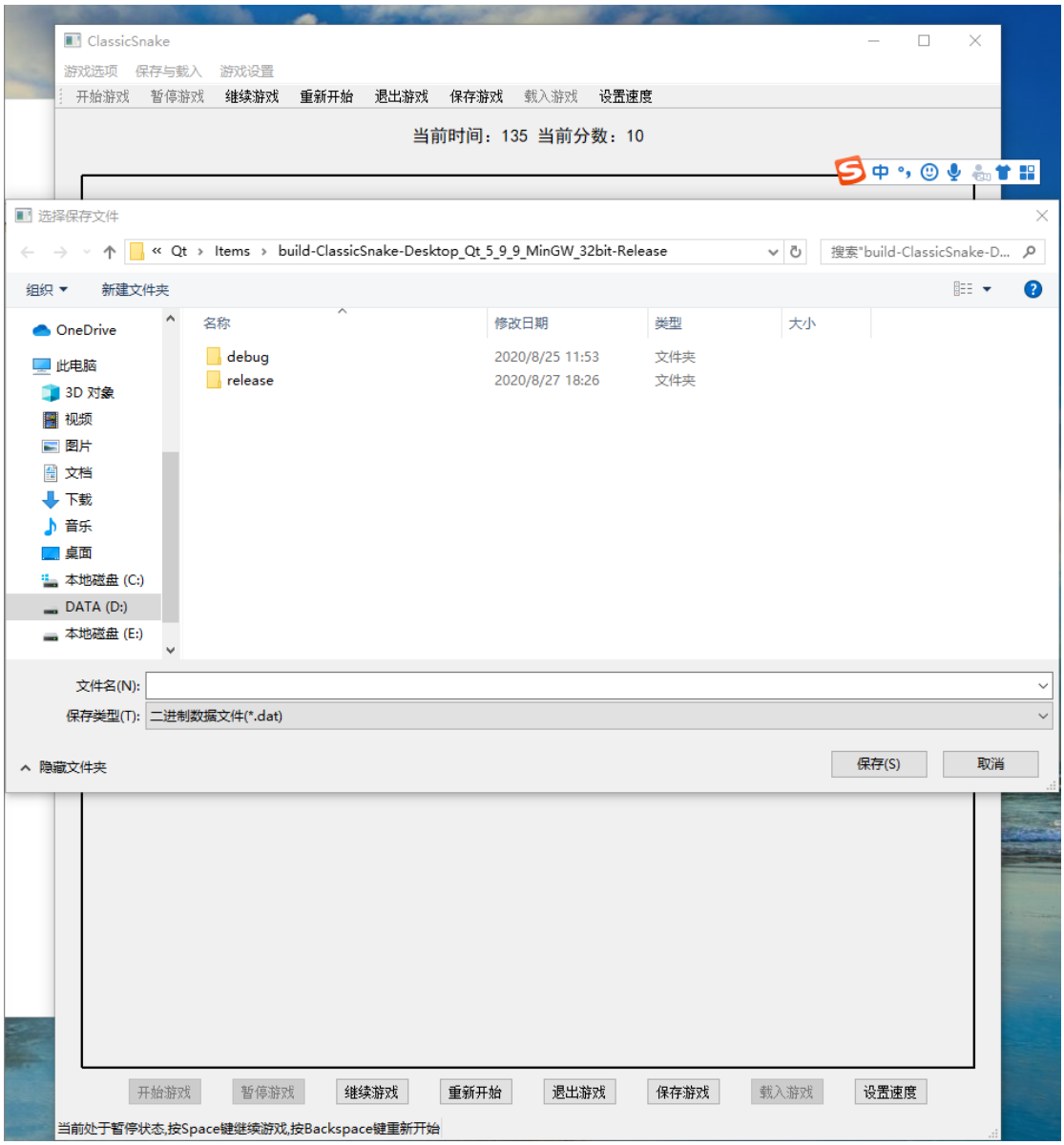
开始游戏


ClassicSnake
游戏选项 保存与载入 游戏设置
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 设置速度
当前时间：107 当前分数：5
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 设置速度
当前处于暂停状态,按Space键继续游戏,按Backspace键重新开始

吃掉果实会长大


ClassicSnake
游戏选项
保存与载入
游戏设置
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
设置速度
当前时间：135 当前分数：10
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
设置速度
当前处于暂停状态,按Space键继续游戏,按Backspace键重新开始

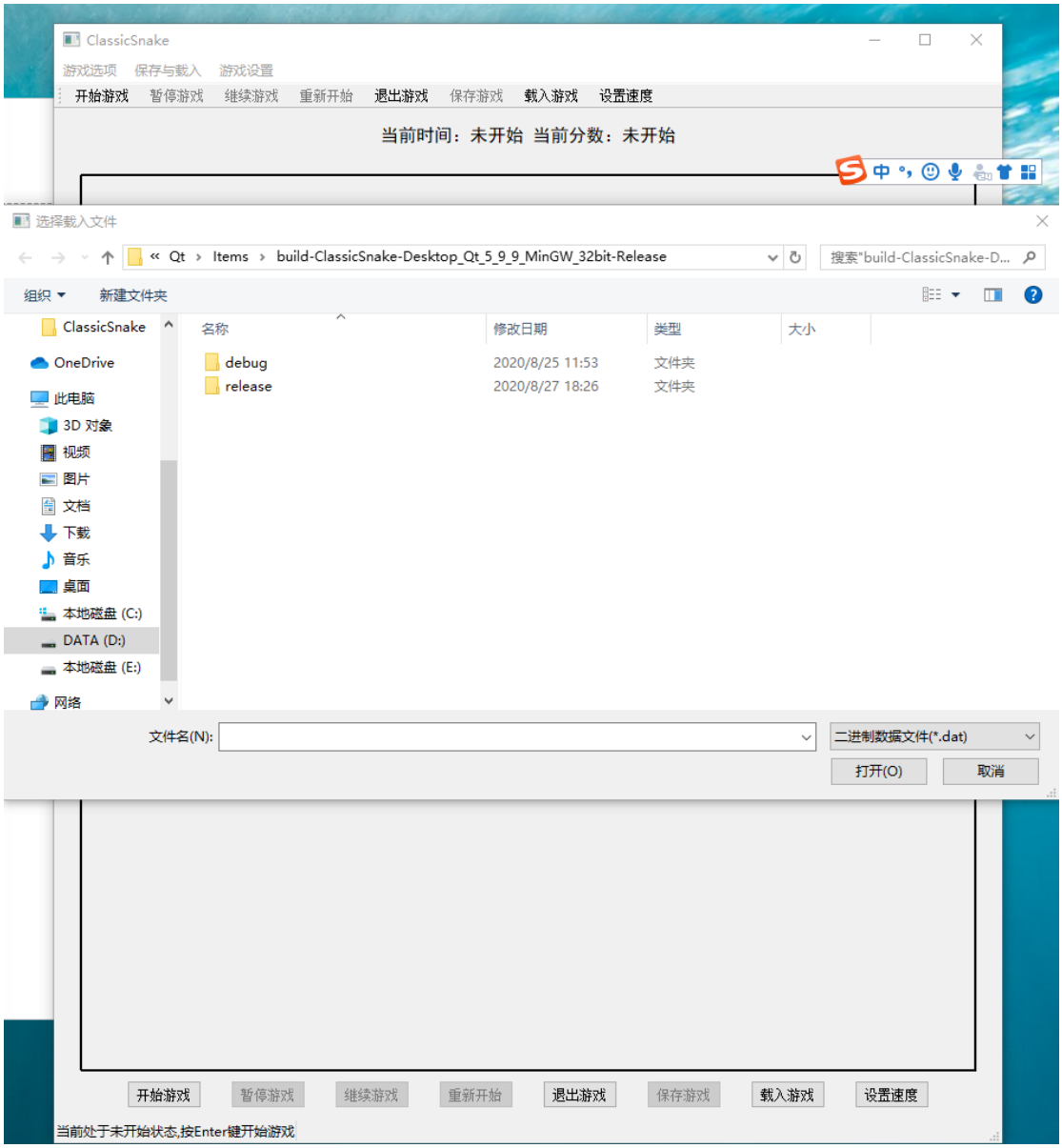
保存游戏

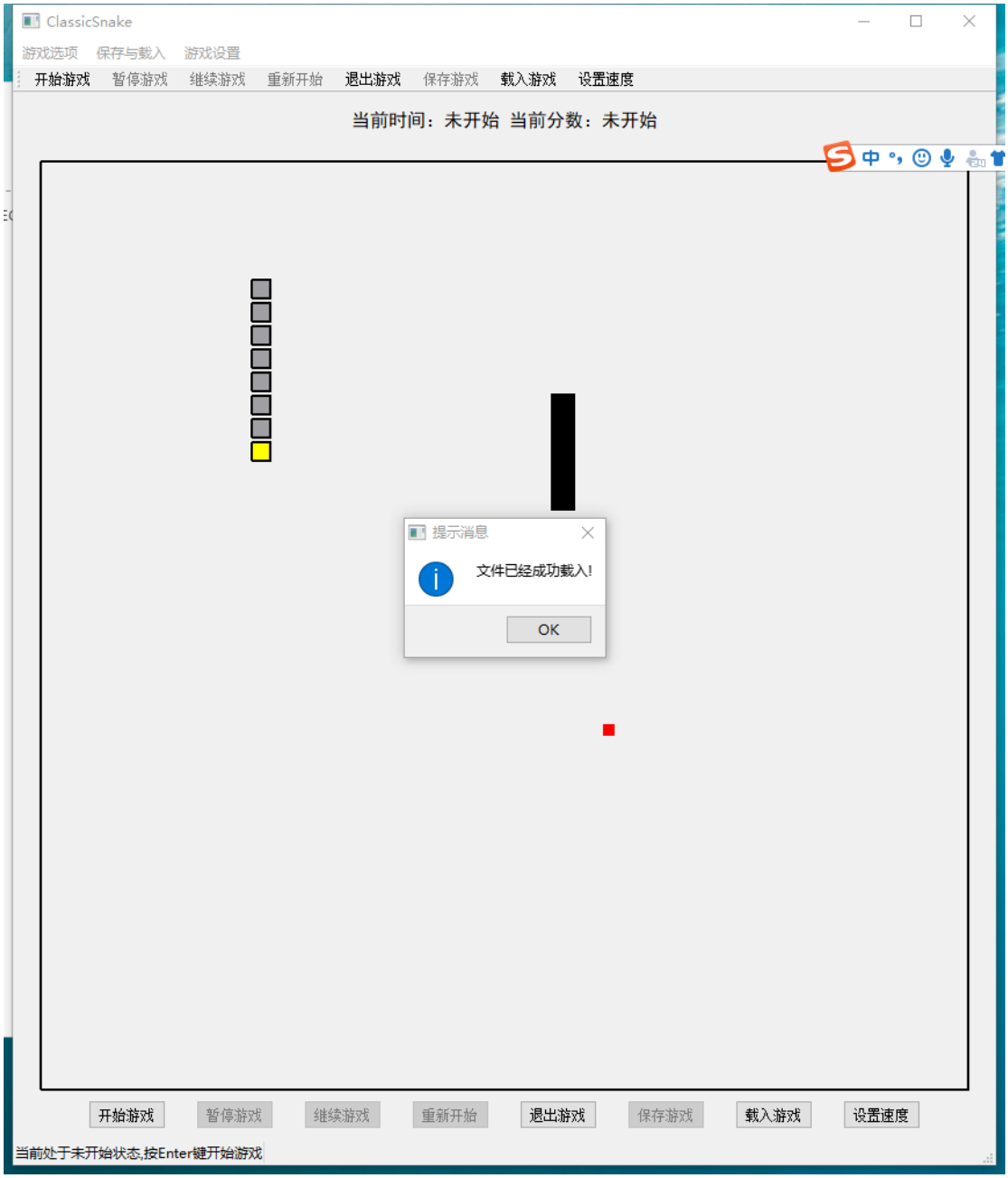

ClassicSnake
游戏选项 保存与载入 游戏设置
开始游戏 暂停游戏 继续游戏 重新开始 退出游戏 保存游戏 载入游戏 设置速度
当前时间：135 当前分数：10
选择保存文件
« Qt › Items › build-ClassicSnake-Desktop_Qt_5_9_9_MinGW_32bit-Release
搜索"build-ClassicSnake-D...
组织 新建文件夹
OneDrive
此电脑
3D 对象
视频
图片
文档
下载
音乐
桌面
本地磁盘 (C:)
DATA (D:)
本地磁盘 (E:)
名称 修改日期 类型 大小
debug 2020/8/25 11:53 文件夹
release 2020/8/27 18:26 文件夹
文件名(N):
保存类型(T): 二进制数据文件(*.dat)
隐藏文件夹
保存(S)
取消
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 设置速度
当前处于暂停状态,按Space键继续游戏,按Backspace键重新开始

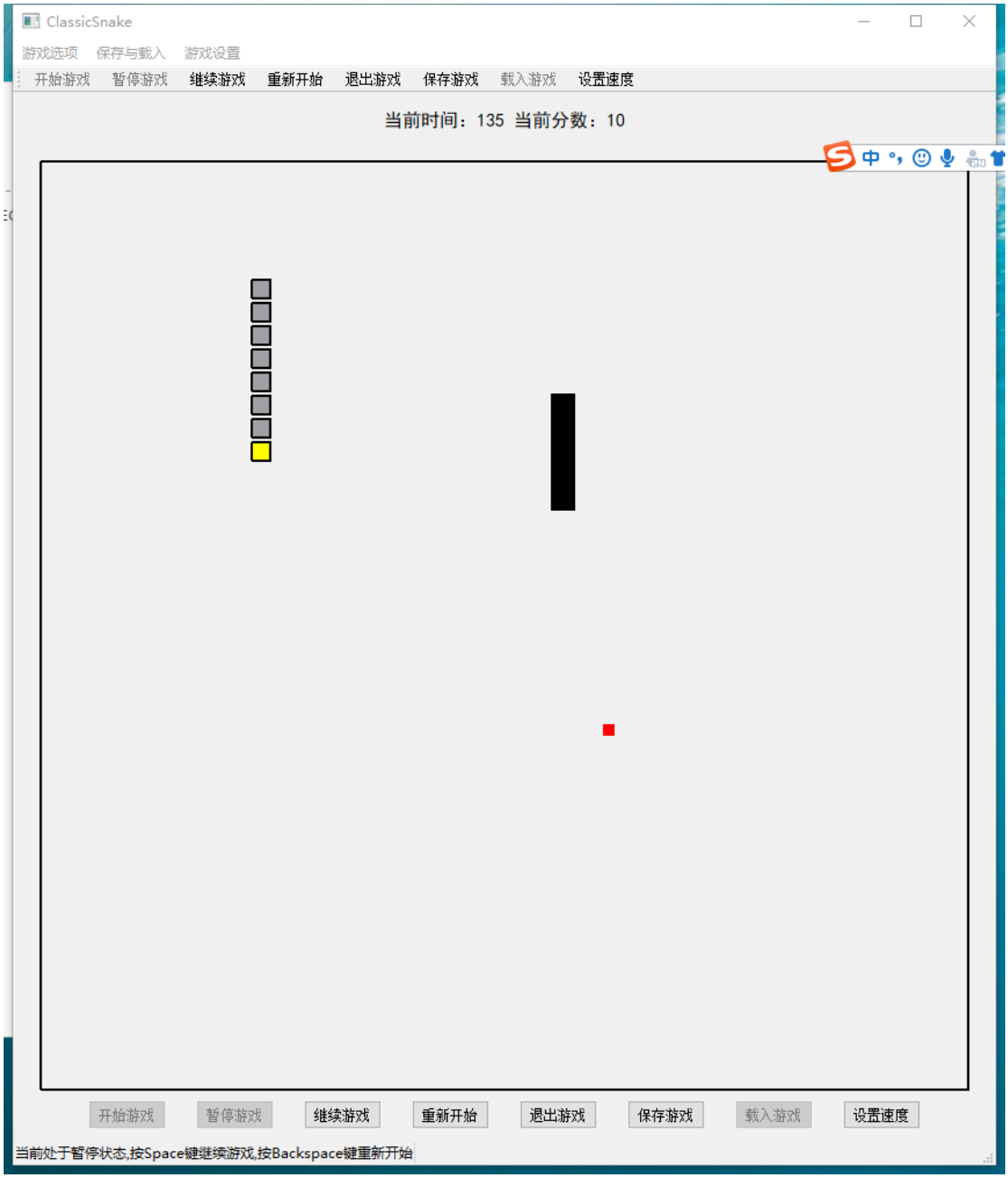
撞到障碍


ClassicSnake
游戏选项 保存与载入 游戏设置
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 设置速度
当前时间：156 当前分数：10
游戏结束
你碰到障碍了
OK
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 设置速度
当前处于终止状态,按Backspace键重新开始

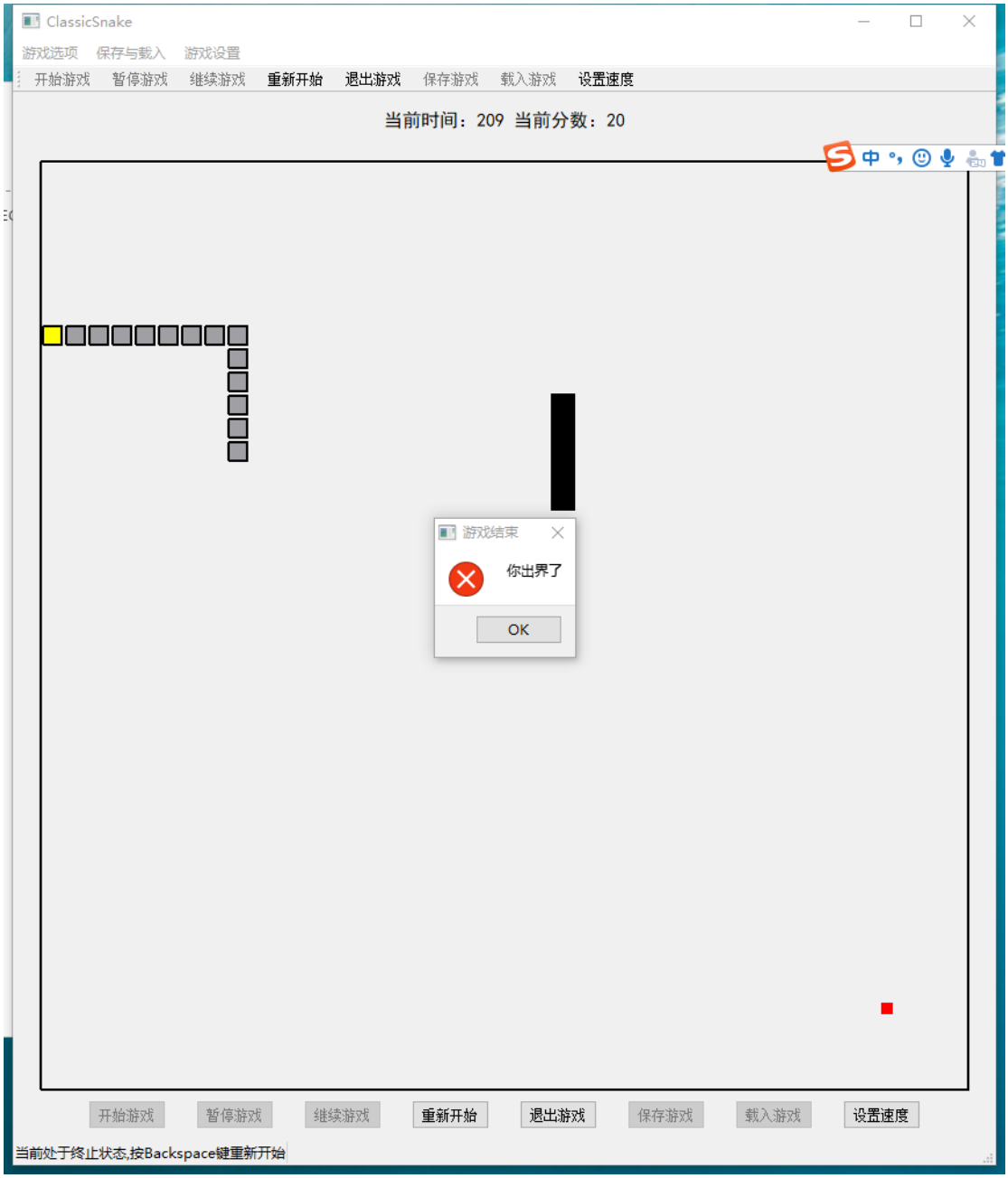
重新开局，载入游戏


ClassicSnake
游戏选项 保存与载入 游戏设置
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 设置速度
当前时间：未开始 当前分数：未开始
选择载入文件
Qt › Items › build-ClassicSnake-Desktop_Qt_5_9_9_MinGW_32bit-Release
搜索"build-ClassicSnake-D...
组织 新建文件夹
ClassicSnake
OneDrive
此电脑
3D 对象
视频
图片
文档
下载
音乐
桌面
本地磁盘 (C:)
DATA (D:)
本地磁盘 (E:)
网络
名称 修改日期 类型 大小
debug 2020/8/25 11:53 文件夹
release 2020/8/27 18:26 文件夹
文件名(N):
二进制数据文件(*.dat)
打开(O)
取消
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 设置速度
当前处于未开始状态,按Enter键开始游戏

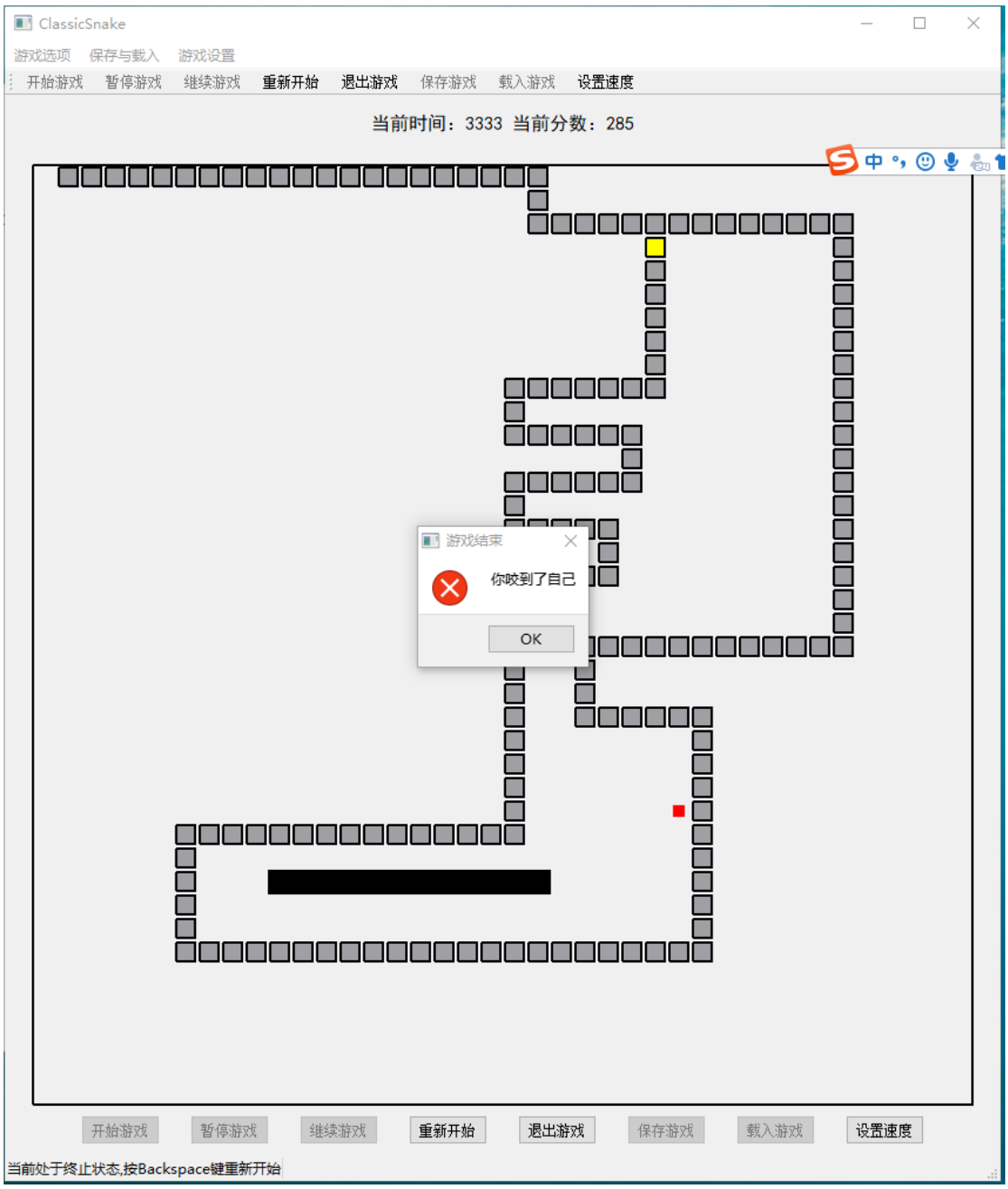

ClassicSnake
游戏选项
保存与载入
游戏设置
开始游戏
暂停游戏
继续游戏
重新开始
退出游戏
保存游戏
载入游戏
设置速度
当前时间：未开始 当前分数：未开始
提示消息
文件已经成功载入!
OK
开始游戏
暂停游戏
继续游戏
重新开始
退出游戏
保存游戏
载入游戏
设置速度
当前处于未开始状态,按Enter键开始游戏


ClassicSnake
游戏选项 保存与载入 游戏设置
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 设置速度
当前时间：135 当前分数：10
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 设置速度
当前处于暂停状态,按Space键继续游戏,按Backspace键重新开始

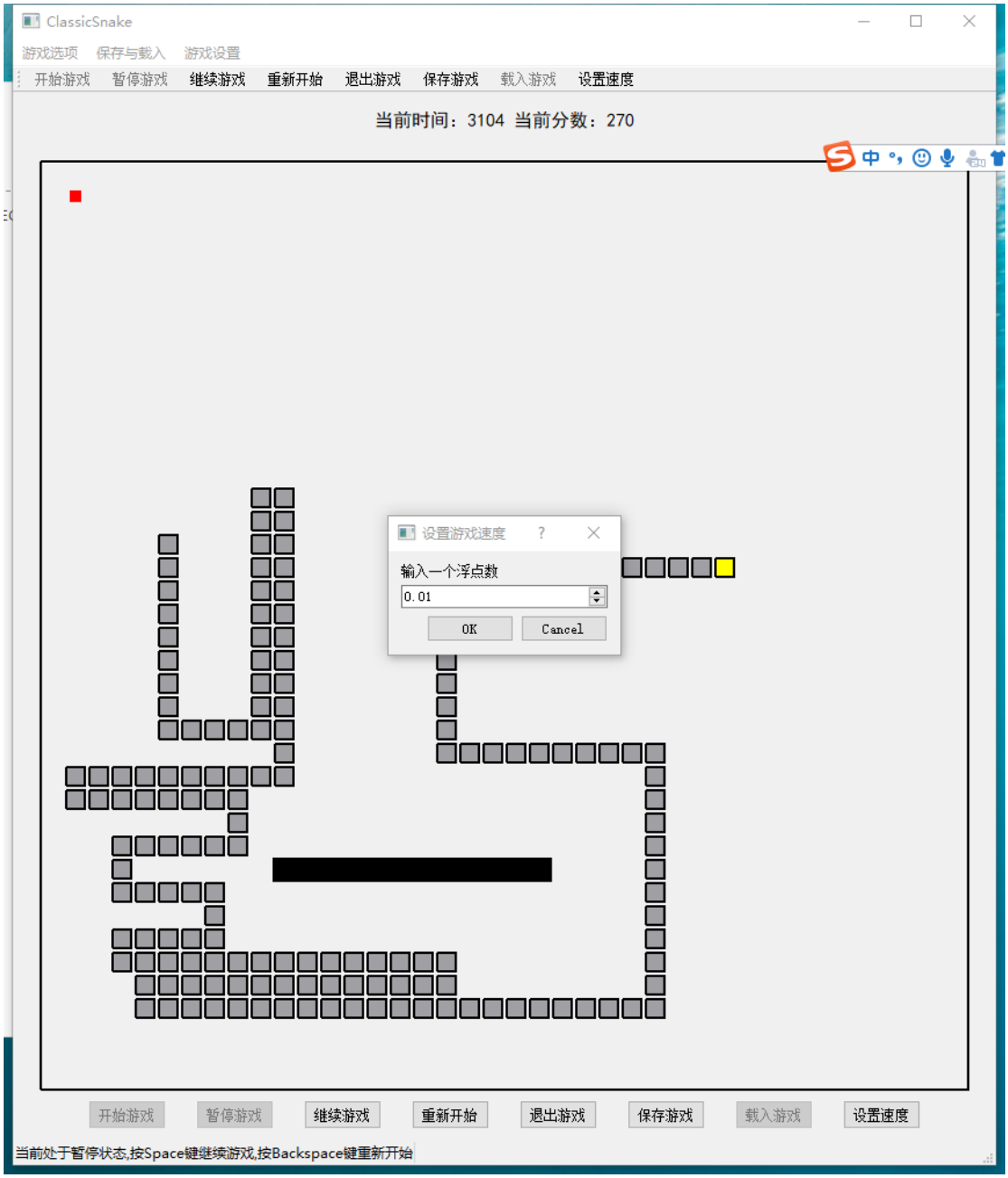
出界，啊这


ClassicSnake
游戏选项 保存与载入 游戏设置
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 设置速度
当前时间：209 当前分数：20
游戏结束
你出界了
OK
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 设置速度
当前处于终止状态,按Backspace键重新开始

咬到自己，啊这


ClassicSnake
游戏选项
保存与载入
游戏设置
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
设置速度
当前时间：3333 当前分数：285
游戏结束
你咬到了自己
OK
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
设置速度
当前处于终止状态,按Backspace键重新开始

设置速度，降低难度，就这



4. 总结

本程序充分利用 Qt 平台及其内置组件，借助面向对象编程和有限自动机的思想，很好地实现了贪吃蛇游戏。总的来说，界面简洁用户友好，代码编写有一定的结构性和可读性，各项功能实现得也比较完备。特别地，本程序在要求基础上还加入了记分、调节速度、状态栏等功能。经过测试，本程序的功能基本符合预期，没有出现不符合预期的行为。