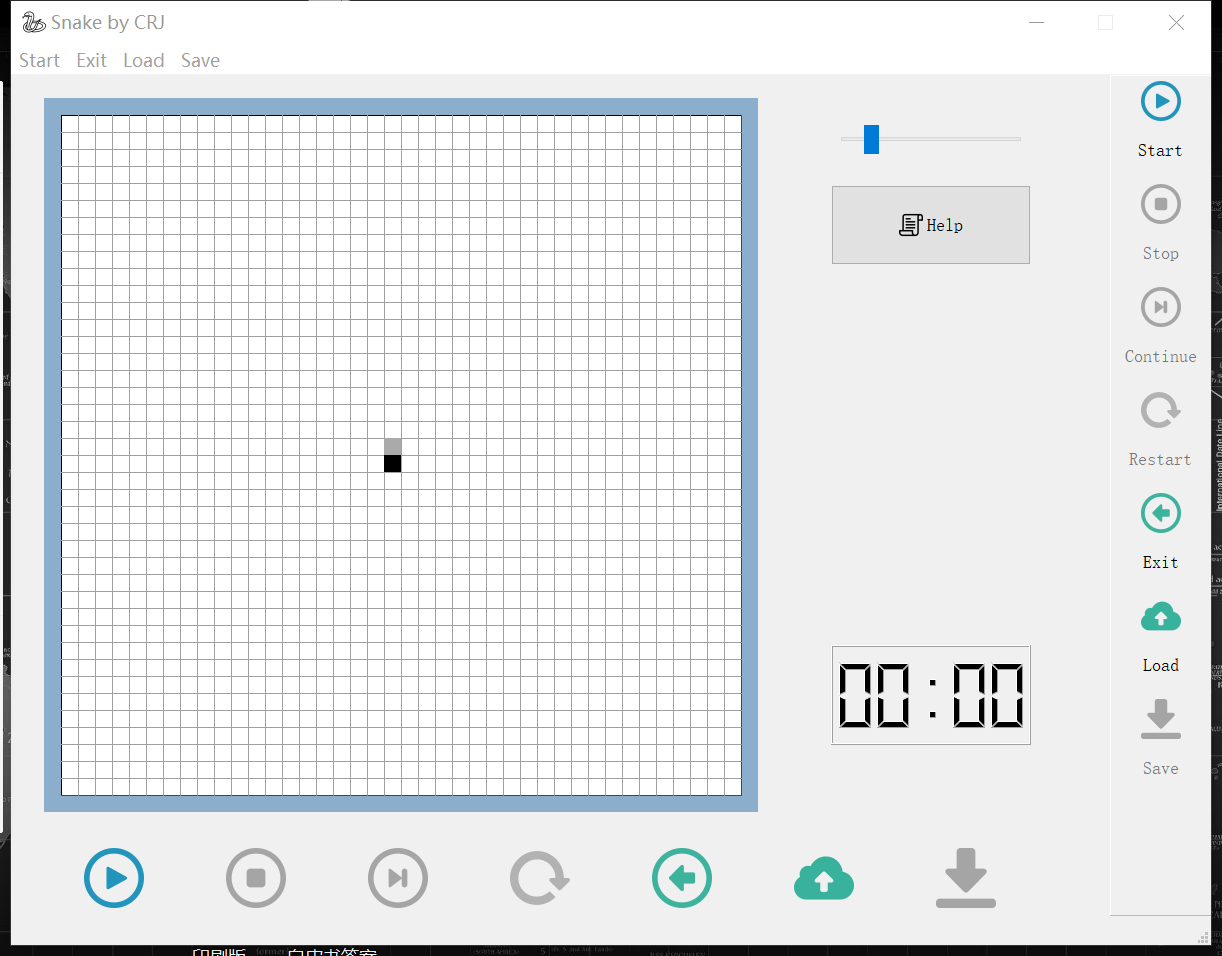
**Qt贪吃蛇大作业 设计文档**

1. **功能介绍**
   1. **未开始状态**

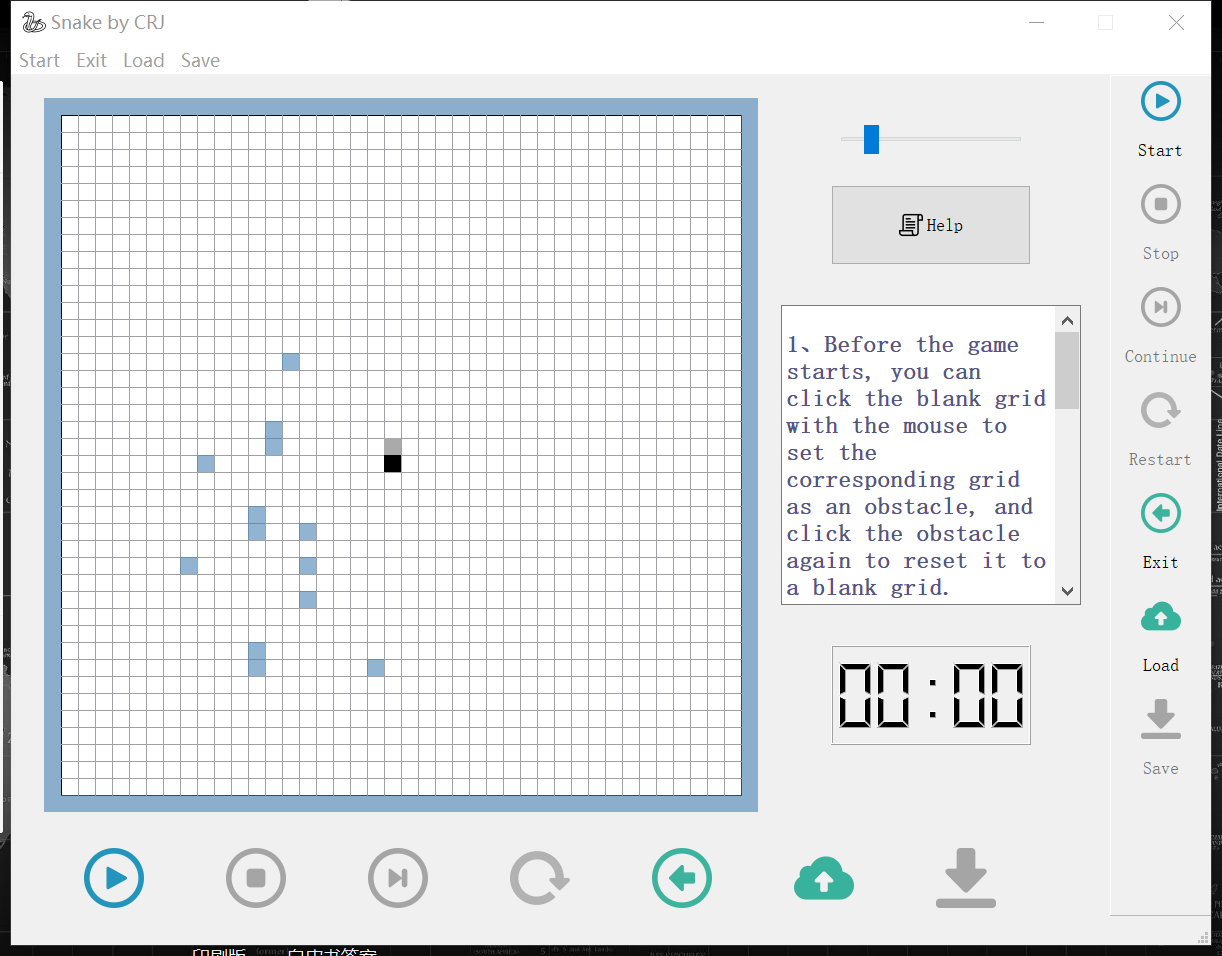
****

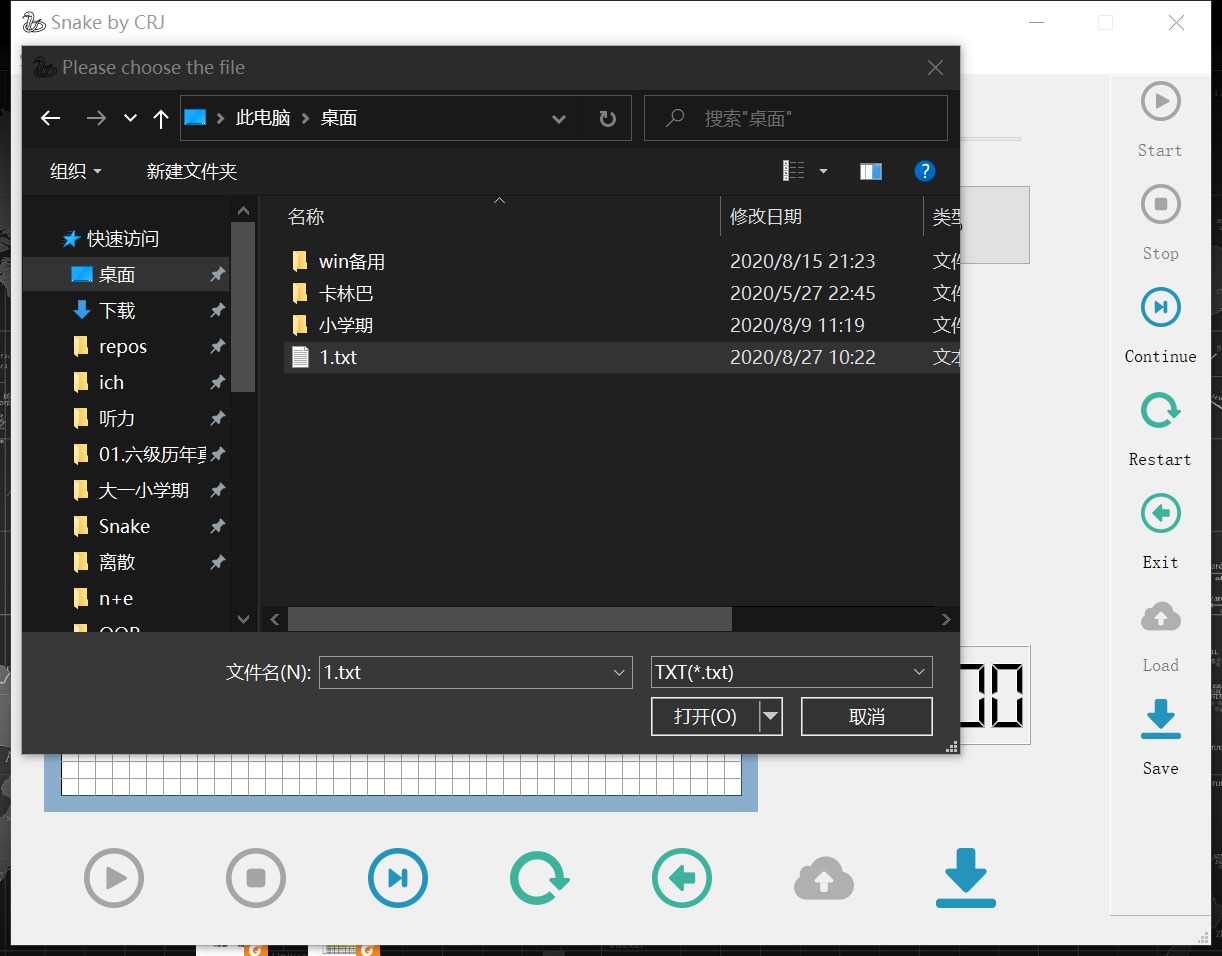
实现游戏界面为40\*40的网格，界面上包括菜单栏（上侧）、工具栏（右侧）和按钮（下侧），三者均实现开始游戏、暂停游戏、继续游戏、重新开始、退出游戏、保存游戏、载入游戏的功能。

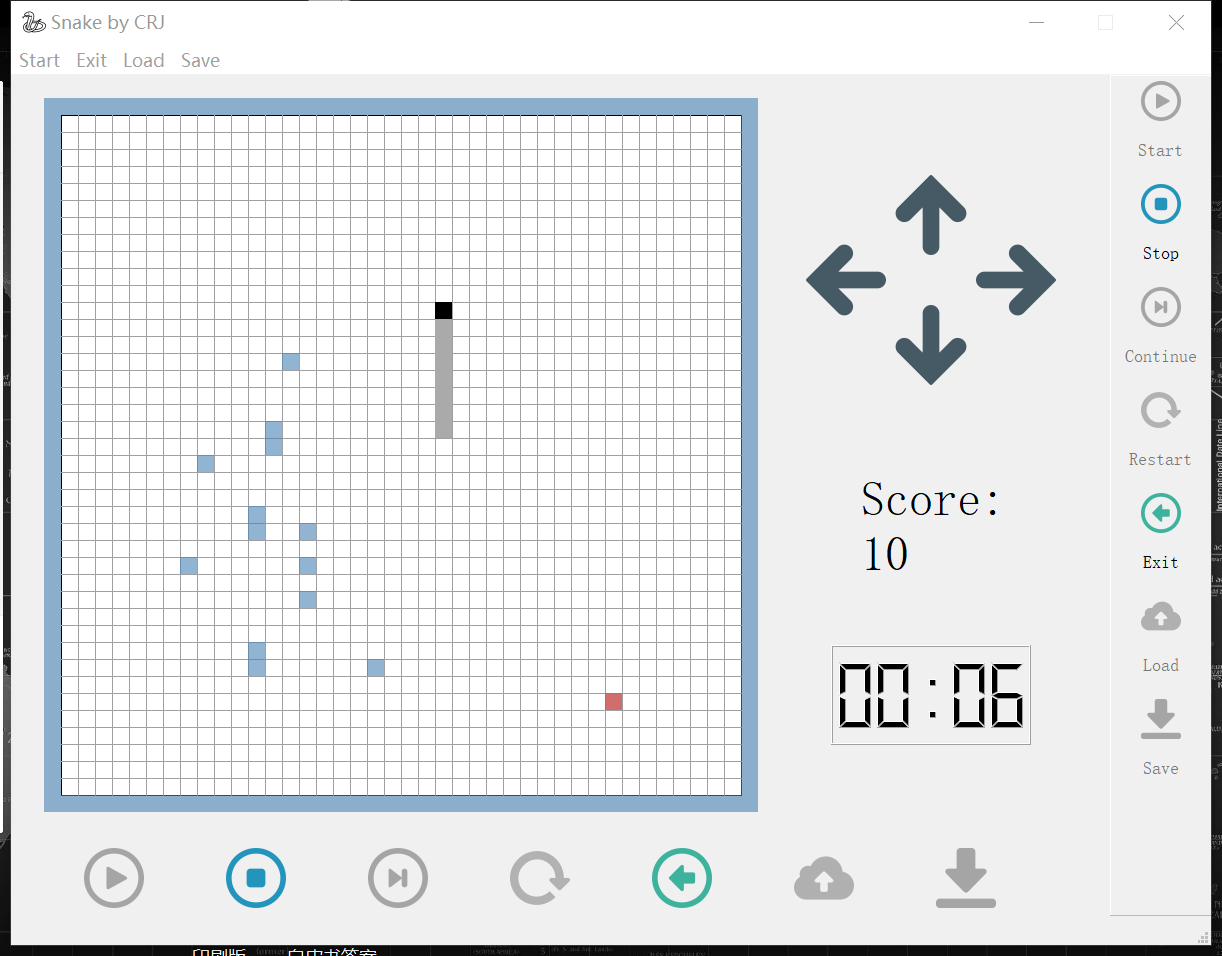
单击“Help”按钮打开帮助文本，再次点击按钮关闭文本。

调节滑动条（右上角）可调节贪吃蛇速度，最左端速度最快。

用户处于未开始状态时，贪吃蛇初始长度为2个网格。蛇头为黑色方格，蛇身为灰色。通过鼠标点击空白网格，可以将对应网格置为障碍（蓝色方格），再次点击障碍可将其重置为空白网格。



此时可以点击“Load”，上传已有的游戏格局。

* 1. **游戏状态**

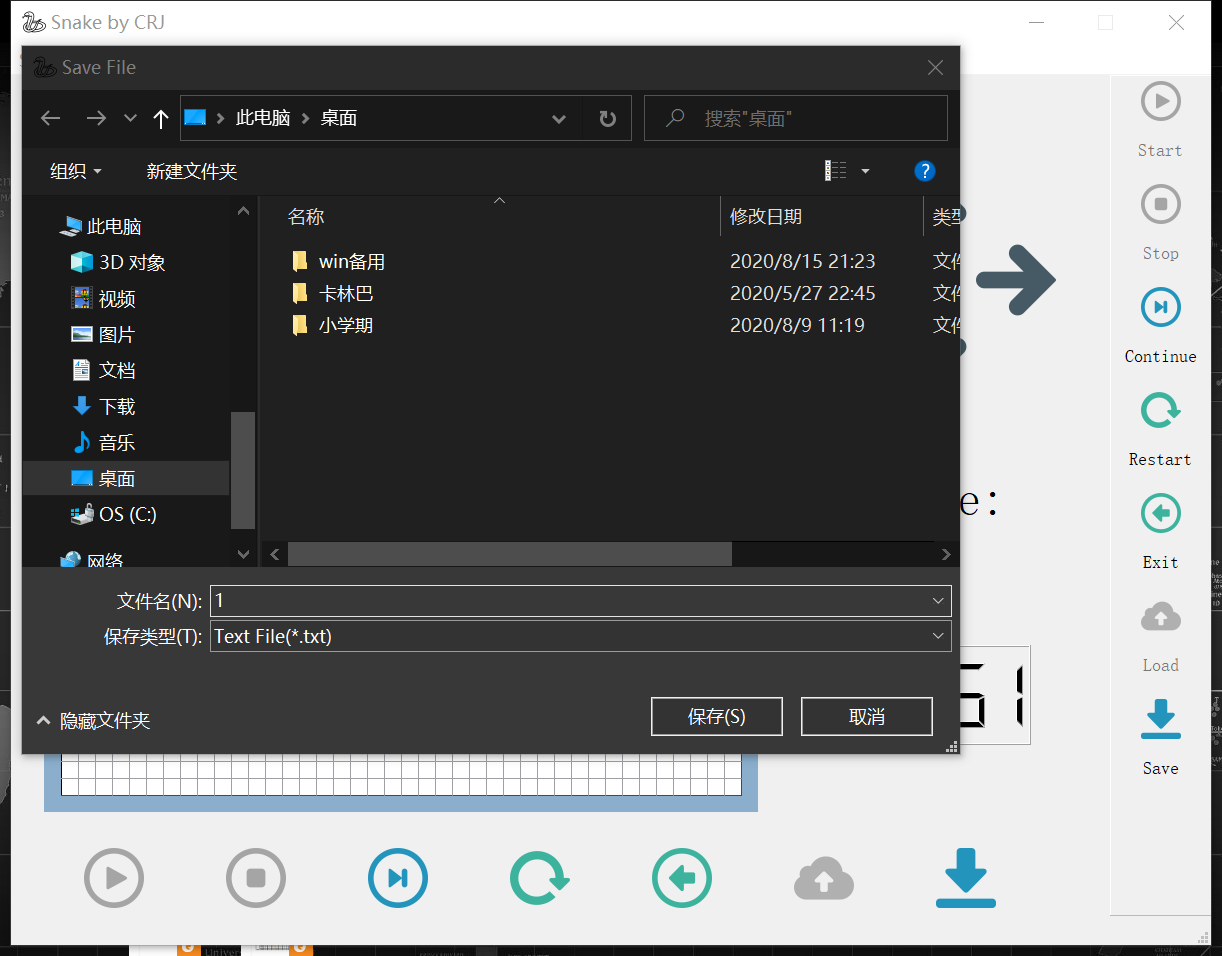
程序随机选择一个空白网格生成一个果实。此后，按任意方向键或是点击右上角的方向箭头，贪吃蛇将向对应方向移动，并开始计时，移动速度由未开始状态时用户的设置决定。每吃到一个果实分数加5分。右下角显示时间（贪吃蛇头部移动的格子数）。

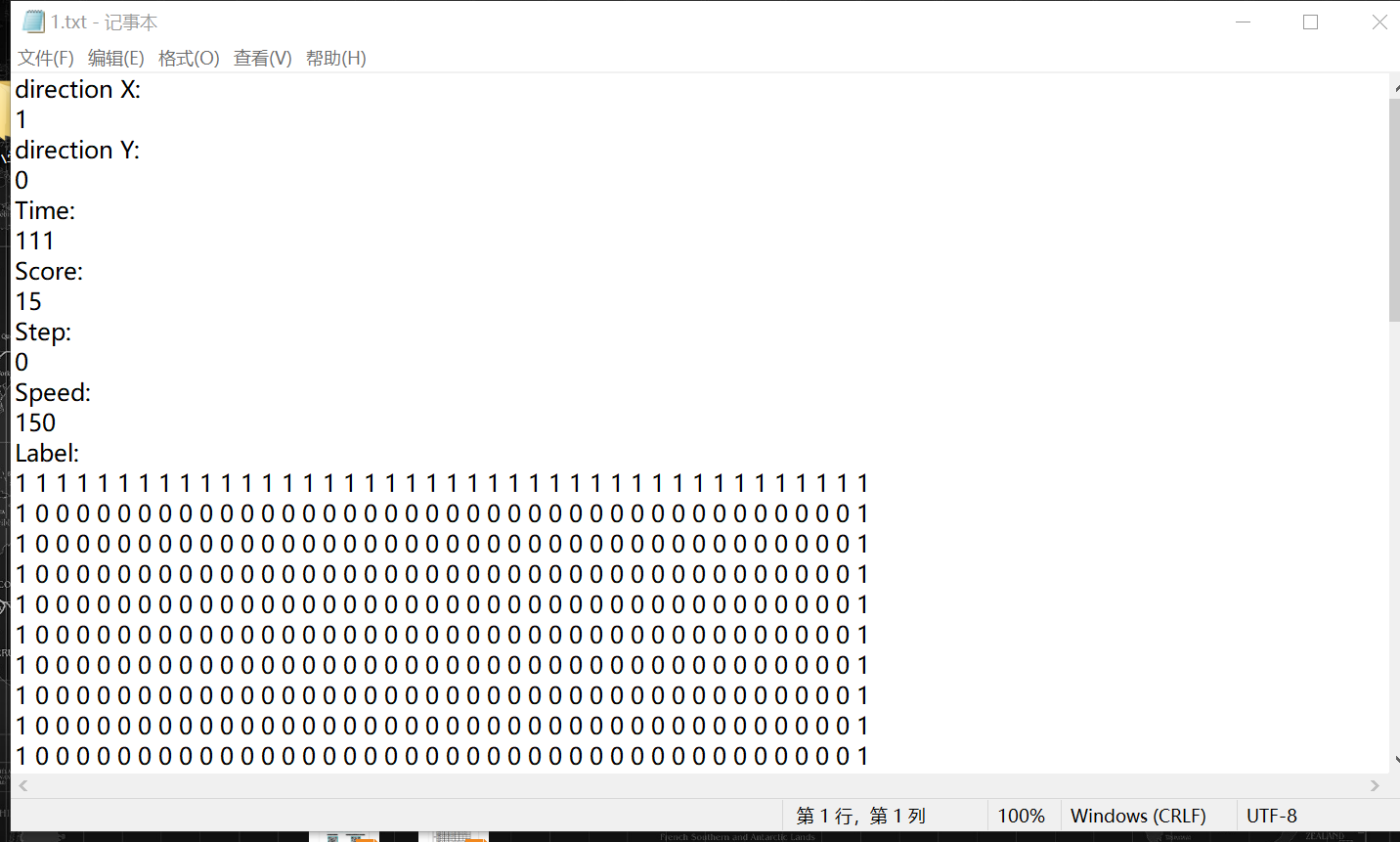
贪吃蛇吃到果实后，在接下来的3步中，头部延长3个格子，与此同时尾部保持不变。

* 1. **暂停状态**

点击“Stop”，贪吃蛇的移动停止，游戏时间停止。此时可以保存游戏格局为txt格式。如果未输入正确保存路径，程序将给予提示。

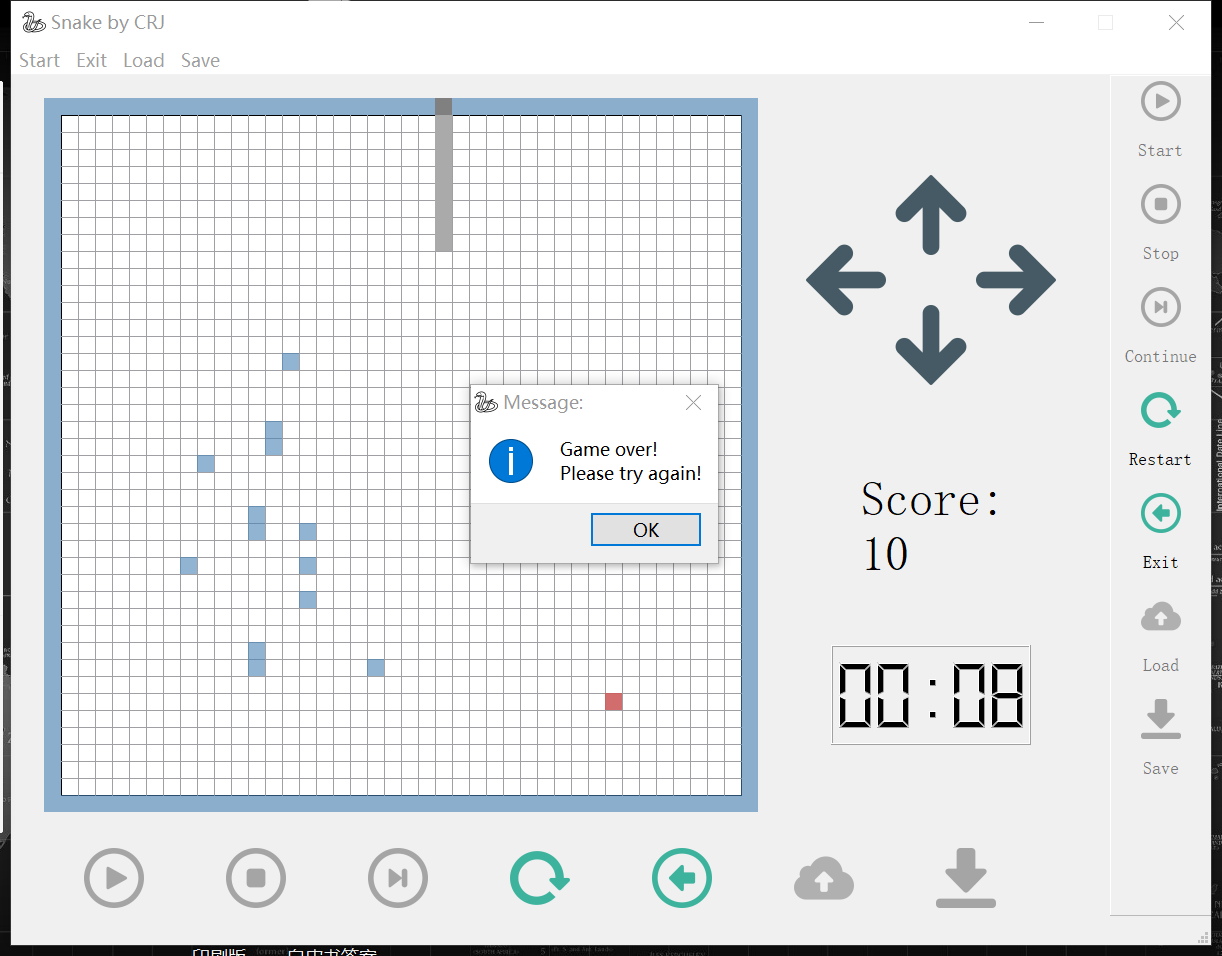
此时点击“Continue”，贪吃蛇恢复移动，恢复计时，进入游戏状态。





* 1. **终止状态**

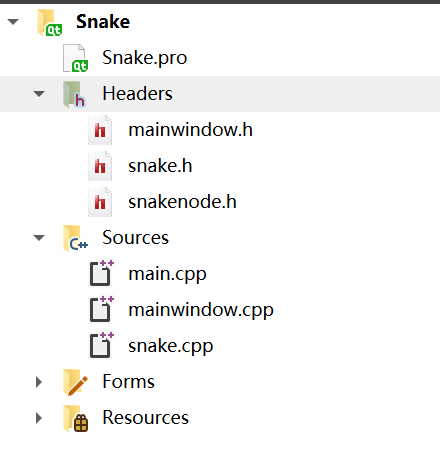
当贪吃蛇碰到周围边界、碰到障碍、碰到自己的身体时，游戏进入终止状态，提示游戏失败，游戏计时停止，贪吃蛇无法移动。



点击“Restart”，用户变为未开始状态，计时清零，格局恢复为初始状态。

任何时候点击“Exit”，游戏直接关闭。

1. **程序架构及设计思路**

****

在我的设计中，ui界面和蛇本身是分离的。我写了一个MainWindow类，继承自QMainWindow，在该类上设计界面，包括菜单类、工具栏、按钮和调整速度的滑动条的位置、样式等。MainWindow类控制着时间和分数的刷新和显示，定时器每发出一次timeout信号，时间和分数刷新一次。MainWindow类还控制着鼠标事件和键盘事件，来改变蛇的方向和设置/取消障碍。MainWindow类中包含一个指向Snake类的指针，以控制蛇的运动和更新。

蛇的结点和背景结点都以标签形式显示。我创建了一个SnakeNode类，来确认标签的位置、类型和样式。标签分为普通、障碍、蛇头、蛇身和食物。不同的标签的结点颜色不同。我还创建了一个Snake类，其中包含一个矩阵，矩阵类型为SnakeNode，来记录游戏界面各个位置标签的类型；Snake类中还包含一个链表（SnakeNode），记录哪些节点是蛇的结点。链表尾部的结点为头结点。

游戏开始后计时器按照设置好的速度开始计时，将显示时间设置蛇头移动的格子数（时间刷新间隔与蛇移动时间间隔相同）。计时器到时后蛇头向对应方向移动一格，即将下一格标签设置为蛇头，原来蛇头格子的标签改为蛇身。然后进行判断，是游戏结束、吃到食物还是正常行进。蛇头移动的格子标签若为蛇身或是障碍，则游戏结束，发出信号，让MainWindow进入终止状态（弹出QMessageBox提示，计时停止）；吃到食物则步长加二（step+=2）；正常行进则判断步长是否为零，若为零则删除尾结点，不为零则步长减一。如此一来，贪吃蛇会按照预想的进行移动。

工具栏和菜单栏中使用同一套QAction，按钮的click（）槽函数分别与这些QAction进行关联，实现开始游戏、暂停游戏、继续游戏、重新开始、退出游戏、保存游戏、载入游戏的功能。这些QAction控制着定时器的暂停和启用、按钮的可用和禁用。Save和Load按钮借助QFile和QTextStream类完成将暂停状态的游戏保存为txt文件，或是将时间、标签、蛇身等数据从txt文件中读出。

**2.1 mainwindow.h/mainwindow.cpp详解**

MainWindow(QWidget \*parent = 0); //初始化窗口设置（按钮、菜单栏、工具栏等位置及样式）

void paintEvent(QPaintEvent \*event); //绘制游戏界面的格子

void init(); //初始化时间，实例化蛇

void keyPressEvent(QKeyEvent \*e); //设置按下键盘方向键，改变蛇的方向

void mousePressEvent(QMouseEvent \*event); //设置在未开始状态点击游戏界面放置障碍；在游戏界面点击箭头改变蛇的方向

void showTime(); //刷新时间和分数的槽函数

void on\_horizontalSlider\_valueChanged(int value); //拖动滑动条改变蛇的速度

void gameOver(); //提示游戏结束，改变游戏状态为终止，禁用按钮并停止计时

int s=0; //时间（蛇移动的格子数）

int status=0; //当前游戏处于的状态（0：未开始；1：游戏状态；2：暂停状态；3：终止状态）

QTimer\* timer; //计时器

QString str; //显示分数的QString

Snake \* snake; //蛇

**2.2 snakenode.h详解**

enum Label； //列举标签类型

class SnakeNode； //保存游戏界面上各结点的标签、类型及位置

**2.3 snake.h/snake.cpp详解**

Snake(); //创建并初始化所有标签为普通结点

void init(); //初始化蛇身为两格

void verseinit(); //若玩家指定初始方向与屏幕显示的蛇的初始方向相反，则调用此函数将蛇头反向

void initBorder(); //设置初始边界

void createFood(); //随机创建食物

void restartinit(); //重新开始游戏，重新初始化

~Snake(); //析构并释放指针，防止内存泄漏

void gameOver(); //游戏结束的信号

int speed=100; //蛇的速度

int step=0; //蛇的步长，用于判断是否是吃到食物后的三步

SnakeNode \*snake\_label[SIZE+2][SIZE+2]; //储存游戏画面

QList<SnakeNode\*> snake\_list; // 蛇身

int dX,dY; //移动方向

int score=0; //游戏分数

SnakeNode \*head，tail // 蛇头和蛇尾指针

void moveSnake(); //槽函数，移动蛇头并判断是游戏结束、吃到食物还是正常行进

1. **心得**

一开始把ui和蛇写在同一个类里，导致找函数的时候十分混乱。后面将界面和内容分开，代码架构清晰了许多。OOP的思想还是很重要的。

写Save和Load功能的时候在保存文件路径名和名字上花了很长时间，后来发现一个函数就能解决。要学会更快速有效地阅读帮助文档，可以先写几个小程序来理解，再用进项目里。

写ui还是以简洁为美，不一定要追求花里胡哨的效果。