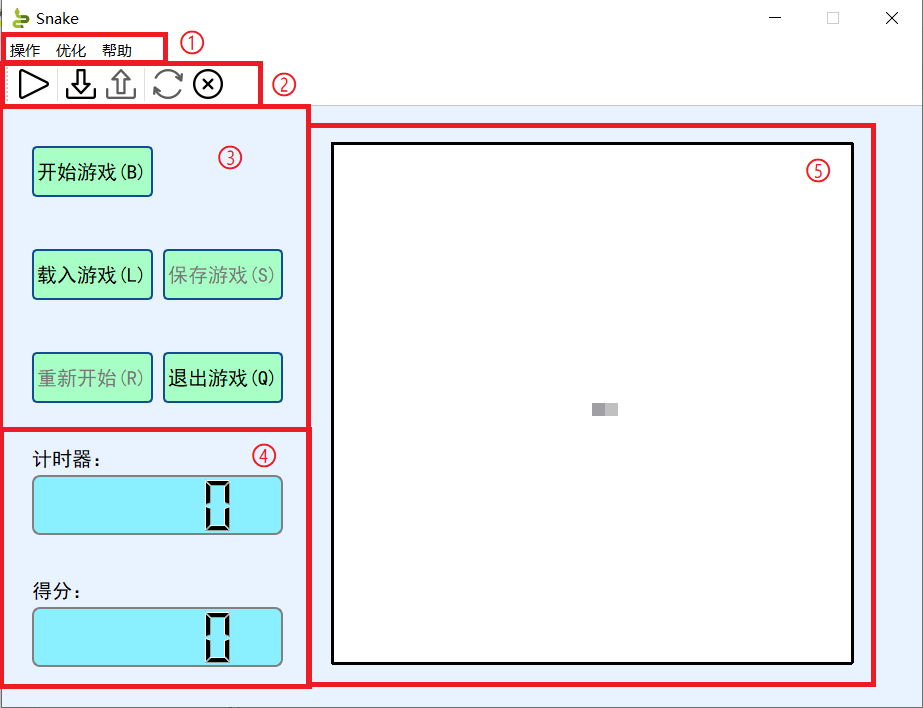
贪吃蛇游戏的设计与实现文档

一、总体简介

本程序是对经典游戏“贪吃蛇”的复刻，以经典黑白方块的款式，提供开始、暂停、继续、保存、载入等基本游戏功能，完成对于QT应用程序编写的初步时间。

二、界面及功能介绍

本程序由一个主界面，以及一系列对话框构成。运行程序，即可进入主界面：



主界面由5部分构成，分别是（1）菜单栏；（2）工具栏；（3）按钮操作区；（4）计时计分区；（5）游戏界面。

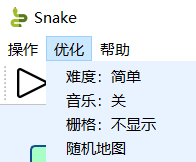
**1.菜单区**



在操作菜单里面，分别是开始游戏、暂停游戏、继续游戏、重新开始、退出游戏、保存游戏、载入游戏7个功能菜单，其中由于开始游戏、暂停游戏、继续游戏在游戏的生命周期中互斥存在，为了提高界面的简洁性，只提供了一个菜单项来进行操作，并在游戏中智能的转化。

游戏进行时，可以暂停游戏，并点击保存游戏来保存当前游戏状态，保存文件为梯形图文件。游戏未开始时，可以选择载入游戏，载入txt文件获得上次保存的游戏信息，注意，由于文件并不会检查输入文件的格式，请务必保证选择游戏自动保存的脚本文件。

为了防止在游戏进行过程中出错，菜单选项并不是一直可用的。



优化菜单为基本实现之外的拓展内容，包含游戏的控制面板以及预定义的游戏脚本，点击对应的菜单项可以设置游戏难度、背景音乐、栅格显示。同时点击随机地图可以随机生成障碍物并开始游戏。注意：使用随机地图会清除地图上已设置的障碍物。

点击帮助菜单项可以查看游戏的帮助文档。

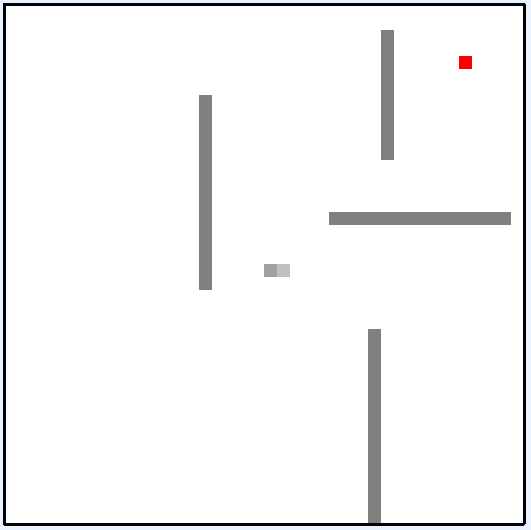
**2.工具栏及按钮区**

工具栏和按钮区提供和操作菜单相同的功能

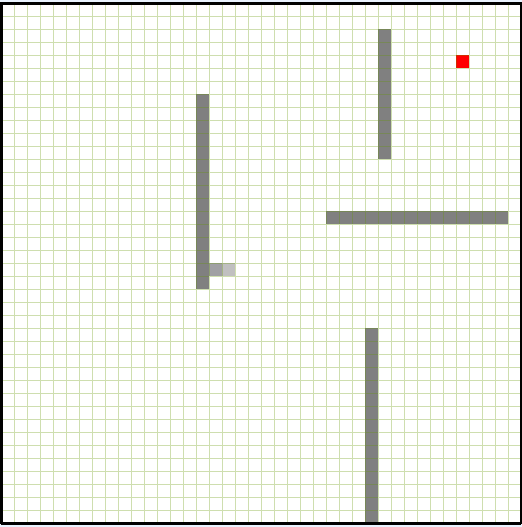
**3. 计时计分区**

包括两个显示屏，在游戏里进行时分别记录游戏时间和游戏得分。其中贪吃蛇移动一格计时器计数加一，贪食蛇吃到一个果实计分器计分加一。

**4.游戏区**



贪吃蛇游戏界面由40\*40的网格构成。其中外围的黑色边框为边界，深灰色方块为障碍物，白色区域为可通行的道路，红色方块为果实（游戏目标），贪吃蛇身体偏深的色块代表贪吃蛇的头。游戏目标为尽可能的吃到更多的果实，同时避免：（1）贪吃蛇碰到周围边界；（2）碰到障碍；（3）碰到自己的身体。注意，没吃到一个果实，贪吃蛇的身体会延长三个单位。满足以上三个条件之一即判定游戏失败。可以开启栅格使得更容易判断贪吃蛇的位置：



可以使用随机地图来生成障碍物，同时用户也可以通过鼠标点击空白网格，可以将对应网格置为障碍，再次点击障碍将其重置为空白网格。改变游戏难度可以调整贪吃蛇的移动速度。

为了防止用户点击开始后不能立刻切换到键盘导致手忙脚乱，在最近一次的更新中，点击开始游戏后也会等待用户从键盘输入信息再真正开始。继续游戏同理。

三、实现方法

程序主要由MainWindow类和Snake类实现完成。其中MainWindow类负责实现程序的交互界面以及处理交互信息，而Snake则主要负责维护游戏逻辑，实现游戏功能。

**1.MainWindow类**

MainWindow中，内部变量包括：

QTimer\* timer;

QMediaPlayer\* player;

Snake map;

bool isBegin;

int currentTime;

int score;

其中timer负责定时，并通过connect(timer, SIGNAL(timeout()), this, SLOT(timer\_clock()));与PaintEvent联系起来，通过定时刷新界面来实现贪吃蛇的移动功能。

player负责维护程序的背景音乐的播放和暂停。

map是MainWindow采用组合的方式内嵌了贪食蛇游戏的逻辑，调用其相应接口，可以完成贪吃蛇的相应操作。

isBegin为游戏是否开始的标记标志。

currentTime和score分别保存计时和计分器的数字值。

MainWindow和按钮、菜单、工具栏之前的相应通过QAction来实现，MainWindow内部实现了12个QAction的相应函数

private slots:

void **on\_startAction\_triggered**();

void **on\_loadAction\_triggered**();

void **on\_saveAction\_triggered**();

void **on\_continueAction\_triggered**();

void **on\_pauseAction\_triggered**();

void **on\_restartAction\_triggered**();

void **on\_quitAction\_triggered**();

void **on\_helpAction\_triggered**();

void **on\_setMapAction\_triggered**();

void **on\_setDifficultyAction\_triggered**();

void **on\_setMusicAction\_triggered**();

void **on\_showGridAction\_triggered**();

对于各私有接口功能的简略说明：

**on\_startAction\_triggered**()负责在用户点击开始后加载障碍物信息，地图在构造函数中已经初始化。

**on\_loadAction\_triggered**()负责弹出加载文件窗口，并进行相应的文件读取、游戏信息加载工作。

**on\_loadAction\_triggered**()负责弹出存储文件窗口，并进行相应的游戏状态存储工作。

**on\_continueAction\_triggered**()负责计时继续。

**on\_pauseAction\_triggered**()负责计时停止。

**on\_restartAction\_triggered**()负责重新初始化MainWindow和Snake类。

**on\_quitAction\_triggered**()负责退出游戏。

**on\_helpAction\_triggered**()负责弹出帮助对话框。

**on\_helpAction\_triggered**()负责调用map随机初始化地图。

**on\_setDifficultyAction\_triggered**()负责改变游戏难度（即改变计时器的间隔）。

**on\_setMusicAction\_triggered**()和**on\_showGridAction\_triggered**()分别负责音乐开关和栅格的显示与否。

同时MainWindow类也重载三个事件函数了

protected:

void ***keyPressEvent***(QKeyEvent\* e);

void ***mousePressEvent***(QMouseEvent \*e);

void ***paintEvent***(QPaintEvent\* e);

***keyPressEvent***负责监听鼠标左键点击事件，可以在游戏开始前设置障碍物。

***mousePressEvent***负责监听按键事件，可以在游戏进行中调用map实现转向。

***paintEvent***负责重绘游戏界面，结合计时器实现动画的效果。

**2.Snake类**

Snake中关于方向、网格状态等均使用宏定义，便于修改，也可以使用自定义的enum类型，这里没有采用。

Snake类中包含的成员有：

int map[SIZE][SIZE];

QQueue<QPoint> snake;

QList<QPoint> area;

int direction;

int extension;

bool isMoved;

其中map记录地图每一格的信息（通道、障碍物、果实、身体、头部）；snake记录贪吃蛇的每一个节点的位置，由于贪吃蛇的移动路径满足先进先出的原则，选用队列是最合适的数据结构；area保存所有的可移动位置（通道和果实），由于Qlist的实现使得查找和插入删除都可以在O(logn)的时间内完成，而如果不用area，游戏后期随机生成果实的平均时间复杂度高达O(n)，且满足几何分布，有可能出现循环很长时间的情况，故额外赠加了area变量来快速的选择果实随机生成的位置，direction保存运动方向。extention辅助完成贪吃蛇吃下果实后的伸长过程，isMoved则是为解决统一帧内贪吃蛇移动两次导致吃到自己的bug，辅助实现贪吃蛇在每次刷新过程中只能改变一次方向。

Snake类包含的方法有：

bool **move\_forward**(int& score);

bool **change\_direction**(int dir);

bool **set\_wallorpath**(int x, int y);

int **get\_status**(int x, int y);

void **init\_map**();

void **clear\_map**();

void **random\_map**();

void **load\_wall**();

void**ve**(QTextStream& out);

void **load**(QTextStream& in);

**move\_forward**使贪吃蛇前进一步，并根据下一个方格确定状态，碰到边界或者自己身体、墙壁会返回false，此时MainWindow会调用end\_game()结束游戏；吃到果实会随机生成下一个果实。

**change\_direction**负责改变方向，注意不可改为同水平/竖直的方向，同一次刷新过程中只会改变一次方向。

**set\_wallorpath**负责用户自定义障碍物。

**init\_map**()负责初始化贪吃蛇的位置。

**clear\_map**();负责清空地图和变量，

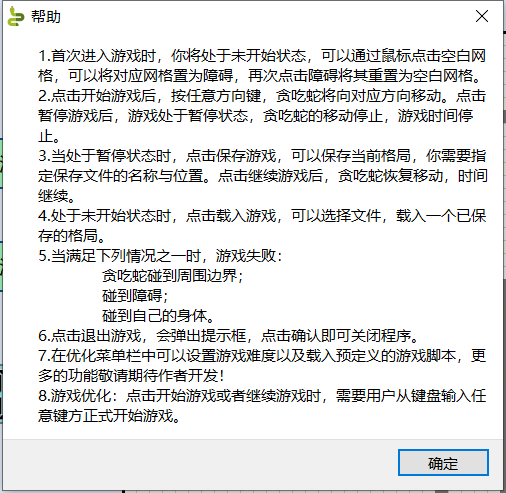
**random\_map**()随机生成地图。

**random\_map**()加载用户设定的障碍物

**save**和**load**负责存取游戏状态的脚本文件。

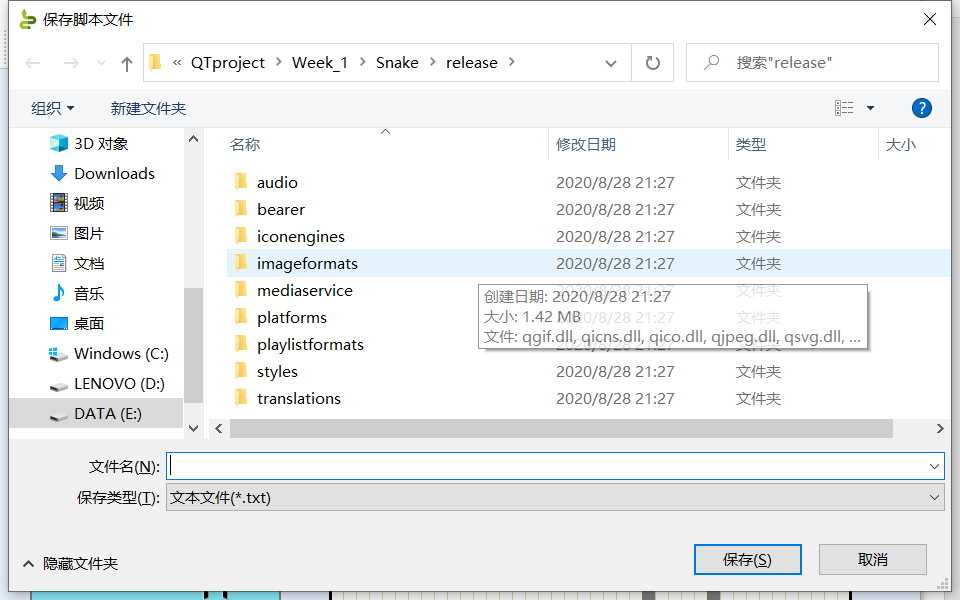
四、对话框以及错误信息介绍

1.点击帮助菜单中的帮助文档，会弹出帮助对话框。



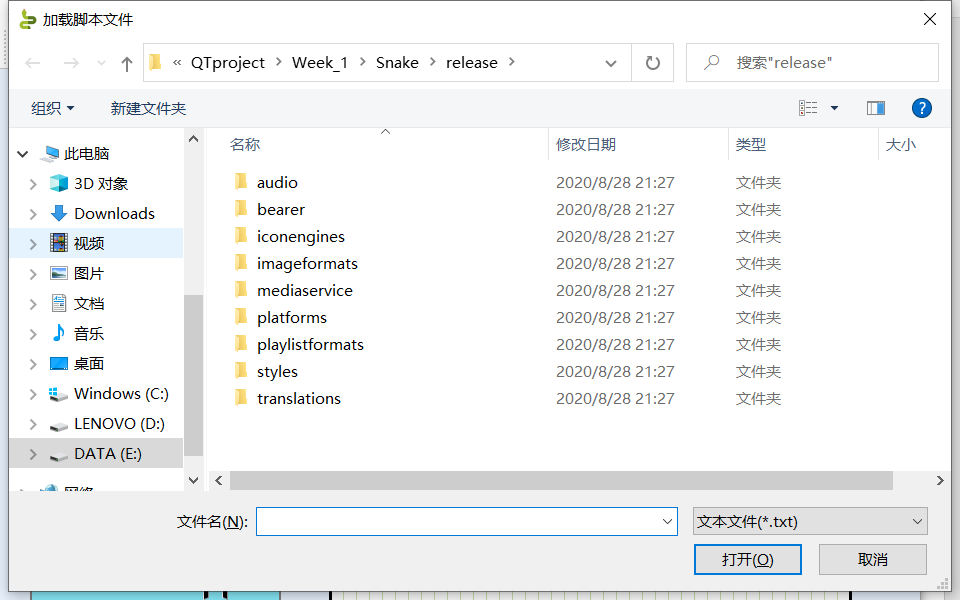
如果帮助文档出错，会提示找不到帮助文档

2.点击保存游戏会弹出保存脚本文件对话框：



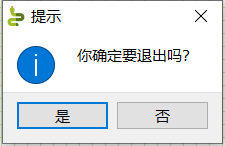
用户需要在这里输入保存位置和文件名称，路径错误会提示：无法在该路径下创建文件。

3，点击载入游戏会弹出加载脚本问价对话框



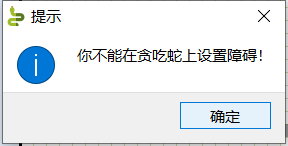
用户需要在这里加载的文件，路径错误会提示：找不到文件。

4.点击退出游戏会弹出：



选择是退出游戏，选择否回到游戏界面。

5.如果在未开始阶段，用户在贪吃蛇身上设置障碍，会弹出错误提示：



五、实践体会

使用Qt编程的最大优点是Qt creator的编辑界面，可以轻松的完成界面布局，并且可以很直观的得到界面预览，方便做进一步的调整。另外Qt整合了各类平台，使得在不同的平台上都可以使用相同的API进行编程，相比于MFC编程开发和学习的压力少了很多。

但是，可能由于Qt的年代已经比较久远，其实现效果稍微有一点老旧。同时Qt的普及度并不是很高，导致相关的学习资料特别是错误问题的解答比较难找，这一定义程度上限制了Qt的发展。最重要的是，使用C++进行图形界面的编程是在是相对比较麻烦，尽管Qt采用了优秀的内存管理机制，可以不用和内存泄露斗智斗勇，但是C++语言的简洁性以及功能显然不如现代语言如Python，使用一些更加高级的编程语言可能使得我们的编程更加轻松。