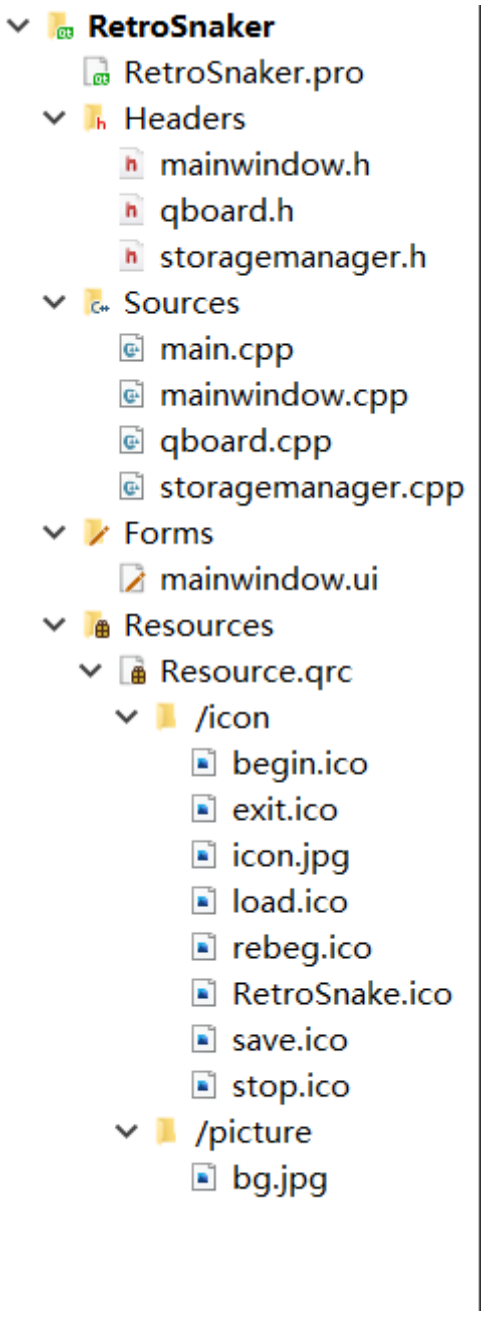
贪吃蛇设计文档

2019011200 计 93 王哲凡

1. 项目基本结构

1.1. 预览



1.2. 项目简介

项目主要由头文件、代码文件、UI 文件、资源文件以及项目配置文件组成。

每个头文件都有与其对应的实现代码文件：

- `mainwindow.h` 和 `mainwindow.cpp`：实现主窗口的设计细节。
- `qboard.h` 和 `qboard.cpp`：实现游戏网格部分的信息与绘制等。
- `storagemanager.h` 和 `storagemanager.cpp`：实现读档存档的功能。

`mainwindow.ui` 中包括了菜单栏、工具栏和按钮等 UI 信息。

资源文件包括图标类（用于按钮和程序图标）和背景类。

---

2. 项目自建类

2.1. `class QBoard`

游戏网格类中包含以下数据成员（均为私有）：

- `quint8 grids[gridSize][gridSize]`：记录网格中每个单位当前是什么状态：
  - `VOID => 0`：空地。
  - `HINDER => 1`：墙体，障碍。
  - `HEAD => 2`：蛇头。
  - `BODY => 3`：蛇身。
  - `FOOD => 4`：食物。
- `quint8 status`：记录当前游戏状态：
  - `BEFORE_BEGIN => 0`：未开始状态。
  - `GAMING => 1`：游戏状态。
  - `STOP => 2`：暂停状态。
  - `END => 3`：终止状态。
- `quint8 dir`：记录当前状态下蛇头方向（不考虑本回合的改变）：
  - `0`：向右。
  - `1`：向上。
  - `2`：向左。
  - `3`：向下。
- `quint8 nextDir`：记录本回合改变后的方向：
  - `0~3`：同上。
  - `4`：未选择或者改变方向无效（与原方向共线）。
- `quint16 grow`：当前回合蛇身待生长长度（每次获得食物增加 3，每次增长减 1）。
- `quint32 time`：当前已进行回合数（蛇头移动长度）。
- `QList<QPair<quint8, quint8>> snake`：用链表记录蛇每节的位置。

包含以下成员函数（均为私有）：

- `void start()`：用于将状态恢复至初始状态，并且随机生成蛇的位置和方向。
- `void addFood()`：在游戏开始和每次蛇吃掉食物时调用，用于在空地随机生成食物。
- `bool legalPosition(int, int)`：判断位置是否在网格内。

包含以下重载函数：

- `void paintEvent(QPaintEvent*)`：用于重绘游戏网格，主要根据上述数据绘制，具体在 UI 设计中介绍。
- `void mouseReleaseEvent(QMouseEvent*)`：用于实现点击空白处变障碍，以及点击障碍变回空白。

包含基本的构造函数与析构函数：

- `QBoard(QWidget *parent = nullptr)`：用于设置随机数种子和网格控件大小，并且调用 `start()` 函数初始化。

除此之外，本类还有两个友元 `class MainWindow` 和 `class StorageManger`，用于在此两类的成员函数中方便调用相关成员。

2.2. `class StorageManager`

存档管理器包含以下成员函数（均为公有）：

- `bool input(const QString&, QBoard*)`：从给定文件中读取格局状态，并写入到游戏窗口的状态，用返回值表示是否读取成功。
- `bool output(const QString&, QBoard*)`：根据游戏窗口的状态，向给定文件写入格局状态，用返回值表示是否写入成功。

本类利用私有成员 `QFile *file` 打开文件，通过 `QDataStream` 类成员进行二进制读写。

考虑到本类作为工具类存在，本类使用了单例模式，即禁用了拷贝构造函数和拷贝赋值函数，并且将默认构造函数设置为私有，唯一的单例对象为类内静态成员 `_instance`，只能通过 `static StorageManager &instance()` 获取引用。

2.3. `class MainWindow`

主窗口类包含以下数据成员（均为私有）：

- `QBoard *board`：游戏网格控件的指针。
- `QTimer *timer`：计时器指针，用于为游戏中每回合的计时。
- `QLabel *showTime`：时间（回合）显示器。
- `QLabel *showScore`：得分显示器（拓展功能）。
- `QLabel *showMaxScore`：历史最高得分显示器（拓展功能）。

包含以下成员函数（均为私有）：

```
void beforeBeginStatus();
void gamingStatus();
void stopStatus();
void endStatus();
```

分别用于对每个游戏状态的相关设定，即为**状态设定函数**。

包含以下重载函数：

- `void keyPressEvent(QKeyEvent*)`：用于在游戏状态中通过“上下左右”或“WASD”控制蛇的方向。

同时还有以下私有槽函数：

```
void begin();
void stop();
void cont();
void rebeg();
void exit();
void save();
void load();
void move();
void about();
void generate();
```

前七个分别与开始游戏、暂停游戏、继续游戏、重新开始、退出游戏、保存游戏、载入游戏功能连接绑定（包括工具栏、菜单栏和按钮）。

`move()` 函数与 `timer` 计时器的 `timeout` 信号绑定，用于蛇每过一段时间的移动行为。

`about()` 和 `generate()` 分别用于加载介绍对话框和自动生成障碍（拓展功能）。

而本类的构造函数则负责了包括 UI 建构、初始状态设定、信号与槽函数连接在内的几项任务。

---

3. 项目实现逻辑

3.1. 游戏状态逻辑

在每次游戏状态变更时，通过调用状态设定函数设置各按钮、工具栏、菜单栏的可用性，并且设定 `board` 中的游戏状态和 `timer` 当前的状态：

- 游戏状态时调用 `timer->start(frame)` 开始计时。
- 其他状态调用 `timer->stop()` 停止计时。

在点击“开始游戏”时，还会调用 `QBoard::addFood()` 函数来随机设置食物，函数中通过获取所有空地位置来随机。

在点击“重新开始”时，还会调用 `QBoard::start()` 函数来重新初始化网格。

在点击“载入游戏”或“保存游戏”时，通过 `QFileDialog::getOpenFileName()` 和 `QFileDialog::getSaveFileName()` 函数获取打开文件以及保存文件的窗口，再调用 `StorageManager::input()` 和 `StorageManager::output()` 载入存档或保存存档。

除此之外，在每次点击按钮或者移动（`move()`）后还需要调用 `board->update()` 使棋盘得以重绘。

3.2. 游戏棋盘绘制逻辑

在 `QBoard::start()` 函数中，通过在非边界网格（内部 38×38）随机蛇头位置以及随机蛇初始朝向，来给出初始蛇的位置。

同时也需要将 `dir`、`nextDir`、`grow`、`time`、`snake`、`grids[40][40]` 等对应初始化。

在绘制函数中，主要根据 `grids` 中的状态来绘图，同时在不同的游戏状态，可以增加不同提示信息，比如在暂停状态下显示“暂停中”字样。

3.3. 鼠标键盘事件逻辑

在 `MainWindow::keyPressEvent()` 中，对于**游戏状态下**的键盘事件，通过 `QKeyEvent::key()` 获取按下的按键：

- `Qt::Key_Up`：上。
- `Qt::Key_Right`：右。
- `Qt::Key_Down`：下。
- `Qt::Key_Left`：左。

通过筛选，来修改 `board->nextDir` 的值。

在 `QBoard::mouseReleaseEvent()` 中，对于**未开始状态下**的鼠标点击放开事件，通过 `QMouseEvent::pos()` 获取点击位置，对于空地和障碍位置进行状态反转。

3.4. 游戏逻辑

在游戏进行中（游戏状态），通过 `timer` 计时器来定时调用 `MainWindow::move()` 函数移动蛇的位置（间隔设置为 $75ms$）。

在 `move()` 函数中，通过 `board->nextDir` 判断是否更换方向，并且向头朝向方向移动一个位置。

如果下一位置是墙体、蛇身或者边界，则进入结束状态（注意将方向恢复至之前）。

如果允许移动，则设置 `grids` 的状态，向 `snake` 头部加入新位置，根据当前的 `grow` 判断是否需要删除尾部位置。

如果下一位置是食物，则将 `grow` 加 3，并且调用 `addFood()` 添加新食物。

注意最后需要调用 `update()` 重绘并更新时间 `time`。

3.5. 存档读档逻辑

通过 `QDataStream astream(file)` 绑定文件指针，在文件开头通过 `readBytes()` 和 `writeBytes()` 读入、写入程序信息，用于判断是否为本项目的存档文件。

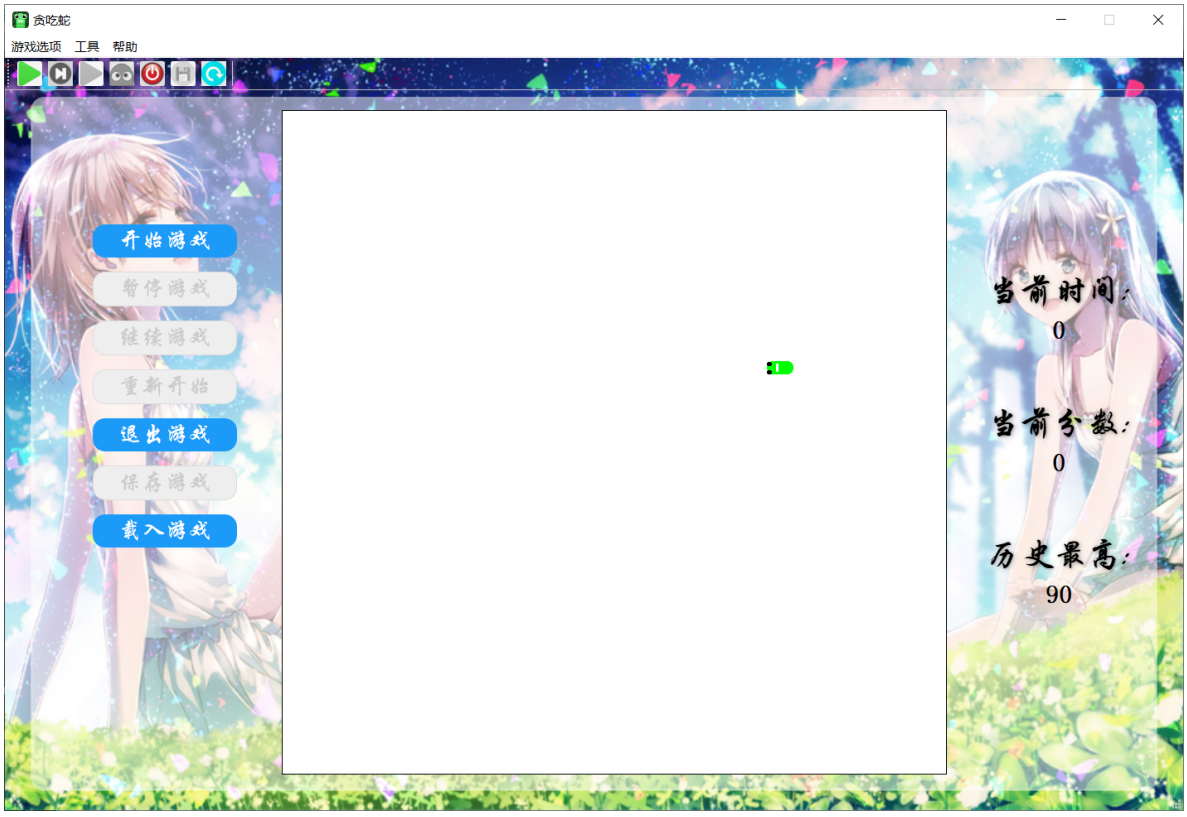
之后通过 `readRawData()` 和 `writeRawData()` 依次读入或写入游戏网格的 `time`、`dir`、`grow`、`grids`、`snake` 等信息。

中间加入对于信息合理度的判断，比如蛇是否连续，蛇头是否只有一个，数据是否符合规则等等。

---

4. UI 设计

4.1. 整体界面



4.2. 控件样式

通过在 UI 文件中添加 qss 控制，比如下面是按钮的样式：

```
QPushButton {
    border-radius: 15px;
    background-color: rgb(27, 154, 247);
    border-color: rgb(27, 154, 247);
    color: #FFF;
    font-size: 28px;
    height: 40px;
```

```
    line-height: 40px;
    padding: 0 30px;
    font-weight: 400;
    font-family: &amp;quot;华文行楷&amp;quot;;
    text-decoration: none;
    text-align: center;
    margin: 5px;
}
QPushButton:visited {
    color: #FFF;
}
QPushButton:hover, QPushButton:focus {
    background-color: #4cb0f9;
    border-color: #4cb0f9;
    color: #FFF;
    text-decoration: none;
    outline: none;
}

QPushButton.disabled, QPushButton.is-disabled, QPushButton:disabled {
    top: 0 !important;
    background: #EEE !important;
    border: 1px solid #DDD !important;
    color: #CCC !important;
    opacity: .8 !important;
}
```

下面是设置主窗口的背景：

```
#MainWindow {
    border-image: url(:/picture/bg.jpg);
}
```

或者直接向 `QLabel` 中填入 HTML 代码，比如：

```
QLabel *showText = new QLabel("<font style='font-family:\"华文行楷\";font-weight:400;font-size:40px;'>" + QStringLiteral("当前时间") + ":</font>", this);
QLabel *scoreText = new QLabel("<font style='font-family:\"华文行楷\";font-weight:400;font-size:40px;'>" + QStringLiteral("当前分数") + ":</font>", this);
QLabel *maxScoreText = new QLabel("<font style='font-family:\"华文行楷\";font-weight:400;font-size:40px;'>" + QStringLiteral("历史最高") + ":</font>", this);
```

对于菜单栏和工具栏，在 UI 文件中加入图标以优化界面。

4.3. 细节部分

通过以下代码可以设置程序的标题和图标（图标位于资源文件中）：

```
setWindowTitle(QStringLiteral("贪吃蛇"));
setWindowIcon(QIcon(":/icon/RetroSnake.ico"));
```

通过以下代码可以使得窗口增加阴影效果：

```
QGraphicsDropShadowEffect *shadow = new QGraphicsDropShadowEffect(this);
shadow->setOffset(0, 0);
shadow->setColor(Qt::gray);
shadow->setBlurRadius(30);
this->setGraphicsEffect(shadow);
ui->horizontalLayout->setMargin(10);
```

4.4. 游戏网格绘图

对于障碍，利用 `QRadialGradient` 类和 `drawRoundedRect()` 函数来绘制渐变色圆角矩形。

对于食物，通过 `drawEllipse()` 来绘制一个红色圆，以此表示。

对于蛇身，通过 `QPainterPath` 类和 `drawPath()` 函数来绘制头部，同方向蛇身，转向蛇身以及蛇尾，对于不同的朝向，通过 `rotate()`、`translate()` 函数调整坐标来完成，并通过 `save()`、`restore()` 保存与恢复。

比如，转向蛇身的绘制方法：

```
QPainterPath turn, stra;
turn.moveTo(-8, -10);
turn.lineTo(8, -10);
turn.arcTo(-28, -28, 36, 36, 0, -90);
turn.lineTo(-10, -8);
turn.arcTo(-12, -12, 4, 4, 270, 90);

```

再通过方向判断即可：

```
p.save();
p.translate(x + 10, y + 10);
p.rotate(180);
p.drawPath(turn);
p.restore();
```

对于暂停状态和终止状态下，还增加了文字标识（拓展功能）。

5. 拓展功能

5.1. 帮助

通过 `QMessageBox::about()` 来显示关于对话框，其中包括游戏功能的介绍。

5.2. 快捷键

为基础的七个功能全部设置了对应的快捷键，比如为开始游戏、暂停游戏、继续游戏均绑定了空格快捷键。

快捷键可以帮助用户减少对于鼠标的依赖。

5.3. 自动生成墙体

在菜单栏增加一个**自动生成墙体**功能，可以自动在未开始状态下，在空地网格中随机选择 20 个变为墙体（仅可使用一次）。

5.4. 得分显示

在时间显示下增加了分数显示，计分方式为吃掉食物数量 $\times 5$。

5.5. 历史最高分显示

在分数显示下增加了历史最高分显示，通过 `QSettings` 类在注册表中保存、更新用户的最高分值。

5.6. 文字标识

在暂停状态增加了**暂停中**标识，并且游戏网格界面整体变暗。

在终止状态增加了**游戏结束**标识和点击提示语，并且游戏网格界面整体变暗。

5.7. 蛇的绘制

通过 `QPainter` 的绘制，使得蛇整体连续，而不是被网格分割成若干个网格。