**设计文档**

2019011332 朱昱熹

主要功能实现原理

1. 菜单栏、工具栏和按钮分别实现相同功能

使用Qt的action机制，实现菜单栏、工具栏和快捷键实现相同功能。将按钮的clicked()信号连接到对应QAction的triggered()信号上，保证按钮也能实现相同功能。

1. 实现游戏中的多个状态和状态转移，实现功能的disable

利用Qt的state machine机制，创建yetToStart（未开始）、playing（游戏）、interrupt（暂停）和stop（终止）四个状态。为了确保上述四个状态都可以退出，将上述四个状态设计为game状态的子状态，并增加finish这一终止状态（final state），增加game向finish的状态转移，并使状态机的finished()信号连接到quit()函数。在四个状态之间增加状态转移，状态转移主要由1中所述的QAction来触发（仅yetToStart向interrupt的状态转移由load成功触发）。

为每个状态的entered()信号和exited()信号连接对应的槽函数，对状态进行有关设置。设置包括了：有关功能的enable和disable，（进入yetToStart状态时）初始化游戏状态，开始或终止（暂停）计时，开始或终止鼠标的追踪和响应/键盘（指方向键）的响应等。

1. 实现游戏主要逻辑

为便于添加和删去自定义障碍物，使用二维数组存储障碍物信息；为便于获取头尾信息，使用队列存储蛇身体有关信息。每到计时器发出timeout()信号时（即经过一帧），计算出下一帧的蛇身体信息、目标物信息等，判断游戏是否已经结束，导出一个二维数组显示所有格子的当前状态，并绘制出来。

对于方向的控制由direction和lastDirection两个变量控制。direction负责主要控制方向，每次计算下一帧时利用direction的信息，两帧之间可用键盘控制修改direction中的信息。为了避免出现蛇掉头自杀的情况，使用lastDirection记录蛇上一帧的移动方向，并不允许direction设置为lastDirection的反方向。

对于自定义障碍物的设置，在yetToStart状态下监听mouseMoveEvent事件，实时计算鼠标当前所在位置对应的格子，存入hover变量中，并在绘制中用特殊颜色显示。当鼠标按下时，改变hover所标示的格子的障碍物存在情况（如果蛇处在该位置则忽略）。

1. 窗口和绘图设计

整个窗口水平分布，左侧放置弹簧，右侧为sideBar，设置layoutStrech为3:1，空出左侧3/4的空间用于绘制贪吃蛇主界面。右侧sideBar垂直分布，上方为LCDNumber，中间为一弹簧，下方为7个按钮，设置layoutStrech为2:3:1:1:1:1:1:1:1。

在绘图时，根据窗口大小，改变painter的单位坐标长度，使得绘制的贪吃蛇主页面可以根据窗口大小动态调整大小。获取鼠标输入时也对应进行调整，使得可以正确计算鼠标所在的格子。

1. 保存和装载游戏

使用Qt提供的QJsonDocument、QJsonArray和QJsonObject类，将游戏信息存储成json格式。

类设计

1. RCPoint类

Qt自带的QPoint使用x和y表示点的两个坐标，然而贪吃蛇游戏中更倾向于使用row和col来表示，且游戏中常常需要做row/col和x/y的转换，使用QPoint容易导致语义混乱，故设计RCPoint类。

RCPoint类还实现了和QJsonArray相互转化的接口，方便游戏状态的存储和导入。

1. PointConverter类

程序中使用row和col表示格子位置，但是绘图时需要转化成x和y坐标，这一转化依赖于贪吃蛇界面在窗口的位置、格子大小等参数。为了避免转化中对参数的硬编码，将这部分逻辑单独设计为一个类。

该类只有静态变量和静态成员函数，将构造函数设计为私有，避免实例化对象。

1. Playground类

记录游戏信息的主要类。将游戏信息和窗口分离开，Playground类负责游戏逻辑的实现，并提供toPlate()接口供窗口绘制。

该类同样重载了和QJsonObject相互转化的接口，方便游戏状态的存储和导入。

1. Snake类

QMainWindow的子类。主要负责状态机的控制，处理玩家输入并传递给Playground类计算，绘制窗口呈现给玩家。