# QSnake设计文档

## 实现功能

## 游戏部分

|  |  |
| --- | --- |
| 功能 | 描述 |
| 地图 | 一个有上下左右边界的长方形，划分为小方格单元。所有的游戏实体都由方格单元组成。 |
| 蛇 | 蛇可以沿上下左右方向移动。玩家可以操控蛇的移动方向。当蛇头移出地图或蛇头撞到蛇身时，游戏结束。一般情况下，蛇以普通速度移动，吃了特殊食物星星（下述）后，一段时间内以高速运动。 |
| 苹果 | 苹果在游戏过程中始终存在，且始终只有一个。蛇吃掉苹果后，长度增长一个单元，普通速度有一定提升（有上界），游戏分数加1，同时随机放置一个新的苹果。如果地图上没有空单元放置新的苹果，若地图上有特殊食物星星（下述），将它变成苹果，否则游戏结束。 |
| 单人/双人 | 玩家可设定单人/双人模式。单人模式下，有一条蛇，地图较小。双人模式下，有两条蛇，地图较大，若两条蛇相撞则游戏结束。 |
| 穿墙 | 玩家可开启穿墙选项。开启穿墙后，方格地图的上下边缘和左右边缘视作联通。 |
| 砖块 | 玩家可开启砖块选项。砖块是方格地图上的固定障碍。蛇撞击砖块后，游戏结束。开启该选项后，游戏开始时会随机读取一张预置的砖块地图。 |
| 路障 | 玩家可设定路障个数（可以为0）。路障是地图上的非固定障碍，每隔一段时间后会重新随机排布。 |
| 星星 | 玩家可开启星星选项。星星是一种特殊食物，它会在地图上出现一段时间，消失一段时间，循环往复。蛇吃掉星星后，长度增长一个单元，普通速度有一定提升（有上界）,并在一段时间内以高速运动，游戏分数加5。 |
| 暂停 | 玩家可以在游戏过程中暂停，并在之后恢复。 |
| 自动 | 暂停时，玩家可以开启/关闭自动模式。只有在单人模式下，路障个数为0且星星选项关闭时，才可以开启自动模式。 |

## 窗口部分

|  |  |
| --- | --- |
| 开始窗口 | 开始窗口是打开游戏后的第一个窗口，有开始按钮与设置按钮。点击开始按钮后，跳转到游戏窗口，开始游戏；点击设置按钮后，弹出设置窗口，进行游戏设置。 |
| 设置窗口 | 在设置窗口可以更改1.1节所述的各类设置。 |
| 游戏窗口 | 在游戏窗口中，玩家可以看到游戏画面，并操控蛇的移动方向。单人模式下，蛇的方向通过WASD键或者上下左右键操控；双人模式下，两条蛇的方向分别通过WASD键和上下左右键操控。玩家可以通过P键暂停游戏，暂停时会弹出暂停窗口。游戏结束后，跳转到游戏结束窗口。 |
| 暂停窗口 | 在暂停窗口中有自动模式复选框和恢复按钮。玩家可以通过自动模式复选框开启/关闭自动模式。若自动模式不可使用（见1.1节详述），自动模式复选框会被禁用。玩家可以通过恢复按钮、P键或是关闭按钮关闭暂停窗口，暂停窗口关闭后游戏继续。 |
| 游戏结束窗口 | 在游戏结束窗口中，有重新开始按钮与设置按钮，二者的功能与开始窗口中的两个按钮类同。在游戏结束窗口中，展示了蛇的游戏得分（单人模式下，一个得分；双人模式下，两个得分）。 |

## 模块设计

在模块设计中，游戏逻辑与UI界面的代码分离。单人与双人模式的游戏逻辑分别由GameSingle与GameDouble两个类实现，UI界面分别由GameWindowSingle与GameWindowDouble两个类（均继承自QWidget类）实现。游戏逻辑类负责推进游戏进程、处理玩家输入，UI界面类负责展示游戏画面、接收用户输入。游戏逻辑类与UI界面类之间通过函数调用以及信号槽机制进行通信。

游戏逻辑类与UI界面类的通信

|  |  |
| --- | --- |
| 单人模式 | |
| 通信内容 | 通信方式 |
| 游戏画面更新 | GameSingle对象发出update()信号，GameWindowSingle对象的repaint()槽响应。 |
| 蛇的转向 | GameWindowSingle对象的keyPressEvent函数响应玩家操作，并调用GameSingle对象的turnSnake函数。 |
| 游戏结束 | GameSingle对象发出gameOver()信号，GameWindowSingle对象的handleGameOver()槽响应。 |
| 游戏暂停/恢复 | GameWindowSingle对象的pause/resume函数响应玩家暂停/恢复请求，并调用GameSingle对象的pause/resume函数。 |
| 自动模式开启/关闭 | GameWindowSingle对象的resume函数中判断自动模式的开启/关闭，并调用GameSingle对象的startAutomatic/closeAutomatic函数。 |
| 双人模式 | |
| 游戏画面更新 | 类同单人模式。 |
| 蛇的转向 | GameWindowDouble对象的keyPressEvent函数响应玩家操作，并调用GameDouble对象的turnSnakeRed/turnSnakeBlue函数。 |
| 游戏结束 | 类同单人模式。 |

为实现代码去重，GameSingle类与GameDouble类继承自父类Game。Game类从两个子类中提取出了随机种子设定、苹果生成、路障移动、星星的产生与消失的相关逻辑。

在UI方面也需实现代码去重。GameWindowSingle类与GameWindowDouble类继承自父类GameWindow。开始窗口类StartWindow、单人游戏结束窗口类GameOverWindowSingle、双人游戏结束窗口类GameOverWindowDouble由于具有相同的按钮逻辑，继承自父类GameOverWindow。

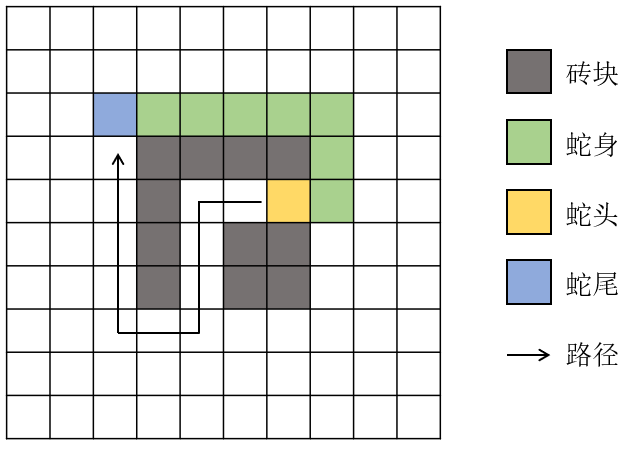
此外，Snake类封装了蛇的运动状态、游戏分数以及玩家对蛇输入的操作序列，并提供一些相关函数。PauseWindow类实现暂停窗口。SettingWindow类实现设置窗口。

## 自动模式算法

## 安全状态

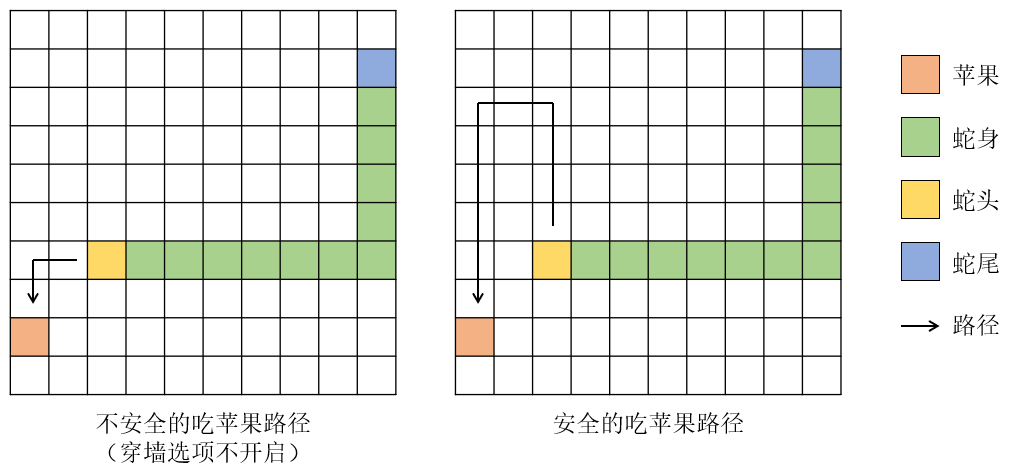
规定蛇的长度大于等于2，即蛇头与蛇尾不得重合。定义蛇身为蛇头和蛇尾中间的部分。

当存在一条从蛇头到蛇尾，不经过蛇身、砖块的路径时，我们称蛇处在安全状态。只要蛇始终沿着自身与该路径构成的环路运动，游戏就永远不会结束。



## 吃苹果路径

假设存在一条从蛇头到苹果，不经过蛇身、蛇尾、砖块的路径，则称其为吃苹果路径。若蛇沿着这条路径运动并吃掉苹果后，处于安全状态，则称该路径为安全的吃苹果路径。

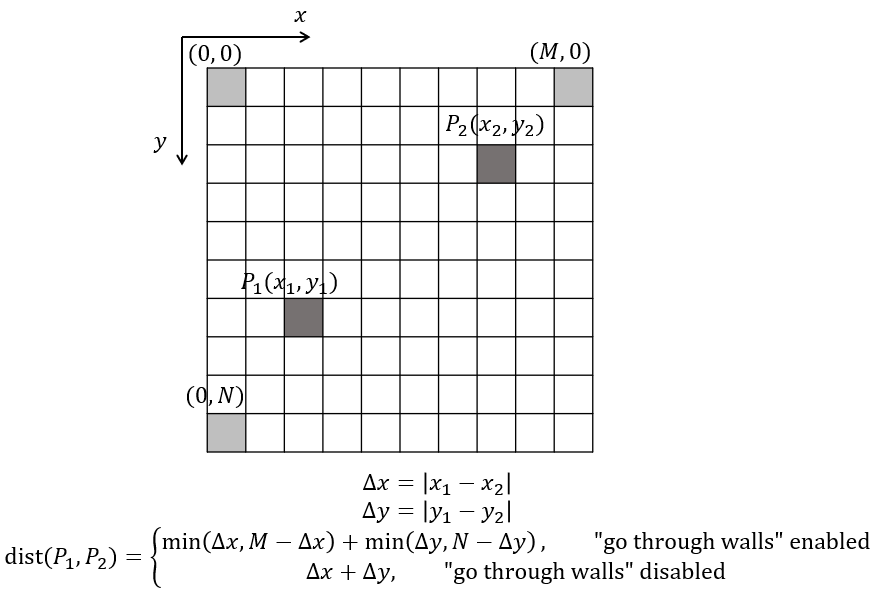


## 算法流程

首先，利用广度优先搜索寻找一条吃苹果路径。如果这条路径是安全的吃苹果路径，则让蛇沿着这条路径去吃苹果。

如果不存在吃苹果路径，或找到的那条吃苹果路径不是安全的吃苹果路径，则利用广度优先搜索判断当前蛇是否处在安全状态。

如果当前蛇处在安全状态，尝试让蛇沿着上下左右方向移动，找出所有能让蛇保持安全状态的方向。在这些方向中，找出走完后蛇头与蛇尾的曼哈顿距离最远的方向，让蛇沿该方向走一步。穿墙选项的开启与否影响了曼哈顿距离的计算方式。



如果当前蛇不处在安全状态，尝试让蛇沿着上下左右方向移动，找出所有不会直接导致游戏结束的方向。如果存在这样的方向，在这些方向中随机挑选一个，让蛇沿该方向走一步。否则，游戏必然在下一步结束。

## 算法解释

本算法的大部分内容是容易理解的。需要解释的部分是，为什么当不存在吃苹果路径，或找到的那条吃苹果路径不是安全的吃苹果路径，但蛇处在安全状态时，要在保持安全状态的前提下，走出使蛇头尾曼哈顿距离最远的一步。

当算法无法得出安全的吃苹果路径，但蛇处在安全状态时，我们要做的是使蛇在保持安全状态的情况下，不断地调整蛇的位置形态，直到算法能够得出一条安全的吃苹果路径。

考虑使用最短路径策略进行蛇的位置形态调整，即每次算法无法得出安全的吃苹果路径时，让蛇沿广度优先搜索所得的从蛇头到蛇尾的路径走一步。那么，蛇头会被蛇尾引导，蛇不会充分利用地图上的空余空间进行位置形态调整，蛇所能呈现的位置形态种类受到局限，容易陷入死循环。

相反，如果在保持安全状态的前提下，走出使蛇头尾曼哈顿距离最远的一步，蛇就能充分利用地图上的空余空间来调整位置形态，算法更有机会在某一时刻得到一条安全的吃苹果路径。

