

设计要点:

首先此程序分为两个窗口，一个是由 `mainwindow` 类来完成，另一个则由 `widget` 类完成。它们在项目中分别在 `mainwindow.h` 和 `widget.h` 中声明。

`mainwindow` 就是点开程序的开始界面，界面中有四个按钮，分别对应单人模式、双人模式、自动模式和退出。点开一个按钮后就会进入相应模式的界面，而这个界面是属于 `widget` 类的。

整个项目采用的是面向对象的编程，一共有 5 个类。除去刚刚介绍过的 `mainwindow` 类和 `widget` 类还有 `snake` 类、`food` 类、`wall` 类。

`snake` 类中存放着蛇的信息，包括：蛇身体各个方块的坐标，蛇头的方向、蛇前进的速度、蛇的条数和蛇的长度。由于存在双人模式，则所有的状态均分成两份存在数组中。在 `snake` 类执行生成函数时，会根据蛇的数量 `snakeNum` 来决定数组分配的大小。

`food` 类中存放的为食物的信息，其中包括：食物方块的坐标 (`foodX foodY`)、特殊食物方块的坐标 (`specialFoodX specialFoodY`)、标记特殊食物究竟是哪种食物的状态、特殊食物效果的持续时间和特殊食物出现的周期。

`wall` 类中存放的是墙的坐标和传送门的坐标（其实传送门可以单独开出一个类出来，但最后还是没有这么做.）。

这 5 个类在项目中的结构为：最上层的为 `mainwindow` 类，其中有一个 `widget` 类的数据成员 `subwidget`，而 `widget` 类中存放有 `snake` 类、`wall` 类和 `food` 类的一个对象指针。`Widget` 下的三个对象之间是可以互相调用对方的数据的，因此可以用一张图来表示它们之间的关系，如图 1.。



图 1. 五个类之间的关系

代码详解:

接下来分别详细介绍各个类中所包含的有关功能的函数。

`snake` 类中有意义的函数只有 `drawSnake()` 函数，用途是在传进来的 `painter` 中画蛇的身体。其中算法的核心就是利用 `for` 语句把存在 `snake` 对象里的蛇身体坐标用方块在地图上画出来，同时根据 `i` 的奇偶给蛇上深浅交替的颜色。

`wall` 类中我存了两个类型的数据，一个是属于障碍物的，还有一个是属于传送门的。传送门是 `project` 要求完成的额外功能。`Wall` 的生成函数是给负责给传送门上色的 `RGB` 变量赋值，同时启动传送门闪烁的计时器（每隔 0.4s 交换相邻两个色块的颜色）。而 `timerEvent()` 则负责每隔 0.4s 交换传送门相邻的色块，达到闪烁的目的。此外，`wall` 类中还有用于生成障碍物坐标的 `wallGenerater()` 函数、生成传送门的 `portalGenerater()` 函数、在 `painter` 中画障碍物和传送门的 `drawWall()` 和 `drawPortal()` 函数、检测蛇头是否撞

上墙的 `checkWall()` 函数、检测蛇头是否撞上传送门的 `checkPortal()` 函数。在此项目中，撞墙后采取的操作是蛇随机选择一个和墙平行的方向，然后蛇的长度减 3。

`food` 类中存有生成食物坐标的 `foodGenerate()` 函数、生成特殊食物的 `specialFoodGenerate()` 函数、在 `painter` 中画食物和特种食物的 `drawFood()` 函数、检测蛇头吃食物的 `checkFood()` 函数和控制特殊食物效果持续时间的 `timerEvent()` 函数。其中 `checkFood()` 函数对于吃到普通食物的操作是重新生成食物坐标，重新生成障碍物的坐标，蛇的长度加 1，蛇的刷新时间减少 20ms，下限为 30ms 刷新一次蛇的位置。如果蛇吃到了特殊食物，加速效果比较好处理，就是刷新时间减半，而按键反转和无敌状态这两种效果需要将存在 `widget` 类中的对应 `bool` 变量 `keyReverse` 和 `invincibility` 赋值为 `true` 即可，之后就交给 `widget` 类的函数去具体执行这两个效果。

`Widget` 类是所有类中最重要的一个，这个类主要负责整合其下三个对象的数据，接收键盘的按键信号，绘制整个地图和游戏的各个内容，控制蛇的运动、蛇的方向，检测游戏是否达到结束的状态，执行特殊食物的效果和蛇的简易自动寻路功能。生成函数负责给 `food`、`snake`、`wall` 等指针分配空间同时将这几个对象按照图 1 的方式串联起来（启动计时器还有生成初始地形啥的代码里看得比较清楚，这里就不再赘述了）。`painterEvent()` 遵从 `update()` 信号每隔 20ms 刷新一遍整个地图，而绘制各个部件的话只要调用对应对象的 `drawxxx()` 函数即可。`KeyPressEvent()` 负责根据按键的不同改变 `snake` 对象的 `state` 变量，即改变蛇的方向。如果 `bool` 变量 `keyReverse` 为 `true`，则对应在 `KeyPressEvent()` 函数中做出调整即可。`move()` 函数遵从 `timerEvent()` 函数每隔 `speed`（`snake` 中的有关蛇移动刷新时间的变量）毫秒调用一次，先将蛇的坐标整体向后移一位，再根据 `state` 的不同分别将蛇头的坐标向对应的方向移动一个单位长度。移动之后执行 `wall->checkPortal()`、`food->checkFood()` 和判断是否为自动模式或者执行 `wall->checkWall()`。如果为自动模式的话则执行 `widget` 中的 `autoMove()` 函数。`check()` 函数检测游戏是否到达终局状态。`autoMove()` 函数则是自动模式的核心。额，在写这个算法前完全没有看过网上的一些经典算法，于是全部的 AI 算法都是 `if` 语句，设置了不知道多少分支，比较的蠢这个算法。主要分成三种情况：1、蛇撞墙 2、蛇与自身相交 3、前两个情况的补集。情况 1 就绕开墙向着靠近食物的方向改变 `state`，情况 2 我只考虑了蛇沿 9 字形撞法然后沿着蛇尾巴边缘移动。情况 3 就是普通的最短路径，策略是先沿 `y` 方向靠近再沿着 `x` 方向靠近。这个算法没有考虑到的方面有很多，而且这个算法也比较笨，后来才知道有追着蛇尾巴跑这个策略。

`Mainwindow` 类主要是提供一个界面进入不同模式。将 `button` 和对应的函数用 `connect()` 连接起来就行了。槽函数也就是改一下蛇的数量和改变 `widget` 类中 `bool` 变量 `autoMove` 的值而已。

最后就是生成本项目所有随机量的函数 `randEx()`。之前有用 `time` 来作为随机种子，但是那个 `time` 是 `ms` 级别的，不能满足在短时间内生成多个随机变量的要求，所以上网查阅了一些资料后采用 CPU 时钟（是这么说的吗）这样微秒级别的时间作为随机数种子。这个函数作为全局函数，因为所有的类都要用到随机数。

操作方法：

`wasd` 控制蛇 1 的方向，`8654` 控制蛇 2 的方向。另外在初期蛇运动速度较慢的情况下不要按方向键按得太快，因为我不知道如何设置两次 `KeyPressEvent()` 的调用间隔时间。