华 中 科 技 大 学

《面向对象的程序设计》

上机实验报告

专业班级：自卓 1901　　专业班级：自动化 1905

学　　号：U201914581　　学　　号：U201914689

学生姓名：屈文杰　　学生姓名：孟繁鹏

目录

一. 实验名称

蚂蚁回家小游戏

二. 实验需求分析

本游戏的最初灵感来源于一款记忆力小游戏。本游戏的基本设定为一只小蚂蚁需要避开所有的蜘蛛，并且吃掉所有的食物，最终回到家中，即判定为游戏胜利。该游戏中玩家共有三条生命，每遇到一次蜘蛛便会减少一条生命，生命值为 0 时即判定失败。

由于这是一款记忆力小游戏，在设定中我们增加了迷雾功能，即玩家需要开局记住随机地图的信息，随后在移动第一步之后地图会被迷雾遮盖，玩家需要凭借记忆完成游戏。为降低游戏难度，我们会适当给予玩家提示，即离玩家最近蜘蛛和食物的距离。

三. 实验分工

为了快速高效地完成实验内容，对小组成员进行了必要的分工，促进成员的相互合作和交流，提高工作效率。具体的分工如下：

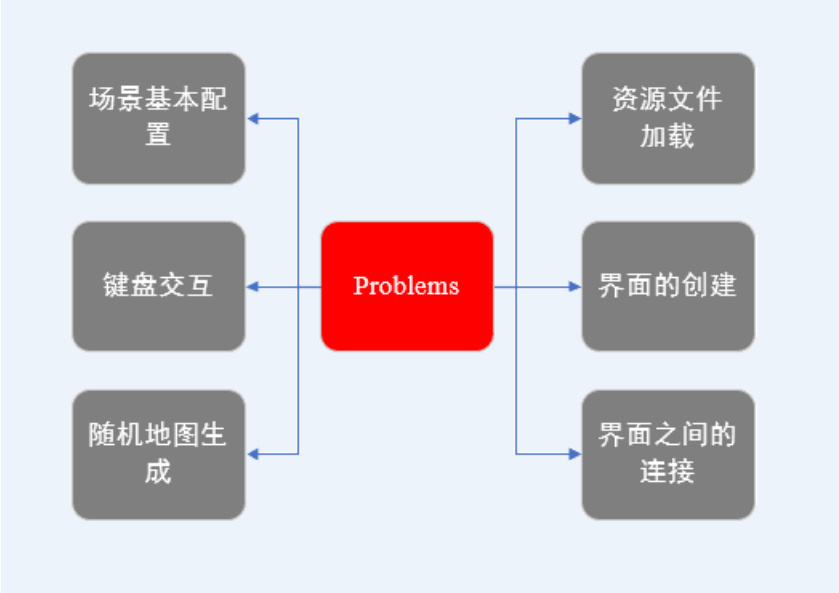
表 1　成员的具体分工

姓　名	任务描述	完成情况
屈文杰	C++游戏逻辑实现 广度优先搜索计算最短路径 美术设计	完成

	报告和文档撰写	
孟繁鹏	游戏玩法和策略的提出 QT 框架学习使用 可视化界面设计开发 并查集+队列优化的随机地图生成	完成

四. 概要设计

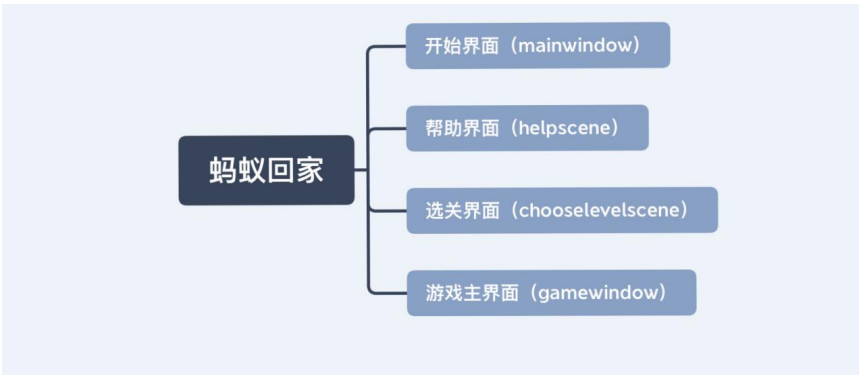
本项目可以拆分出如下几个子问题



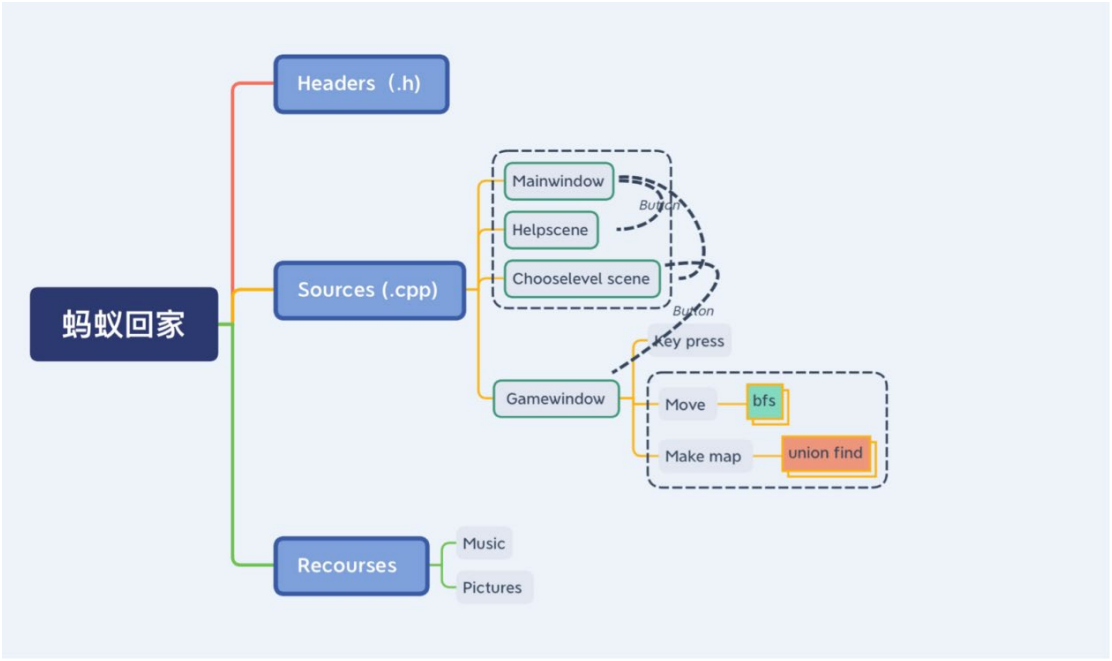
左边三个问题可归结为游戏逻辑的实习游戏主场景算法的设计。

右边三个问题可归结为 QT 的应用，利用 QT 中自带的库来进行游戏各个界面的初始化创建。

游戏的各个界面：


蚂蚁回家
开始界面（mainwindow）
帮助界面（helpscene）
选关界面（chooselevelscene）
游戏主界面（gamewindow）

游戏整体框架设计：


蚂蚁回家
Headers（.h）
Sources (.cpp)
Recourses
Music
Pictures
Mainwindow
Helpscene
Chooselevel scene
Button
Button
Gamewindow
Key press
Move
bfs
Make map
union find

五. 详细设计

1. 可视化界面的创建

1.1 按钮的封装

为了代码的简洁美观，我们创建了 mypushbutton.h 和 mypushbutton.cpp 这两个文件，以封装游戏中出现的所有按钮，我们定义了自己的按钮的一个类，它继承于 QPushbutton,同时定义了自己的成员函数来实现上下弹跳的功能，从而在需要使用按钮挂件时直接使用 mypushbutton 这个类来执行操作。

1.2 各个界面的连接

在界面中我们设置了几个按钮，用信号和槽来进行窗口与窗口之间的连接，connect 函数可以将点击按钮的信号和隐藏这一界面、显示下一界面联系到一起，我们看到在开始和帮助按钮的 connect 函数的槽函数中还有一个 connect 函数，它把监听到的返回信号和槽函数联系到一起，退出游戏按钮的的槽函数是 close，可以将点击按钮和关闭窗口联系到一起。

2. 游戏逻辑的设计

2.1 游戏基本场景的配置——iniscene 函数的使用

我们通过 iniscene 函数设置了界面的大小，图标，音效标题等基本配置，我们还设置了一个返回按钮。在原有配置的基础上添加了一个清空地图数组的函数，并且通过 QPainter 绘图事件设计了动画曲线，设置了胜利界面以及失败界面的动画。

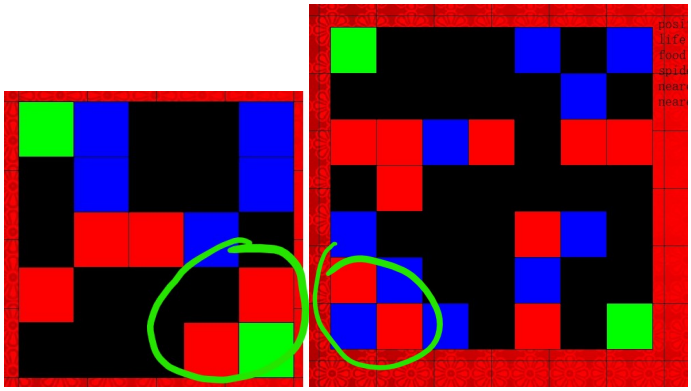
2.2 随机地图的生成——makemap 函数

这里提到的随机地图其实并不准确，因为这个函数并没有将地图画出来，我们实际上是通过构建一个二维数组，并在二维数组中添加随机数，生成了随机地

图的坐标。

在上一个界面中我们传入了一个 level 数据，我们拿他去规定生成矩阵的大小，我们设置一个随机种子，调用系统时间，进行迭代生成了随机矩阵，她由 0，1，2，3，4 组成，分别代表背景食物蜘蛛蚂蚁以及终点。

但是我们会发现，地图中经常出现包围圈的问题，其中主要有两种包围圈：第一种是蜘蛛将终点或起点堵死在角落，如下图左；另一种是存在一个由蜘蛛围成的包围圈，如下图右。



对于第一种包围圈：

我们采用了并查集的方法，将相邻的两个蜘蛛归入同一个并查集，并进行判断，如果一个并查集中有两个点都和图中的边界相连，则说明这一群蜘蛛与墙壁一起堵死了图中的某个区域，那么这种地图存在包围圈

对于第二种包围圈：

在此我们借助了一种判断图中是否存在环的算法，通过队列实现了包围圈的查找以及排除，具体过程如下：

首先我们以蜘蛛为节点，将相临的两个蜘蛛连一条边建图。

1. 将所有出度为 0 的点删除
2. 将所有出度为 1 的点推进队列

循环以下操作：

1. 取队列头（此点出度一定为 1）
2. 删除该点和所有该点所连的边
3. 删除操作二后出度为 0 的点
4. 将操作二后出度为 1 的点推进队列

直到队列为空

如果结束循环还有点没被删除，则说明有环。

注意：上文提到的相邻指其中一个蜘蛛在以另一个蜘蛛为中心的 3*3 方格内。

判断函数会输出该地图是否存在包围圈，于是我们不断生成随机地图，直到判断函数返回值为 1，也就是地图中不存在包围圈

2.3 绘制地图

在这个函数中，我们利用了 QT 自带的 QPainter 绘图事件按照之前生成的随机地图数组绘制我们的随机地图。但由于我们添加了迷雾功能，为了实现它我们设置了一个迷雾变量 f，并为其设置了一个初值 0。当游戏初始打开时，f=0，绘制去掉迷雾的全地图，随后当我们按下键盘时，将 f 的值改为 1。在执行这一函数之前，我们用一个 if 语句判断一下 f 是否为 0。若为 0，则绘制全地图，若为 1，则绘制带迷雾的地图。

2.4 处理操作对象移动带来的变化——move 函数

我们构造了一个 move 函数，当我们的蚂蚁移动到一个位置时，我们先判断地图矩阵中该位置有没有道具如食物，蜘蛛之类，如果有，我们对这些道具变量进行操作，如果没有我们会更新地图数组。

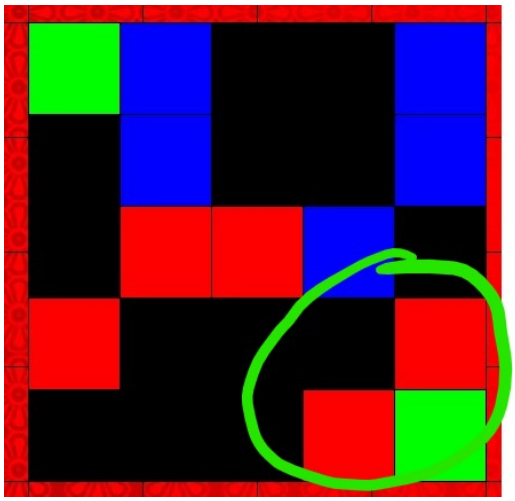
2.5 利用键盘操作对象——keypress 键盘操作事件

这里，我们采用 QT 中自带的 keypress 键盘按下事件，通过 WSAD 来实现其上下左右的移动，并在每次移动后重新更新地图。

由于我们最高难度使用了 10*10 的大地图，为了提高玩家的游戏体验，我们会适当地在旁边一栏给出提示，即剩余食物数量，剩余蜘蛛数量，离蚂蚁最近的食物和蜘蛛的距离分别是多少。为了实现这样的功能，我们在此使用了求最短路径的 bfs 广度优先搜索算法，借助队列，实现了最短路径的计算，

六. 调试与改进

1. 在随机地图的生成的过程中，我们遇到了一个问题，直接向地图中填入随机数，生成的地图有可能存在包围圈，如下图所示。



我们运用数据结构课上学习的图论知识，和并查集、队列的数据结构去除了包围圈。

2. 在实际测试中我们发现，如果不给玩家更多的提示，游戏体验并不好。为了提高玩家的游戏体验，我们实现在旁边一栏给出提示，即剩余食物数量，剩余蜘蛛数量，离蚂蚁最近的食物和蜘蛛的距离分别是多少。为了求出这些数值，我们在此使用了求最短路径的 bfs 广度优先搜索算法，借助队列，实现了最短路径的计算。

七. 参考文献

[1] 严蔚敏, 吴伟民. 数据结构: C 语言版[M]. 清华大学出版社有限公司, 2002.

[2] Cormen T. 算法导论:第 2 版[M]. 机械工业出版社, 2007.

[3] Lippman S B, Lajoie J. C++ primer[M]. Posts & Telecom Press, 2005.