**上海应用技术大学课程设计报告**

课程名称 算法设计与应用课程设计

设计题目 走迷宫问题

院系 计算机科学与技术 专业 软件工程 班级 19104221

姓名 张智强 学号 1910400740 指导教师 姜丽 日期 2021/7/8

目录

[1. 目的与要求 3](#_Toc19108)

[2. 课程设计内容说明 3](#_Toc2305)

[2.1. 需求分析 3](#_Toc861)

[2.2. 概要设计 4](#_Toc30108)

[2.3. 详细设计 4](#_Toc13786)

[2.3.1. 菜单(Menu) 4](#_Toc16224)

[2.3.2. 地图(Map) 5](#_Toc22136)

[2.3.3. 标记地图(MarkMap) 6](#_Toc32481)

[2.3.4. 地图生成器(MapGenerator) 6](#_Toc25220)

[2.3.5. 路径规划器(RoutePlanner) 7](#_Toc28467)

[2.3.6. GUI模块 9](#_Toc30673)

[2.3.7. 游戏启动器(MazeGame) 11](#_Toc4516)

[2.3.8. 主模块(main) 11](#_Toc10777)

[2.4. 调试与测试 12](#_Toc27680)

[2.4.1. 菜单测试 12](#_Toc3142)

[2.4.2. 迷宫生成测试 12](#_Toc8149)

[2.4.3. 迷宫路径规划器的测试 13](#_Toc21776)

[2.4.4. 主程序的测试 14](#_Toc16852)

[2.4.5. 运行时菜单测试 16](#_Toc414)

[2.5. 待改进方向 17](#_Toc28085)

[2.5.1. Prim迷宫生成算法 17](#_Toc11777)

[2.5.2. DFS迷宫偶数问题 17](#_Toc2713)

[2.5.3. 最小转弯路径 17](#_Toc23241)

[2.6. 使用说明 17](#_Toc15613)

[3. 结论和体会 18](#_Toc20487)

[附录1：参考文献 18](#_Toc5163)

# 目的与要求

猪八戒与唐僧身处一个m×n的迷宫中，如图所示。每一个方格表示迷宫中的一个房间。这m×n个房间中有一些房间里面有妖精居住，不允许进入。在迷宫中任何房间均可沿8 个方向进入相邻的安全房间。猪八戒位于迷宫的(p，q)方格中，他必须找出一条通向唐僧所在的(t，s)方格的路。请设计和实现一个算法帮助猪八戒找出这样一条道路。

程序的主界面如下：

迷宫问题

1. 查看迷宫地图

2. 迷宫设置

3. 开始走迷宫

0. 退出

2. 扩展要求

（1）求最短的路径

（2）每改变一次前进方向算作转弯一次，求转弯次数最少的路径

（3）要求猪八戒在抵达唐僧之前，必须走遍所有住有妖精的房间各一次，铲除妖精后，再找到唐僧。

3.提示

考虑迷宫的数据结构

考虑路径搜索算法的策略

# 课程设计内容说明

## 需求分析

实现一个可视化的迷宫小游戏，允许用户键盘操控，并且允许用户动态添加障碍物，妖怪等游戏元素。能够自动地路径规划，完成自动导航，能够自动消灭所有的妖怪，并解救唐僧。使用贴图元素使得界面尽量好看。

## **概要设计**

程序计划分为如下几大模块

**地图生成模块**

使用不同算法实现地图的程序生成

**路径规划模块**

使用不同算法实现动态的实时地路径规划

**用户交互模块**

实现可视化迷宫与可视化路径规划与导航，提供一定的用户交互性和可玩性。

## **详细设计**

### **菜单(Menu)**

本程序的菜单模块位于util/Menu.hpp中，这是一个基于确定型有穷自动机理论的一个菜单的实现，可轻易地实现n级复杂菜单，以及动态菜单的创建与显示。

在游戏的开局菜单中，使用到了如下菜单定义，本菜单模块用法极其简单，只需要构造时传入菜单名，添加菜单项，启动菜单循环三步即可在控制台实现菜单输出。

多级菜单的状态迁移图如图2.3.2-1所示，

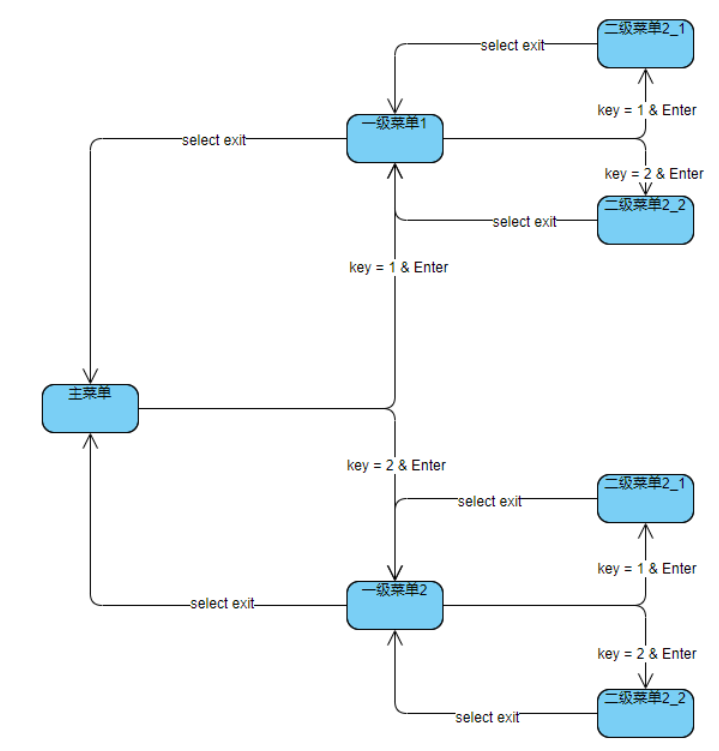


图 2.3.2-1 菜单状态迁移图

### **地图(Map)**

地图实际上是一个二维数组，具体实现参考代码"map/Map.hpp"中定义的Map类。它底层使用动态开辟的一维数组，通过面向对象封装，隐藏了底层数据结构，对上层只提供二维的访问方式，实现了动态创建二维数组。并且提供了一些便捷的批量操作方法。该类使用需要传入一个模板参数，表示二维数组的元素类型。后续的标记地图类便派生于此类。

### **标记地图(MarkMap)**

标记地图为本课设提出的一个概念，它实际上是继承自Map类，模板参数为bool的一个派生类。它用于标记一副指定尺寸的地图，能够进行标记，清除，判定等操作。实际上，障碍物地图(ObstacleMap)，妖怪地图(MonsterMap)，均为MarkMap换了一个名字。

#### **多层标记地图(MultiMarkMap)**

由于本项目课设中的基础功能需要将妖怪看做障碍物，避开妖怪行进，而扩展功能中又要求击杀所有的妖怪，故提出多层标记地图的概念通过C++的动态多态性实现路径规划所需地图的动态切换，大大地提高了项目的灵活性。MultiMarkMap继承自MarkMap，重写了hasMark方法，添加了addMarkMap(MarkMap \*markMap)方法，实现了逻辑上的地图融合。addMarkMap方法的实现非常简单，就是将另一个MarkMap的指针放入一个链表即可。hasMark的重写方法也非常简单，即遍历所有的地图层，只要任意一层地图在该位置存在标记，那就整个判定为存在标记。

### **地图生成器(MapGenerator)**

MapGenerator类实现了所有地图生成器的一个基类，其构造函数需要传入一张空白的MarkMap，起点，终点信息。其中存在一个纯虚函数generate方法，该方法用于地图的生成，交给派生自MapGenerator类的子类实现。

#### **DFS迷宫生成**

DfsMapGenerator类继承自MapGenerator类，实现了通过DFS算法生成迷宫的算法，该算法实现良好。

#### **随机Prim迷宫生成**

PrimMapGenerator类继承自MapGenerator类，该类预计实现随机Prim算法生成迷宫，该算法暂未实现。

### **路径规划器(RoutePlanner)**

首先，我们需要引入路径的数据结构，一条路径是一个由起点到终点的所有顶点构成的一组有序的点集。

Path类定义了路径的数据结构，为一个顶点链表，该类包含了一些常用方法，如计算该路径的各个位移段，计算转弯次数，计算路径长度(路径长度规定水平竖直为2，斜方向为3)。

RoutePlanner类为一个抽象类，其中构造函数传入路径规划所需的障碍地图，存在一个纯虚函数Path plan(const Point &start,const Point &end)用于交给子类实现其中的路径规划算法。

#### **DFS路径规划器**

DFS路径规划器实现位于类DfsRoutePlanner，该类继承于RoutePlanner抽象类，实现了基于DFS回溯法的路径规划算法。该类实现的算法为非递归实现的回溯法，使用了STL的list链表作为栈。

#### **BFS路径规划器**

BFS路径规划器实现位于类BfsRoutePlanner，该类继承于RoutePlanner抽象类，实现了基于BFS的路径规划算法，该算法使用了优先队列(PriorityQueue)，实现了最短路径的规划。

##### **优先队列(PriorityQueue)**

类PriorityQueue实现了一个优先队列，该优先队列的优先规则为优先数越小，优先级越高。

该类定义为一个模板类，实现了对任意类型的数据存取。

有三个方法

* push(T t,int priority)

向优先队列中添加元素，并且赋予优先级

* T pop()

从优先队列中取出优先级最高的元素

* bool empty()

判断优先队列是否为空

该类目前是直接封装了STL的priority\_queue，使其更加易用，后续可考虑手工实现优先队列。

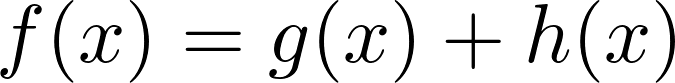
#### **A\*路径规划器**

本项目迷宫的数据结构采用二维数组表示地图，1表示存在障碍物，0表示空闲地带，整个迷宫可以看做一个无向图，其中每个空闲的网格都算图中的一个结点，因为迷宫可沿着8个方向前进，故每一个结点最多有8个度。

对于迷宫问题而言，每个相邻结点的边权值相同，如果使用Dijkstra算法，其实无法按照原有的贪心依据进行快速地最短路径规划，此时Dijkstra其实已经退化为了BFS算法。

而A\*算法补充了其贪心依据，使得到终点的预估距离成为了其贪心依据的一部分，故其又算作Dijkstra算法的改进版本。由于A\*算法需要根据预估距离作为贪心依据的一部分，故该算法又算作为启发式搜索算法。

A\*算法通过下面这个函数来计算每个结点的优先级



其中，

* f(x)表示总代价，当我们下一次选取要遍历的结点时，优先选择总代价f(n)最小的结点。
* g(x)为已经花费了的当前代价，即走到该结点花费的总代价。
* h(x)为预计要耗费的预估代价，到终点的预计抵达的代价，这也是A\*作为启发式搜索的启发函数。

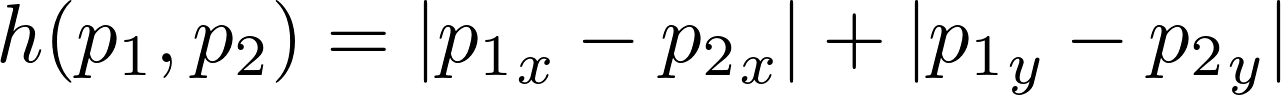
每次沿着启发函数f(x)更小的方向前进，每前进一次再次计算四周的启发函数f(x)。

##### **关于启发函数h(x)的选取**

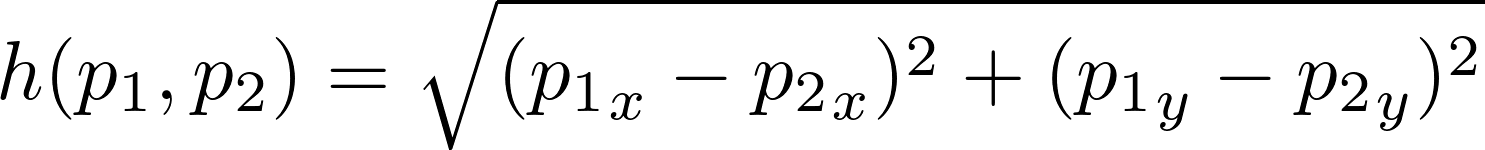
启发函数h(x)需要预估当前点抵达终点所花费的代价，该代价可以看作是抵达终点的距离。

* 当地图上仅能够上下左右四个方向移动时，我们可以使用曼哈顿距离作为启发函数。

曼哈顿距离即两点x,y坐标之差的绝对值之和，即公式如下：



* 当地图上除了能够上下左右移动时，还能够四角移动，此时可以使用对角距离。
* 当地图上能够任意方位移动时，启发函数可使用欧几里得距离，即公式如下：



### **GUI模块**

#### **精灵(Spirit)**

该类借鉴了游戏开发领域中的精灵概念，即游戏中一块可独立控制的区域，拥有自己的位置，姿态，贴图等，该类实现了猪八戒和唐僧的渲染与运动。

#### **绘图仪(MazePlotter)**

该类的主要作用为在一块cv::Mat对象（画布）上绘制游戏中的所有元素，如地图，精灵等。

#### **游戏控制器(MazeController)**

该类中封装了游戏的各个应用逻辑，包括人物的运动逻辑，添加，清除障碍物和妖怪的行为，对待妖怪的行为模式切换与路径规划器的切换等等。

如图2.3.6-1定义了障碍物地图，与妖怪地图，还定义了MultiMarkMap类型的指针，该指针在其构造函数中被赋值为障碍物地图与妖怪地图的融合标记地图的对象指针。

如图2.3.6-2 定义了几个路径规划器的切换函数，\_routePlanner的类型为RoutePlanner基类指针，其指向一个子类对象，通过动态多态性，实现了路径规划器的动态切换。

如图2.3.6-3，同理，可以通过切换路径规划与碰撞检测的地图指针来实现是否击杀与避开妖怪的判定。



图 2.3.6-1 地图定义



图 2.3.6-2 路径规划器切换



图 2.3.6-3 击杀妖怪的模式切换

### **游戏启动器(MazeGame)**

游戏入口中初始化了游戏运行时所需的各种资源，并初始化默认键位映射设置，生成迷宫图，加载游戏等等。

### **主模块(main)**

在main.cpp中，主要功能为显示主菜单，启动MazeGame模块，引导入游戏。

## **调试与测试**

### 菜单测试

如图2.4.1-1 使用基于状态机的菜单，可方便地定义出一个菜单，得到如图2.4.1-2的输出结果。

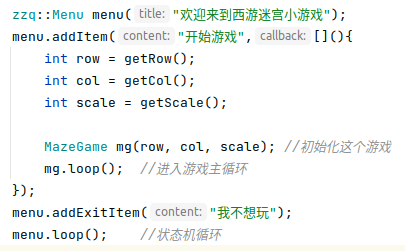


图 2.4.1-1 菜单定义代码

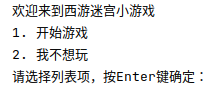


图 2.4.1-2 菜单运行结果

### 迷宫生成测试

如图2.4.2-1展示了迷宫地图的定义，地图生成器的定义与地图生成并打印的代码。

实现了如图2.4.2-2展示的运行结果，迷宫生成结束。

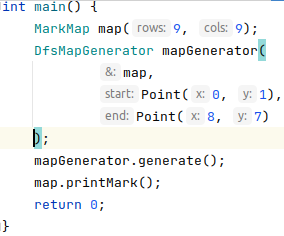


图 2.4.2-1 迷宫生成代码

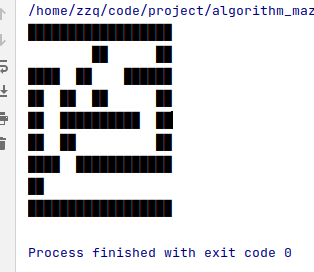


图 2.4.2-2 迷宫生成运行结果

### 迷宫路径规划器的测试

如图2.4.3-1为迷宫路径规划的代码，基于之前的迷宫生成的结果，配合DFS路径规划器求解出一系列迷宫的关键顶点数据。

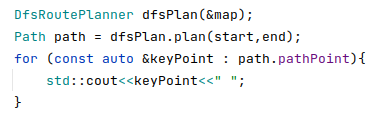


图 2.4.3-1 迷宫路径规划代码

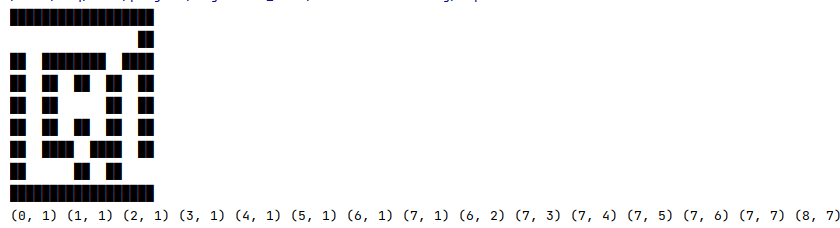


图 2.4.3-2 迷宫路径规划运行结果

### 主程序的测试

进入游戏主界面后，如图2.4.4-1所示，单机鼠标中间可添加妖怪如图2.4.4-2所示

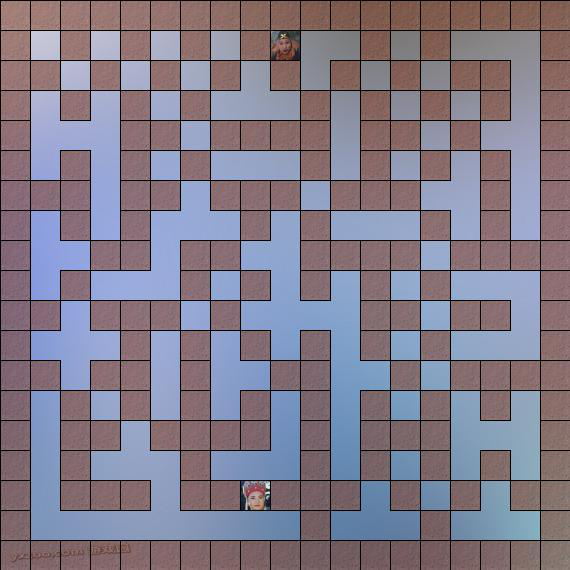


图 2.4.4-1 游戏主界面

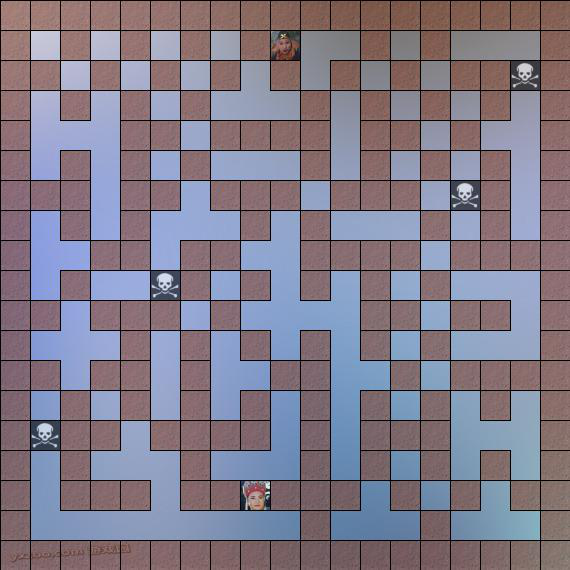


图 2.4.4-2 添加妖怪

启动避让妖怪模式且开始自动导航，如图2.4.4-3所示，避让了一个妖怪。

当允许击杀妖怪后，如图2.4.4-5所示，将弹出击杀妖怪的窗口。

最后，如图2.4.4-4所示，游戏胜利。

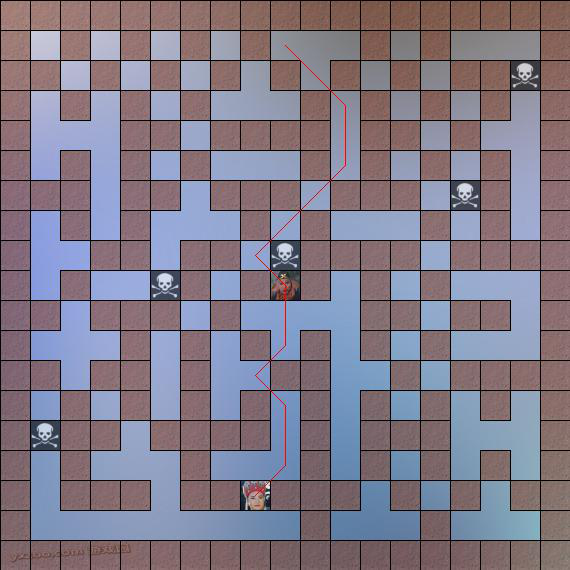


图 2.4.4-3 避让妖怪模式



图 2.4.4-4 游戏胜利



图 2.4.4-5 猪八戒击杀妖怪

### 运行时菜单测试

游戏运行时，按下ESC键，游戏暂停，可在控制台处显示如图2.4.5-1,图2.4.5-2,图2.4.5-3的菜单输出。

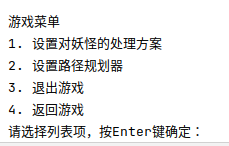


图 2.4.5-1 游戏运行时菜单

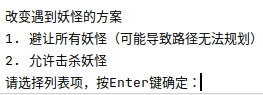


图 2.4.5-2 设置对妖怪的处理方案

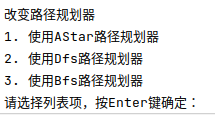


图 2.4.5-3 动态设置路径规划器

## **待改进方向**

### Prim迷宫生成算法

随机Prim迷宫生成算法可以生成可玩性更强的，迷惑程度更高的迷宫，目前框架已经搭建在了PrimMapGenerator类中，但是由于时间有限，该算法尚未实现。

### DFS迷宫偶数问题

当输入偶数的迷宫尺寸时，迷宫生成算法将会可能生成无解的迷宫，出现这一局限性的原因主要是由于需要生成围墙。目前，解决方法之一就是对用户输入的偶数迷宫+1转换为奇数迷宫。

### 最小转弯路径

由于时间原因，该功能暂未实现，以下给出一些思路

**DFS深度优先搜索**

使用DFS深度优先搜索，图中的每个结点最多有8个分支，可以求解出所有的可行路径，并求出其中最小转弯的路径，但是该方案当迷宫规模较大时，该方案搜索空间变得异常庞大，几乎不可用。

**BFS宽度有限搜索**

对八个方向进行BFS，但不是像最短路径那样一步步地搜索，而是搜索到一个方向后就一直沿着这个方向搜索到尽头，这样就能保证转弯次数最小。

## **使用说明**

键盘按下q w e a d z x c可沿着8个方向移动，p开始路径规划，l开始自动导航，

u,i切换是否击杀妖怪模式，o击杀所有妖怪并解救唐僧，鼠标左键添加障碍物，鼠标右键清除障碍物，鼠标中键添加妖怪，ctrl+鼠标中键可导航到任意地点。

# 结论和体会

本次算法设计与应用课程设计，本人选用了C++语言和OpenCV图像处理库（仅用来实现界面渲染）实现了一款有趣的迷宫小游戏，对各种路径规划算法DFS,BFS,AStar启发式搜索算法有了更深的理解，对一些迷宫生成算法也有了一些了解，对STL中的vector，list，priority\_queue的用法也有了一定的了解。

本次算法设计与应用课程设计中，我也学习到了除了算法之外的很多知识，如将C++的动态多态性第一次完美地使用在了本次课设项目中，实现了运行时动态切换各种方案。

同时，本次课设使用OpenCV还熟悉了很多OpenCV中的绘图API的使用。

当完成本次课程设计时，能够玩起自己亲手编写的小游戏，感觉还是相当有意思的一件事。

# 附录1：参考文献

1. [优先队列] <https://blog.csdn.net/weixin_36888577/article/details/79937886>
2. [A\*算法在线学习与可视化] <https://www.redblobgames.com/pathfinding/a-star/introduction.html>
3. [DFS迷宫生成算法] <https://blog.csdn.net/jjwwwww/article/details/82872922>
4. [Prim迷宫生成算法] <https://blog.csdn.net/jjwwwww/article/details/82891213>
5. [BFS最小转弯路径] <https://www.cnblogs.com/fzl194/p/8677699.html>
6. [OpenCV图片缩放] <https://www.jianshu.com/p/3ababd718030>