# 设计说明书

目录

[一、 系统需求分析 1](#_Toc137651785)

[二、 总体设计 1](#_Toc137651786)

[三、 详细设计 2](#_Toc137651787)

[四、 系统调试 3](#_Toc137651788)

[五、 结果分析 4](#_Toc137651789)

[六、 总结 4](#_Toc137651790)

## 系统需求分析

在本项目中，我们的目标是实现一个寻找最短路径的算法，并在图形界面上显示出障碍物和最终路径。系统需要满足以下需求：

* 随机产生二维障碍物环境，包括路径随机起点、终点，以及障碍物和边界信息（形状、个数、大小、位置都随机，要避免太简单环境）。
* 设计算法，实现从起点到终点的最短路径的求解。
* 输入输出障碍物环境信息、最短路信息。

## 总体设计

整个系统由四个类构成：Point、Map、Findpath和Show。其中，Point类表示一个二维点的坐标，Map类表示地图信息和障碍物集合，Findpath类负责寻找最短路径，Show类用于显示地图和路径。

* **Point类:**
* **数据成员:**

x（横坐标）

y（纵坐标）

* **成员函数:**

构造函数

* **Map类:**
* **数据成员:**

obstacles（障碍物点集合）

is\_obstacles（标记是否为障碍物的哈希表）

w（地图宽度）

h（地图高度）

* **成员函数:**

P2I（将点转换为索引）

GetRandObstacles（随机生成障碍物）

GetIs\_obstacles（将障碍物加入到is\_obstacles集合中）

* **Findpath类:**
* **数据成员:**

w（地图宽度）

h（地图高度）

mymap（Map对象）

pass\_path（经过的路径集合）

final\_path（最终路径）

* **成员函数:**

Find（寻找最短路径）

Isxline（判断是否可以向右移动）

Isyline（判断是否可以向下移动）

Isslash（判断是否可以斜向移动）

* **Show类:**
* **数据成员:**

w（地图宽度）

h（地图高度）

* **成员函数:**

Show\_back（显示背景网格）

Show\_obsandline（显示障碍物和路径）

Show\_all（显示全部内容）

## 详细设计

* **Point类:**
* **数据成员:**

x（横坐标）

y（纵坐标）

* **成员函数:**

构造函数：用于初始化Point对象的坐标。

* **Map类:**
* **数据成员:**

obstacles（障碍物点集合）：用vector<Point>存储障碍物点的坐标。

is\_obstacles（标记是否为障碍物的哈希表）：使用unordered\_set<int>存储障碍物点的索引。

w（地图宽度）

h（地图高度）

* **成员函数:**

P2I(Point p)：将Point对象转换为索引，使用公式key = p.y \* w + p.x。

GetRandObstacles()：根据地图尺寸，在合适的范围内生成随机的障碍物坐标，并存储到obstacles中。

GetIs\_obstacles()：遍历obstacles，将障碍物点的索引存储到is\_obstacles中。

* **Findpath类:**
* **数据成员:**

w（地图宽度）

h（地图高度）

mymap（Map对象）

pass\_path（经过的路径集合）：使用unordered\_set<int>存储已经经过的路径点的索引。

final\_path（最终路径）：用vector<Point>存储最短路径上的点坐标。

* **成员函数:**

Find()：在地图上寻找从起点(0,0)到终点(w-1,h-1)的最短路径。使用广度优先搜索算法进行遍历，通过判断是否可以向右移动、向下移动或斜向移动来确定下一个点，直到找到终点或无法继续搜索。

Isxline(Point now)：判断当前点是否可以向右移动，即判断右侧的点是否为障碍物并且未经过。

Isyline(Point now)：判断当前点是否可以向下移动，即判断下方的点是否为障碍物并且未经过。

Isslash(Point now)：判断当前点是否可以斜向移动，即判断右下方的点是否为障碍物并且未经过。

* **Show类:**
* **数据成员:**

w（地图宽度）

h（地图高度）

* **成员函数:**

Show\_back()：使用图形库初始化画布，并绘制背景网格和边框。

Show\_obsandline()：创建Findpath对象，调用Find方法获取最短路径，并在画布上绘制障碍物和路径。

Show\_all()：调用Show\_back()和Show\_obsandline()，显示全部内容。

## 系统调试

在系统调试阶段，我们需要确保以下功能正常运行：

地图的生成和障碍物的随机放置。

寻找最短路径算法的正确性。

图形界面的显示。

## 结果分析

在完成系统调试后，我们可以进行结果分析，主要包括以下方面：

随机生成的地图和障碍物是否符合预期。

最短路径算法能否正确找到从起点到终点的路径。

图形界面能否正确显示地图、障碍物和路径。

## 总结

在设计说明书的最后，我们可以对整个项目进行总结和反思，包括：

是否满足了系统需求分析中的所有功能。

设计过程中遇到的问题和解决方案。

对系统性能、可扩展性和用户体验的评估。

后续可能的改进和优化方向。

+-----------------+ +-------------------+

| Point | | Map |

+-----------------+ +-------------------+

| - x: int | | - obstacles: vector<Point> |

| - y: int | | - is\_obstacles: unordered\_set<int> |

+-----------------+ | - w: int |

| - h: int |

| |

| + P2I(p: Point): int |

| + GetRandObstacles() |

| + GetIs\_obstacles() |

+-------------------------+

^ ^

| |

| |

| |

| |

| |

v v

+-------------------+ +-------------------+ +-------------------+

| Findpath | | Show | | Main |

+-------------------+ +-------------------+ +-------------------+

| - w: int | | - w: int | | |

| - h: int | | - h: int | | |

| - mymap: Map | | | | |

| - pass\_path: unordered\_set<int> | | | |

| - final\_path: vector<Point> | | | |

| | | | | |

| + Find() | | + Show\_back() | | |

| + Isxline(now: Point): bool | | + Show\_obsandline() | | |

| + Isyline(now: Point): bool | | + Show\_all() | | |

| + Isslash(now: Point): bool | | | | |

+-------------------+ +-------------------+ +-------------------+