|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования  FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра параллельных вычислительных технологий | | |
| Расчетно-графическое задание | | |
| по дисциплине «Методы построения и анализа алгоритмов» | | |
| **Построение пути для мобильного робота в двумерном пространстве** | | |
|  | | |
|  | Факультет | ПМИ |
|  | Группа | ПМИ-11 |
| Студент | Рогальский Михаил |
| Вариант | 2 |
|  |  |
|  |  |
| Преподаватель | ЩУКИН ГЕОРГИЙ АНАТОЛЬЕВИЧ |
|  |  |
| Новосибирск  2023  Оглавление  [Постановка задачи 3](#_Toc122541450)  [Описание решения 4](#_Toc122541451)  [GUI 4](#_Toc122541452)  [Составляющее 4](#_Toc122541453)  [Подход к решению 4](#_Toc122541454)  [Отрисовка 6](#_Toc122541455)  [Результат работы 7](#_Toc122541456) | | |

# Постановка задачи

Мобильный робот движется по плоскости. Робот - материальная точка с координатой (Cx, Cy); робот может двигаться в любом направлении без ограничений. Также на плоскости находятся объекты-препятствия. Заданы начальная и конечная координаты робота Init=(Initx, Inity) и Goal=(Goalx, Goaly), дополнительно может быть задано несколько промежуточных координат.

Задача: найти кратчайший путь для робота из начальной координаты в конечную, не приводящий к столкновению робота с препятствиями. При наличии дополнительных координат путь должен проходить через каждую дополнительную координату минимум по одному разу. Если путь не возможен - сообщить об этом.

Дано:

* Тип: разбиение на клетки;
* Препятствия: прямоугольники (клетки);
* Наличие промежуточных координат: без промежуточных координат.

В ходе задачи требуется реализовать основные возможности:

* интерактивное задание и изменение сцены: препятствий и координат начальной, конечной точек пути;
* отображение сцены: препятствий, начальной, конечной точек пути;
* возможность сохранения сцены в файл и загрузки ее из файла;
* отображение построенного пути или сообщения о том что путь не удалось построить;
* задание параметров алгоритма (через конфигурационный файл).

В качестве GUI была выбрана мультимедийная библиотека SFML.

Пространство конфигураций разбивается на регулярные клетки. Клетки одного фиксированного размера.

Клетки делятся на две группы - свободные и занятые, свободные клетки полностью лежат в Cfree, занятые - частично или полностью пересекаются с Cobs. Вводится граф, где вершинами являются свободные клетки. Соседние свободные клетки связываются ребром. Ребром связываются соседние клетки по горизонтали и вертикали, а также по диагонали, если все 4 клетки в этом “квадрате” свободны; ребра имеют вес - 1 для движения по горизонтали и вертикали и sqrt(2), т.е 1.4, для движения по диагонали. Для полученного графа решается задача поиска кратчайшего пути. Считается, что робот передвигается от центра одной свободной клетки к центру другой.

## Описание решения

### GUI

Установка SFML для Visual Studio (x64, Release method):

<https://www.sfml-dev.org/tutorials/2.5/start-vc.php>

Используемые библиотеки: sfml-graphics-2.dll, sfml-system-2.dll, sfml-window-2.dll.

### Составляющее

В решении использованы: классы (точка(Point), прямоугольник (Rectangle), граф(Graph)), алгоритм поиска кратчайшего пути (алгоритм Дейкстры с вершинами класса точек), и другие функции, необходимые для решения задачи (см. описание ниже).

### Подход к решению

Идея заключается в том, чтобы поделить пространство окна программы (поле) на клетки фиксированного размера (задается ширина клетки), в зависимости от разрешения сторон программы. При этом необходимо создать рамку (зону), за которую нельзя выступать роботу при передвижении. Расставив препятствия и начальную с конечной точкой, программа сформирует кратчайший путь для заданной конфигурации (метод удаления центра препятствия и всех путей к нему из графа, и нахождение пути по тем центрам, что не являются препятствиями).

Для задания параметров используется файл settings.txt, а выходные параметры записываются в output.txt после каждого запуска и построения пути.

(см. main.cpp).

Для начала в файле settings.txt пропишем параметры (см. README.txt):

ширина\_окна(пиксели) высота\_окна(пиксели)

ширина\_клетки(пиксели)

столбец\_клетки\_старта строка\_клетки\_старта

столбец\_клетки\_финиша строка\_клетки\_финиша

режим\_для\_расставления\_препятствий

столбец\_препятствия строка\_препятствия (Для режима 1, перечисляются ниже построчно все номера).

Режимы:

0 – расстановка препятствий случайным образом;

1 – расстановка на основе номеров столбцов и строк клеток, куда будут выставлены препятствия;

2 – левой кнопкой мыши кликать для установки препятствий после запуска программы.

Нумерация клеток по столбцам (строкам) начиная с единицы до максимального кол-ва - 1 , сколько всего клеток может вместиться с такой шириной клетки и такой шириной окна (высотой окна).

Левый угол первой клетки разбиения устанавливается на x = unit , y = unit (пиксели).

Занесем крайние точки рамки в sf::VertexArray lines в кол-ве 5 штук, рисуется линией на основе sf::LineStrip. Краевые точки для рамки-прямоугольника устанавливаются на углы крайних клеток разбиения, причем по X максимальная величина unit\*(win\_width/unit), где выражение в скобках определяет целое значение (отсекается дробная часть) количества клеток по горизонтали, умноженное на величину одной ширины клетки. А по Y максимальная величина unit\*(win\_height/unit).

Далее в sf::Text text\_info и text\_hasnotShortPath установим шрифт, сам текст, размер, цвет и стиль. В первой переменной хранится информация по началу программы (нажать по Enter для показа пути), во второй информация, что не имеется пути (в нужной ситуации будет выведено).

В граф vertices будем заносить центральные точки наших клеток-разбиения (центры определяют начало движения для каждой клетки) через двойной цикл for, где в std::vector <sf::Points> topLeftPoints пойдут левые углы для каждых клеток. Причем, если мы находимся не на последнем столбце по Х и не на последней строке по У, мы и будем добавлять в граф центры от данных клеток методом centerPoint(), для определения центра прямоугольника(клетки) которого определяется смещение вправо-вниз, что недоступно для краевых клеток из-за ограничения рамки.

После формируются начальная и конечная точки по указанным номерам столбца и строк из файла. Координаты определяются смещением через первый левый верхний центр клетки, добавляя номер строки или столбца – 1, умноженный на ширину одной клетки.

Для отображения точки старта и финиша нарисуем в соответствующем центре клетки круг небольшого радиуса и какого-либо цвета (в задаче желтый и синий).

Построим все дороги, то есть ребра с весами, (по горизонтали, вертикали, диагонали) для каждых из центров клеток через функцию buildPath, обрабатывая следующие случаи нахождения рамки от центра клетки: сверху и слева, сверху и справа, сверху, справа снизу, справа, снизу слева, снизу, слева, и если нет рамки поблизости. То есть заносим добавление путей, проходя по часовой стрелке у краев (удобно представить квадрат 3х3, начиная проверку условия от левого верхнего угла и заканчивая центром). Причем передобавления одних и тех же путей и вершин в методе Point::add\_edge(…) не будет, так как проверяется условие, есть ли там вершины и сами пути, или совпадает ли вес ребра прошлый с текущим заносимым.

Если режим построения препятствий равен 0, то запускаем функцию genRectangles, а если 1 – chooseRectangles.

В первой функции происходит генерирование 15% от общего числа клеток (оптимальное количество) препятствий, при этом удаляем вершину препятствия (центра клетки) из графа и все дороги к нему, а для последующей отрисовки методом sf::Quads заносим в std::vector <sf::Vertex> 4 точки препятствия, полученные через Rectangle::getPoints().

Во второй функции происходят аналогичные действия, только в начале из файла считываются клетки на основе номеров столбцов и строк.

Запускается алгоритм Дейкстры с приоритетной очередью (находятся для каждой вершины ближайшая по расстоянию смежная вершина в массив parent, при чем были заданы условия, что если смежная вершина находится от текущей точки по диагонали в какую-либо из двух сторон (например, вправо-вверх), то должны иметься пути в обе стороны по отдельности (вправо и вверх), то есть что там нет препятствий по обе стороны от движения по диагонали (значит пути не удалялись ранее).

После формируется путь через функцию drawPathMode в sf::VertexArray result\_line\_path для отрисовки по sf::LineStrip.

### Отрисовка

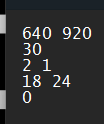
С помощью window.draw рисуются: рамка, краевые точки клеток разбиения, препятствия в виде закрашенных клеток, текст и окружности стартовой и конечной точек соответствующего цвета.

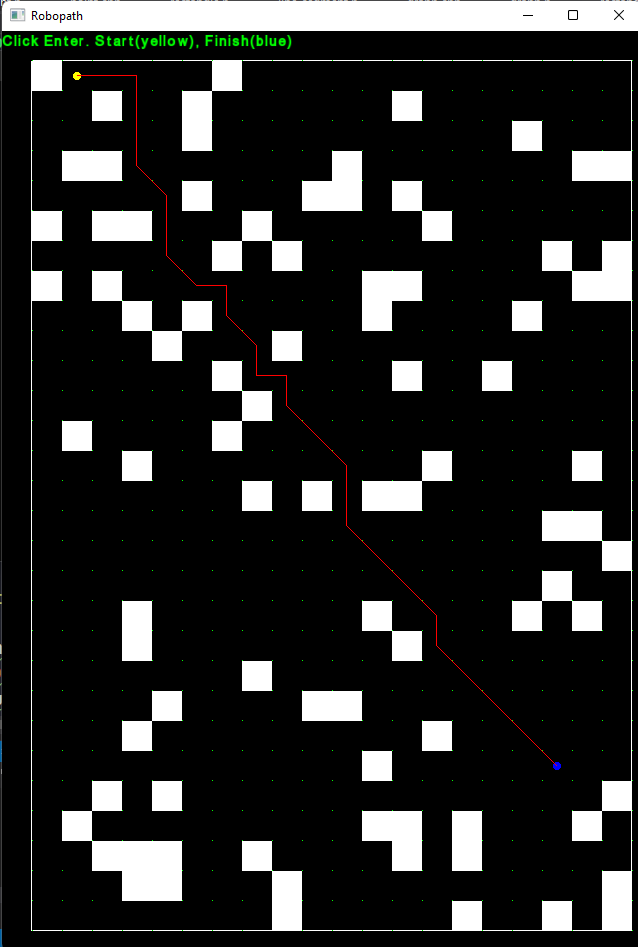
Если режим 1 или 0 и если имеется путь и был нажат Enter, нарисуется путь, иначе если (нет пути) – сообщение, что нет пути.

Иначе если режим равен 2 (сами препятствия расставляем) и нажат Enter, и есть путь, нарисуется путь. Перед этим через sf::Event будет проверено, что если нажата была левая кнопка мыши, то считывается место нажатия в точку, для всех центров клеток (кроме старта и финиша) найдется тот, который пересекается с этой точкой. После центр удалится из графа, как и дороги к нему, а краевые точки этого препятствия занесутся в points\_of\_genRectangles, и номер столбца и строки в файл output.txt.

# Результат работы

1) Режим генерации клеток (0), окно 640х920, ширина клетки 30.Файл settings.txt:





2) Режим прописывания точек в файле (1). Данные settings.txt:

300 300

20

1 1

13 13

1

12 13

13 12

2 2

1 3

4 2

4 1

5 3

6 4

7 5

7 6

7 7

7 8

7 9

6 14

6 13

5 12

13 11

13 10

13 9

14 6

14 7

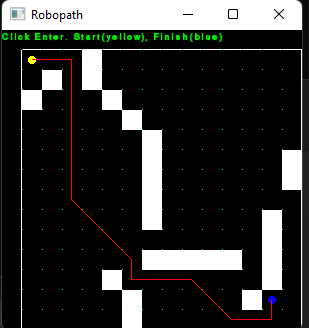
7 11

8 11

9 11

10 11

11 11



3) Режим выбора препятствий нажатием кнопки (2). Данные settings.txt:

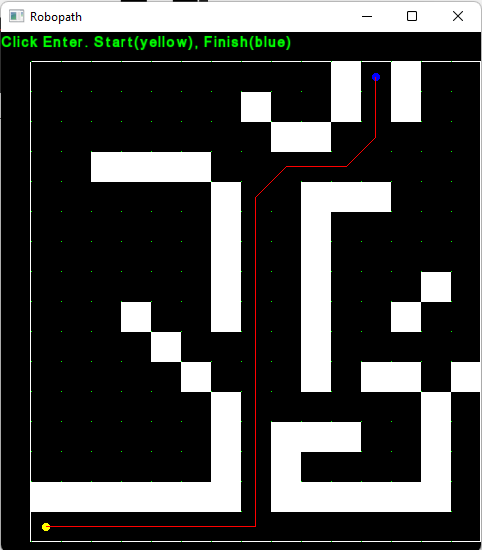
480 520

30

1 16

12 1

2



Данные файла output.txt (для последующей загрузки в settings сменить режим с 2 на 1):

480 520

30

1 16

12 1

2

1 15

2 15

3 15

4 15

5 15

6 15

7 15

7 14

9 14

9 15

10 15

11 15

12 15

13 15

14 15

14 14

14 13

14 12

15 11

9 13

10 13

11 13

11 1

13 1

13 2

11 2

10 3

9 3

10 5

11 5

12 5

10 6

10 7

10 8

10 9

10 10

10 11

12 11

13 11

13 9

14 8

7 13

7 12

6 11

5 10

4 9

7 9

7 8

7 7

7 6

7 5

6 4

5 4

4 4

3 4

8 2

4)Режим 2. Нет пути. Файл settigns.txt из предыдущего пункта.

