МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий
Базовая кафедра №234 — Управляющих ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Системный анализ информационных технологий»
(наименование дисциплины)

Тема курсовой работы: Моделирование системы планирования движения наземного робота

Студент группы: ИКМО-05-22,
Белов Александр Владимирович _____
(учебная группа, фамилия, имя, отчество) (подпись студента)

Руководитель курсовой работы: к. т. н., доцент базовой кафедры №234,
Бочаров Никита Алексеевич _____
(должность, звание, ученая степень, фамилия, имя, отчество) (подпись руководителя)

Рецензент (при наличии)**:** _____ _____
(должность, звание, ученая степень, фамилия, имя, отчество) (подпись рецензента)

Работа предоставлена к защите до « ___ » декабря 2024 г.

Допущен к защите до « ___ » декабря 2024 г.

Москва 2024 г.

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий
Базовая кафедра №234 — Управляющих ЭВМ

Утверждаю

Заведующий кафедрой: _____
(Подпись)

_____
(фамилия, имя, отчество)

ЗАДАНИЕ
на выполнение курсовой работы по дисциплине
«Системный анализ информационных технологий»

Студент: Белов Александр Владимирович **Группа:** ИКМО-05-22
Тема: Моделирование системы планирования движения наземного робота

Исходные данные: Разработать программную модель, которая будет строить кратчайший маршрут по карте местности, по модели создать отчёт

Перечень вопросов, подлежащих разработке, и обязательного графического материала:

1. Разработать программную модель, которая будет строить кратчайший маршрут по карте местности
2. Создать отчёт
3. Защитить работу

Срок представления к защите курсовой работы: до «___» _____ 2024 г.

Задание на курсовую работу выдал _____ _____
(подпись руководителя) (фамилия, имя, отчество)

Задание на курсовую работу получил: до «___» _____ 2024 г.

_____ _____
(подпись студента) (фамилия, имя, отчество)

Содержание

1. Задача

Разработать программную модель, которая будет строить кратчайший маршрут по карте местности, согласно варианту.

1.1. Декомпозиция

1.1.1. Модель окружающего пространства

Одной из важных задач данной курсовой работы, является построение описания модели окружающего пространства. Такое описание включает в себя геометрическое представление мира (т. е. составление карты местности), локализацию относительно выделенных объектов (т. е. определение своего местоположения) и описание выделенных объектов с целью их идентификации, классификации для целенаправленного перемещения робота.

Построение карты местности будет выполнено с помощью такого математического инструмента как граф. Он будет направленным, у пользователя будет возможность устанавливать на карте местности начальную и конечную точку маршрута и алгоритм поиска будет искать между ними путь [1].

1.1.2. Модель робота

Существуют разные модели роботов, но для программной модели понадобиться только геометрическая модель. Она используется для визуализации движения робота, поэтому размер робота будет влиять на представление карты местности. Согласно выданному варианту, размер робота должен быть шестиугольный, поэтому ячейки карты местности будут разделены на них, для удобства [2].

1.1.3. Модель движения робота

Существуют разные типов устройств [3] передвижения роботов, но, так как для выполнения курсовой работы нужна программная модель, будем считать, что робот колёсный, а значит может перемещаться по ячейкам карты местности, классифицированные как дорога.

1.1.4. Модель технического зрения

Техническое зрение робота – это система компьютерного зрения, которая позволяет роботу получать информацию об окружающей среде и выполнять задачи, связанные с обработкой и анализом изображений. Техническое зрение робота включает в себя различные датчики и устройства, а также различные алгоритмы обработки изображений и программное обеспечение для управления этими компонентами [4]. Но в данной курсовой работе разрабатывается программная модель и, согласно варианту, для определения маршрута робот видит всю карту местности.

1.2. Анализ

1.2.1. Провести анализ алгоритмов хранения карты местности в памяти (графы, матрицы смежности)

Граф – это топологическая модель, которая состоит из множества вершин и множества соединяющих их рёбер. При этом значение имеет только сам факт, какая вершина с какой соединена (рис. 1).

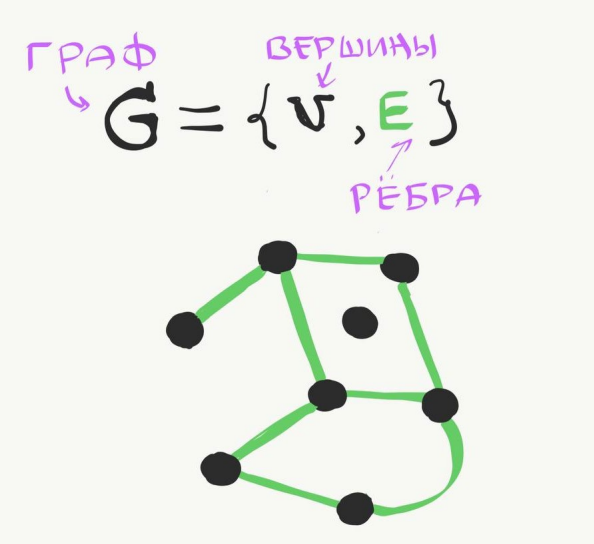

Рисунок 1. Граф, состоящий из 8 вершин и 8 рёбер.

Вершина – это точка, через которую проходят ребра. В общем случае граф состоит из множества вершин и множества ребер, соединяющих вершины.

Ребро – это неупорядоченная пара двух вершин, которые связаны друг с другом. Эти вершины называются концевыми точками или концами ребра. При этом важен сам факт наличия связи, каким именно образом осуществляется эта

связь и по какой дороге – не имеет значения. Также рёбрам может быть присвоен вес, что позволит говорить о взвешенном графе и решать задачи оптимизации.

Если вершина для ребра является концевой, то и вершина и ребро являются *инцидентными* (рис. 2).

Две *вершины* называются *смежными*, если они инцидентны одному ребру.

Два *ребра* называются *смежными*, если они инцидентны одной вершине.

Иными словами, две вершины смежные, если они соединены ребром, а два ребра смежные - если они соединены вершиной (рис. 2).

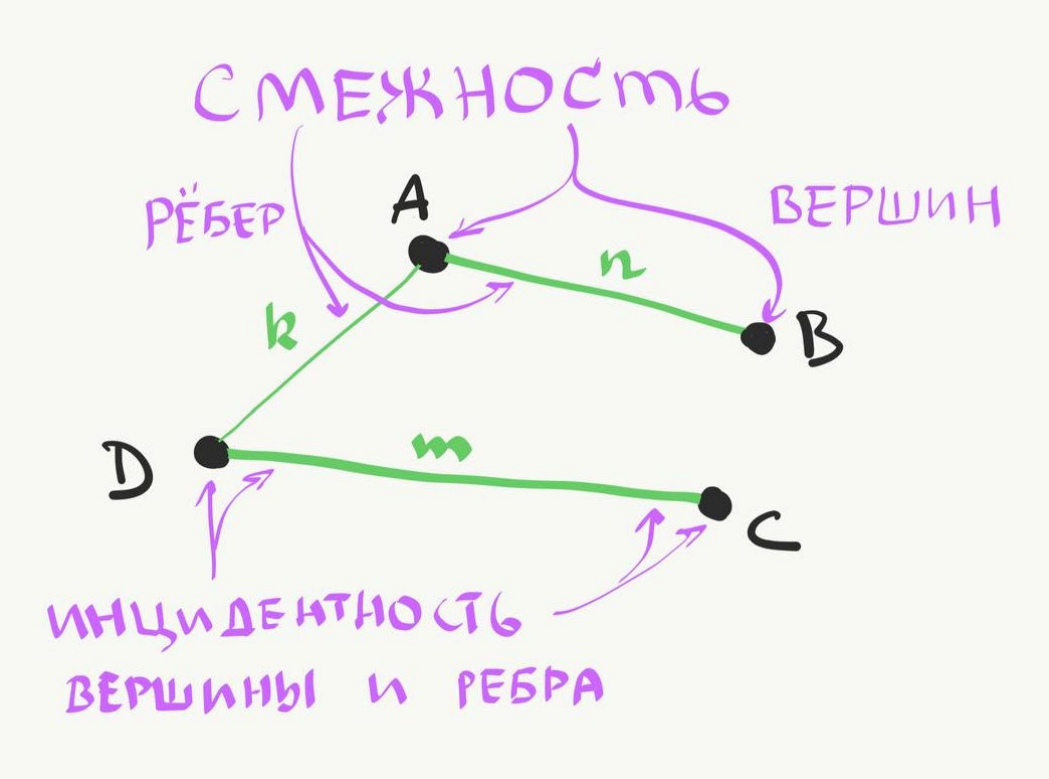

Рисунок 2. Отражение понятий про граф.

Петля – это ребро, инцидентное одной вершине. Ребро, которое замыкается на одной вершине.

Путь – это последовательность смежных рёбер. Обычно путь задаётся перечислением вершин, по которым он пролегает (рис. 3).

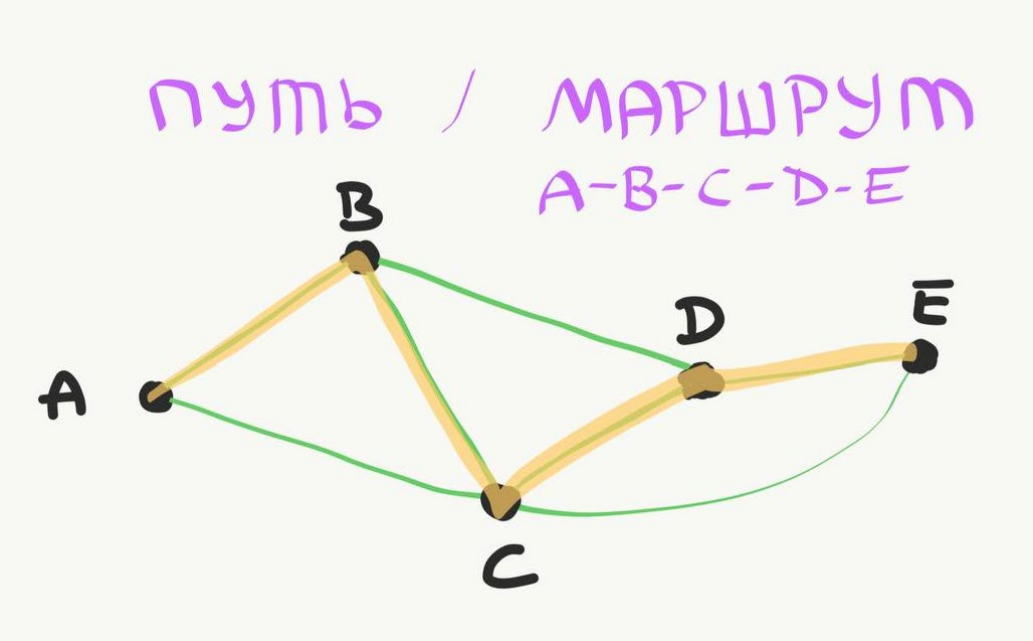

Рисунок 3. Путь на графе.

Очень многие задачи могут быть решены, используя богатую библиотеку алгоритмов теории графов. Для этого достаточно лишь принять объекты за вершины, а связь между ними - за рёбра, после чего весь арсенал алгоритмов теории графов к вашим услугам: нахождение маршрута от одного объекта к другому, поиск связанных компонент, вычисление кратчайших путей, поиск сети максимального потока и многое другое [5].

Матрица смежности – это самый популярный и расточительный способ представления графа в памяти. Его уместно использовать, если количество рёбер велико, порядка V^2.

Для хранения рёбер используется двумерная матрица размерности $[V, V]$, каждый $[a, b]$ элемент которой равен 1, если вершины a и b являются смежными и 0 в противном случае (рис. 4).

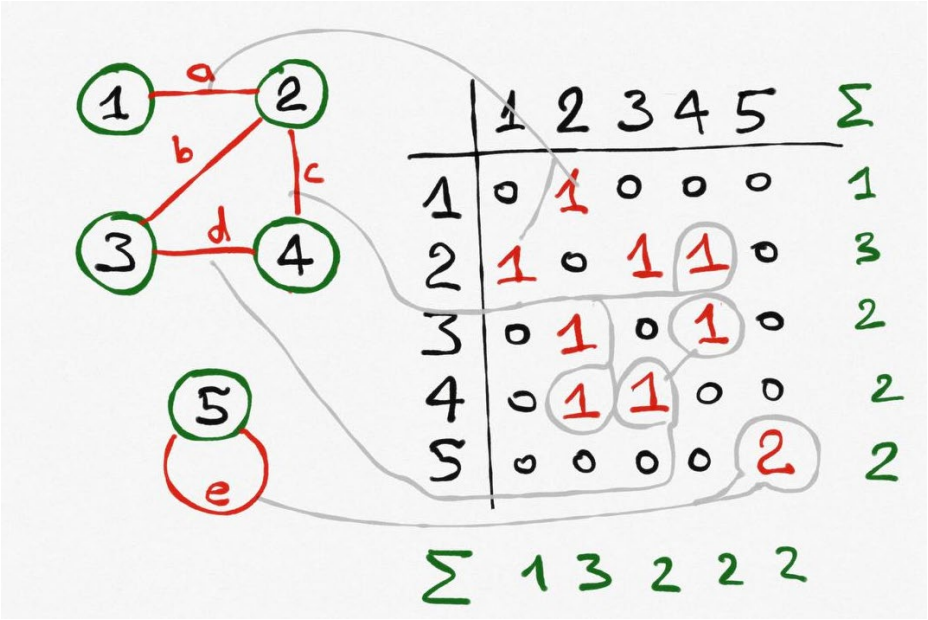

Рисунок 4. Матрица смежности.

В случае неориентированного графа матрица является симметричной относительно главной диагонали, а сумма каждой строчки и каждого столбца равна степени вершины. В связи с этим, при записи рёбер-петель в матрицу необходимо записывать число 2.

Матрица инцидентности – это самый расточительный способ хранения графа, его уместно использовать, если количество рёбер невелико.

Для хранения используется двумерная матрица размера $[V, E]$, в каждом столбце которой записано одно ребро таким образом: напротив вершин, инцидентных этому ребру, записаны 1, в остальных случаях 0 (рис. 5).

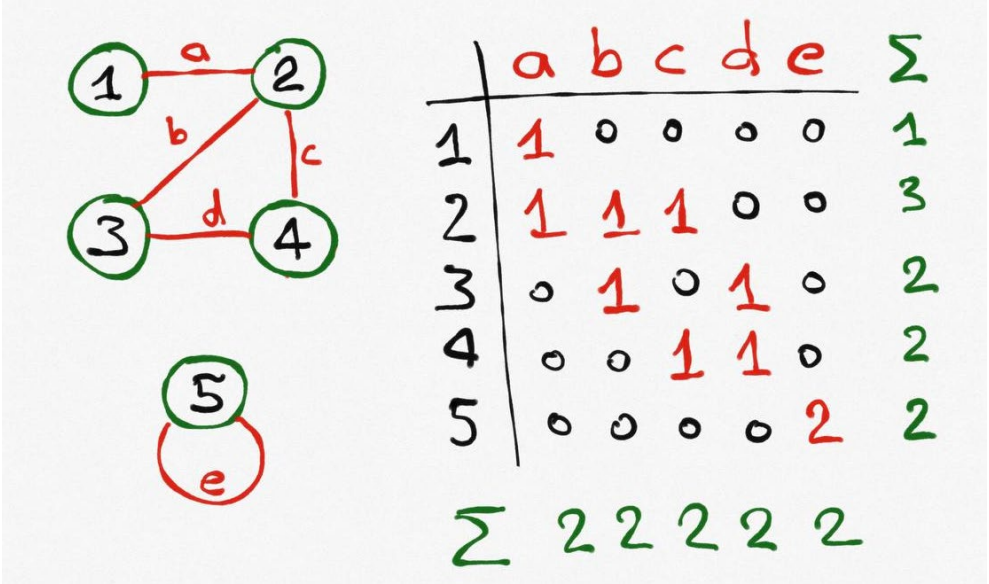

Рисунок 5. Матрица инцидентности.

Таким образом, сумма чисел в каждом столбце равна 2, а сумма чисел в строчке a равна степени вершины a [6].

В программной модели для курсовой работы будут использованы списки для хранения информации о смежных вершинах графа. Сам граф будет строиться с учётом проходимости.

1.2.2. Провести анализ методов поиска пути на карте местности

Поиск пути – это построение компьютерным приложением кратчайшего маршрута между двумя точками на графе. Эта область исследований в значительной степени основана на алгоритме Дейкстры для нахождения кратчайшего пути на взвешенном графе.

Поиск пути тесно связан с проблемой нахождения кратчайшего пути в рамках теории графов, в которой исследуется, как определить путь, который наилучшим образом соответствует некоторым критериям (кратчайший, дешевый, самый быстрый и т.д.) Между двумя точками в большой сети.

По своей сути метод поиска пути выполняет поиск по графу, начиная с одной вершины и исследуя соседние узлы до тех пор, пока не будет достигнут конечный узел, обычно, с целью найти самый дешевый маршрут.

Есть разные методы поиска пути на графе, самый распространённый метод – это поиск в ширину, но в тоже время существует много методов, которые ищут путь намного лучше, идя в направлении конечного узла и отклоняясь только для того, чтобы уклониться от препятствий.

Две основные проблемы поиска пути:

- найти путь между двумя узлами в графе;
- найти оптимальный кратчайший путь.

Базовые алгоритмы, такие как поиск в ширину и поиск в глубину, решают первую проблему, исчерпывая все возможности; начиная с данного узла, они перебирают все возможные пути, пока не достигнут узла назначения. Эти алгоритмы выполняются за:

$$O(|V| + |E|),$$

или линейное время, где V - количество вершин, а E - количество ребер между вершинами.

Более сложной проблемой является нахождение оптимального пути. Исчерпывающий подход в этом случае известен как алгоритм Беллмана–Форда, который дает временную сложность:

$$O(|V||E|)\},$$

или квадратичное время. Однако нет необходимости изучать все возможные пути, чтобы найти оптимальный. Такие алгоритмы, как A^* и алгоритм Дейкстры, стратегически устраняют пути либо с помощью эвристики, либо с помощью динамического программирования. Устраняя невозможные пути, эти алгоритмы могут достигать такой низкой временной сложности [7], как:

$$O\big(|E|log(|V|)\big)$$

Согласно варианту задания нужно реализовать в программной модели алгоритм двунаправленного поиска пути на графе, который и рассмотрим более подробно.

Двунаправленный поиск – это алгоритм поиска по графу, который находит кратчайший путь от начальной вершины к конечной вершине в графе. Он выполняет два одновременных поиска:

- прямой поиск от начальной вершины к конечной;
- обратный поиск от конечной вершины к начальной вершине [8].

Он заменяет один график поиска (который, вероятно, будет расти экспоненциально) двумя подграфами меньшего размера – один начинается с начальной вершины, а другой – с целевой вершины. Поиск завершается при пересечении двух графиков.

Использовать двунаправленный поиск следует, когда и начальное и конечное состояние известны и определены на графе, как в данном случае и есть, так как видимость у робота бесконечная и он видит всё карту местности.

Показатели производительности:

Полнота: Двунаправленный поиск считается завершённым, если в обоих поисках используется BFS.

Оптимальность: оптимально, если для поиска используется BFS и пути имеют одинаковую стоимость.

Сложность во времени и пространстве: Сложность во времени и пространстве равна:

$$O\ (b^{\frac{d}{2}}),$$

где b – это коэффициент ветвления дерева, d – это расстояние целевой вершины от источника [9]. И по сути, тратиться в двое меньше времени, чем при простом поиске в ширину.

Чаще всего при двунаправленном поиске используется алгоритм поиска в ширину и так как это конфигурация и будет применяема в данной работе, рассмотрим также алгоритм поиска в ширину.

Поиск в ширину (BFS) – это алгоритм обхода графа, который исследует все вершины графа на текущей глубине, прежде чем перейти к вершинам на следующем уровне глубины. Он начинается с указанной вершины и посещает всех своих соседей, прежде чем перейти к следующему уровню соседей. BFS обычно используется в алгоритмах поиска путей, связанных компонентов и задачах о кратчайших путях в графах.

Алгоритм поиска в ширину:

1. *Инициализация*. Помещаем начальный узел в очередь и помечаем его как посещенный.
2. *Исследование*. Пока очередь не пуста:
   - Извлекаем узел из очереди и посещаем его.
   - Для каждого не просмотренного соседнего узла, который исключается из очереди
     - Ставим соседа в очередь
     - Отмечаем соседа как посещённого
3. *Завершение*. Повторяем шаг 2, пока очередь не опустеет.

Этот алгоритм гарантирует, что все узлы графа будут посещаться в ширину, начиная с начального узла [10].

Описание работы алгоритма.

При работе алгоритма поиска в ширину (Breadh-first search, BFS) сетка представляется в виде графа, где из каждой вершины возможен переход в одну из 8 соседних вершин (кроме граничных). Для работы алгоритма требуется организовать очередь, в которую будут последовательно добавляться вершины. Помимо этого, чтобы иметь возможность восстановить пройденный путь, понадобится массив «предков», где будет указано, из какой вершины ведет путь в данную. Необходим также номер стартовой вершины (координаты стартовой ячейки). Сам алгоритм можно понимать как процесс условного «поджигания» графа: на нулевом шаге «поджигаем» только стартовую вершину. На каждом следующем шаге «огонь» с уже «горящей» вершины перекидывается на всех её соседей. Таким образом, за одну итерацию алгоритма происходит расширение «кольца огня» в ширину на единицу (отсюда и название алгоритма).

Более строго это можно представить следующим образом. Создадим очередь, в которую будут помещаться «горящие» вершины, а также заведем булевский массив, в котором для каждой вершины будем отмечать, «горит» она или нет (иными словами, была ли она посещена). Изначально в очередь помещается только стартовая вершина. Затем реализуется цикл: пока очередь не пуста, достать из её головы одну вершину, просмотреть все ребра, исходящие из этой вершины, и, если какие-то из просмотренных вершин ещё не «горят», «поджечь» их и поместить в конец очереди.

В итоге, когда очередь опустеет, алгоритм BFS обойдет все достижимые из стартовой вершины, причем до каждой дойдет кратчайшим путем. Также можно посчитать длины кратчайших путей (для чего просто надо завести массив длин путей) и компактно сохранить информацию, достаточную для восстановления всех этих кратчайших путей (завести массив «предков», в котором для каждой вершины необходимо хранить номер вершины, из которой мы в неё попали).

При достижении конечной вершины алгоритм прекращает свою работу и восстанавливается общий путь до конечной вершины.

Как говорилось ранее, двунаправленный поиск улучшает простой поиск BFS тем, что запускаются два одновременных поиска в ширину из стартового и конечного узлов, и процесс останавливается, когда узел из одного фронта поиска находит соседний узел из другого фронта. Это может улучшить простой поиск в ширину, который остается при этом не очень эффективным.

Одним из главных достоинств алгоритма BFS является то, что он гарантированно находит кратчайший путь из начальной вершины в конечную. Однако недостатком является необходимость исследования большого числа сторонних точек, что требует излишних затрат памяти для хранения информации, а также дополнительного времени [11].

1.2.3. Провести анализ методов учета дополнительных параметров (размер робота, радиус поворота, радиус видимости, топливо, и т.д.)

Как было сказано выше карта проходимости будет плоской, а её ячейки будут размером с робота для удобства. Робот видит сразу всё карту проходимости, а также начальную и конечную точку маршрута, установленные пользователем. Также будет присутствовать допущение, что робот поворачивается любой гранью без дополнительных штрафов. Согласно варианту, топливо роботу в данной работе не нужно.

1.2.4. Провести анализ методов обнаружения препятствий

Для обнаружения препятствий существуют разные методы. Как отмечалось ранее в данной программной модели было принято решения выбрать гексагональную (шестиугольную) карту местности в проекциях на две оси x и y.

Для примера на рисунке 6 показана карта проходимости, похожая на ту, что нужно получить.

Так как работ видит всю карту, ему на вход будет поступать карта местности с «подписанными» объектами, на основании которой он будет прокладывать свой маршрут, с помощью двунаправленного поиска пути.

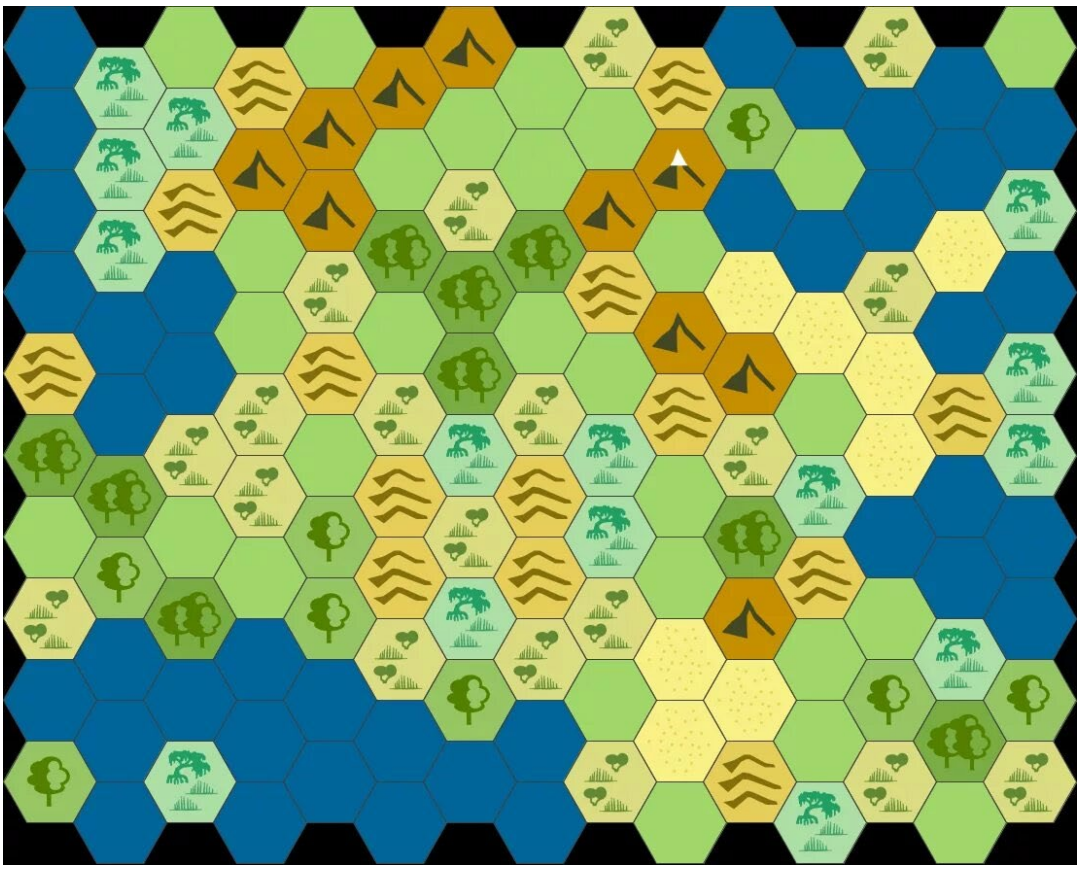

Рисунок 6. Пример гексагональной карты проходимости.

1.3. Синтез

1.3.1. Выбор параметров

Разрабатываемая программная модель должна обеспечивать следующие функциональные возможности:

- У пользователя должна быть возможность изменить карту местности:
  - поменять диаметр ячейки;
  - количество строк;
  - количество колонок;
  - создать случайную карту местности;
  - очистить карту;
  - изменить условия проходимости для робота.
- Пользователь должен иметь возможность устанавливать начальную точку пути и конечную

– Для пользователя должна быть наглядна и понятна информация о найденном пути
– Пользователь должен иметь возможность узнать о возможностях приложения
– Интерфейс приложения должен быть понятен и приятен пользователю.

1.3.2. Разработка модели

Согласно функциональным требованиям, рассмотренным в предыдущем пункте, была создана диаграмма, отражающая внутренние устройство разрабатываемой программной модели (рис. 7).

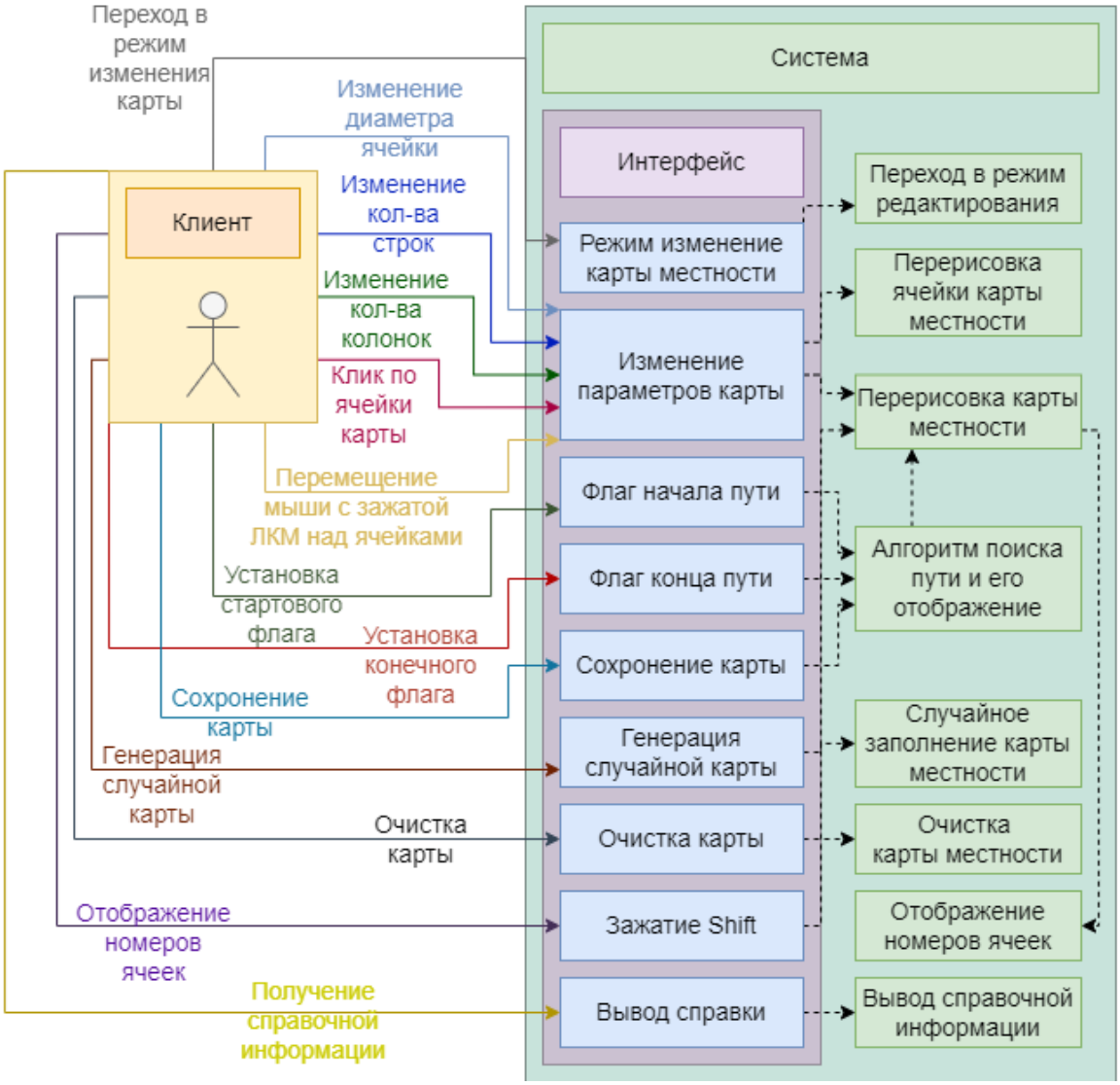

Рисунок 7. Диаграмма системы планирования движения наземного робота.

На основе составленной диаграммы к программной модели были выделены несколько возможных реализаций данного проекта:

- используя Unity и работать с языком программирования C#;
- использую язык программирования python, с использованием библиотеки pygame для работы с интерфейсом;
- используя windows forms и языка программирования C#.

После многочисленных экспериментов был сделан вывод, что наиболее целесообразно реализовать проект третьим способом.

2. Решение

2.1. Разработать средство генерации окружающего пространства робота

2.1.1. Входные параметры

2.1.1.1. Метод создания карты

Согласно варианту метод создания карты ручной.

2.1.1.2. Размер карты (m, n)

Размер карты регулирует сам пользователь, но для удобства были выбраны следующие значения по умолчанию:

Диаметр = 70;

Количество строк = 8;

Количество колонок = 17;

2.1.1.3. Учёт проходимости

Согласно варианту задания, нужно учитывать проходимость робота. В программной модели у робота есть две классификации объектов это:

- Дорога (рис. 8);
- Гора (рис. 9);

По дороге робот может перемещаться, а по горам нет.

Рисунок 8. Ячейка карты местности - дорога.

Рисунок 9. Ячейка карты местности - горы.

2.1.2. Результат – карта проходимости с дополнительными данными

В результате получилась карта местности с разным уровнем проходимости (рис. 10). Когда пользователь после главного меню переходит в основную форму, сразу в случайном месте выставляются флаги начала и конца пути и между ними строиться кратчайший маршрут, при помощи алгоритма двунаправленного поиска пути.

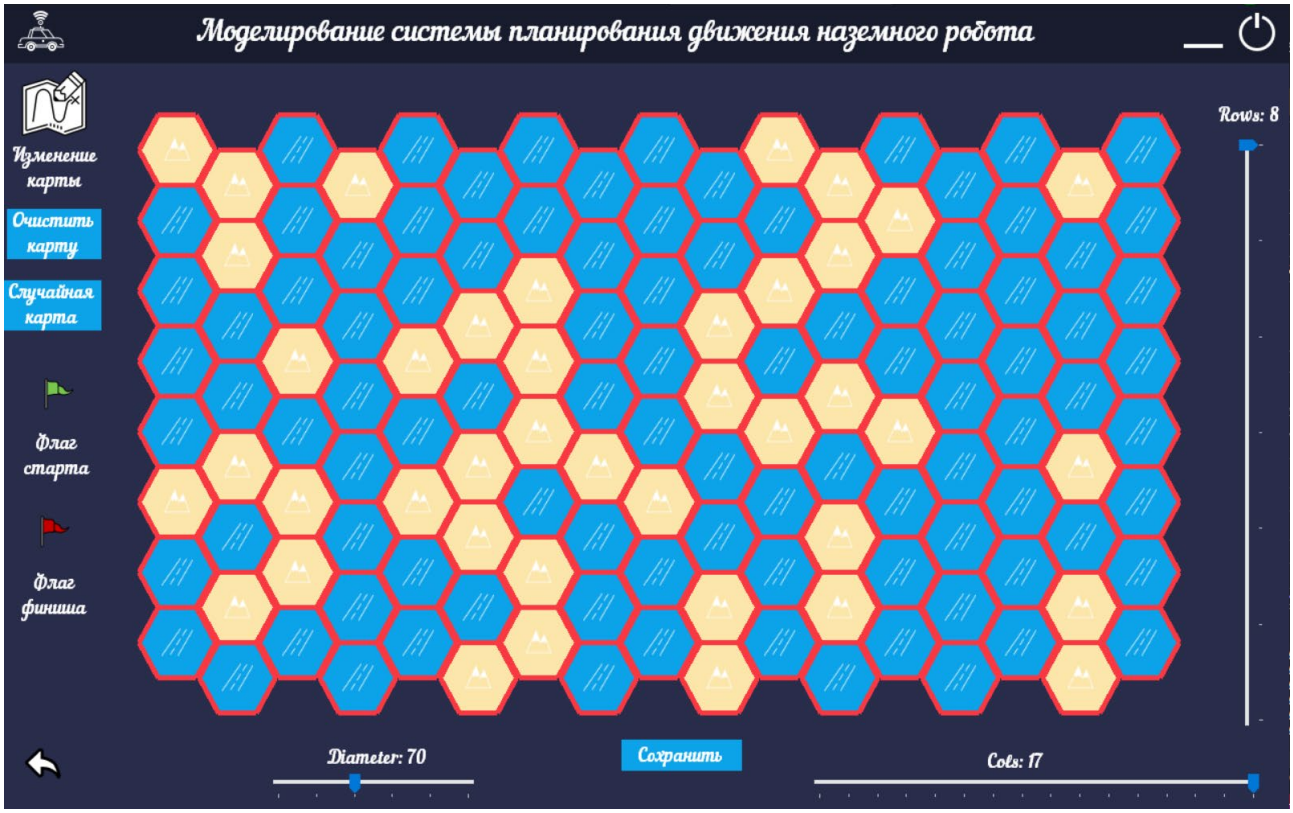

Рисунок 10. Карта местности с разным уровнем проходимости.

При желании пользователь может нажать на кнопку, в левом верхнем углу, и перейти в режим изменения карты местности, где будут сразу отображены условные обозначения. Пользователь может изменить значения диаметра ячеек, количества строк и колонок с помощью удобных trackbar-ов, расположенных внизу и справа экрана. Ещё пользователь может (и это даже нужно для выполнения поиска пути) поставить в любую ячейку карты местности, обозначенную как дорога, флаги начала и окончания пути. Ячейки карты местности пользователь может изменять с помощью клика мышью на интересующую клетку, либо с помощью удержания левой кнопки мыши и перемещения по клеткам, тогда будут изменяться все ячейки, попавшие на пути у курсора. Также у пользователя есть возможность очистить всю карту, и тогда все ячейки превратятся в дорогу, или создать случайный вариант карты местности.

После того как пользователь определиться с картой местности, нужно нажать на кнопу «сохранить», тогда произойдёт сохранение и переход в автоматический режим поиска пути.

Если пользователь захочет увидеть номера ячеек карты местности, то в режиме поиска пути нужно нажить и удерживать клавишу Shift. Все ячейки, кроме флагов, будут подписаны соответствующими цифрами (рис. 11).

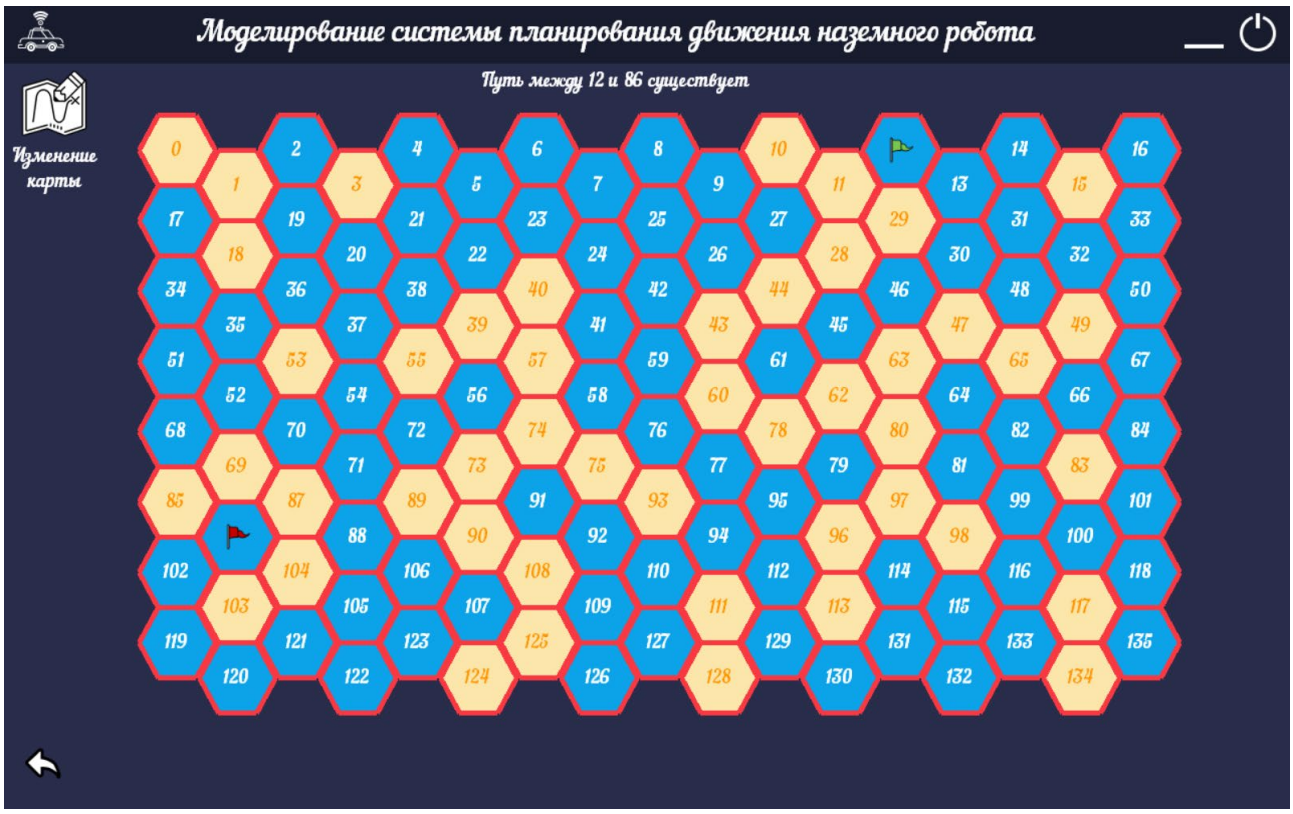

Рисунок 11. Отображение нумерации ячеек карты местности, при нажатии клавиши Shift.

2.2. Разработать модель движения робота

2.2.1. Входные параметры

2.2.1.1. Радиус поворота

Согласно варианту, нужно учитывать радиус поворота, но как было отмечено ранее в программной модели будет считаться, что вокруг своей оси робот поворачивается без дополнительных штрафов.

2.2.1.2. Радиус видимости

Радиус видимости в данном варианте бесконечный, то есть робот видит сразу всю карту, а также стартовый и финишный флаги, что позволяет ему находить путь с помощью алгоритма двунаправленного поиска.

2.2.1.3. Потребность в топливе

Согласно заданию, потребность в топливе в данной программной модели у робота – отсутствует.

2.2.1.4. Размер робота

Размер робота, как об этом было сказано ранее, соответствует ячейки карты местности, то есть робот имеет шестиугольную форму.

2.2.1.5. Алгоритм движения с учетом параметров

В пункте 1.2.2 был рассмотрен алгоритм движения робота по карте местности – это двунаправленный поиск. Он состоит из двух алгоритмов поиска в ширину, которые берут своё начало от стартового и финишного флагов.

После настройки пользователем карты местности и нажатии на кнопку «сохранить», происходит выполнение алгоритма поиска пути. Сам алгоритм был описан выше. В результате пользователь должен увидеть либо путь, либо его отсутствие. Если путь отсутствует пользователь увидит сообщение, говорящее о невозможности построения пути (рис. 12). Также для информирования о текущем состоянии пути, есть текстовое поле сверху карты местности.

Рисунок 12. Диалоговое окно, сообщающее о невозможности построения пути.

При успешном построении пути пользователю на ячейках будут нарисованы точки, соответствующие найденному пути (рис. 14). Точка встречи

двух алгоритмов поиска в ширину, отмечается особенным жёлтым цветом (рис. 13).

Рисунок 13. Ячейка карты местности, обозначающее встречу двух циклов.

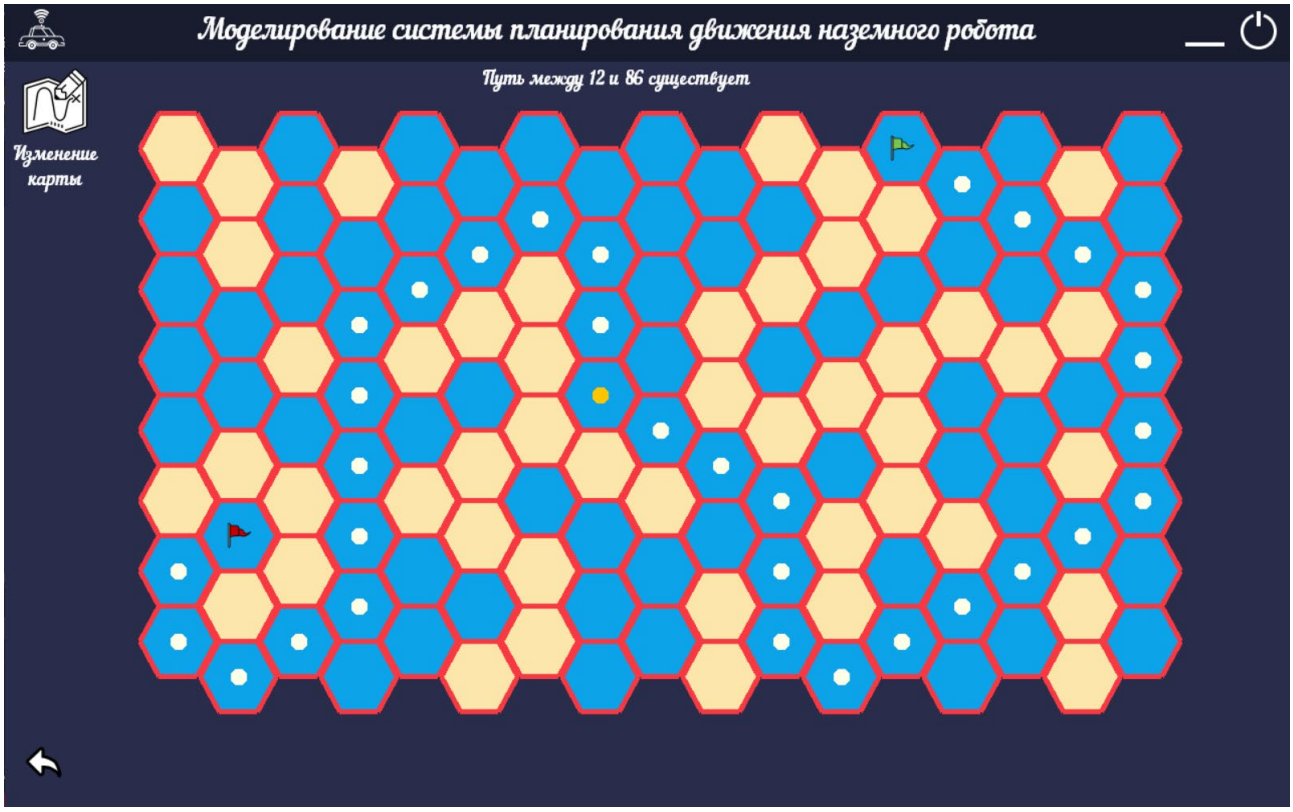

Рисунок 14. Отображение найденного пути.

Как говорилось ранее пользователь может изменять диаметр, количество строк и колонок. Для примера на рисунке 15 приведена карта местности с диаметром = 100, количеством колонок = 5 и количеством строк = 4.

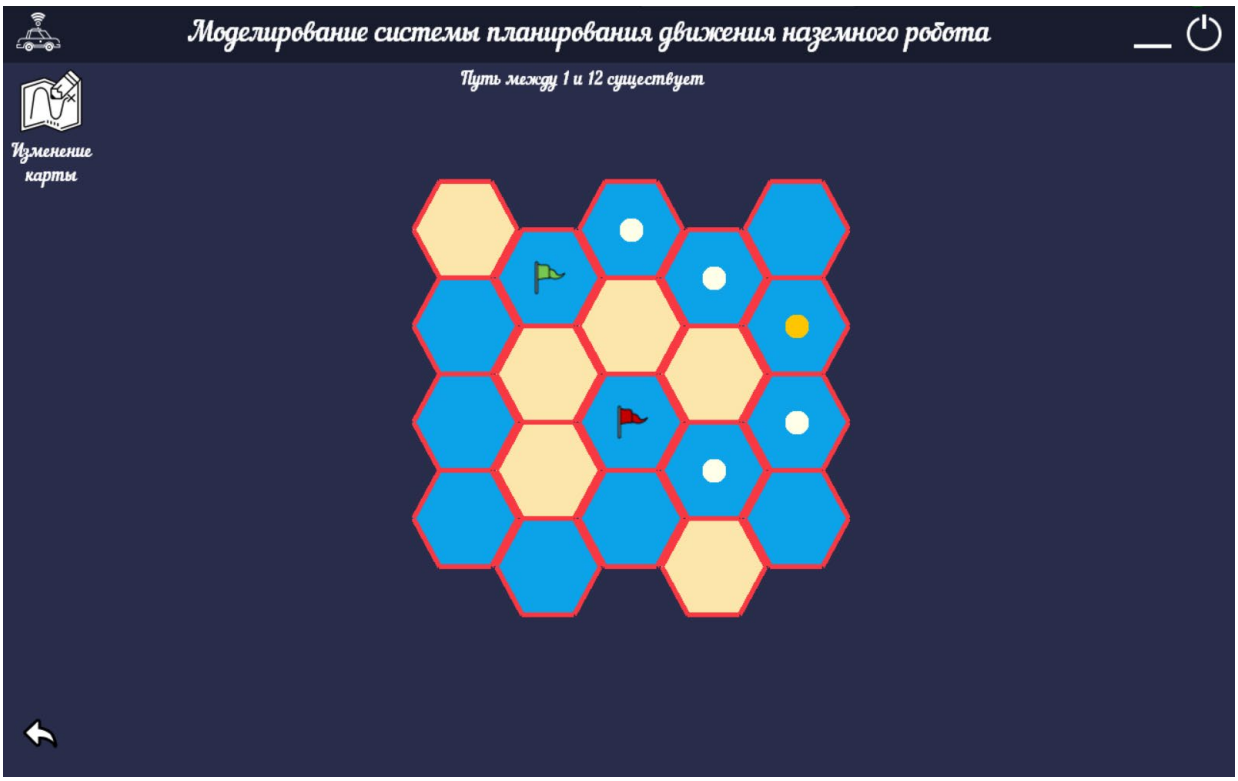

Рисунок 15. Карта местности, с найденным путём.

Также для удобства пользователя была создана начальная форма, в которой пользователь может получить справочную информацию (рис. 17), перейти в основное приложение, с помощью кнопки «start» или выйти из приложения – кнопка «Quit» (рис. 16).

Рисунок 16. Начальное приветственное окно.

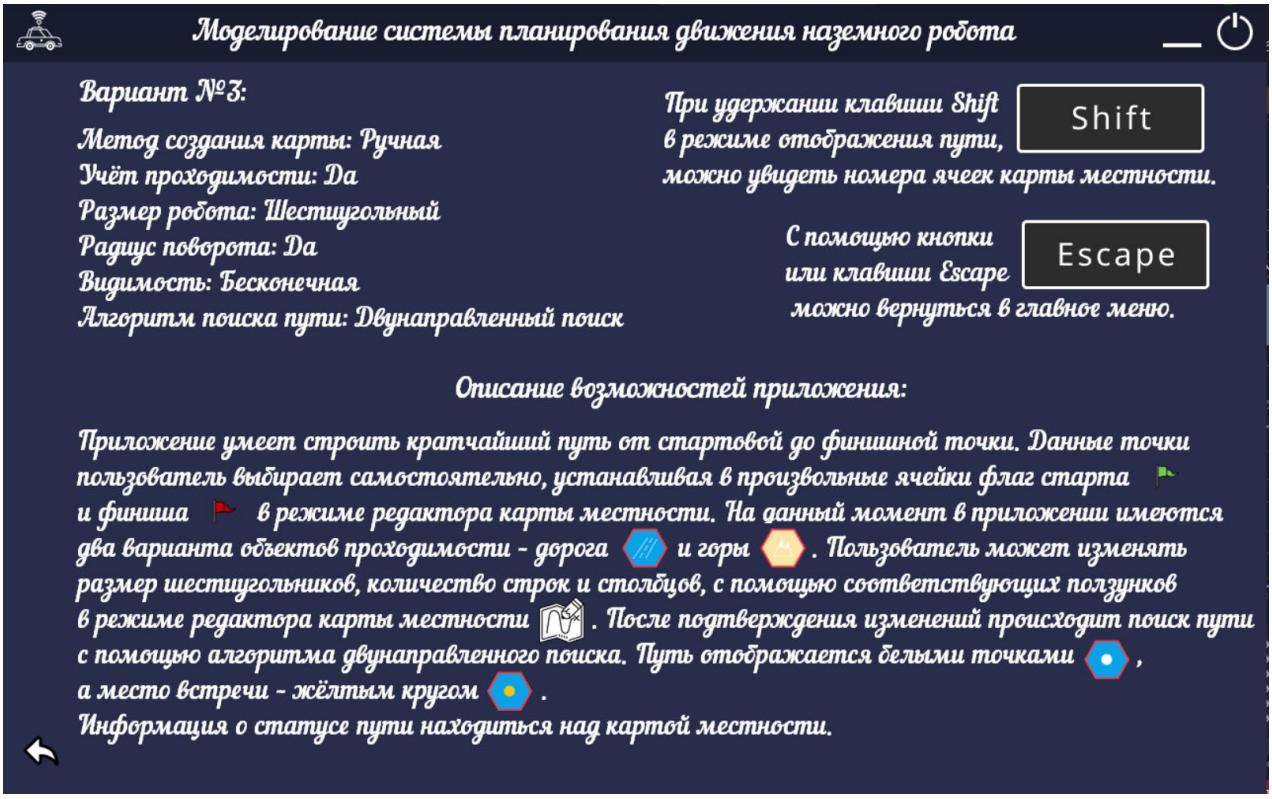

Рисунок 17. Окно со справочной информацией.

3. Заключение

В результате выполнения курсовой работы, была разработана программная модель, которая справляется с задачей построения кратчайшего маршрута по карте местности, используя алгоритм двунаправленного поиска пути.

4. Ссылка на использованную литературу

1. Статья «Представление окружающего пространства на основе векторных динамических структур в мобильных робототехнических системах» [Электронный ресурс] URL: https://cyberleninka.ru/article/n/predstavlenie-okruzhayuschego-prostranstva-na-osnove-vektornyh-dinamicheskih-struktur-v-mobilnyh-robototehnicheskih-sistemah Дата обращения (12.01.2023);
2. Статья «Моделирование и исследование роботов» [Электронный ресурс] URL: https://studfile.net/preview/3642488/ Дата обращения (15.01.2023);
3. Статья «Устройства передвижения мехатронных систем/роботов» [Электронный ресурс] URL:

https://ru.wikiversity.org/wiki/Устройства_передвижения_мехатронных_систем/роботов Дата обращения (20.01.2023);

4. Статья «Системы технического зрения» [Электронный ресурс] URL: https://studfile.net/preview/774683/ Дата обращения (30.01.2023);
5. Статья «Теория графов. Термины и определения в картинках» [Электронный ресурс] URL: https://habr.com/ru/companies/otus/articles/568026/ Дата обращения (14.06.2023);
6. Статья «Способы хранения графа в памяти компьютера» [Электронный ресурс] URL: https://habr.com/ru/companies/otus/articles/675730/ Дата обращения (15.06.2023);
7. Статья Википедии «Поиск пути» [Электронный ресурс] URL: https://en.wikipedia.org/wiki/Pathfinding Дата обращения (20.06.2023);
8. Статья Википедии «Двунаправленный поиск» [Электронный ресурс] URL: https://en.wikipedia.org/wiki/Bidirectional_search Дата обращения (21.06.2023);
9. Статья «Двунаправленный поиск» [Электронный ресурс] URL: https://www.geeksforgeeks.org/bidirectional-search/ Дата обращения (22.06.2023);
10. Статья «Поиск в ширину или BFS для графика» [Электронный ресурс] URL: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/ Дата обращения (23.06.2023);
11. Басараб М.А., Домрачева А.Б., Купляков В.М. Алгоритмы решения задачи быстрого поиска пути на географических картах. Инженерный журнал: наука и инновации, 2013, вып. № 11. URL: http://engjournal.ru/catalog/it/hidden/1054.html Дата обращения (30.06.2023);
12. Статья «Алгоритмы поиска пути» [Электронный ресурс] URL: https://pmg.org.ru/ai/stout.htm#listing1 Дата обращения (01.07.2023);
13. Документация по платформе .NET Extensions 8 [Электронный ресурс] URL: https://learn.microsoft.com/en-us/dotnet/api/system.drawing?view=dotnet-plat-ext-8.0 Дата обращения (07.02.2024);

14. Сайт-руководство по способу создания шестиугольных сеток [Электронный ресурс] URL: https://www.redblobgames.com/grids/hexagons/#coordinates Дата обращения (17.11.2023);
15. Различные уроки по C# «Очередь Queue» [Электронный ресурс] URL: https://metanit.com/sharp/tutorial/4.7.php; https://metanit.com/sharp/windowsforms/4.16.php Дата обращения (08.02.2024);
16. Урок «Как рисовать на C# с помощью класса Graphics – Программирование CSharp» [Электронный ресурс] URL: https://stardevstudio.com/softwaredev/how-to-draw-in-c-using-the-graphics-class-csharp-programming/ Дата обращения (10.02.2024);
17. Форум «Кастомный контрол "Шестиугольный Panel". Возможно ли?» [Электронный ресурс] URL: Дата обращения (15.02.2024);
18. Форум «Визуализация графа по матрице смежности» [Электронный ресурс] URL: https://www.cyberforum.ru/windows-forms/thread2149073.html Дата обращения (10.03.2024);
19. Подбор палитры цветов [Электронный ресурс] URL: https://coolors.co/fcecc9-fcb0b3-f93943-7eb2dd-445e93 Дата обращения (20.02.2024);
20. Статья «Как добавить графику в приложение C# Windows Form» [Электронный ресурс] URL: https://www.makeuseof.com/c-sharp-windows-form-graphics/ Дата обращения (12.03.2024);
21. Урок 7. Рисование графики в C# .NET [Электронный ресурс] URL: https://www.ictdemy.com/csharp/winforms/c-sharp-tutorial-drawing-graphics-windows-forms Дата обращения (21.02.2024);
22. Видео-урок «Графика на C# - Нарисуйте карту шестиугольника с помощью Graphics и drawPolygon» [Электронный ресурс] URL: https://ya.ru/video/preview/16333083894863595540 Дата обращения (15.02.2024).