МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программного обеспечения

РЕКОМЕНДОВАНО К ЗАЩИТЕ В ГЭК
И ПРОВЕРЕНО НА ОБЪЕМ
ЗАИМСТВОВАНИЯ
Заведующий кафедрой
д.п.н., профессор
_____И.Г. Захарова
_____29 июня 2018 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

РАЗРАБОТКА СИСТЕМЫ СОГЛАСОВАНИЯ ДВИЖЕНИЯ ГРУППЫ РОБОТОВ В ЗАМКНУТОМ ПРОСТРАНСТВЕ

02.04.03. Математическое обеспечение и администрирование информационных систем
Магистерская программа «Высокопроизводительные вычислительные системы»

Выполнил работу
Студент 2 курса
очной формы обучения
Белик
Анатолий
Александрович

Научный руководитель
к.т.н., старший преподаватель
Пушкарев
Александр
Николаевич

Рецензент
Кафедра программной
и системной инженерии,
старший преподаватель
Кропотин
Александр
Александрович

Тюмень 2018

Содержание

Введение

Для решения многих задач могут использоваться роботы. Однако при использовании некоторого числа роботов возникает задача в согласовании их действий. Действия роботов должны быть согласованы по времени и в пространстве. Поскольку могут возникать ситуации, в которых найти решение становится невозможным. Решением данной задачи является либо поиск заранее найденной последовательности действий роботами группы, либо в оперативном формировании рациональной последовательности действий и одновременной или последующей реализации этой последовательности [1].

Целью данной работы является разработка системы согласования движения группы роботов в замкнутом пространстве.

- При разработке системы основными задачами являются:
- исследование существующих систем управления группой роботов;
- изучение стратегий группового управления роботами;
- изучение алгоритмов поиска путей в замкнутом пространстве;
- программная реализация алгоритма согласования движения группы роботов в замкнутом пространстве;
- визуализация работы реализованного алгоритма.

Глава 1. Согласование движения группы роботов

1.1 Группа роботов

Группа роботов бывает гомогенной или гетерогенной.

Гомогенная группа роботов представляет собой набор роботов имеющих одинаковую конструкцию и одинаковые функциональные задачи и возможности.

Гетерогенная группа представляет собой набор роботов имеющих разные конструкции и разные функциональные задачи и возможности.

Для того что бы поставленная для группы роботов цель была достигнута, в случае определяемой и детерминированной среды, каждому роботу необходимо выполнять четко заданную для него последовательность действий. В случае не определяемой и недетерминированной изменяющейся среды последовательность действий необходимо находить с помощью системы управления, которая используются для управления группой роботов в ходе достижения ими поставленной цели. В этом случае важнейшим аспектом становиться правильная и качественная координация работы группы роботов как единой связанной системы.

1.2 Обзор существующих систем управления группой роботов

Наиболее известной системой управления группой роботов является система разработанная и созданная компанией Amazon Robotics (ранее KIVA Robotics).

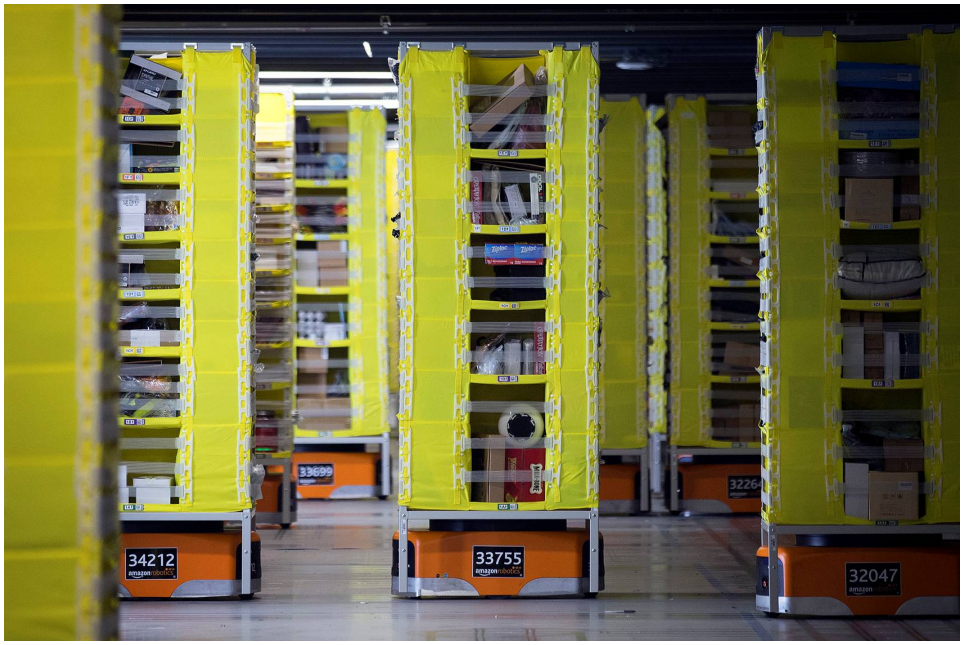

Рис 1.1 Amazon Robotics

Данная система распределяет заказы между ближайшими свободными роботами и рассчитывает маршруты, по которым роботы должны передвигаться. Для определения робота в пространстве используются QR-коды, расклеенные по полу склада. Также система отдает команды для остановки роботов в случае возникновения препятствия.

Столкновения роботов полностью исключаются благодаря специальным датчикам. Роботы в такой системе перемещают товары в зависимости от их местонахождения до конечной точки, где происходит дальнейшая обработка заказа. Вся информация по местоположениям определенных товаров хранится в базе данных. За работой системы наблюдает оператор, который может управлять роботами в случае возникновения нестандартной ситуации.

Подобную систему также использует Alibaba Group.

Другая система используется компанией Ocado — онлайн-супермаркете в Великобритании.

Рис 1.2 Склад компании Ocado

В этой системе роботы перемещаются по сетке, каждая зона является отдельной корзиной с различными товарами. Местонахождение каждого товара хранится в базе данных. Столкновения также полностью исключаются благодаря работе датчиков и работы системы. Система определяет местонахождение необходимого товара и направляет ближайшего робота. Затем робот направляется к зоне разгрузки, где происходит дальнейшая обработка заказа.

Также существует множество систем управления дронами.

1.2. Стратегии группового управления

Стратегии группового управления группой объектов можно разбить на два класса (рис 1.3):

- Централизованное управление
- Децентрализованное управление

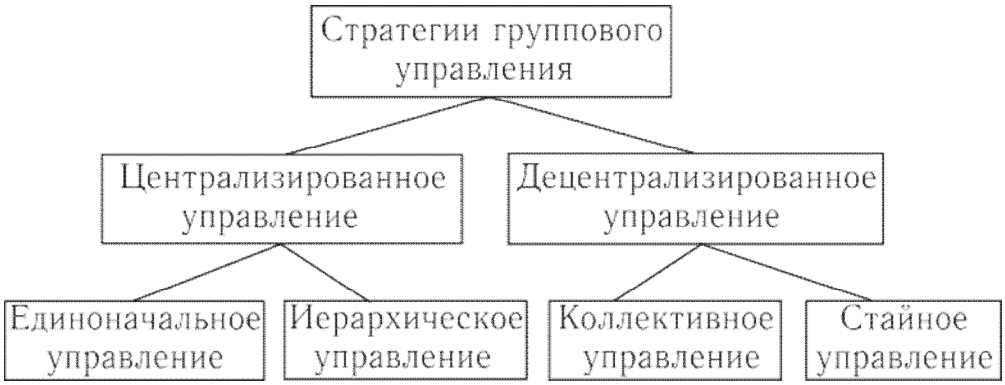

Рис. 1.3. Стратегии группового управления

Основной смысл централизованного управления заключается в существовании некого устройства, которое осуществляет непосредственное управление группой, в частности планирование возможных действий и контроль за выполнением поставленных задач.

В свою очередь, централизованное управление можно разбить на следующие подклассы:

- Единоначальное управление
- Иерархическое управление

Основной смысл централизованного управления заключается в существовании некого устройства, которое осуществляет непосредственное управление группой, в частности планирование возможных действий и контроль за выполнением поставленных задач. (см. рис. 1.4).

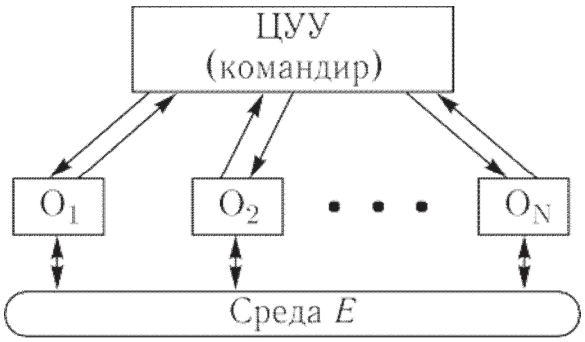

Рис. 1.4. Единоначальное управление

К преимуществам единоначального управления можно отнести простоту организации и алгоритмизации всей системы в целом. К недостаткам можно отнести сложность задачи оптимизации членов группы в случае большого количества объектов в группе.

Поэтому при использовании этой стратегии задача проверяется до начала действий, а затем запланированные действия выполняются в соответствии с программой без учета непредвиденных изменений в окружающей среде.

Недостаток оптимизации задачи в случае большого количества объектов, возможно частично устранить при использовании иерархического управления (рис. 1.5), где устройство управления управляет небольшим количеством подчиненных, которые, в свою очередь, являются устройствами управления второго уровня, в подчинении каждого из которых состоит своя подгруппа объектов группы и т.д.

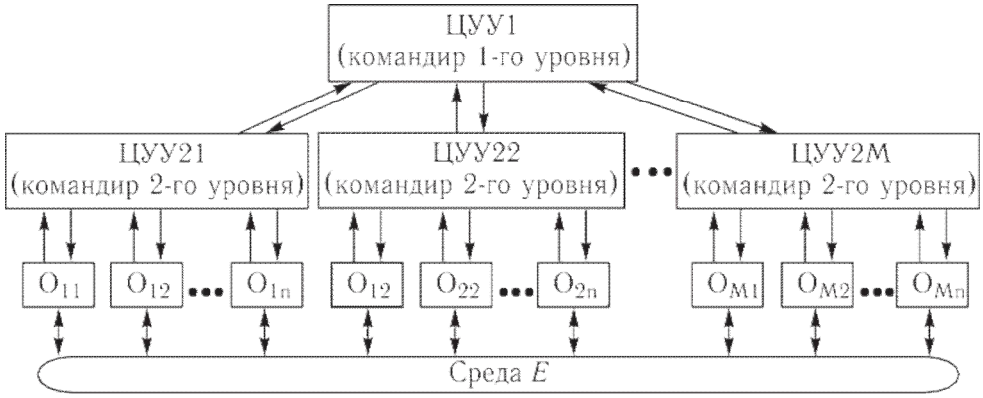

Рис. 1.5. Иерархическое управление

Недостаток оптимизации задачи в случае большого количества объектов возможно частично устранить при использовании иерархического управления (рис. 1.5), где устройство управления управляет небольшим количеством подчиненных, которые, в свою очередь, являются устройствами управления второго уровня, в подчинении каждого из которых состоит своя подгруппа объектов группы и т.д.

Непосредственно следует подчеркнуть, что все системы группового управления, использующие централизованную стратегию, имеют недостаток — это низкая устойчивость системы. Действительно, выход из строя управляющего устройства приводит к выходу из строя всей системы в целом или её части. Чтобы уменьшить риск такой ситуации, как правило, используются резервные методы, когда существует несколько взаимозаменяемых центральных управляющих устройств, и если один из них не работает, его функция передается другой [1].

Недостаток управляющего устройства, а именно выход его из строя, отсутствует в системах группового управления, которые используют децентрализованную стратегию управления. Суть этой стратегии в том, что система не имеет центрального контрольного устройства или командира, и каждый член группы самостоятельно решает свои действия, пытаясь внести максимально возможный вклад в достижение общей цели.

Преимущества децентрализованной стратегии группового управления следующие. Во-первых, задача, решаемая каждым членом группы, будет несложной, поскольку он решает задачу оптимизации только своих действий, не пытаясь оптимизировать действия всей группы в целом. Во-вторых - высокая устойчивость системы. Все члены группы равны, поскольку у них нет управляющего устройства, и поэтому сбой или уничтожение любого из них не приводит к отказу всей группы в целом.

К недостаткам стратегии децентрализованного группового управления можно отнести значительную сложность при её алгоритмизации, это все из-за того, что у каждого члена группы должно быть четко представление общей групповой задачи и умение отталкиваясь от нее соответственно так определять свои действия, что бы они приводили к получению самого эффективного решения для нее. Данная ситуация подразумевает наличие высокого «интеллектуального уровня» каждого члена группы, что так же является сложной для реализации задачей.

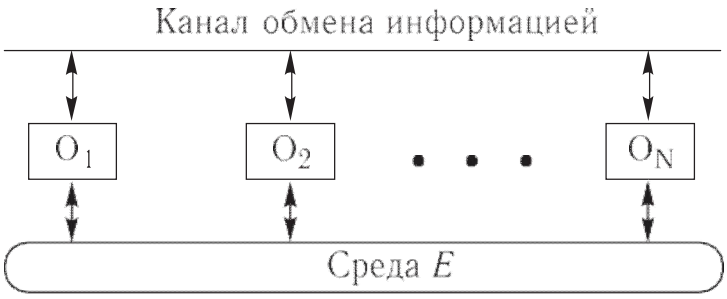

Рис. 1.6. Коллективное управление

Преимуществом стратегии коллективного управления является существующая возможность по совместной оптимизации общегрупповых действий за счет обмена информацией о действиях, которые определены, выбраны и происходят между объектами группы. Необходимо взять во внимание то, что при такой организации управления группами необходим надежный канал связи и возможный отказ данного канала может привести к потере группового взаимодействия.

Группы с наиболее высокой устойчивостью это использующие стратегию стайного управления (рис. 1.7).

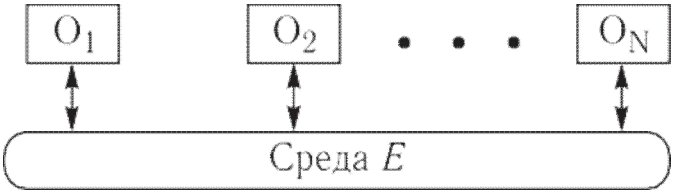

Рис. 1.7. Децентрализованное стайное управление

Суть стайной стратегии управления в том, что у каждого объекта группы отсутствует информационное соединение с остальными объектами, и более того, он может даже не иметь информации о структуре и характеристиках остальных объектах, которые находятся в его группе.

Но с учетом этого у каждого объекта остается возможность на основании информации об изменениях в состоянии окружающей среды, которые происходят из-за действий других объектов, входящих в его группу, осуществлять и корректировать свои текущие действия таким образом, что бы они позволяли достигать общие цели поставленные группе в целом.

1.3 Алгоритмы поиска пути

Суть алгоритма поиска пути заключается в поиске пути, который берет свое начало в начальной точке, и на основании исследования соседних узлов, длится до тех пор, пока не будет достигнут конечный узел. В алгоритмах поиска пути, самой главное задачей в большинстве случаев является найти оптимальный и одновременно кратчайший путь. Использование некоторых методов решения задачи поиска на графе, такого, к примеру, как поиск в ширину, позволяют найти путь, если нет значительного ограничения по времени поиска. Другие методы, которые «исследуют» граф, могут достигать необходимой точки назначения в несколько раз быстрее. Важно перед началом построения пути заранее уделить время для изучения всех характеристик и препятствий в графе, рассчитать оптимальный маршрут и только после этого начать движение. Также можно сразу начать движение по приблизительному или предполагаемому направлению цели, а потом, уже в ходе движения по пути, делать «живые» корректировки движения для того, что бы обходить препятствия, возникающие на пути.

К самым распространенным алгоритмам поиска пути относят:

- Алгоритм поиска A*
- Алгоритм Дейкстры
- Алгоритм волновой трассировки

Алгоритм волновой трассировки

Алгоритм поиска пути, алгоритм поиска кратчайшего пути на графе. Принадлежит к алгоритмам, основанным на методах поиска в ширину.

Алгоритм предназначен для поиска кратчайшего пути от начальной до конечной ячейки, если же такой путь отсутствует и его не возможно найти, алгоритм выдает сообщение об невозможности прохода.

Работа данного алгоритма состоит из трех этапов: инициализация, распространение волны и восстановление пути.

На первом этапе, во время инициализации, каждой ячейке их множества приписываются значения проходимости/непроходимости, определяются и фиксируются значения стартовой и финишной ячейки.

Далее, во время распространения волны, от стартовой ячейки осуществляется проверка проходимости соседней ячейки и в случае положительного результата происходит шаг в эту соседнюю ячейку, при этом учитывается момент того, что ячейка, в которую осуществляется шаг, не принадлежит ранее отмеченной в пути ячейке.

При выполнении условий проходимости и непринадлежности её к ранее отмеченным в пути ячейкам, в ячейку записывается число, равное количеству шагов от стартовой ячейки. Каждая ячейка, отмеченная числом шагов от начальной ячейки, становится исходной ячейкой, и из нее создаются следующие шаги в соседние ячейки. Можно сказать, что по осуществлении такого поиска, найден будет путь от стартовой до финишной ячейки или шаг не возможно будет осуществить из-за отсутствия такой возможности.

Формирование кратчайшего пути происходит в обратном направлении: при проходе от финишной ячейки к стартовой на каждом шаге выбирается та ячейка, которая имеет значение расстояния от стартовой на единицу меньше текущей ячейки. Таким образом находится кратчайший путь между парой заданных ячеек.

Алгоритм Дейкстры.

Алгоритм находит кратчайшее расстояние от одной из вершин графа до всех остальных и работает только для графов без ребер отрицательного веса.

Каждой вершине присваивается число - это вес пути от начальной вершины к конечной. Также можно выбрать каждую вершину. Если выбрана вершина, то путь от нее к начальной вершине является кратчайшим , если нет, то временным. Обходя граф, алгоритм считает для каждой вершины маршрут, и, если он оказывается кратчайшим, выделяет вершину. Весом данной вершины становится вес пути. Для всех соседей данной вершины алгоритм также рассчитывает вес, при этом ни при каких условиях не выделяя их. Алгоритм заканчивает свою работу, дойдя до конечной вершины, и весом кратчайшего пути становится вес конечной вершины.

Алгоритм Дейкстры:

1. Всем вершинам, за исключением первой, присваивается вес равный бесконечности, а первой вершине – 0.

2. Все вершины не отмечены.

3. Первая вершина объявляется текущей.

4. Вес всех не отмеченных вершин пересчитывается по формуле: вес не отмеченных вершины - это минимальное количество старого веса данной вершины, сумма веса текущей вершины и вес края, соединяющего текущую вершина с не отмеченных вершиной.

5. Среди не отмеченных вершин ищется вершина с минимальным весом. Если таковая не найдена, то есть вес всех вершин равен

бесконечности, и маршрут не существует. Следовательно, выход. Иначе, текущей становится найденная вершина. Она же отмечается.

6. Если текущей вершиной оказывается конечная, то путь найден, и его вес есть вес конечной вершины.

7. Переход на 4.

Алгоритм поиска A.*

Алгоритм поиска для первого наилучшего совпадения на графике, который находит маршрут с самым меньшем весом пути от начальной вершины до конечной.

Этот алгоритм по сути расширяет алгоритм Дейкстры. Этот алгоритм обеспечивает намного большую производительность (по времени) благодаря эвристическому анализу.

В начале работы узлы, смежные с начальным просматриваются, выбирается тот из них, который имеет минимальное значение, после чего этот узел раскрывается. На каждом этапе алгоритм оперирует с множеством путей из начальной точки до всех ещё не раскрытых вершин графа — множеством частных решений, — которое размещается в очереди с приоритетом. Алгоритм продолжает свою работу до тех пор, пока значение целевой вершины не окажется меньшим, чем любое значение в очереди, либо пока весь граф не будет просмотрен. Из множества решений выбирается решение с наименьшей стоимостью.

1.4. Постановка задачи

Дана схема склада. Необходимо разработать систему, которая будет управлять группой роботов, предотвращая возможные столкновения и возникновение значительных простоев в доставке товаров со стеллажей.

Также система должна показывать местонахождение каждого робота и его информацию о его текущем состоянии в режиме реального времени.

Для разработки системы допустимо использовать следующие параметры работы роботов:

- Время, необходимое для передвижения в следующий сектор – 1 ед. времени.
- Время, необходимое для загрузки товара – 5 ед. времени.
- Время, необходимое для разгрузки товара – 5 ед. времени.

Пользователь имеет возможность создавать заявку на необходимый ему товар, наблюдать за статусом обработки задания.

Также должна существовать возможность изменять количество стеллажей и количество роботов.

Для выполнения каждого задания создается отдельный Task, который завершается при завершении роботом задания.

Если робот завершил текущее задание, проверяется, существуют ли еще задания на доставку. В случае если задания существуют, робот выполняет следующее задание в очереди. В случае если задания отсутствуют, робот занимает свободную позицию в стартовой локации у стены.

Если при создании нового задания все роботы свободны, выбирается ближайший к точке назначения робот.

Движение роботов должно происходить в соответствии с правилами согласования.

Глава 2. Разработка системы согласования движения группы роботов в замкнутом пространстве

2.1 Инструменты для разработки приложения

Приложение написано на языке программирования C#.

В качестве среды разработки была использована Microsoft Visual Studio. Microsoft Visual Studio предоставляет различные способы разработки программных приложений для ОС Windows.

2.2 Алгоритмы решения задачи

Для решения данной задачи было решено использовать централизованную стратегию управления.

Поскольку требуется искать маршрут на постоянно меняющемся поле, что представляет собой динамическую задачу, её можно представить в виде раздельных статических задач, а именно задача поиска пути в каждый отдельный момент времени для каждого отдельного робота. Чем чаще будет происходить пересчет пути робота, тем будет выше согласованность движения.

Однако стоит отметить что могут возникать спорные ситуации, в которых алгоритм поиска пути не найдет решения. Для решения таких проблем используются согласующие правила, которые определяют движение робота. Данная проверка производится перед перемещением на следующую возможную позицию.

Правила согласования:

1. Каждый робот имеет свой приоритет;
2. Робот, перевозящий товар, имеет наивысший приоритет. Робот, который едет к складу, имеет средний приоритет. Свободный робот и

робот возвращающийся в стартовую локацию имеет наименьший приоритет.

3. В случае если два робота стоят друг напротив друга по направлению движения, то уступить дорогу робот с наименьшим приоритетом. Такая ситуация возможна только при движении с зоны выгрузки. В данном случае, робот с наименьшим приоритетом двигается назад, до тех пор, пока слева или справа от него не будет свободной позиции, куда он и переместится.
4. В случае если такая ситуация происходит и роботы имеют одинаковый приоритет, то уступает дорогу робот с наименьшим количеством итераций.
5. Также существует вероятность, когда в одну точку в один и тот же момент времени могут перемещаться разные роботы. Для решения данной проблемы дорогу уступает тот робот, который движется по вертикале схемы склада в программе. Для определения данной ситуации система проверяет диагональные позиции и учитывает направления движения роботов.

Алгоритм работы системы:

1. Проверяется текущий список заданий.
2. Если список заданий не пуст, то ближайшему к необходимому стеллажу роботу назначается новое задание.
3. При передвижении на следующую позицию, робот пересчитывает весь маршрут целиком и проверяются правила согласования.
4. При направлении к стеллажу занимается соседняя свободная позиция относительно стеллажа.
5. Происходит загрузка товара, изменяется текущее состояние робота.
6. Происходит передвижение робота на одну из точек в зоне разгрузки.

7. Если список заданий после выполнения данного задания пуст, то робот направляется в стартовую локацию, в обратном случае приступает к выполнению следующего задания.

Алгоритм поиска маршрута A* (Приложение 2):

1. Создается 2 списка вершин — ожидающие рассмотрения и уже рассмотренные. В список ожидающих рассмотрения добавляется точка старта, список рассмотренных пока пуст.
2. Для каждой точки рассчитывается F = G + H. G — расстояние от старта до точки, H — примерное расстояние от точки до цели. Так же каждая точка хранит ссылку на точку, из которой в нее пришли.
3. До тех пор пока список точек на рассмотрение не пуст, из этого списка выбирается точка с наименьшим F. Обозначим ее X.
4. Если X — цель, то маршрут найден.
5. Переносим X из списка точек ожидающих рассмотрения в список уже рассмотренных.
6. Для каждой из точек, соседних для X (обозначим эту соседнюю точку Y), делаем следующее:
7. Если Y уже находится в списке рассмотренных — пропускаем ее.
8. Если Y еще нет в списке на ожидание — добавляем ее туда, запомнив ссылку на X и рассчитав Y.G (это X.G + расстояние от X до Y) и Y.H.
9. Если же Y в списке на рассмотрение — проверяем, если X.G + расстояние от X до Y < Y.G, значит достигнута точка Y более коротким путем, заменяем Y.G на X.G + расстояние от X до Y, а точку, из которой пришли в Y на X.
10. Если список точек на рассмотрение пуст, а цель так и не достигнута — маршрут не существует.

2.3 Основные классы и методы

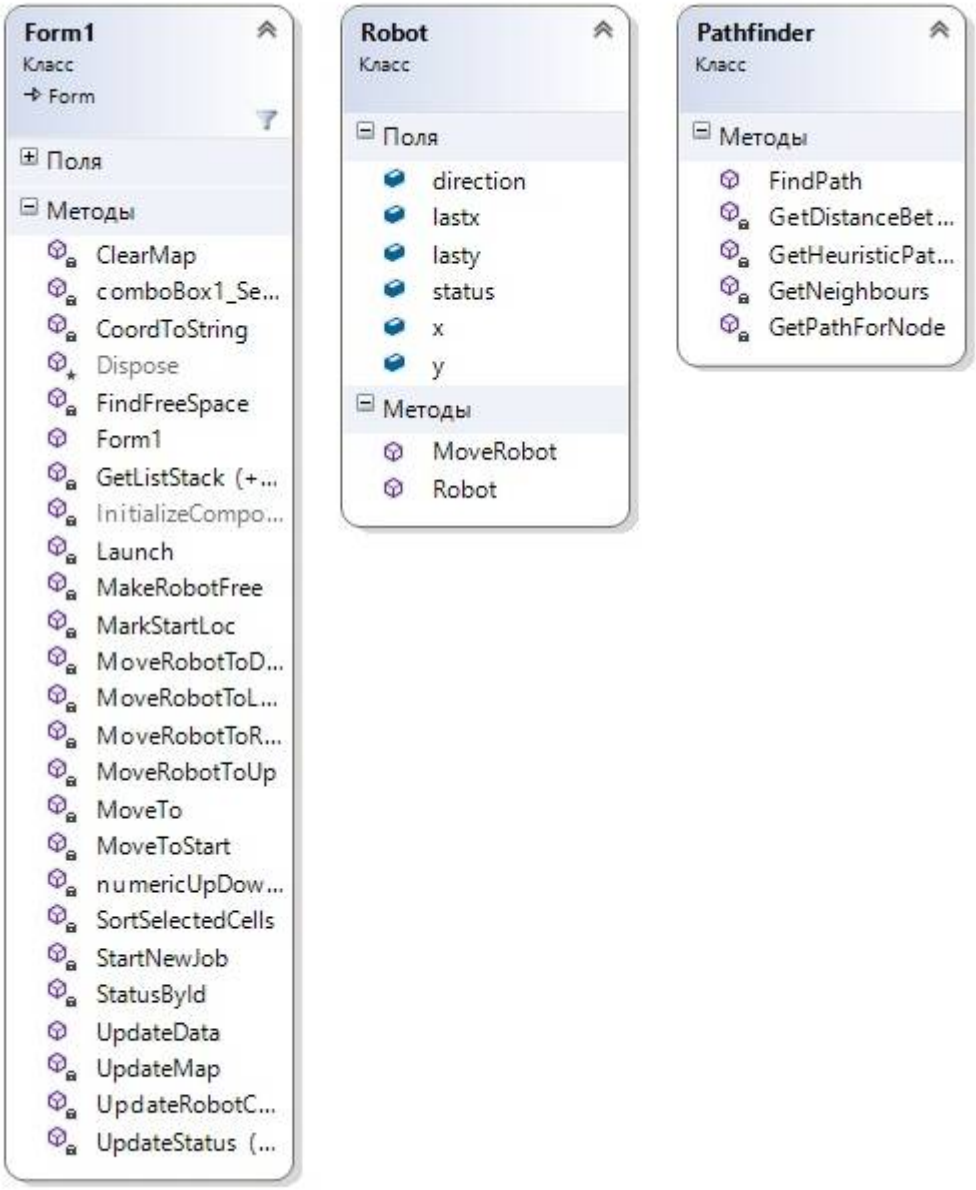

Рис. 2.1. Диаграмма классов

Класс Robot

Данный класс представляет собой робота, который хранит информацию о текущей и прошлой позиции, направления, а также метод, который организует движение робота на следующую позицию по текущему направлению.

Класс Form1

В данном классе происходит управление роботами, а также описываются события элементов интерфейса программы.

- private void MoveTo(Robot robot, int tx, int ty, int[,] field) — метод реализующий движение робота до заданной точки с координатами (tx,ty). Происходит перемещение робота на следующую позицию, относительно текущего местоположения. Перед перемещением происходит проверка правил согласования, если проверка пройдена происходит перемещение на следующую позицию по направлению движения. Если проверка не пройдена, выполняются предписанные правилами действия. После перемещения на следующую позицию происходит новый поиск пути до конечной точки.
- private void StartNewJob(Robot robot, int tx, int ty, int[,] field) — метод, реализующий создание нового потока для выполнения заказа, доставки заказа в зону выгрузки и возврата в стартовую локацию. При завершении потока, робот добавляется в список свободных роботов.
- private void UpdateData() - метод для обновления информации, хранящейся в DataGridView, а также для повторной окраски ячеек в зависимости от значений хранящейся в матрице поля.
- private void ClearMap() - метод для очистки главной части склада для дальнейшей генерации местоположения стеллажей.
- private void UpdateMap(int count) — метод для автоматической генерации стеллажей. Позволяет сгенерировать стеллажи определенных размеров.

- private void Launch() - метод для запуска роботов. Вызывается в управляющем потоке.
- private Point FindFreeSpace(int x,int y) — метод для поиска свободных позиций рядом со стеллажом. Проверяются соседние позиции у стеллажа, исключая диагональные позиции.
- private List<DataGridViewCell> SortSelectedCells() - метод для получения коллекции выделенных ячеек в виде отсортированного списка.
- private void UpdateStatus() - метод для обновления текущего состояния робота.
- private void UpdateRobotCount() - метод для обновления количества роботов.
- private void MarkStartLoc() - метод для обозначения стартовой локации.
- private void GetListStack() - метод для получения списка заданий.
- Редактирование поля производится в событиях соответствующих кнопок. В данном случае изменяются списки, хранящие стеллажи, стены, стартовую локацию и зону выгрузки. При изменении поля, происходят соответствующие изменения в списках и поле перерисовывается.

Класс Pathfinder

Данный класс реализует алгоритм поиска пути A*, а также дополнительные методы необходимые для работы алгоритма.

- public static List<Point> FindPath(int[,] field, Point start, Point goal) — основной метод поиска пути. При поиске пути учитывается текущее местоположение всех роботов, в случае если на предполагаемом маршруте находится другой робот, то вес пути увеличивается дополнительно на единицу. Информация о текущих позициях роботов хранится в двумерном массиве field. Данный метод вызывается для каждой следующей точки маршрута, что позволяет найти наиболее оптимальный путь на динамическом поле.
- private static int GetHeuristicPathLength(Point from, Point to) — метод для расчета эвристического расстояния до конечной точки маршрута. Эвристическим расстоянием является длина пути без препятствий, которая рассчитывается по следующей формуле: |x1-x2|+|y1-y2| , где (x1,y1) – координата текущей точки (x2,y2) – координата конечной точки.
- private static Collection<Point> GetNeighbours(Point point, Point goal, int[,] field) — метод для получения списка соседей для точки.
- private static List<Point> GetPathForNode(Point point) — метод для получения маршрута, маршрут представлен в виде списка координат точек.

2.4 Руководство пользователя

Запустив данное приложение, пользователь попадает на главную форму. В частности он может наблюдать схему склада и иметь возможность создания нового заказа. Свободные роботы по умолчанию отображаются зеленым цветом на форме.

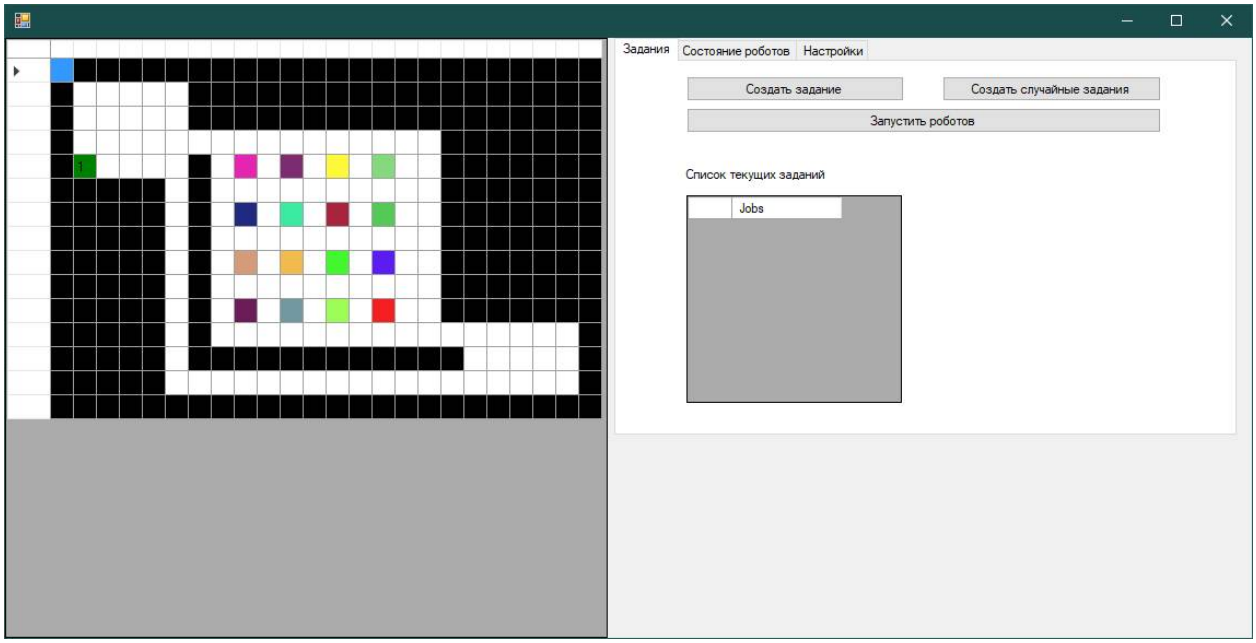

Рис. 2.2. Главная форма

Предварительно можно настроить количество роботов, а также размер стеллажей на вкладке «Настройки». В случае создания задания вручную к выполнению задания приступает ближайший робот. В случае случайной генерации заданий, необходимо запустить роботов вручную. Для генерации задания необходимо выделить необходимый стеллаж и нажать кнопку «Создать задание». Для создания нового задания необходимо выделить необходимый стеллаж и нажать кнопку «Создать задание». Ближайший к данному стеллажу свободный робот начнет выполнение задания. После того как робот возьмет товар с необходимого стеллажа, цвет робота изменится на цвет стеллажа и робот направится к зоне выгрузки. Стеллажи могут иметь

похожие цвета, поскольку цвет каждого стеллажа генерируется случайно. Однако, необходимо помнить, что каждый стеллаж представляет собой разный товар, даже если получилось так, что цвета похожи.

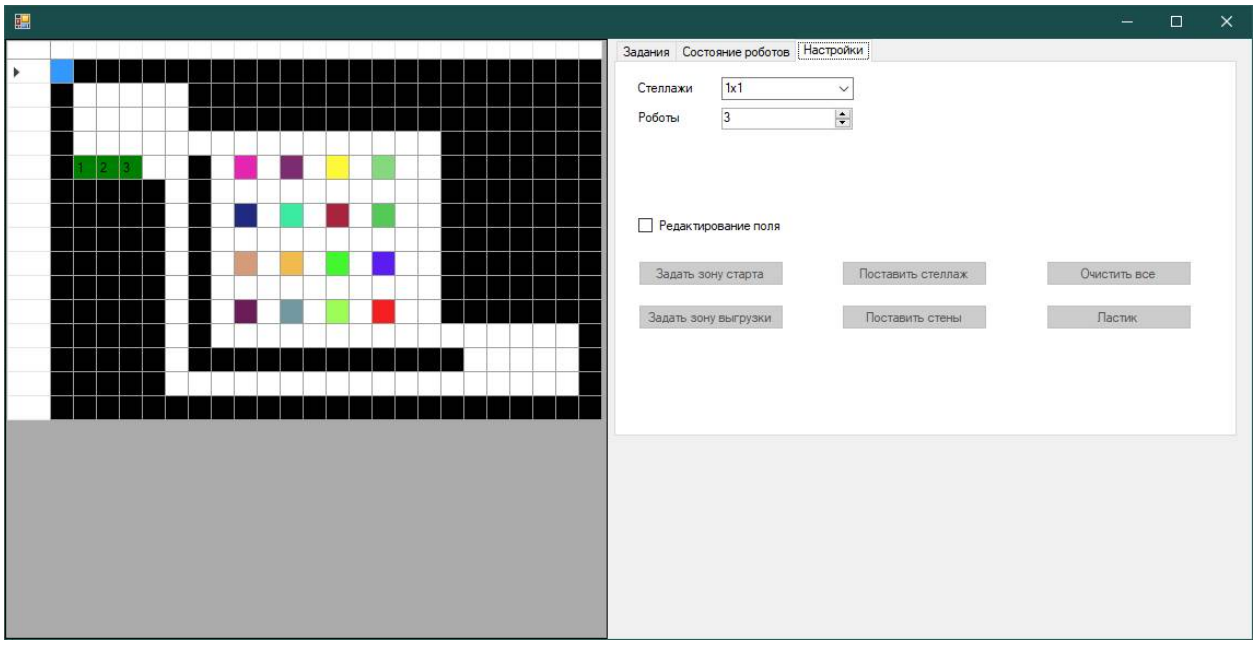

Рис. 2.3. Вкладка «Настройки»

На вкладке «Настройки» возможно изменять количество действующих роботов и расположение и количество стеллажей на поле. Также на данной вкладке можно настроить рабочее поле. Существует возможность добавления стен и стеллажей, а также установка стартовой локации и зоны выгрузки. Для добавления необходимо сначала выделить необходимую область на рабочем поле, а потом нажать на кнопку. В случае необходимости удаления, существует 2 варианта, очистить всё рабочее поле, или очистить выделенный участок. В случае включения режима редактирования, цвет поля меняется, а также отмечаются стартовая локация и зона выгрузки

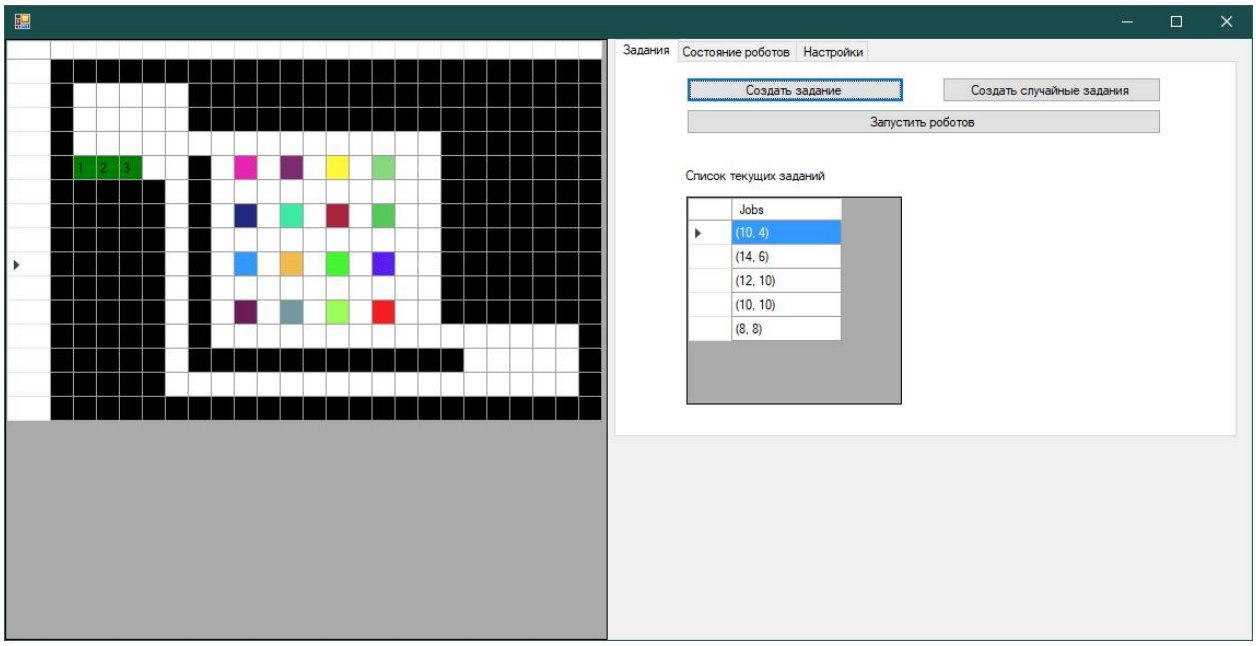

Рис. 2.4. Главная форма с заданиями

Для того чтобы начать работу системы необходимо нажать кнопку «Запустить роботов». При нажатии данной кнопки будет создан управляющий поток, который будет распределять роботов на задания до тех пор пока список заданий не будет пуст, и будет находить координату следующей позиции для каждого робота. Так же управляющий поток будет следить за состояниями роботов, и назначать им новые задания, если они уже выполнили свои текущие задания.

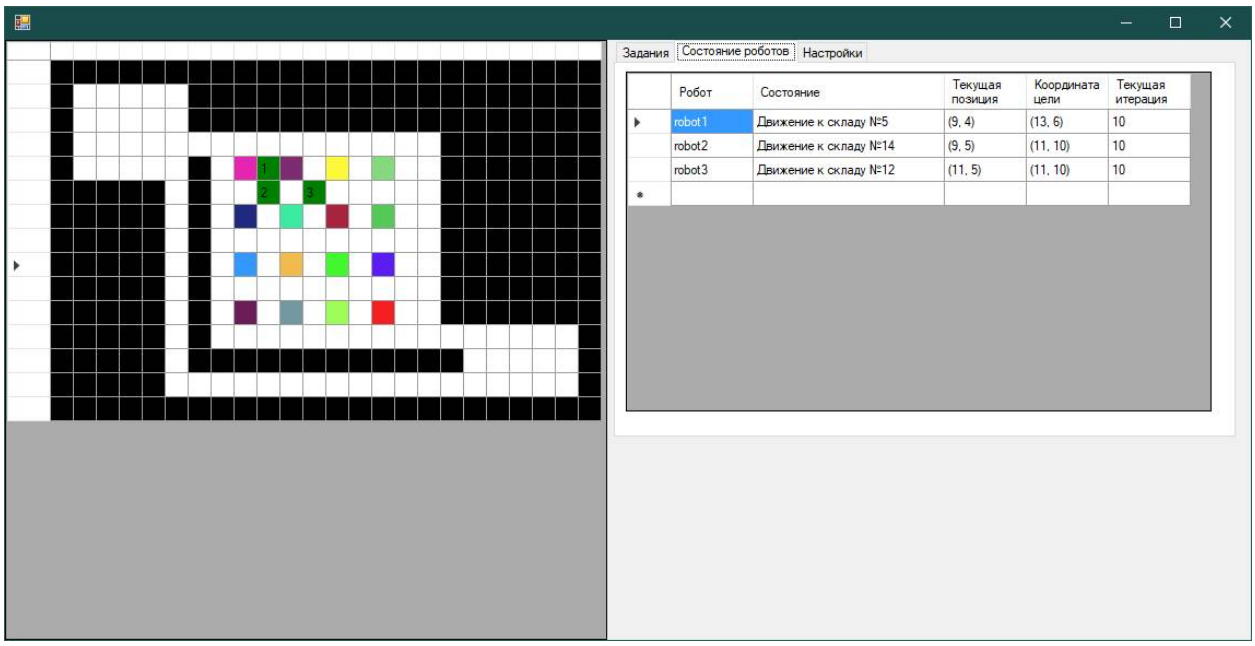

Рис. 2.5. Вкладка «Состояние роботов»

На вкладке «Состояние роботов» можно наблюдать текущее состояние роботов. Возможные значения: «Движение к стеллажу №N», «Доставка товара N», «Свободен», где N — номер стеллажа. Также можно увидеть текущую координату робота, координату цели и количество пройденных координат в текущий момент времени.

2.5 Статистика работы

В ходе работы было принято решение замерить количество времени необходимое системе для выполнения одного и того же количества заданий при различном количестве роботов Дабы исключить фактор случайности, задания были сгенерированы следующим образом:

Сначала создаются задания начиная с левого верхнего угла, и затем следующие по координате Y, и затем следующие ряды по координате X. Таким образом, получается одинаковый список заданий в каждом случае

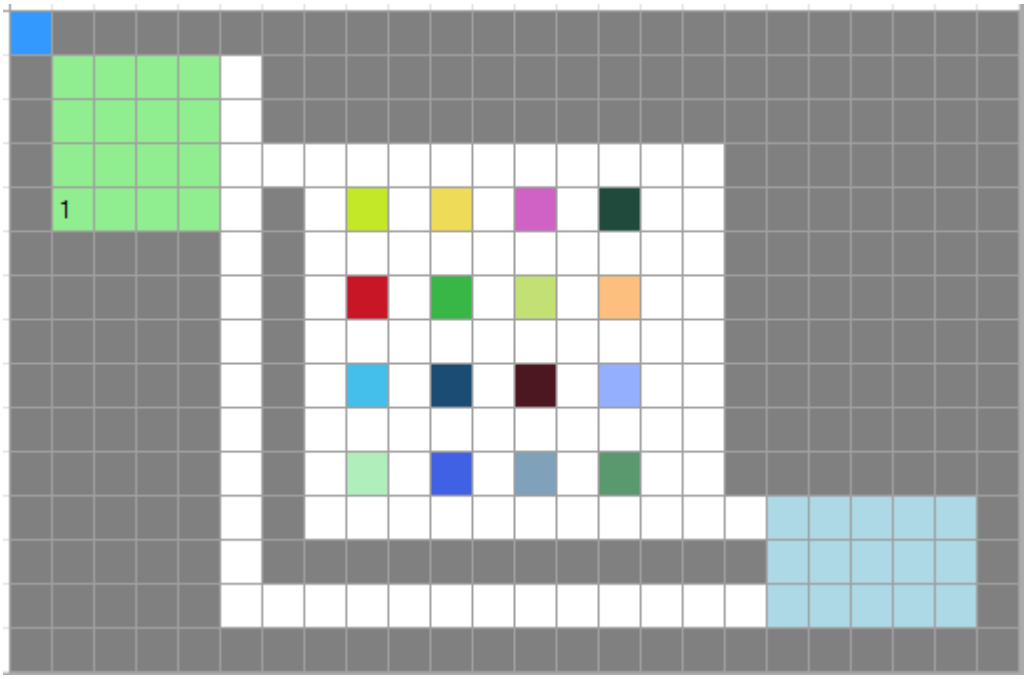

Рис. 2.6. Схема поля №1.

Таблица 1.

Количество роботов	Время выполнения 16 заданий (Количество тактов)
1	472
2	241

3	176
4	143
5	118
6	107
7	98
8	95

Поле №2 позволяет добиться наименьшего времени на большом количестве роботов, поскольку количество ситуации когда один робот ждет, пока проедет другой сведено к минимуму.

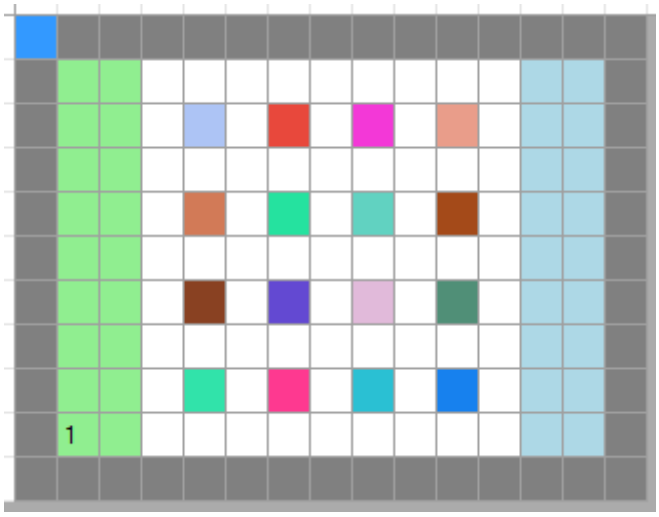

Рис. 2.6. Схема поля №2.

Таблица 2.

Количество роботов	Время выполнения 16 заданий (Количество тактов)
1	462
2	229
3	153
4	116
5	95
6	72
7	68
8	61

Полученные результаты показывают экспоненциальную зависимость между количеством роботов и времени необходимым для выполнения заданного количества заказов.

Заключение

Для разработки математической модели использован комплексный подход на основании результатов научных исследований, изложенных в публикациях в различных отечественных и зарубежных изданиях.

Реализация системы согласования движения группы роботов в замкнутом пространстве получена на основе изучения теоретического учебного материала, научных публикаций, в том числе на английском языке.

В ходе выполнения дипломной работы выполнены следующие поставленные задачи:

- Рассмотрены существующие стратегии группового управления;
- Изучены алгоритмы поиска пути в графе;
- Реализован алгоритм A*;
- Разработано программное приложение на языке C#.

Перспективы дальнейшего развития приложения:

- Создание отчетности о доставке товаров;
- Интеграция с существующими системами учета товаров, в целях информирования пользователя и роботов об оставшемся количестве товаров;
- Возможность управления роботом в течении выполнения задания;

Список литературы

1. Каляев И.А., Гайдук А.Р., Капустян С.Г. Модели и алгоритмы коллективного управления в группах роботов. М.: Физматлит, 2009. 280с.

2. Норсеев С. А., Багаев Д. В. Обзор алгоритмов группового управления робототехническими комплексами // Электротехнические системы и комплексы. 2013. №21. С.137-145

3. Интеллектуальные роботы: учебное пособие для вузов / под общей ред. Е.И. Юревича / И.А. Каляев, В.М. Лохин, И.М. Макаров и др. - М.: Машиностроение, 2007. - 360 с.

4. Алгоритмы нахождения кратчайшего пути — ИНТУИТ //URL:https://www.intuit.ru/studies/courses/648/504/lecture/11475

5. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. Introduction to Algorithms. — 3rd edition. — The MIT Press, 2009.

Приложение 1.

```
private void StartNewJob(Robot robot,int tx, int ty, Grid grid)
        {
            bool flag = false;
            int tmp = 2;
            int rx=0, ry = 0;
            for (int i = 0; i < StackList.Count; i++)
            {
                if (StackList[i].ColumnIndex == tx)
                    if (StackList[i].RowIndex == ty)
                        tmp += i;
            }
            var ts = new CancellationTokenSource();
            CancellationToken ct = ts.Token;
            Task T = Task.Factory.StartNew(() =>
            {
                while (true)
                {
                    Random rand = new Random();
                    robot.status = tmp;
                    UpdateStatus(tx,ty);
                    MoveTo(robot, tx, ty,grid);
                    Thread.Sleep(1000);
                    ry= rand.Next(300) / 100;
                    rx=rand.Next(400) / 100;
```

```
            robot.status = -tmp;
            UpdateStatus(18 + rx, 11 + ry);
            MoveToEx(robot, 18+rx, 11+ry, grid);
            Thread.Sleep(1000);
            robot.status = -1;
            UpdateStatus();
            flag = true;
            if (flag)
            {
               FreeRobots.Add(robot);
               robot.status = 0;
               UpdateStatus();
            }
            if(Jobs.Count == 0)
            {
               MoveToStart(robot, 4, 4, grid);
            }
            if (ct.IsCancellationRequested)
            {
               break;
            }
         }
      }, ct);
      Thread.Sleep(100);
      ts.Cancel();
   }
```

Приложение 2.

```
public static List<Point> FindPath(int[,] field, Point start, Point goal)
    {
      // Шаг 1.
      var closedSet = new Collection<PathNode>();
      var openSet = new Collection<PathNode>();
      // Шаг 2.
      PathNode startNode = new PathNode()
      {  Position = start,
         CameFrom = null,
         PathLengthFromStart = 0,
         HeuristicEstimatePathLength = GetHeuristicPathLength(start, goal)
      };
      openSet.Add(startNode);
      while (openSet.Count > 0)
      {
         // Шаг 3.
         var currentNode = openSet.OrderBy(node =>
           node.EstimateFullPathLength).First();
         // Шаг 4.
         if (currentNode.Position == goal)
            return GetPathForNode(currentNode);
         // Шаг 5.
         openSet.Remove(currentNode);
```

```
            closedSet.Add(currentNode);
            // Шаг 6.
            foreach (var neighbourNode in GetNeighbours(currentNode, goal, field))
            {
                // Шаг 7.
                if (closedSet.Count(node => node.Position ==
neighbourNode.Position) > 0)
                    continue;
                var openNode = openSet.FirstOrDefault(node => node.Position ==
neighbourNode.Position);
                // Шаг 8.
                if (openNode == null)
                    openSet.Add(neighbourNode);
                else
                  if (openNode.PathLengthFromStart >
neighbourNode.PathLengthFromStart)
                {
                    // Шаг 9.
                    openNode.CameFrom = currentNode;
                    openNode.PathLengthFromStart =
neighbourNode.PathLengthFromStart;
                }
            }
        }
        // Шаг 10.
        return null;
    }
```