САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И СИСТЕМ

Михайлов Илья Евгеньевич

Выпускная квалификационная работа бакалавра

Разработка мультиагентной системы управления перемещениями в виртуальной среде

Направление 010400

Прикладная математика и информатика

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Жабко Н. А.

Санкт-Петербург

2016

Содержание

Введение

Проблемы искусственного интеллекта на протяжении многих лет занимают умы многих ученых. Одной из важнейших составляющих теории искусственного интеллекта являются агенты – сущности, обладающие интеллектом. Системы, которые способны управлять более чем одним агентом, называются мультиагентными. Данные системы имеют обширные области применения, такие как робототехника, системы виртуальной реальности, компьютерные игры, симуляторы толпы, моделирование транспортных потоков. В этих системах агентами могут являться роботы, виртуальные люди, а также транспорт.

Важной и актуальной проблемой разработки мультиагентных систем является управление перемещениями агентов в реальном времени. Моделирование перемещений занимает особое место в создании систем искусственного интеллекта, поскольку множество сложных моделей поведения агентов базируется именно на способности к перемещениям. Агенты, воплощенные в некоторой виртуальной среде, которая может моделироваться по образу реально существующей местности, могут выполнять задачи, связанные с передвижением из одной точки виртуальной среды в другую. При этом, поскольку агенты воплощены в данной среде, они подчиняются ее ограничениям, среди которых можно выделить неподвижные препятствия – непроходимые участки виртуальной среды, а также самих агентов, которые являются динамическими препятствиями друг для друга.

Под мультиагентной системой управления перемещениями в виртуальной среде будем понимать определенный набор средств, обеспечивающий передвижение агентов из одной точки виртуальной среды в другую без столкновения с неподвижными препятствиями и с другими агентами.

Цель работы: разработать мультиагентную систему управления перемещениями в виртуальной среде.

При разработке мультиагентной системы управления перемещениями в первую очередь была проанализирована проблема глобального планирования пути – прокладки маршрута для агента в виртуальной среде без учета присутствия в этой среде других агентов. Это объясняется тем, что при перемещении агента из некоторой начальной точки виртуальной среды в конечную (целевую) агент должен придерживаться кратчайшего пути между начальной и конечной точками с учетом неподвижных препятствий для более быстрого и эффективного выполнения своих задач. При этом планирование кратчайшего пути, которого должен придерживаться агент, не зависит от других агентов по причине того, что долгосрочное прогнозирование столкновений агента с другими агентами требует большого количества вычислительных ресурсов, что неприемлемо при работе системы, в которой одновременно участвует множество агентов в реальном времени. В связи с этим было решено использовать графы, характеризующие проходимые участки виртуальной среды, на которых будет осуществляться поиск кратчайшего пути. Проблема глобального планирования пути рассматривается в главе 1.

После того, как был найден кратчайший путь в качестве результата глобального планирования, возникает проблема трассировки пути – отслеживания возможности передвижения агента из своего текущего местоположения напрямую к каждой вершине графа, составляющей путь (данной проблеме посвящена глава 2), а также проблема предотвращения столкновений при локальном взаимодействии между агентами, о которой говорится в главе 3.

Постановка задачи

Введем в рассмотрение двумерное евклидово пространство $\mathbb{E}^2$. Рассмотрим в этом пространстве прямоугольник, поделенный на равные квадраты, которые назовем клетками. Введем правостороннюю прямоугольную Декартову систему координат. Начало координат установим в нижнем левом углу рассматриваемого прямоугольника. На основе данного прямоугольника образуется граф

$$G = G(V, E)$$

в виде равномерной сетки с множеством вершин V, которыми являются центры клеток, и множеством ребер E. Соседние клетки (смежные вершины) выбираются из окрестности Мура. Это означает, что соседними клетками считаются 8 клеток (по вертикали, по горизонтали и по диагонали) или меньше, если какие-то из них отсутствуют. Соседние клетки (смежные вершины) можно называть соседями. Ортогональными соседями будем называть соседей, расположенных по вертикали или по горизонтали относительно друг друга. Диагональными соседями будем называть соседей, расположенных по диагонали относительно друг друга. Расстояния между ортогональными соседями примем равными 1, а между диагональными соседями – $\sqrt{2}$. Каждая клетка (как и соответствующая ей вершина) может быть проходимой или непроходимой. Непроходимые клетки являются неподвижными препятствиями. В дальнейшем рассматриваемый граф можно называть картой.

Путь p между вершинами $v_1, v_n \in V$ – это упорядоченный набор вершин графа G, в котором каждая последующая вершина является смежной с предыдущей:

$$p(v_1, v_n) = (v_1, v_2, \dots, v_n) \mid \{v_i, v_{i+1}\} \in \tilde{E}, i = \overline{1, n-1},$$

где $\tilde{E}$ – такое подмножество ребер графа G, что между концевыми вершинами любого ребра $e \in \tilde{E}$ можно в обе стороны проложить путь напрямую. Будем говорить, что путь нельзя проложить от вершины до ее

соседа напрямую, если он является непроходимой вершиной, и, кроме того, путь нельзя проложить напрямую от вершины:

- до ее верхнего левого соседа, если ее верхний или левый сосед непроходим;
- до ее верхнего правого соседа, если ее верхний или правый сосед непроходим;
- до ее нижнего правого соседа, если ее нижний или правый сосед непроходим;
- до ее нижнего левого соседа, если ее нижний или левый сосед непроходим.

Иными словами, путь не должен проходить сквозь углы неподвижных препятствий. Длина d пути p между вершинами $v_1, v_n \in V$ определяется следующим образом:

$$d\big(p(v_1, v_n)\big) = \sum_{i=1}^{n-1} \rho(v_i, v_{i+1}),$$

где $(v_1, v_2, \dots, v_n)$ – упорядоченный набор вершин графа G, являющийся путем $p(v_1, v_n)$, $\rho(v_i, v_{i+1})$ – расстояние между смежными вершинами $v_i, v_{i+1} \in V$, $i = \overline{1, n-1}$. Совокупность путей между вершинами $v_1, v_n \in V$ образует множество путей $P(v_1, v_n)$ между этими вершинами. Путь p_{short} между вершинами $v_1, v_n \in V$ называется кратчайшим путем между данными вершинами, если его длина не превосходит длины других путей между этими вершинами:

$$p_{short}(v_1, v_n) = \underset{p \in P(v_1, v_n)}{\operatorname{argmin}} d(p).$$

Агенты представляют собой круги одинакового радиуса. Диаметры агентов равны четвертой части стороны каждого квадрата, на основе которых построен граф G. Скорости агентов ограничены одинаковой величиной. Изначально центр каждого агента совпадает с определенной вершиной графа, при этом для каждой вершины существует не более одного агента, центр которого совпадает с данной вершиной.

Требуется обеспечить выполнение каждым агентом задачи о передвижении из одной вершины графа в другую, используя кратчайший путь на графе между этими двумя вершинами, без столкновения с неподвижными препятствиями и с другими агентами (динамическими препятствиями). Будем считать, что между агентом A и препятствием O (неподвижным или динамическим) произошло столкновение, если существует хотя бы одна точка $x \in \mathbb{E}^2$, которая принадлежит одновременно и агенту A, и препятствию O.

Агенты должны достигать вершины при условии, что существует путь на графе до этой вершины и в момент прибытия агента к этой вершине она не занята другим агентом. Вершина считается занятой агентом, если хотя бы одна точка этого агента принадлежит соответствующей этой вершине клетке.

Обзор литературы

Мультиагентные системы управления перемещениями – это достаточно обширная тема, поэтому литература, касающаяся поставленной задачи, освещает отдельные составляющие рассматриваемой проблемы.

Алгоритмы поиска кратчайшего пути на графах, необходимые для планирования путей для агентов в виртуальной среде, представлены в книгах [1] и [2]. В данной работе присутствуют ссылки на мои статьи о порядке обхода вершин графа в алгоритме волновой трассировки [3] и о разработке модификации алгоритма волновой трассировки [4]. Идеи использования поведения, заключающегося в изменении направления движения с помощью управляющих воздействий, изложены в работе [5].

Подходы к предотвращению столкновений между агентами в основном отражены в виде статей. Подходы VO и RVO рассматриваются в работе [6]. Подходы HRVO и ORCA предложены соответственно в статьях [7] и [8].

Глава 1. Глобальное планирование пути

В данной главе рассматриваются алгоритмы поиска кратчайшего пути на графах – алгоритм A* и алгоритм волновой трассировки, приведены результаты практического сравнения эффективности этих алгоритмов по быстродействию, а также представлена разработанная модификация алгоритма волновой трассировки вместе с результатами экспериментов, демонстрирующими превосходство в быстродействии созданной модификации над алгоритмом волновой трассировки.

В пределах этой главы будем отождествлять понятия «вершина» и «клетка», поскольку между множествами вершин и клеток устанавливается взаимно-однозначное соответствие, и, кроме того, вершины будем также называть ячейками или узлами.

1.1. Алгоритм A*

Алгоритм A* – это алгоритм поиска по первому наилучшему совпадению на графе, который находит кратчайший путь от одной вершины (начальной) к другой (целевой, конечной).

Это информированный метод поиска, то есть порядок обхода вершин определяется эвристической функцией $f(x)$, которая представляет собой сумму двух других: функции $g(x)$, определяющей длину пути, найденного на предыдущих этапах алгоритма, из начальной вершины к рассматриваемой вершине x, и эвристической оценки $h(x)$ длины кратчайшего пути из рассматриваемой вершины x в конечную [1].

Алгоритм A* рассматривается в данной работе по той причине, что этот алгоритм является допустимым, так как он никогда не упустит возможности минимизировать длину пути, полным (в том смысле, что он всегда находит решение, если таковое существует) и обходит при этом минимальное количество вершин.

1.2. Алгоритм волновой трассировки

Алгоритм волновой трассировки – это алгоритм поиска кратчайшего пути на графе, основанный на методах поиска в ширину [2].

В связи с тем, что соседние клетки выбираются из окрестности Мура, в рассматриваемом алгоритме возникают неоднозначности при восстановлении пути после распространения волны. Это чревато нахождением пути, который не является кратчайшим, поэтому необходима конкретизация. При возникновении такой ситуации будем устанавливать наивысший приоритет у ортогональных соседей текущего узла при восстановлении пути. Такой приоритет будет обеспечивать нахождение кратчайшего пути исходя из того, что расстояние между диагональными соседями превышает расстояние между ортогональными соседями, а также исходя из правил распространения волны, определяемых данным алгоритмом.

Алгоритм волновой трассировки был выбран для практического сравнения с алгоритмом A*, поскольку он также находит кратчайший путь и всегда находит решение, если оно существует, однако совершает меньшее, по сравнению с алгоритмом A*, количество вычислительных операций при рассмотрении каждой вершины графа, хотя и обходит при этом достаточно большое количество вершин.

1.3. Практическое сравнение эффективности алгоритма A* и алгоритма волновой трассировки по быстродействию

Алгоритмы были реализованы на языке программирования C++. В каждом эксперименте алгоритм A* сравнивается с алгоритмом волновой трассировки при одинаковых условиях, т. е. на одинаковых картах и при одних и тех же начальных и целевых узлах. Все эксперименты проводились на картах размером 90 на 90.

Результат первого эксперимента с алгоритмом A* и алгоритмом волновой трассировки изображен на рис. 1.1. Аналогичным образом результат второго эксперимента показан на рис. 1.2, а результаты третьего – на рис. 1.3 и рис. 1.4 (в третьем эксперименте с помощью сравниваемых алгоритмов были получены разные пути, одинаковые по длине). Рисунки иллюстрируют карту, начальную и целевую клетки, отмеченные синим и красным цветами соответственно, и найденный кратчайший путь между ними, который обозначается зеленым цветом. Неподвижные препятствия выделены черным цветом. Результаты всех экспериментов приведены в таблице 1.1.

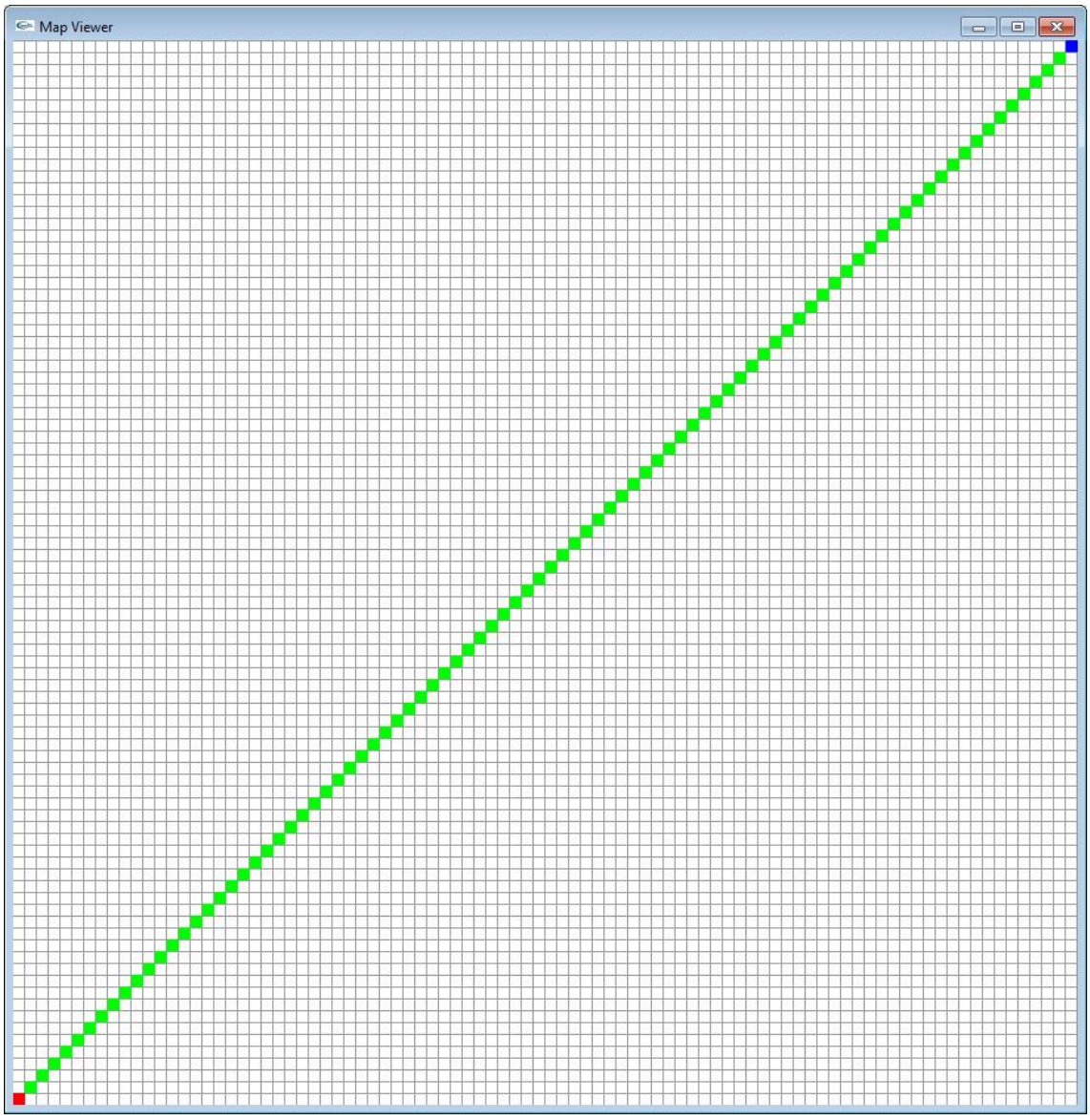

Рис. 1.1. Первый эксперимент с алгоритмом A* и алгоритмом волновой трассировки

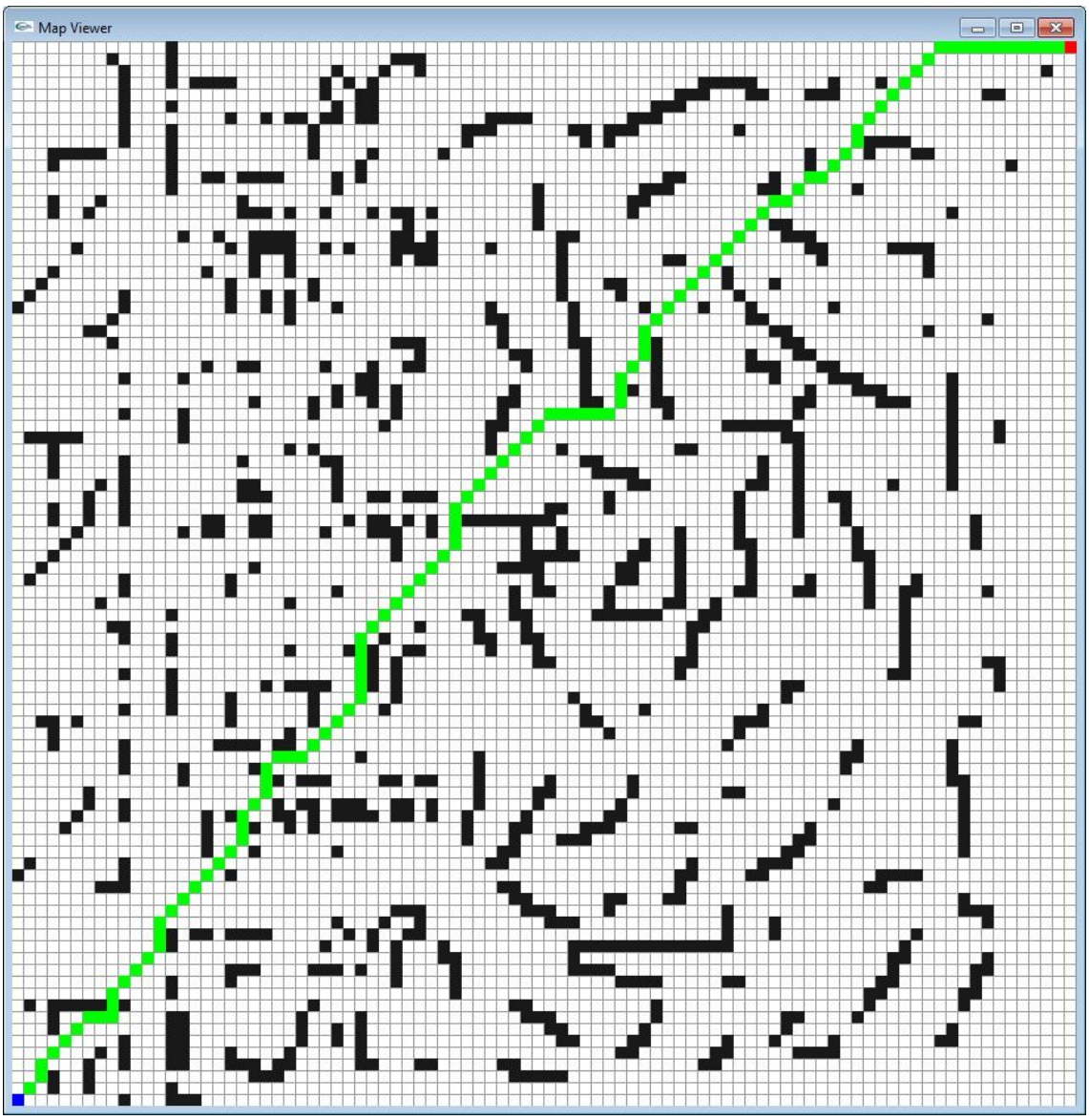

Рис. 1.2. Второй эксперимент с алгоритмом A* и алгоритмом волновой трассировки

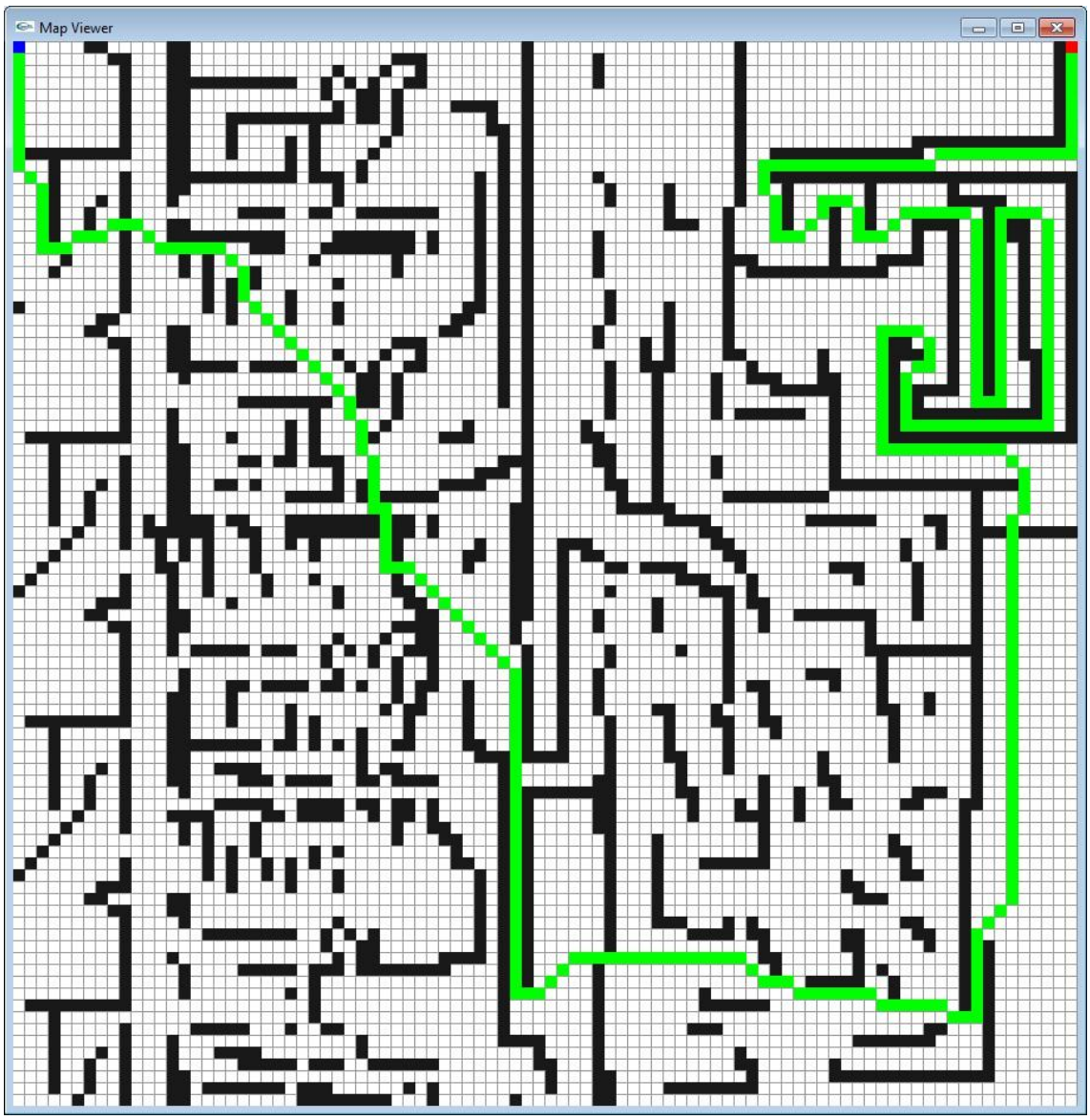

Рис. 1.3. Третий эксперимент с алгоритмом A*

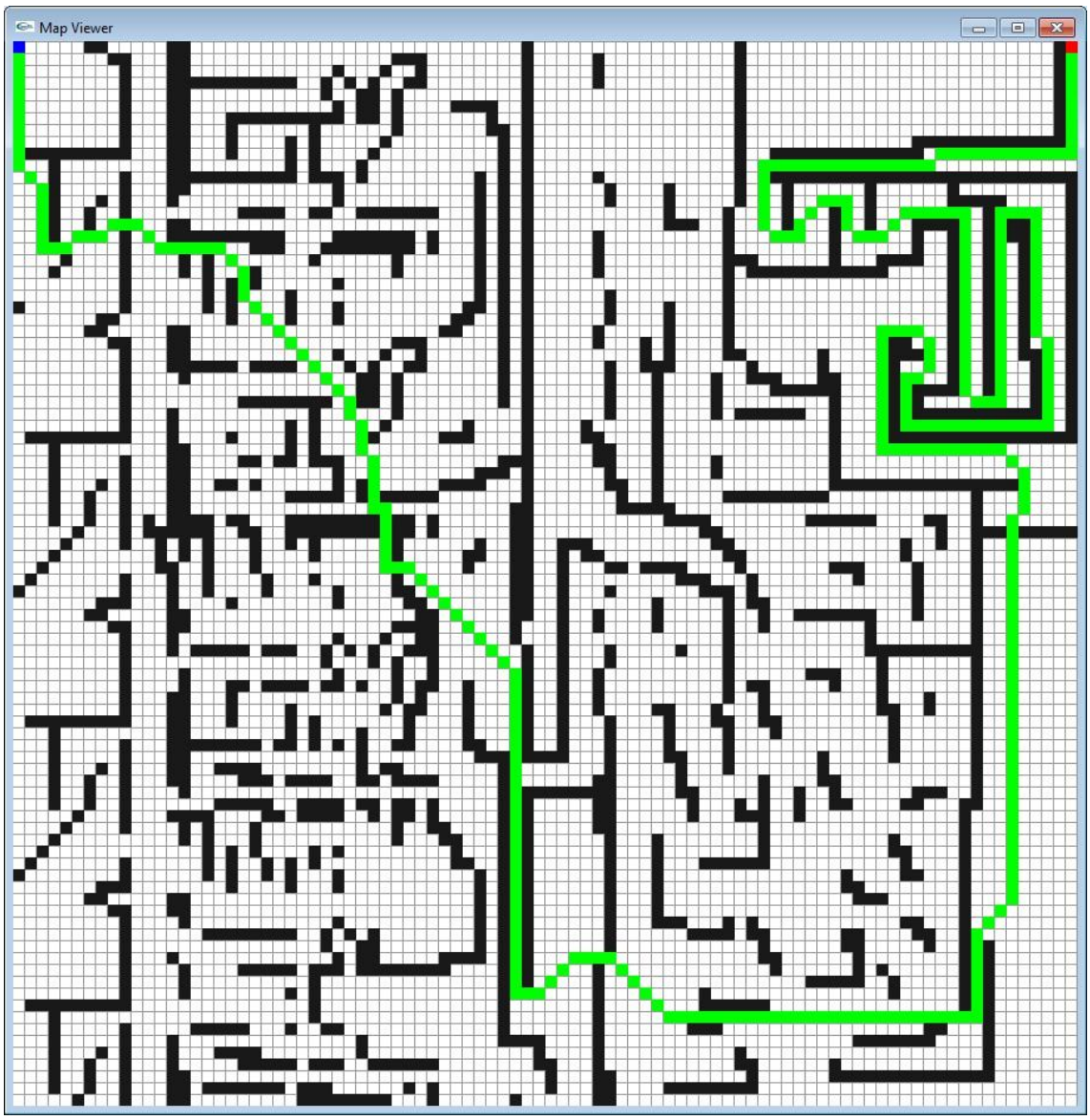

Рис. 1.4. Третий эксперимент с алгоритмом волновой трассировки

Таблица 1.1 – Результаты практического сравнения алгоритма A* с алгоритмом волновой трассировки

	Время работы в первом эксперименте	Время работы во втором эксперименте	Время работы в третьем эксперименте
Алгоритм A*	4 мс	5 мс	56 мс
Алгоритм волновой трассировки	3 мс	3 мс	3 мс

Эксперименты показали, что алгоритм волновой трассировки находит кратчайший путь быстрее, чем алгоритм A*. В связи с этим на основе

алгоритма волновой трассировки было целесообразно разработать модификацию, которая улучшит показатели алгоритма.

1.4. Разработка модификации алгоритма волновой трассировки

Идея модификации алгоритма заключается в реализации двунаправленного поиска, что позволит обходить меньшее количество вершин графа при поиске кратчайшего пути по сравнению с алгоритмом волновой трассировки.

В ходе работы модифицированного алгоритма одновременно будет происходить распространение волны не только от начального узла, но и от конечного. Волну, распространяющуюся от начальной клетки, будем называть прямой, а волну, распространяющуюся от конечной клетки, – встречной.

Основной проблемой создания модификации алгоритма являлась выработка правила построения пути после встречи волн. Для решения этой проблемы был разработан порядок обхода вершин графа, который играет важную роль в модификации алгоритма волновой трассировки. Рассмотрим разработанный способ обхода вершин подробно на примере.

Рассмотрим карту, приведенную на рис. 1.5.

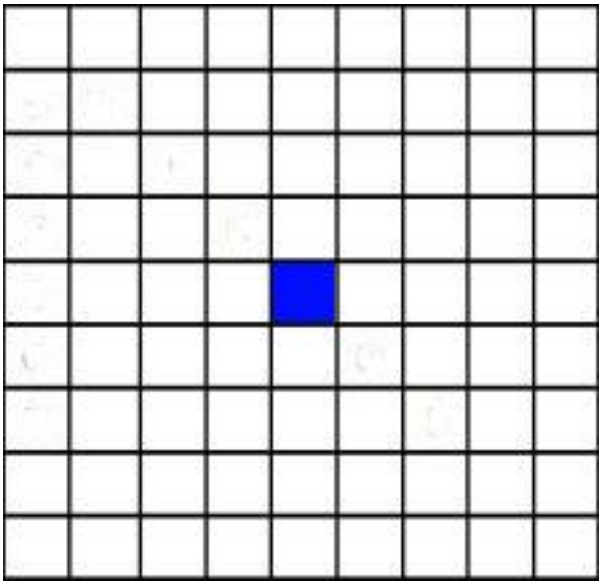

Рис. 1.5. Карта с отмеченной синим цветом начальной клеткой

Пусть волна начинает распространяться из начальной клетки, отмеченной на рис. 1.5 синим цветом. На первом шаге отметим соседей этой клетки в следующем порядке: ортогональных соседей по часовой стрелке, начиная от верхнего, а затем диагональных соседей по часовой стрелке, начиная от верхнего левого. Порядок изображен на рис. 1.6 с помощью порядковых номеров. Красными линиями соединена начальная ячейка с ее соседями.

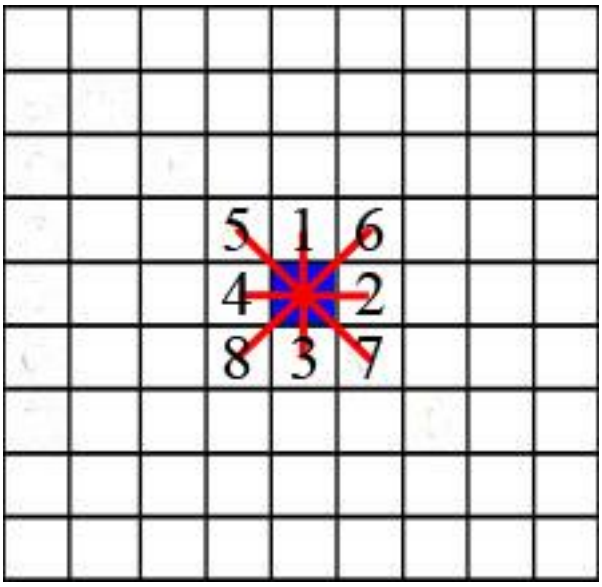

Рис. 1.6. Первый этап распространения волны

Перейдем ко второму шагу. Будем поочередно рассматривать каждую вершину, отмеченную на рис. 1.6 номером, в порядке, соответствующем порядку номеров узлов, и помечать ее соседей, не просмотренных ранее, в таком же порядке, в котором рассматривали соседей начальной клетки. Для наглядности пронумеруем отмеченные на данном шаге ячейки в порядке их просмотра и соединим их красными линиями с теми вершинами, при рассмотрении которых они были отмечены. Результат второго шага проиллюстрирован на рис. 1.7.

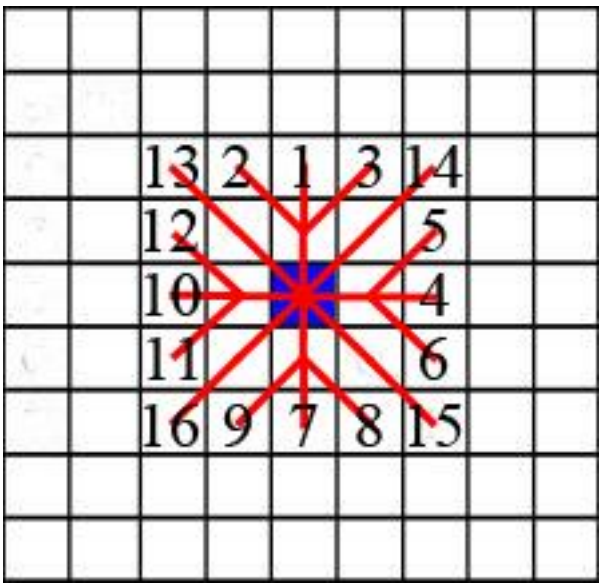

Рис. 1.7. Второй этап распространения волны

На рис. 1.7 можно увидеть, что красные линии образуют кратчайшие пути от начальной клетки до каждой из рассмотренных ячеек. Аналогично сделаем третий шаг и покажем его результат на рис. 1.8 таким же образом, как и на предыдущем шаге.

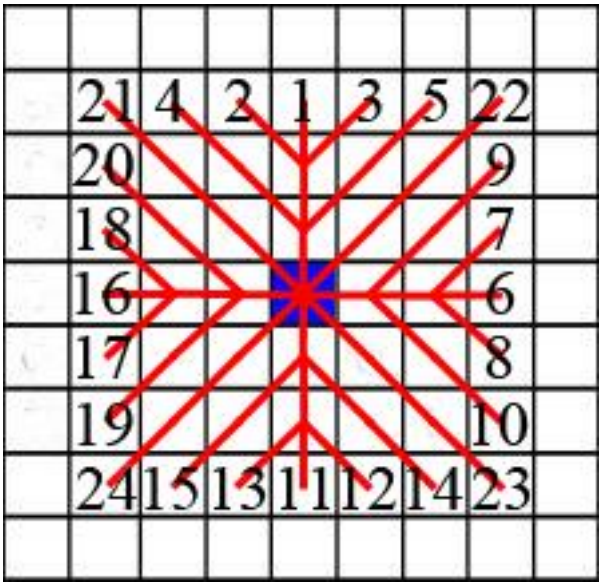

Рис. 1.8. Третий этап распространения волны

Продолжая этот процесс, узлы будут просматриваться таким образом, что красные линии будут образовывать кратчайшие пути от каждой рассмотренной клетки до начальной.

Преимущество данного порядка обхода вершин графа заключается в том, что появляется возможность подсчитывать длину кратчайшего пути от узла, из которого начинается распространение волны, до каждой

просмотренной в процессе распространения волны клетки. При этом расчет длины происходит немедленно при рассмотрении очередной ячейки, не требуя пересчета в дальнейшем. Найденные длины используются для построения искомого кратчайшего пути после встречи волн, что решает вышеупомянутую проблему разработки модификации алгоритма. Данный порядок обхода вершин графа был предложен в моей статье [3].

Будем полагать, что каждый узел имеет атрибут «Длина пути», в котором указывается длина кратчайшего пути до него от начальной вершины, если этот узел был просмотрен при распространении прямой волны, или от конечной вершины, если этот узел был рассмотрен при распространении встречной волны. Перед началом работы алгоритма будем считать, что атрибуты «Длина пути» у всех узлов пусты, то есть в них ничего не записано. Для удобства будем полагать, что фронт прямой волны, фронт встречной волны, новый фронт прямой волны, новый фронт встречной волны и путь представляют собой структуры данных, аналогичные спискам.

Алгоритм разработанной модификации описан ниже по пунктам.

1. Присвоить атрибутам «Длина пути» начальной и целевой клеток значение 0, добавить эти клетки в новые фронты прямой и встречной волн соответственно.
2. Добавить клетки, содержащиеся в новом фронте прямой волны, во фронт прямой волны в порядке их добавления в новый фронт прямой волны. Удалить все клетки из нового фронта прямой волны.
3. Добавить клетки, содержащиеся в новом фронте встречной волны, во фронт встречной волны в порядке их добавления в новый фронт встречной волны. Удалить все клетки из нового фронта встречной волны.
4. Для каждой клетки, находящейся во фронте прямой волны, в порядке их добавления во фронт прямой волны повторить следующие действия:

   а) Проверить, не находится ли какой-либо сосед рассматриваемой клетки, до которого может быть проложен путь из рассматриваемой клетки

напрямую, во фронте встречной волны; если находится, то немедленно перейти к пункту 7.

б) Определить таких ортогональных соседей рассматриваемой клетки, для которых одновременно выполняются следующие условия: до этих соседей может быть проложен путь из рассматриваемой клетки напрямую и у этих соседей атрибут «Длина пути» является пустым.

в) Добавить этих соседей в новый фронт прямой волны.

г) Присвоить атрибутам «Длина пути» этих соседей значение атрибута «Длина пути» рассматриваемой клетки, увеличенное на расстояние от рассматриваемой клетки до ортогонального соседа.

д) Проделать аналогичные действия с диагональными соседями рассматриваемой клетки, увеличивая значение атрибута «Длина пути» рассматриваемой клетки на расстояние от рассматриваемой клетки до диагонального соседа.

5. Для каждой клетки, находящейся во фронте встречной волны, в порядке их добавления во фронт встречной волны повторить следующие действия:

а) Проверить, не находится ли какой-либо сосед рассматриваемой клетки, до которого может быть проложен путь из рассматриваемой клетки напрямую, в новом фронте прямой волны; если находится, то немедленно перейти к пункту 9.

б) Определить таких ортогональных соседей рассматриваемой клетки, для которых одновременно выполняются следующие условия: до этих соседей может быть проложен путь из рассматриваемой клетки напрямую и у этих соседей атрибут «Длина пути» является пустым.

в) Добавить этих соседей в новый фронт встречной волны.

г) Присвоить атрибутам «Длина пути» этих соседей значение атрибута «Длина пути» рассматриваемой клетки, увеличенное на расстояние от рассматриваемой клетки до ортогонального соседа.

д) Проделать аналогичные действия с диагональными соседями рассматриваемой клетки, увеличивая значение атрибута «Длина пути» рассматриваемой клетки на расстояние от рассматриваемой клетки до диагонального соседа.

6. Если новые фронты прямой и встречной волн не являются пустыми – удалить все клетки из фронта прямой волны и из фронта встречной волны, перейти к пункту 2; иначе – завершить выполнение алгоритма, пропустив следующие четыре пункта.
7. Построить путь следующим образом:

   а) Из всех клеток, находящихся во фронте прямой волны, выбрать те, у которых существует хотя бы один сосед, для которого одновременно выполняются следующие условия: этот сосед находится во фронте встречной волны и до него может быть проложен путь из рассматриваемой клетки напрямую.

   б) Из выбранных клеток найти ту, у которой значение атрибута «Длина пути» минимально, назвать ее центральным узлом.

   в) Добавить центральный узел в начало пути, сделать его текущей клеткой.

   г) Найти таких соседей текущей клетки, для которых одновременно выполнены следующие условия: до этих соседей может быть проложен путь из текущей клетки напрямую и эти соседи содержались во фронте прямой волны. Выбрать из таких соседей клетку, у которой значение атрибута «Длина пути» минимально, добавить ее в начало пути, сделать ее текущей клеткой.

   д) Если начальная клетка не добавлена в начало пути – перейти к предыдущему подпункту.

   е) Сделать центральный узел текущей клеткой.

   ж) Найти таких соседей текущей клетки, для которых одновременно выполнены следующие условия: до этих соседей может быть проложен путь из текущей клетки напрямую и эти соседи содержались или

содержатся во фронте встречной волны. Выбрать из таких соседей клетку, у которой значение атрибута «Длина пути» минимально, добавить ее в конец пути, сделать ее текущей клеткой.

з) Если целевая клетка не добавлена в конец пути – перейти к предыдущему подпункту.

8. Вернуть путь в качестве результата работы алгоритма и завершить выполнение алгоритма, пропустив следующие два пункта.
9. Построить путь следующим образом:

а) Из всех клеток, находящихся во фронте встречной волны, выбрать те, у которых существует хотя бы один сосед, для которого одновременно выполняются следующие условия: этот сосед находится в новом фронте прямой волны и до него может быть проложен путь из рассматриваемой клетки напрямую.

б) Из выбранных клеток найти ту, у которой значение атрибута «Длина пути» минимально, назвать ее центральным узлом.

в) Добавить центральный узел в начало пути, сделать его текущей клеткой.

г) Найти таких соседей текущей клетки, для которых одновременно выполнены следующие условия: до этих соседей может быть проложен путь из текущей клетки напрямую и эти соседи содержались или содержатся во фронте прямой волны или содержатся в новом фронте прямой волны. Выбрать из таких соседей клетку, у которой значение атрибута «Длина пути» минимально, добавить ее в начало пути, сделать ее текущей клеткой.

д) Если начальная клетка не добавлена в начало пути – перейти к предыдущему подпункту.

е) Сделать центральный узел текущей клеткой.

ж) Найти таких соседей текущей клетки, для которых одновременно выполнены следующие условия: до этих соседей может быть проложен путь из текущей клетки напрямую и эти соседи содержались во фронте

встречной волны. Выбрать из таких соседей клетку, у которой значение атрибута «Длина пути» минимально, добавить ее в конец пути, сделать ее текущей клеткой.

з) Если целевая клетка не добавлена в конец пути – перейти к предыдущему подпункту.

10. Вернуть путь в качестве результата работы алгоритма и завершить выполнение алгоритма.

Разработанная модификация алгоритма волновой трассировки была представлена в моей статье [4].

1.5. Практическое сравнение эффективности разработанной модификации и алгоритма волновой трассировки по быстродействию

Алгоритм разработанной модификации был реализован на языке программирования C++. В каждом эксперименте созданная модификация сравнивается с алгоритмом волновой трассировки при тех же условиях, что и в соответствующих экспериментах с алгоритмом A* и алгоритмом волновой трассировки.

Изображение результата первого эксперимента с разработанной модификацией такое же, как на рис. 1.1, результат второго эксперимента показан на рис. 1.9, а результат третьего – на рис. 1.10. Рисунки иллюстрируют карту, начальную и целевую клетки, отмеченные синим и красным цветами соответственно, и найденный кратчайший путь между ними, который обозначается зеленым цветом. Неподвижные препятствия выделены черным цветом. Результаты всех экспериментов приведены в таблице 1.2.

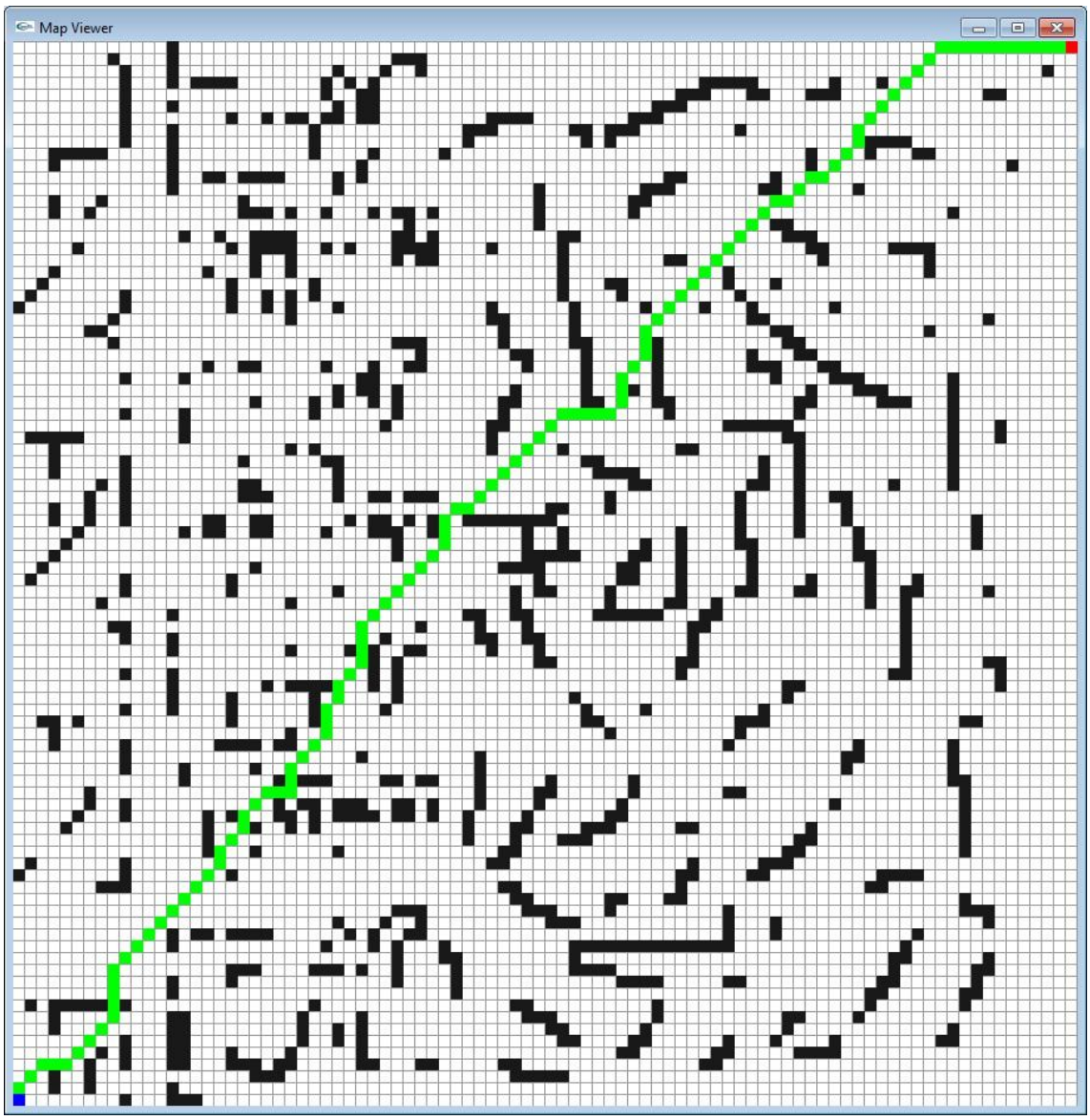

Рис. 1.9. Второй эксперимент с разработанной модификацией

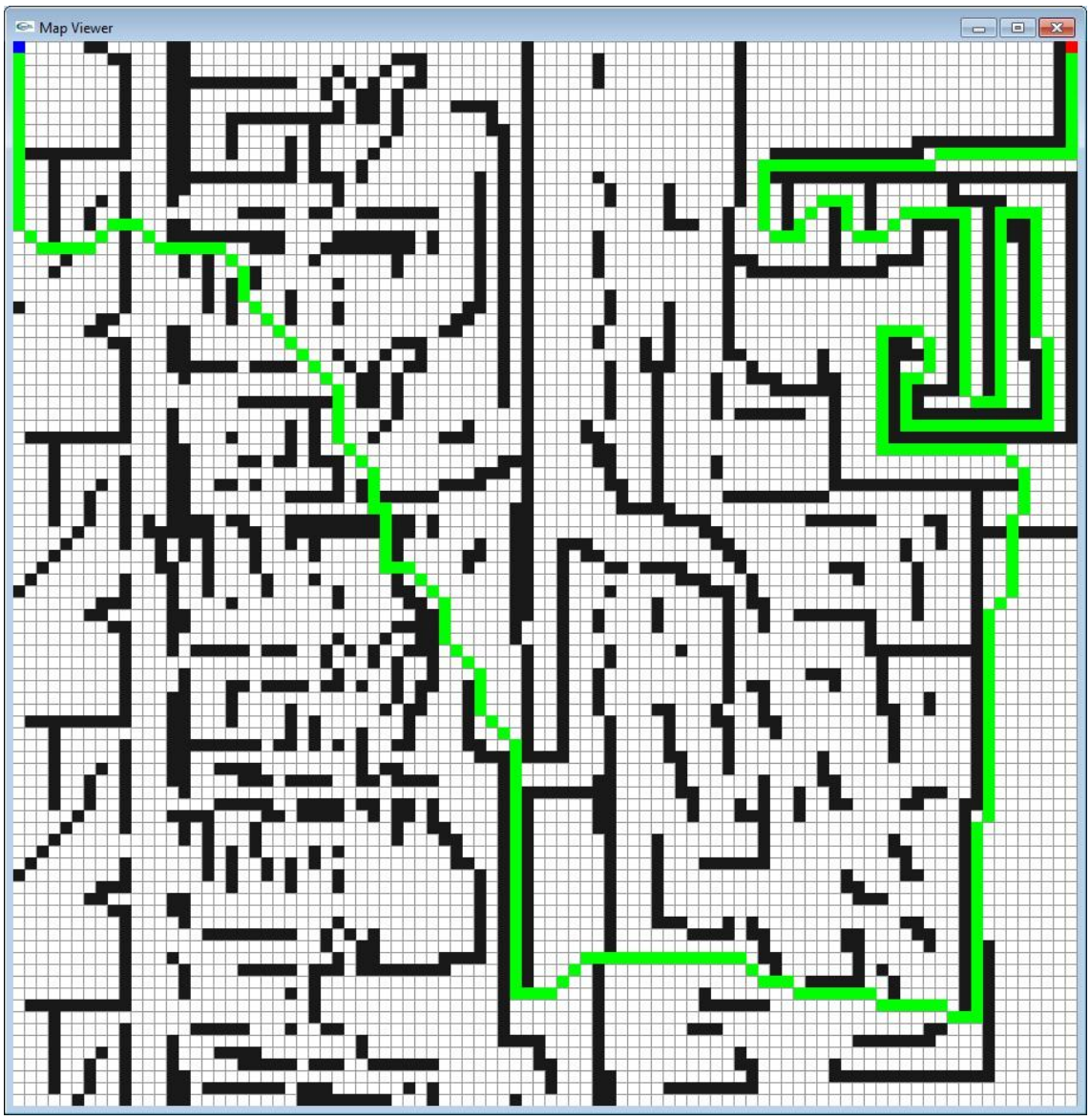

Рис. 1.10. Третий эксперимент с разработанной модификацией

Таблица 1.2 – Результаты практического сравнения разработанной модификации с алгоритмом волновой трассировки

	Время работы в первом эксперименте	Время работы во втором эксперименте	Время работы в третьем эксперименте
Разработанная модификация	1 мс	1 мс	2 мс
Алгоритм волновой трассировки	3 мс	3 мс	3 мс

Результаты экспериментов продемонстрировали превосходство разработанной модификации над алгоритмом волновой трассировки в среднем в два раза по скорости работы. Разработанная модификация предлагается в качестве решения проблемы глобального планирования пути в мультиагентной системе управления перемещениями.

Глава 2. Трассировка пути

В результате глобального планирования определяется кратчайший путь на графе в виде равномерной сетки, однако на открытых участках виртуальной среды зачастую полученный путь фактически не является кратчайшим. Пример такой ситуации проиллюстрирован на рис. 2.1.

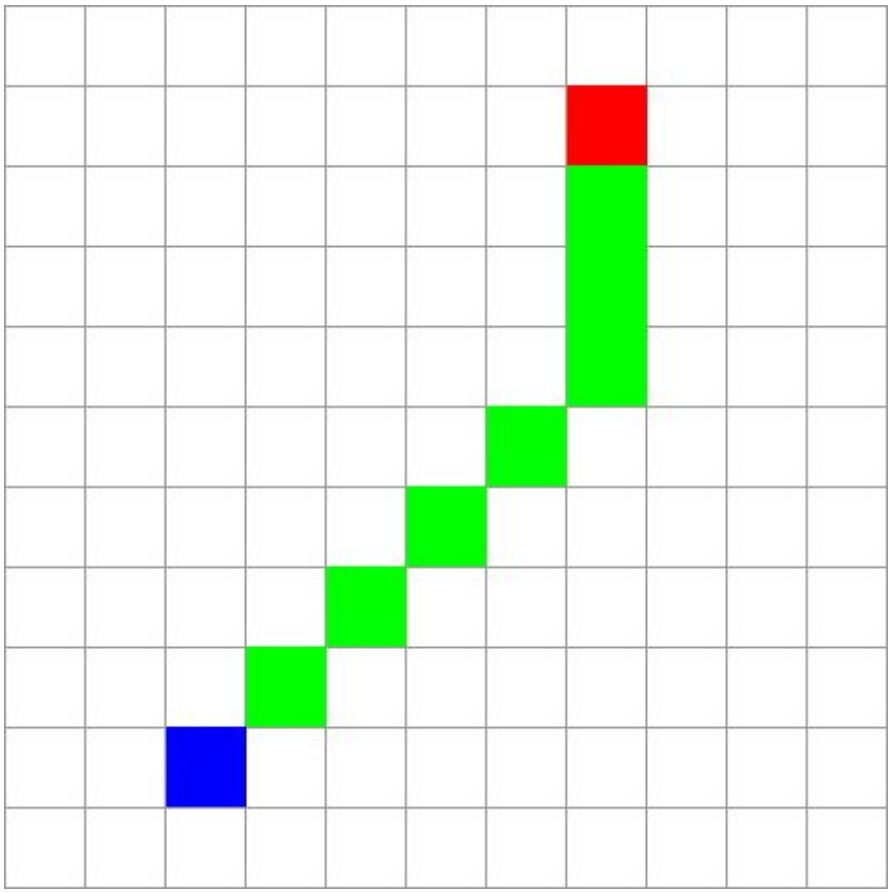

Рис. 2.1. Пример ситуации, при которой найденный с помощью глобального планирования кратчайший путь на графе в виде равномерной сетки фактически не является кратчайшим

По этой причине возникает необходимость трассировки пути – отслеживания возможности передвижения агента из своего текущего местоположения напрямую к каждой вершине графа, составляющей путь.

В рамках рассматриваемой в данной работе проблемы разработки мультиагентной системы управления перемещениями был разработан алгоритм трассировки пути, описание которого приведено ниже.

Будем проводить трассировку от положения агента, заданного координатами его центра (x_1, y_1), до вершины графа, имеющей координаты (x_2, y_2).

Введем в рассмотрение угловой коэффициент k прямой, проходящей через точки с координатами (x_1, y_1) и (x_2, y_2), и число b. Сравним величины $|x_2 - x_1|$ и $|y_2 - y_1|$. Если первая из них больше второй, то выполним поворот графа вместе с системой координат так, чтобы ось абсцисс была направлена строго вверх, вычислим значения k и b по формулам

$$k = \frac{y_2 - y_1}{x_2 - x_1},$$

$$b = y_1 - kx_1,$$

и будем определять значения y в точках пересечения рассматриваемой прямой с горизонтальными линиями, образующими равномерную сетку и расположенными между положением агента и рассматриваемой вершиной графа:

$$y = kx + b,$$

где x – это абсцисса точки, через которую проходит соответствующая горизонтальная прямая. В противном случае поворот не осуществляется, значения k и b рассчитываются по формулам

$$k = \frac{x_2 - x_1}{y_2 - y_1},$$

$$b = x_1 - ky_1,$$

и по аналогии с предыдущим случаем определяется величина x в точках пересечения:

$$x = ky + b,$$

где y – это ордината точки, через которую проходит соответствующая горизонтальная линия.

Далее для более краткого и удобного описания алгоритма трассировки пути проведем дальнейшие преобразования с целью свести несколько случаев взаимного расположения агента и вершины графа к одному. Если

рассматриваемая вершина графа оказалась ниже агента, то отразим граф вместе с системой координат по вертикали. Если в результате вершина графа находится слева от агента, то выполним отражение по горизонтали. После этого агент должен быть не сверху и не справа относительно рассматриваемой вершины.

Проделав вышеуказанные операции, будем двигаться по прямой от положения агента до рассматриваемой вершины графа, определяя точки пересечения с горизонтальными линиями так, как было указано ранее.

Рассмотрим случай, при котором выполняется либо

$$|x_2 - x_1| > 2|y_2 - y_1|,$$

либо

$$|y_2 - y_1| > 2|x_2 - x_1|.$$

При пересечении первой горизонтальной линии будем проверять, являются ли выделенные на рис. 2.2 клетки проходимыми, в том случае, если точка пересечения прямой с горизонтальной линией расположена ближе к правой относительно точки вертикальной линии, образующей равномерную сетку. Этот случай назовем случаем 1. Рассматриваемая прямая и точка пересечения также изображены на рис. 2.2. Иначе (случай 2) проверяются клетки, отмеченные на рис. 2.3.

Для этих двух случаев при пересечении остальных горизонтальных линий, вплоть до предпоследней, проверяемые клетки выделены на рис. 2.4 (случай 3) и рис. 2.5 (случай 4) соответственно.

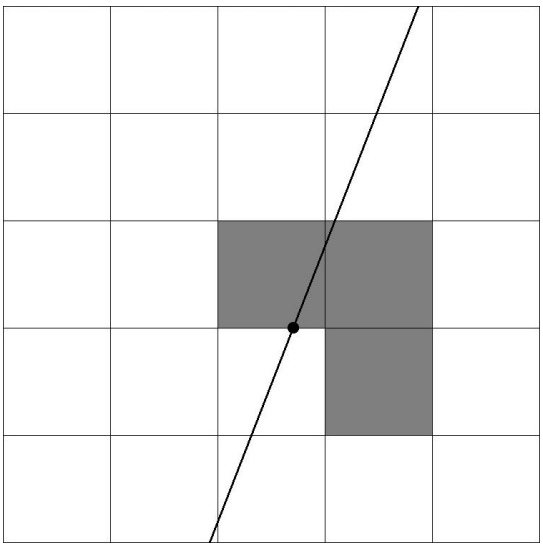

Рис. 2.2. Случай 1

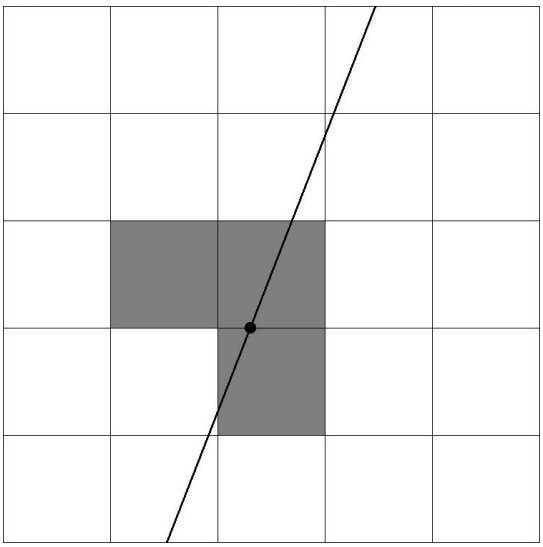

Рис. 2.3. Случай 2

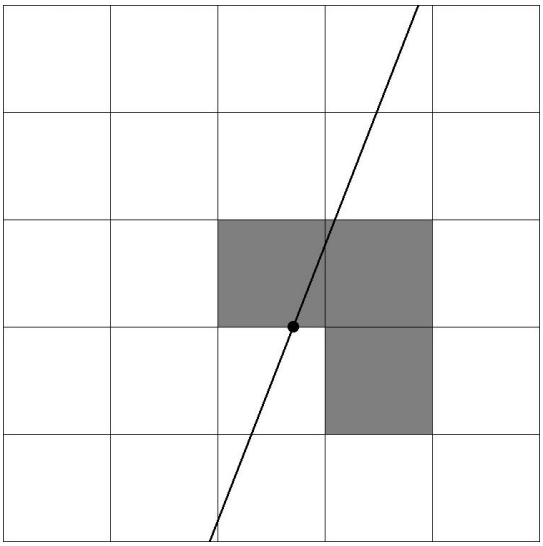

Рис. 2.4. Случай 3

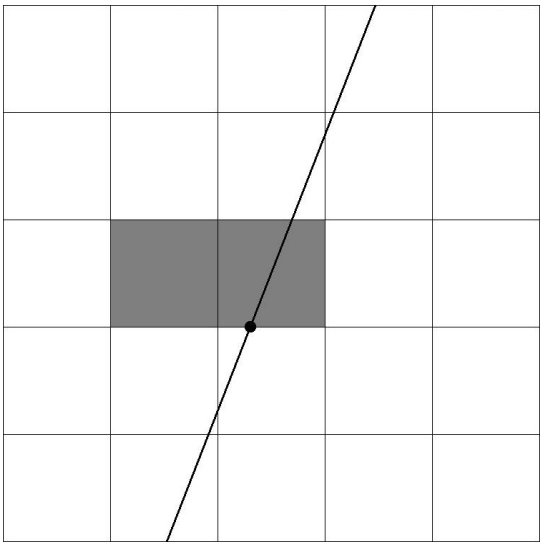

Рис. 2.5. Случай 4

В конце проверим левого соседа соответствующей рассматриваемой вершине клетки на проходимость.

Для случая, когда выполняются неравенства

$$|x_2 - x_1| \leqslant 2|y_2 - y_1|$$

и

$$|y_2 - y_1| \leqslant 2|x_2 - x_1|,$$

аналогичным образом ситуации с пересечением первой горизонтальной линии представлены на рис. 2.6 (случай 5) и рис. 2.7 (случай 6), а ситуации с пересечением остальных вплоть до предпоследней горизонтальной линии проиллюстрированы на рис. 2.8 (случай 7) и рис. 2.9 (случай 8).

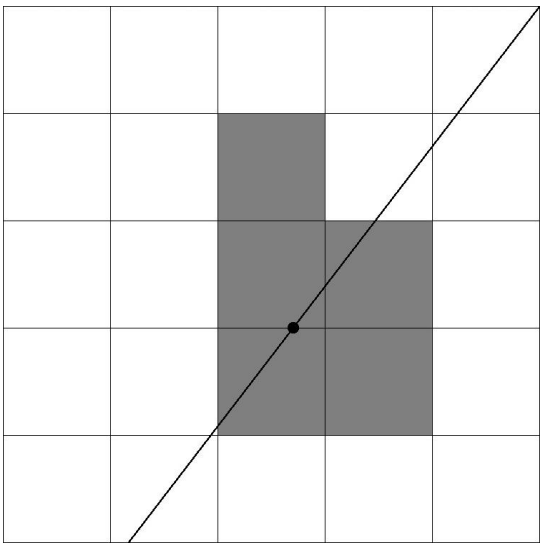

Рис. 2.6. Случай 5

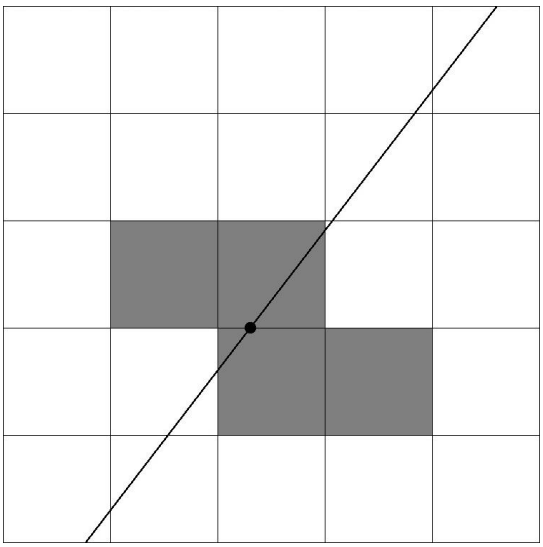

Рис. 2.7. Случай 6

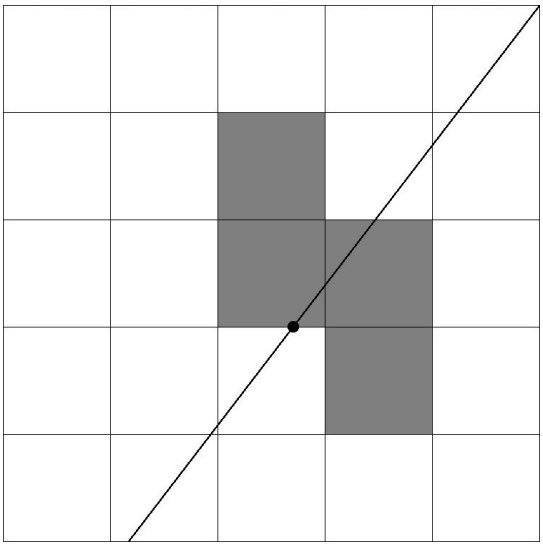

Рис. 2.8. Случай 7

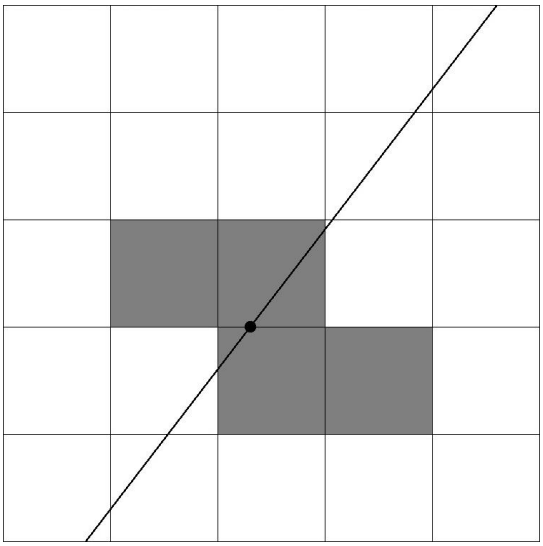

Рис. 2.9. Случай 8

В конце необходимо проверить левого и нижнего соседа соответствующей рассматриваемой вершине клетки.

Данный алгоритм трассировки пути для каждого агента успешно определяет возможность сократить путь, полученный на этапе глобального планирования, и двигаться при этом в достаточно свободном пространстве, обеспечивающем выполнение агентом необходимых маневров. При этом данный алгоритм совершает сравнительно небольшое количество вычислительных операций, что крайне положительно отразилось на результатах его испытаний на практике.

Глава 3. Моделирование движения агентов в виртуальной среде и предотвращение столкновений при локальном взаимодействии между агентами

При движении агента по найденному глобальному кратчайшему пути требуется обеспечить его перемещение без столкновений с другими агентами. Существует множество подходов для решения этой проблемы, которые будут рассмотрены в этой главе.

Для моделирования движения агентов в разрабатываемой мультиагентной системе управления перемещениями используется метод моделирования поведения, заключающегося в изменении направления движения с помощью управляющих воздействий (steering behaviors) [5], который характеризуется простотой реализации и невысокой требовательностью к вычислительным ресурсам, что очень важно при управлении большим количеством агентов в реальном времени.

В соответствии с данным подходом определяется желаемая скорость $\overrightarrow{v_{des}}$ агента, с которой он может достичь целевой точки, заданной радиус-вектором $\overrightarrow{r_p}$:

$$\overrightarrow{v_{des}} = \frac{\overrightarrow{r_p} - \vec{r}}{\left|\overrightarrow{r_p} - \vec{r}\right|} v_{max} ,$$

где $\vec{r}$ – радиус-вектор, задающий центр агента, v_{max} – максимальная скорость агента. Управляющее воздействие $\overrightarrow{F_{steer}}$ для движения агента к целевой точке вычисляется с использованием текущей скорости $\vec{v}$ агента и желаемой скорости $\overrightarrow{v_{des}}$:

$$\overrightarrow{F_{steer}} = \overrightarrow{v_{des}} - \vec{v}.$$

Ускорение $\vec{a}$ агента рассчитывается по формуле

$$\vec{a} = \frac{\overrightarrow{F_{steer}}}{\left|\overrightarrow{F_{steer}}\right|} a_{max} ,$$

где a_{max} – максимальное ускорение агента. После этого обновляются скорость $\overrightarrow{v_{new}}$ и положение $\overrightarrow{r_{new}}$ агента:

$$\overrightarrow{v_{new}} = \vec{v} + \vec{a}\,\Delta t,$$
$$\overrightarrow{r_{new}} = \vec{r} + \overrightarrow{v_{new}}\,\Delta t.$$

В мультиагентной системе управления перемещениями агенты проходят через вершины, составляющие их пути. В целях моделирования плавных поворотов и получения гладких траекторий агентов, точка (за исключением конечной вершины пути агента) считается достигнутой агентом, если расстояние между его центром и данной точкой меньше увеличенного в два раза радиуса агента.

3.1. Результаты исследования подходов к предотвращению столкновений

Steering behaviors. Данный подход был описан в начале этой главы. В контексте избегания столкновений одного агента с другими выбираются соответствующие управляющие воздействия, направленные от центров динамических препятствий и уводящие агента в сторону. Несмотря на указанные плюсы метода, в некоторых случаях возможны столкновения агентов из-за неверно определенных управляющих воздействий, что особенно свойственно ситуациям с большим количеством агентов.

Velocity obstacle (VO). Пусть агенты A и B представляют собой круги радиусами R_A и R_B с центрами в точках, заданных радиус-векторами $\overrightarrow{r_A}$ и $\overrightarrow{r_B}$ соответственно. Множество $VO_B^A(\overrightarrow{v_B})$ – это совокупность таких скоростей $\overrightarrow{v_A}$ агента A, при движении с которыми в некоторый момент времени агенты A и B столкнутся при условии, что агент B будет двигаться со скоростью $\overrightarrow{v_B}$ [6]:

$$VO_B^A(\overrightarrow{v_B}) = \{\overrightarrow{v_A} \mid \exists t > 0 : (\overrightarrow{v_A} - \overrightarrow{v_B})t \in D(\overrightarrow{r_B} - \overrightarrow{r_A},\ R_A + R_B)\},$$

где $D(\vec{r}, R)$ – круг радиусом R с центром в точке, заданной радиус-вектором $\vec{r}$.

Для предотвращения столкновений скорость агента A выбирается так,

чтобы она не принадлежала $VO_B^A(\overrightarrow{v_B})$. Аналогично выбирается скорость агента B при использовании $VO_A^B(\overrightarrow{v_A})$.

Подход позволяет точно определять скорости, которые приведут к столкновению, и успешно огибать неподвижных агентов, однако использование VO для избегания столкновений с движущимися агентами может привести к резким и очень частым изменениям направления движения агентов, напоминающим колебательное движение.

Reciprocal velocity obstacle (RVO). Подход базируется на применении VO и решает проблему, связанную с колебанием агентов [6].

RVO определяется следующим образом:

$$RVO_B^A(\overrightarrow{v_B}, \overrightarrow{v_A}) = \{\vec{v} \mid 2\vec{v} - \overrightarrow{v_A} \in VO_B^A(\overrightarrow{v_B})\}.$$

Использование RVO позволяет агентам разделить между собой усилия при огибании друг друга и разрешает все ситуации, касающиеся предотвращения столкновений между двумя агентами. К сожалению, в некоторых случаях, например при угрозе столкновения сразу нескольких агентов, снова возможны резкие и частые изменения направления движения агентов, но теперь уже по причине того, что агенты не могут достичь согласия между собой в том, с какой стороны им следует друг друга обходить.

Hybrid reciprocal velocity obstacle (HRVO). В данном подходе решена проблема RVO, связанная с недоговоренностью о выборе стороны обхода, при помощи комбинирования VO и RVO [7].

Недостатком подхода является большая требовательность к вычислительным ресурсам по сравнению с предыдущими подходами.

Optimal reciprocal collision avoidance (ORCA). Подход является альтернативным решением проблемы выбора стороны обхода и заключается в построении полуплоскости, состоящей из скоростей, при движении с которыми агенты смогут обходить друг друга [8].

Данный подход превосходит HRVO в быстродействии, однако количество столкновений агентов при применении ORCA больше, чем при

использовании HRVO.

3.2. Разработка подхода к предотвращению столкновений при локальном взаимодействии между агентами

Анализ подходов, результаты которого были продемонстрированы в предыдущем параграфе, показал, что они не лишены недостатков. Одни лишь частично решают проблему предотвращения столкновений (steering behaviors, VO, RVO). Другие же требуют достаточно большого числа вычислительных операций (HRVO, ORCA). Поэтому в данной работе предложен иной подход к предотвращению столкновений при локальном взаимодействии между агентами, использующий результаты исследования существующих методов.

В разработанном подходе при угрозе столкновения между агентами принимается решение, соответствующее определенной ситуации в виртуальной среде. При этом данное решение принимается однократно, в отличие от рассмотренных подходов, где одинаково большое количество вычислений совершается на каждой итерации. Кроме того, в этих подходах предполагается постоянное движение агентов без остановок, в то время как в разработанном подходе предлагается останавливать агентов в случаях, когда более целесообразно пропустить других агентов.

Приведем описание разработанного подхода. В рамках этого подхода вводится понятие блокирующей области – зоны в виде круга, в которой могут находиться только определенные агенты, а остальные, если данная зона расположена у них на пути, останавливаются и стоят на месте до тех пор, пока им не будет разрешено двигаться дальше. В соответствии с разработанным подходом два агента друг для друга могут являться напарниками. Понятие «напарник» используется лишь в качестве обозначения для описания подхода, не требуя соответствующего строгого определения.

Если модуль разности координат любой пары агентов и по оси абсцисс,

и по оси ординат меньше увеличенного в шесть раз радиуса агента и ни один агент из данной пары не является напарником для любого другого агента, то необходимо проверить, произойдет ли столкновение между этой парой агентов. Проверка на угрозу столкновения осуществляется с помощью подхода VO. Назовем агентов данной пары агентами A и B. Пусть центры этих агентов заданы радиус-векторами $\overrightarrow{r_A}$ и $\overrightarrow{r_B}$ соответственно, $\overrightarrow{v_A}$ и $\overrightarrow{v_B}$ – это соответствующие векторы скоростей агентов. Если угроза столкновения выявлена, то агенты друг для друга становятся напарниками, устанавливается блокирующая область радиусом, равным увеличенному в три раза радиусу агента, с центром в точке, расположенной посередине между центрами агентов A и B, находящихся в тех местах, где между ними произойдет столкновение (эти места определяются с использованием VO), и в каждом конкретном случае будет принято определенное решение для данной пары агентов. Рассмотрим эти случаи и соответствующие им принимаемые решения.

В первом случае ни один из агентов рассматриваемой пары не является неподвижным или останавливающимся, угол между векторами скоростей этих агентов не меньше 135 градусов, и нет ни одного агента из данной пары такого, что на полпути к напарнику существует вершина, в которой путь этого агента меняет свое направление. Агенты должны обойти друг друга, двигаясь к вспомогательным точкам. Для нахождения этих точек построим вектор $\overrightarrow{m}$ с началом в точке, заданной радиус-вектором $\overrightarrow{r_A}$, и концом в точке, заданной радиус-вектором $\overrightarrow{r_B}$. Определим сторону обхода с помощью косого произведения векторов. Если выполняется

$$(\overrightarrow{v_A} - \overrightarrow{v_B}) \wedge \overrightarrow{m} \geqslant 0,$$

то обход будет справа. Иначе – слева.

Определим вектор, ортогональный вектору $\overrightarrow{m}$. Если обход будет справа, то ортогональный вектор будет иметь координаты $(m_y, -m_x)$. В противном случае – $(-m_y, m_x)$. Придадим этому вектору длину, равную умноженному на коэффициент 3,8 радиусу агента, не изменяя направление

вектора, и назовем его вектором $\vec{s}$.

Получим радиус-вектор $\overrightarrow{r_q}$ по формуле

$$\overrightarrow{r_q} = \overrightarrow{r_A} + \vec{m} + \vec{s}.$$

Введем в рассмотрение вектор с началом в точке, заданной радиус-вектором $\overrightarrow{r_A}$, и концом в точке, заданной радиус-вектором $\overrightarrow{r_q}$. Изменим длину этого вектора, умножив ее на 0,65, и назовем его вектором $\vec{h}$.

Найдем радиус-вектор $\overrightarrow{r_{pA}}$, который задает вспомогательную точку для агента А:

$$\overrightarrow{r_{pA}} = \overrightarrow{r_A} + \vec{h}.$$

Вспомогательная точка для агента В задается радиус-вектором $\overrightarrow{r_{pB}}$:

$$\overrightarrow{r_{pB}} = \overrightarrow{r_B} - \vec{h}.$$

При достижении агентами своих вспомогательных точек они возвращаются к своим маршрутам и больше не будут являться друг для друга напарниками (пока снова не возникнет угроза столкновения между ними).

Рассмотрим второй случай, в котором ни один из агентов рассматриваемой пары не является неподвижным или останавливающимся, угол между векторами скоростей этих агентов не меньше 135 градусов и хотя бы для одного агента из данной пары на полпути к напарнику существует вершина, в которой путь этого агента меняет свое направление. В данной ситуации этот агент продолжает движение, а другой агент должен остановиться и уступить. Когда агент, которому разрешено движение, пройдет через вершину, в которой меняется направление пути, другому агенту разрешается начать движение, агенты этой пары больше не будут являться друг для друга напарниками (пока снова не возникнет угроза столкновения между ними).

В третьем случае ни один из агентов рассматриваемой пары не является неподвижным или останавливающимся, угол между векторами скоростей этих агентов меньше 135 градусов. Здесь один из агентов данной пары останавливается и уступает другому. Когда агент, которому разрешено

движение, пройдет через место, где он должен был быть в момент столкновения с другим агентом (место определяется при помощи VO), другому агенту разрешается начать движение, агенты этой пары больше не будут являться друг для друга напарниками (пока снова не возникнет угроза столкновения между ними).

Перейдем к четвертому случаю, в соответствии с которым один из агентов рассматриваемой пары является неподвижным или останавливающимся и либо угол между векторами скоростей этих агентов не меньше 135 градусов (в качестве вектора скорости неподвижного или останавливающегося агента используется скорость, с которой данный агент будет двигаться после начала движения), либо у неподвижного агента отсутствует задача о передвижении. Требуется запретить неподвижному или останавливающемуся агенту начинать движение. Для другого агента определяется вспомогательная точка по такому же принципу, как и в первом случае, с тем отличием, что непосредственно перед получением вектора $\vec{h}$ изменение длины вектора не происходит. При достижении агентом своей вспомогательной точки он возвращается к своему маршруту, другому агенту разрешается начать движение, агенты этой пары больше не будут являться друг для друга напарниками (пока снова не возникнет угроза столкновения между ними). В случае если для движущегося агента существует вершина, занятая неподвижным агентом, в которой путь движущегося агента изменяет свое направление на 90 градусов, сторона обхода совпадает с той стороной, в которую поворачивает путь; при этом добавляется еще одна вспомогательная точка, расположенная на пути, которого придерживается движущийся агент, к вершине на расстоянии от вершины, равном увеличенному в шесть раз радиусу агента.

В пятом случае один из агентов рассматриваемой пары является неподвижным или останавливающимся, угол между векторами скоростей этих агентов меньше 135 градусов (в качестве вектора скорости неподвижного или останавливающегося агента используется скорость,

с которой данный агент будет двигаться после начала движения). В этой ситуации агенты не становятся друг для друга напарниками, движущийся агент останавливается и стоит до тех пор, пока другой агент не покинет блокирующую область, установленную по причине угрозы столкновения между агентами из данной пары.

В каждом из указанных случаев, когда оба агента покинут свою блокирующую область, она удаляется, при этом другим агентам, остановленным из-за этой области, разрешается начать движение, если оно не запрещено в связи с другими предусмотренными данным подходом условиями.

Пример использования разработанного подхода к предотвращению столкновений при локальном взаимодействии между агентами представлен на рис. 3.1. В данном примере агенты A и B обходят друг друга, в то время как агент C ждет, пока другие агенты не освободят ему путь. Агенты изображены с помощью оранжевых кругов, траектории движения агентов A и B обозначены черными линиями.

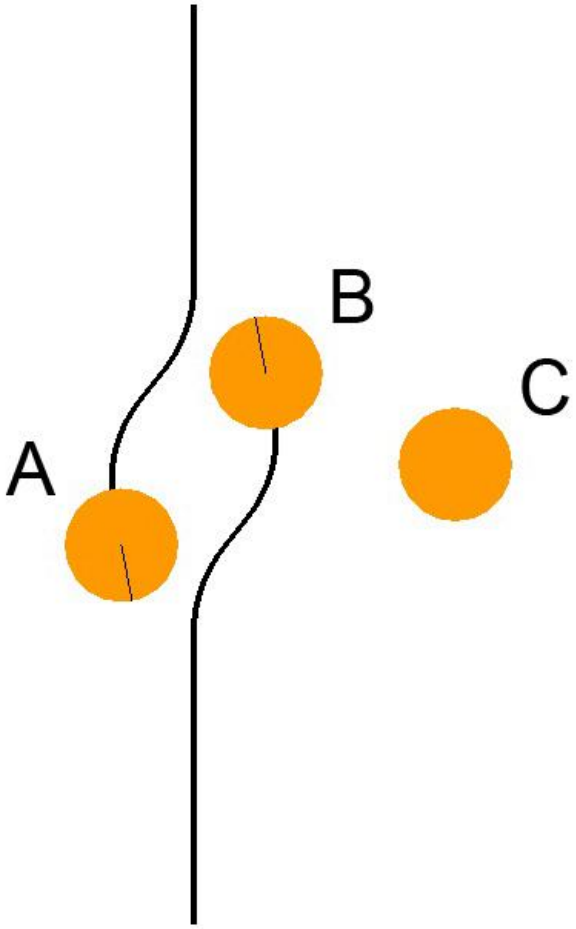

Рис. 3.1. Пример использования разработанного подхода к предотвращению столкновений при локальном взаимодействии между агентами

Выводы

В данной работе в рамках разработки мультиагентной системы управления перемещениями в виртуальной среде были решены проблемы глобального планирования пути, трассировки пути, а также предотвращения столкновений при локальном взаимодействии между агентами.

Проведено практическое сравнение эффективности алгоритма A* и алгоритма волновой трассировки по быстродействию, в ходе которого было установлено, что наиболее быстрым по скорости работы является алгоритм волновой трассировки. На основе этого алгоритма разработана его модификация, которая в среднем в два раза превзошла эффективность алгоритма волновой трассировки по быстродействию.

Разработан алгоритм трассировки пути, который для каждого агента успешно определяет возможность сократить путь, полученный на этапе глобального планирования, и двигаться при этом в достаточно свободном пространстве, обеспечивающем выполнение агентом необходимых маневров.

Проанализированы подходы, предназначенные для предотвращения столкновений между агентами. При использовании результатов исследования этих подходов разработан подход к предотвращению столкновений при локальном взаимодействии между агентами, в котором при угрозе столкновения между агентами принимается решение, соответствующее определенной ситуации в виртуальной среде. При этом данное решение принимается однократно, в отличие от рассмотренных подходов, где одинаково большое количество вычислений совершается на каждой итерации. Кроме того, в этих подходах предполагается постоянное движение агентов без остановок, в то время как в разработанном подходе предлагается останавливать агентов в случаях, когда более целесообразно пропустить других агентов.

Совокупность полученных в данной работе решений образует решение поставленной в работе задачи.

Заключение

Существует множество способов создания мультиагентных систем, каждый из которых требует досконального изучения. Различные комбинации, составленные из множества методов решения указанных в работе проблем, дают разные результаты, которые в совокупности позволяют найти лучший подход к проектированию мультиагентных систем.

В данной работе в рамках разработки мультиагентной системы управления перемещениями в виртуальной среде были рассмотрены и решены проблемы глобального планирования пути, трассировки пути, а также предотвращения столкновений при локальном взаимодействии между агентами.

Проведено практическое сравнение эффективности алгоритма A* и алгоритма волновой трассировки по быстродействию. Разработана модификация алгоритма волновой трассировки, которая в среднем в два раза превзошла эффективность алгоритма волновой трассировки по быстродействию.

Разработан алгоритм трассировки пути, который для каждого агента успешно определяет возможность сократить путь, полученный на этапе глобального планирования, и двигаться при этом в достаточно свободном пространстве, обеспечивающем выполнение агентом необходимых маневров.

Проанализированы подходы, предназначенные для предотвращения столкновений между агентами. Разработан подход к предотвращению столкновений при локальном взаимодействии между агентами.

В результате выполненной работы получены алгоритмы и подходы, которые в совокупности решили поставленную в данной работе задачу.

Список литературы

1. Лорьер Ж.-Л. Системы искусственного интеллекта / пер. с франц. М.: Мир, 1991. 568 с.
2. Кормен Т. Х., Лейзерсон Ч. И., Ривест Р. Л., Штайн К. Алгоритмы: построение и анализ / пер. с англ. 3-е изд. М.: Вильямс, 2013. 1328 с.
3. Михайлов И. Е. Порядок обхода вершин графа в алгоритме волновой трассировки (алгоритме Ли) // Наука, техника и образование, 2016. № 2 (20). С. 9–11.
4. Михайлов И. Е. Разработка модификации алгоритма волновой трассировки (алгоритма Ли) // Наука, техника и образование, 2016. № 3 (21). С. 60–62.
5. Reynolds C. W. Steering Behaviors For Autonomous Characters [Электронный ресурс] // Reynolds Engineering & Design. URL: http://www.red3d.com/cwr/papers/1999/gdc99steer.pdf (дата обращения: 12.03.2016).
6. Van den Berg J., Lin M., Manocha D. Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation // IEEE International Conference on Robotics and Automation, 2008. P. 1928–1935.
7. Snape J., van den Berg J., Guy S. J., Manocha D. The Hybrid Reciprocal Velocity Obstacle // IEEE Transactions on Robotics, 2011. Vol. 27(4). P. 696–706.
8. Van den Berg J., Guy S. J., Lin M., Manocha D. Reciprocal n-body collision avoidance // Robotics Research: The 14th International Symposium ISRR. Berlin, Germany: Springer, 2011. P. 3–19.

Приложение А. Структура реализованной мультиагентной системы управления перемещениями в виртуальной среде

Мультиагентная система управления перемещениями состоит из пяти модулей, которые схематично представлены на рис. А.1.

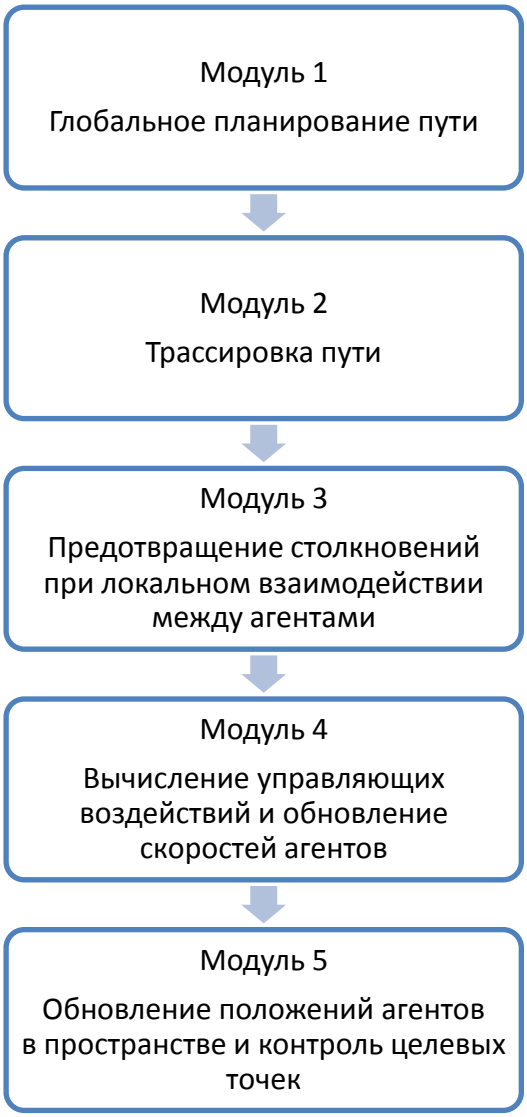

Рис. А.1. Модули мультиагентной системы управления перемещениями в виртуальной среде

Поочередно в каждом модуле, начиная с первого, проводятся определенные операции. После выполнения операций пятого модуля все повторяется сначала.

В первом модуле содержится очередь из идентификаторов агентов, для

которых осуществляется глобальное планирование пути с помощью разработанной модификации алгоритма волновой трассировки.

Во втором модуле осуществляется трассировка пути. Для каждого агента проверяется ограниченное количество вершин графа, составляющих путь, для экономии времени вычислений на текущей итерации.

Третий модуль предназначен для предотвращения столкновений при локальном взаимодействии между агентами. В данном модуле проверяется необходимость принятия решения для каждой пары агентов и при необходимости принимается соответствующее решение.

В четвертом модуле хранятся идентификаторы агентов, для которых вычисляются управляющие воздействия и обновляются скорости. В целях быстродействия хранение идентификаторов обеспечивается только для тех агентов, которым необходимо обновление скорости.

Аналогично организован последний модуль, в котором обновляются положения агентов в пространстве. В этом модуле также проверяется удаленность агентов от их целевых точек; в случае достижения агентом точки, для него задается новая цель.