УДК 681.5

ТРАССИРОВКА МАРШРУТА ДВИЖЕНИЯ РОБОТА ПРИ УСЛОВИИ ОБХОДА ПРЕПЯТСТВИЙ

Трунова Т.О., студент, кафедра МСТ ХНУРЭ
Табакова И.С., ассистент, кафедра МСТ ХНУРЭ

Аннотация. *Проведен анализ существующих способов расчета траектории перемещения по плоскости работа с учетом преград в виде прямоугольных фигур, многогранников принадлежащих плоскости.*

Ключевые слова: *ТРАССИРОВКА, ТОЧКА ЦЕЛИ, ТОЧКА СТАРТА, НАПРАВЛЕНИЕ ОБХОДА, ПРЕПЯТСТВИЯ.*

Одна из основных функции робота состоит в выполнении движения в заданную точку в зависимости от геометрических форм препятствий. Однако не всегда трассировка ставит перед собой цель найти кратчайший путь; зачастую это просто невозможно. Программирование данной функции упирается в проблему, которая сводится к алгоритму обхода препятствий [1].

В основу программ трассировки могут быть положены алгоритмы:

- с дискретным представлением зоны движения;
- основанные на теории графов;
- основанные на методах потенциальных полей;
- основанные на эвристических моделях;
- с представлением информации в виде уравнений.

Рассмотрим простейшую задачу трассировки: обход двумерных лабиринтов по заданной цифровой (растровой) карте местности в виде двумерной плоскости, разделенной на клетки (квадратные, треугольные или гексагональные), с отмеченными точкой старта (точки, из которой мы должны провести маршрут), точкой цели (точка, в которую мы должны провести маршрут) и препятствиями различной геометрической формы.

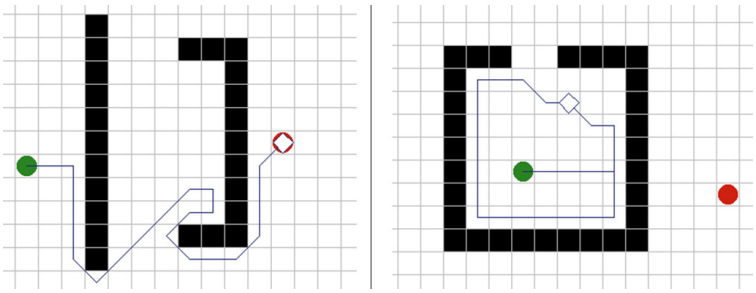
Вначале необходимо выбрать стратегию обхода препятствия. Мы можем пойти двумя путями: первый – прокладывать путь «на ходу», игнорируя препятствия до столкновения с ними; второй – заранее спланировать путь до начала перемещения.

Наиболее простые алгоритмы основаны на предположении, что препятствие можно «касаться рукой» и следовать его контуру, пока оно не будет обойдено. Первоначально выбираем направление для движения к цели. Движемся до столкновения с препятствием. При столкновении выбираем другое направление в соответствии со стратегией обхода, например:

- перемещение в случайном направлении. Делаем небольшое смещение в случайном направлении при встрече препятствия;

- трассировка вокруг препятствия. При встрече препятствия идем по его контуру до некоторого момента, определяемого эвристикой (например количество перемещений в одном направлении; остановить обход препятствия при возможности передвижения в направлении, которое было желаемым в начале трассировки).

Этот алгоритм хорошо работает с протяженными препятствиями (рис. 1, а), однако траектория далеко не кратчайшая. При этом показывает (рис. 1, б), что может возникнуть зацикливание, при неверном подборе стратегии.



а б

Рисунок 1 – Демонстрация работы алгоритмов перемещения в случайном направлении и перемещения вокруг препятствий

Можно методом случайного перебора найти обход, но тогда на достаточно сложной территории перемещение из точки цели в конечную точку может занять неопределенно долгое время, и потребовать много ресурсов.

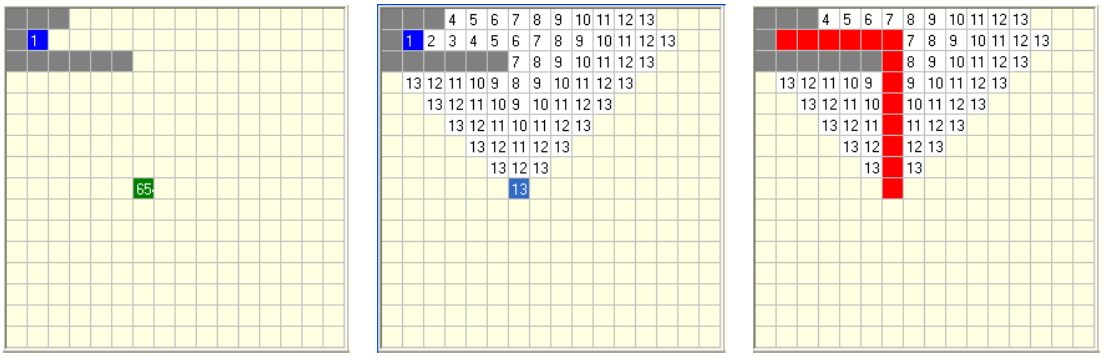
Заметим, что наш лабиринт можно представить в виде графа, соответственно, можно воспользоваться некоторыми известными алгоритмами на графах для предварительного планирования пути. Существуют достаточно простые, но эффективные алгоритмы поиска пути с обходом препятствий.

При реализации многих алгоритмов на графах возникает необходимость организовать систематический перебор вершин графа, при котором каждая вершина просматривается точно один раз. Такой перебор можно организовать двумя способами: поиском в глубину или поиском в ширину.

Рассмотрим алгоритм, называемый поиск в ширину (альтернативное название - волновой алгоритм). Волновой алгоритм является одним из самых уникальных алгоритмов трассировки. Он позволяет построить трассу (путь) между двумя элементами в любом лабиринте. Суть его в том, что для поиска пути используется принцип расходящихся на воде кругов от брошенного в воду камня. Работу алгоритма проиллюстрируем на примере лабиринта из пары стен. Точка старта выделена цифрой 1, точка финиша – 65 (рис. 2, а).

Переходим из клетки в клетку, считая длину пути от нее до точки начала расчета. В точку старта помещаем единицу. Это - начало пути. Препятствия (стены) заполнены некими заранее заданными значениями (любое число, заведомо отличимое от пустой клетки). На следующем шаге осматриваем соседние клетки, на которые можно перейти. Их не больше четырех: по одной клетке справа, слева, сверху и снизу. Перейти можно на поля, не занятые препятствиями и не пройденными на одном из предыдущих шагов. Если такие поля находятся среди

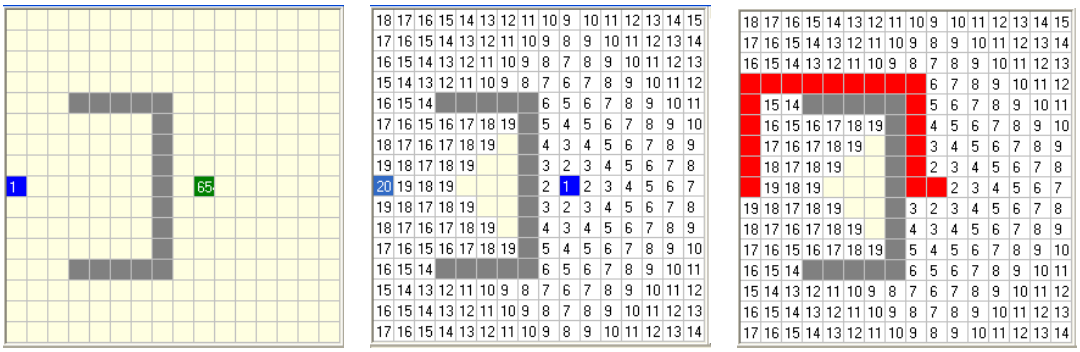
соседей клетки, мы пишем в них расстояние от исходной точки расчета. После этого поочередно для каждой изученной на предыдущем шаге клетки ищем незанятые соседние клетки, в которые заносим значение длины пути. Таким образом, с каждым шагом мы заполняем все большее и большее число клеток, расширяя границу изученной территории. Расчет заканчивается, когда мы приходим в точку цели (рис. 2, б, в). Как видно (рис. 2, в), часть территории остается неизученной, что говорит об эффективности алгоритма при использовании его для лабиринтов.



а б в

Рисунок 2 – Демонстрация работы волнового алгоритма

Эффективность работы алгоритма можно повысить еще больше, для этого применяется принцип обратной трассировки. Поиск пути ведется из точки финиша в точку старта. Это позволяет избежать трудностей, связанных с поиском способов определения тупиков и выхода из "карманов" и циклов, появляющихся в случае прямой трассировки из точки старта в точку финиша (рис. 3).



а б в

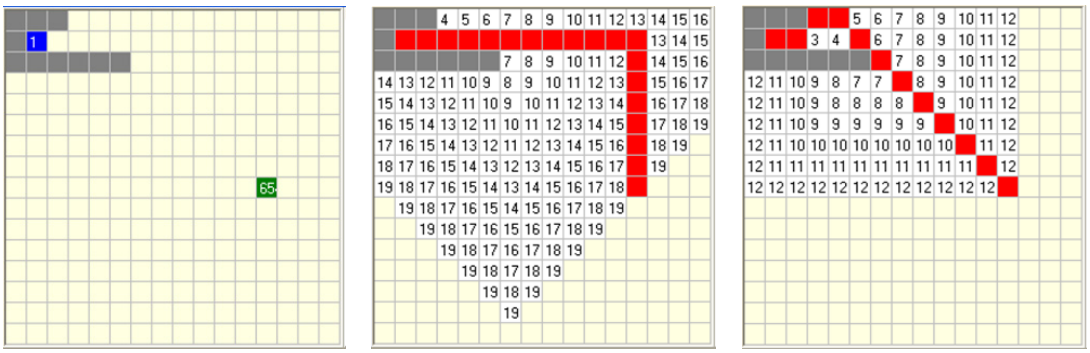
Рисунок 3 – Демонстрация работы волнового алгоритма обратной трассировки

Также возможно усовершенствование алгоритма, за счет расширения числа рассматриваемых им клеток-соседей учитывая угловые. Этот алгоритм называется волновым алгоритмом по 8 точкам (рис. 4, в) и он позволяет получить большую экономию расчетных ресурсов, что продемонстрировано на рис. 4.

Для совмещения достоинств волнового алгоритма прямой и обратной трассировки целесообразна применять двунаправленный поиск. Запускается две волны: из стартовой и целевой точек. Алгоритм работает до встречи двух волн.

Проанализировав волновой алгоритм и его модификации выделим рад преимуществ и недостатков при их реализации.

Преимущества: простота реализации; всегда находит кратчайший путь.



а б в

Рисунок 4 – Демонстрация работы волнового алгоритма прямой трассировки по 4 и 8 точкам

Недостатки: поиск идет равномерно во всех направлениях, вместо того, чтобы быть направленным к цели; не всегда все шаги равны (например, диагональные шаги должны быть длиннее ортогональных).

Рассмотрим алгоритм поиска в глубину. Основная идея алгоритма состоит в последовательном движении из заданной вершины вдоль одного из ребер вглубь графа до тех пор, пока не дойдем до вершины, из которой нельзя попасть ни в какую не просмотренную вершину. Такую вершину назовем обработанной. После этого возвращаемся в предыдущую вершину и повторяем поиск. Если после возврата в начальную вершину и ее обработки останутся не просмотренные вершины, то повторим поиск, начиная из любой оставшейся вершины.

Также возможно усовершенствование алгоритма поиска в глубину, например:

- на каждую ячейку добавляем метку с длиной, найденного к ней кратчайшего пути; больше не посещаем ячейку, пока к ней не будет найден путь с меньшей стоимостью;
- выбираем сначала соседей, которые находятся ближе к цели;
- делаем остановку на определенной глубине, сначала равной расстоянию от старта до цели и постепенно увеличиваем (алгоритм последовательных приближений).

Беря за основу рассмотренные алгоритмы можно придумать много достаточно продвинутых алгоритмов. К сожалению идеального алгоритма не существует и поэтому ради использования того или другого требуется жертвовать либо производительностью либо интеллектом который нужен для него.

Недостатки рассмотренных алгоритмов заключаются в следующем: существует жесткая привязка к сетке разбивки плоскости; размер сетки влияет на скорость обработки информации; алгоритмы чувствительны к изменению формы препятствий; найденный путь не всегда является кратчайшим. Поэтому для ряда задач считаем необходимым использовать графо-аналитические способы описания, как препятствий, так и пути перемещения робота (представление в виде уравнений).

Литература.

1. Селифонов, Е. Path Tracing / Е. Селифонов. – Режим доступа: www / URL: http://robotics.bstu.by/mwiki/images/4/47/PathFinding – 20.03.2016. – Загл. с экрана.