§6 МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ НОВЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Винокурова С.Е.

МОДИФИКАЦИЯ МЕТОДА НАВИГАЦИОННОГО ГРАФА ДЛЯ ПОИСКА ПУТИ В ТРЕХМЕРНОМ ПРОСТРАНСТВЕ

Аннотация: *Поиск пути это задача нахождения наилучшего, оптимального маршрута между двумя точками пространства. Применение алгоритмов поиска пути позволяет управлять перемещением персонажей в трехмерном пространстве с автоматическим обходом препятствий, что позволяет пользователю полностью погрузиться в моделируемую 3D реальность за счет обеспечения решения проблем перемещения программной средой. В статье предлагается модификация метода навигационного графа для поиска пути в трехмерном пространстве за счет задания отдельного навигационного графа для каждого 3D объекта. В этом случае навигационный граф задает пути перемещения внутри и вокруг составного 3D-объекта или вокруг одиночного 3D-объекта. Модифицированный алгоритм в значительной степени требует меньших затрат для задания навигационного графа, позволяет рассчитывать путь с более естественной траекторией передвижения и позволяет производить поиск пути с обходом перемещающихся объектов. Предложенный метод подходит для применения в реальном времени, в том числе благодаря предложенным в данной статье оптимизациям.*
Ключевые слова: *поиск пути, навигационный граф, алгоритм, оптимизация, расширенный метод, алгоритм A*, navigation mesh, препятствия, динамические объекты, трехмерное пространство*

1. Введение

Поиск пути (ПП) – задача нахождения наилучшего, оптимального маршрута между двумя точками пространства. В зависимости от контекста критерии оптимальности могут быть различные, но во всех случаях на найденном пути не должно встречаться препятствий.

Поиск пути - важная часть виртуальной 3D среды. Пользователь не должен отвле-

DOI: 10.7256/2305-6061.2014.1.11346

каться от основной цели (например, обучения [1] или игрового процесса) на решение проблем перемещения в пространстве, и если он хочет переместиться из одной точки в другую, то путь разумной длины должен быть обязательно найден автоматически. При появлении на выбранном пути какого-либо препятствия система поиска пути должна автоматически скорректировать найденный путь.

Поиск пути также используется в компьютерных играх, различных симуляторах и тренажерах, чтобы обеспечивать корректное перемещение персонажей, управляемых компьютером.

2. Обзор и анализ алгоритмов поиска пути

Для поиска пути получили распространение четыре основных метода:

- алгоритм A*;
- навигационный граф;
- метод сочетания эвристик в трехмерном пространстве;
- метод навигационных сеток (navigation mesh).

Алгоритм A* [2] подразумевает разбиение пространства на одинаковые клетки, для каждой из которых задано, проходима клетка или нет. Объект, для которого идет поиск пути занимает одну клетку. По полученной матрице проходимости волновым алгоритмом или одной из его модификаций находится путь из одной клетки в другую. На рис. 1 изображен пример реализации алгоритма A*, где серые круги – точки начала и конца пути, черные клетки – препятствия, окружности – найденный путь.

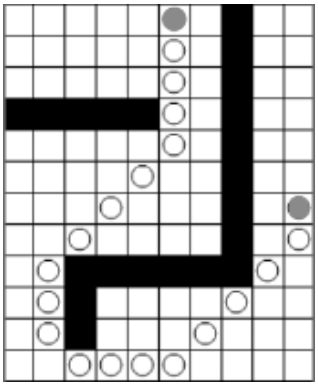

Рис. 1. Пример реализации алгоритма A*.

Метод получил большое распространение в 2D-пространстве (в играх-стратегиях и т.д), однако при применении в 3D-пространстве имеет серьезные недостатки:

- 3D-мир состоит из объектов произвольной формы, которые плохо ложатся на клеточную структуру поиска;
- трудно учитывать перемещающиеся препятствия, так как они могут занимать до 4 клеток сразу (если размер препятствия 1 клетка);
- перемещение по клеткам выглядит неестественно в трехмерном пространстве.

DOI: 10.7256/2305-6061.2014.1.11346

Методы сглаживания пути сплайнами могут привести к проблемам с непроходимостью отдельных участков пути;

- большое время поиска пути при большом количестве клеток.

Метод навигационного графа [3] предполагает поиск пути на графе, где его вершины - трехмерные точки, ребра - отрезки, соединяющие эти точки, с ценой равной длине отрезка. Все ребра графа являются проходимыми (персонаж может пройти по ним, не наткнувшись на препятствие). Задача поиска пути сводится к нахождению ближайших вершин к начальной и конечной точке, а затем к поиску пути на графе между этими вершинами с использованием критерия минимального веса общего пути (рис. 2).

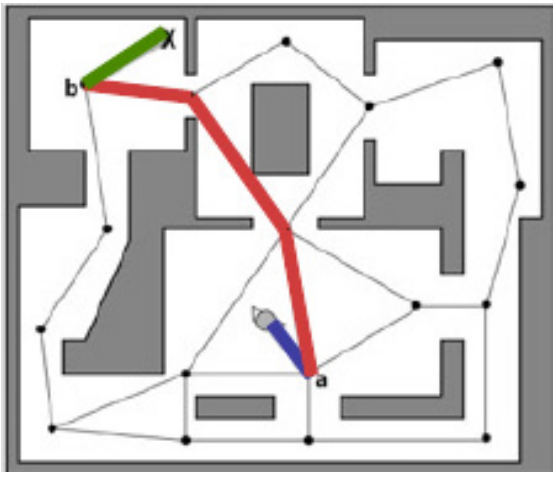

Рис. 2. Пример поиска пути с помощью навигационного графа.

Метод хорошо подходит для 3D-пространства, однако можно принять во внимание следующие недостатки метода навигационного графа [4]:

- большая трудоемкость задания графа, т.к. в некоторых случаях требуется задать чрезмерное число вершин графа;
- перемещающие при помощи поиска пути объекты проходят пустое пространство по неестественной траектории, т.к. следуют по заданным ребрам графа. Проявление проблемы можно смягчить, если увеличить число вершин графа, что усугубит предыдущий недостаток (рис. 3);
- в ситуации, когда динамический объект находится на ребре навигационного графа, нет корректного способа построить путь через это ребро;
- навигационный граф плохо подходит для поиска пути для перемещающихся объектов разных размеров, т.к. при размещении вершин графа приходиться ориентироваться только на 1 конкретный размер персонажа.

DOI: 10.7256/2305-6061.2014.1.11346

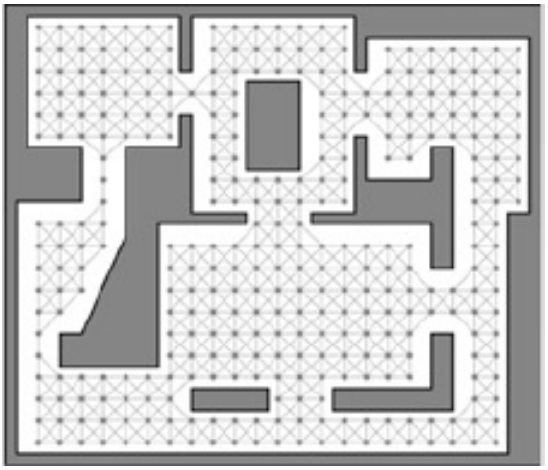

Рис. 3. Увеличение числа вершин графа для улучшения естественности найденного пути.

Метод сочетания эвристик в 3D-пространстве [5] предполагает применение специального алгоритма разрешения проблемной ситуации для небольшого числа распространенных случаев необходимости обхода препятствия. При его помощи полноценный поиск пути организовать невозможно.

Возможным примером можно назвать попытку при столкновении с препятствием найти луч с минимальным отклонением от вектора взгляда персонажа, который бы не пересекался с полигонами 3D-мира. В случае нахождения такого луча, персонаж перемещается по нему, а затем пытается следовать к конечной точке по прямому пути (рис. 4). В приведенном примере серый прямоугольник – препятствие, черный круг – персонаж, стрелка - направление взгляда персонажа, пунктирная линия – неоптимальный путь обхода, черная линия – найденный путь.

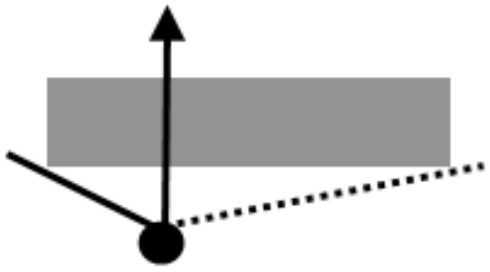

Рис. 4. Метод сочетания эвристик.

Основные недостатки:

- в большом числе ситуаций метод не может найти правильный путь, так как при разрешении коллизии путь ищется локально;
- в случае столкновения с препятствием сложность разрешения коллизии может быть очень высокой и напрямую зависеть от детальности трехмерного мира. В приведенном примере сложность расчетов напрямую зависит от числа полигонов 3D-мира.

DOI: 10.7256/2305-6061.2014.1.11346

Метод навигационных сеток [6] предполагает задание полигональной 3D-модели проходимого пространства (рис. 5). Далее действует правило, что между любыми двумя вершинами выпуклого полигона есть путь, находящийся внутри этого полигона. С помощью этого метода трудно учитывать динамические объекты, так как это требует больших вычислительных затрат. Также нужно хранить большое количество полигонов, если 3D-мир имеет большое количество объектов сложной формы. Метод хорошо работают как на открытых пространствах, так и в закрытых. Автоматическая генерация navigation mesh значительно затруднена и не может осуществляться в полной мере корректно.

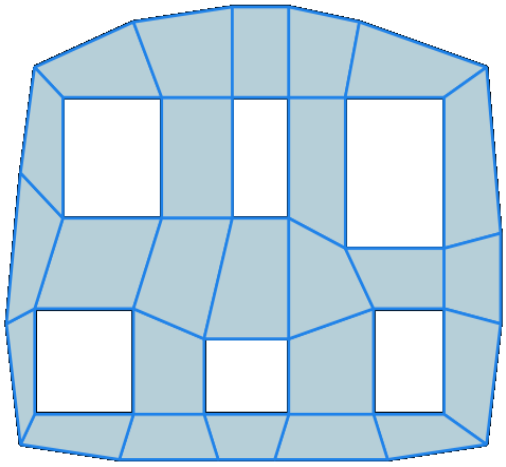

Рис. 5. Пример задания навигационных сеток.

В методе navigation mesh имеется выбор использования центров полигонов, центров ребер и вершин полигонов в качестве контрольных точек пути. Центры полигонов обеспечивают разумный набор узлов для графа поиска пути (рис. 6). В приведенном примере зеленая линия - идеальный путь, желтая линия – найденный путь. Кроме того, точки начала и конца добавляются наряду с точной центра многоугольника, в котором мы находимся.

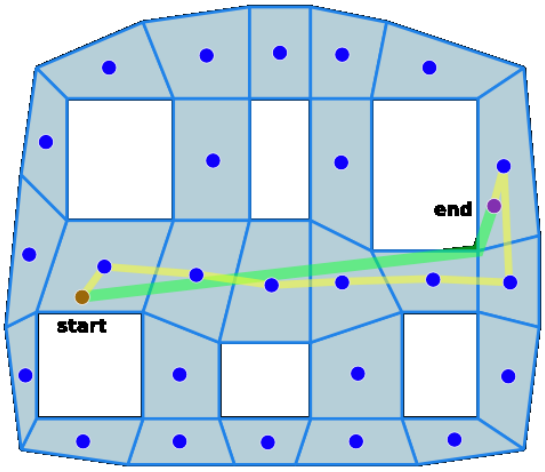

Рис. 6. Пример поиска пути методом navigation mesh с использованием центра полигонов в качестве контрольных точек пути.

DOI: 10.7256/2305-6061.2014.1.11346

Navigation mesh является наиболее полным представлением информации о проходимости пространства, однако естественность найденного пути не всегда является достаточной. Чтобы исправить этот недостаток можно применить методы сглаживания пути сплайнами.

Перемещение через центры многоугольников является обычно ненужным. Вместо этого можно передвигаться через центры ребер смежных многоугольников (рис. 7).

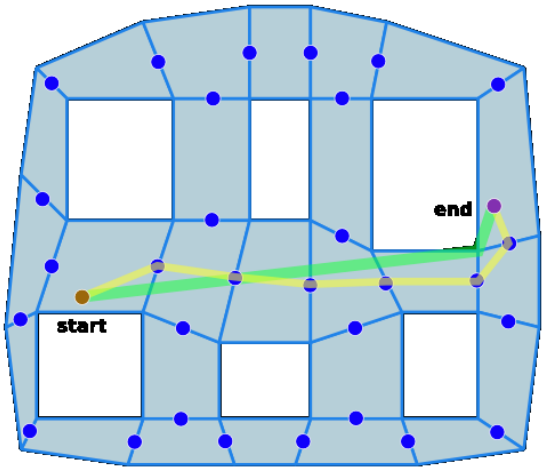

Рис. 7. Пример поиска пути методом navigation mesh с использованием центров ребер в качестве контрольных точек пути.

Можно увеличить количество точек на ребрах, приближая тем самым найденный путь к оптимальному, но возрастет стоимость пути.

Самый короткий путь вокруг препятствия состоит в том, чтобы перемещаться по вершинам многоугольников (рис. 8).

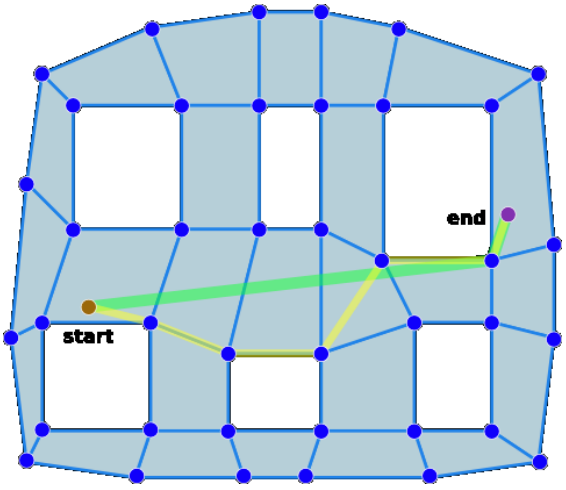

Рис. 8. Пример поиска пути методом navigation mesh с использованием вершин в качестве контрольных точек пути.

DOI: 10.7256/2305-6061.2014.1.11346

Есть только один недостаток в этом примере. Когда обходиться препятствие, найденный путь совпадает с кратчайшим, но тогда как у кратчайшего пути была бы прямая линия от начальной точки до угла препятствия, то найденный путь идет строго по ребрам, что выглядит неестественно.

Можно совместить все контрольные точки в один гибридный метод (рис. 9). Центры полигонов обычно не являются нужными. При обходе препятствия путь частично идет через центры ребер, а частично через вершины.

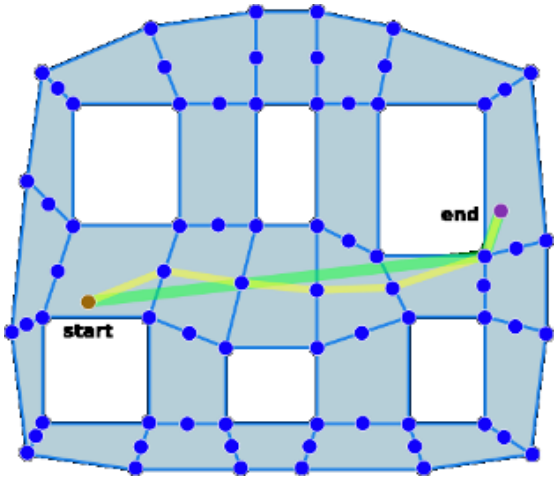

Рис. 9. Пример поиска пути методом navigation mesh с использованием ребер, вершин в качестве точек перемещения.

Для оптимизации поиска пути без потери качества найденного пути существует иерархический поиск пути A* или на навигационном графе – иерархия пространств поиска [7]. В первом случае сначала начала ищем путь по крупным клеткам, потом, когда путь найден, уточняем по более маленьким. Во втором случае, уточняется навигационный граф.

3. Модифицированный алгоритм навигационного графа

На основе собственного расширения метода навигационного графа была разработана система поиска пути для 3D-мира. Для поиска пути используется множество графов

$A = \{A_i, i = 1..n\}$, (1)

где A_i – навигационный граф одного объекта,

$A_i = \{V, R\}$, (2)

заданный множеством вершин

$V = \{V_k, k= 1..s\}$ (3)

и множеством ребер

$R = \{R_j, j = 1..m\}$, (4)

где $R_j = \{V_{j1} \in V, V_{j2} \in V, T_A, T_D, j_1 \neq j_2\}$, (5)

$T_A \in \{0,1\}$ – признак проходимости, T_D - контекстная информация о проходимости пространства вокруг ребра.

DOI: 10.7256/2305-6061.2014.1.11346

Поиск пути выполняется по известному отрезку пути S, соединяющему начальную и конечную точки $P_н$ и $P_к$. При поиске выполняются следующие шаги алгоритма:

1. Поиск множества $A_{изм}$ динамических объектов, у которых изменились поворот или позиция. Замена объединенного графа на его составляющие, если найденный объект находился в составе объединенного графа.

   $A_{изм} = \{A_i: p_i \neq p_i' \cup q_i \neq q_i'\}$,где (6)

   p_i, q_i – позиция и поворот объекта, соответствующего A_i, в текущий момент времени, а p_i', q_i' – в момент предыдущего поиска пути.

2. Пересчет координат ребер навигационного графа найденных динамических объектов в соответствии с новой позицией и поворотом.

3. Поиск множества $A_{пер}$ динамических объектов, навигационные графы которых пересекаются друг с другом.

4. Объединение пересекающихся навигационных графов динамических объектов и создание временного навигационного графа, который учитывается при поиске пути вместо пересекающихся.

5. Поиск множества точек пересечения P отрезка пути с навигационным графом объектов. Первоначально выполняется проверка пересечения отрезка пути с bounding box - наименьшим прямоугольником, включающим в себя все ребра навигационного графа объекта, и, только если проверка прошла успешно, выполняется поиск пересечений с ребрами графа. Это делает подход не менее эффективным, чем иерархические методы поиска пути [8].

   $P = \{P_i: P_i \in S$ и существуют j, k: $Pi \in R_j, R_j \in A_k\}$ (7)

6. Если найдена всего 1 точка пересечения, то идет поиск сегментов навигационного графа, перпендикуляр к которым из начальной или конечной точки имеет длину < ε. Из числа найденных сегментов выбирается тот, длина перпендикуляра к которому меньше других, и к множеству P добавляется P_1 – точка пересечения с перпендикуляром, образуя расширенное множество точек пересечения P′:

   $P' = P \cup P_1$ (8)

7. Рассчитывается сортированное по удаленности от начальной точки пути множество точек пересечения P_s.

   $P_s = \{P_i: D(P_i, P_н) < D(P_i + 1, P_н), i \in 1..N_p - 1\}$ (9)

   где D(A, B) – расстояние от точки A до точки B, N_p – количество элементов множества P′.

8. Множество точек пересечения разбивается на несколько частей P_{sk}, в каждой из которых идут подряд точки одного и того же графа.

   $P_{sk} = \{P_i: P_i \in P_s, P_i \in A_k, P_{i+1} \in A_k, i \in 1..N_p - 1\}$ (10)

9. Внутри получившихся частей списка поиск пути ведется отдельно, а крайние точки соседних частей соединяются прямым отрезком. При поиске пути внутри одного навигационного графа рассматриваются варианты:

   1 точка пересечения — путь касается навигационного графа и поиск оптимального пути не требуется;

DOI: 10.7256/2305-6061.2014.1.11346

2 и более точки пересечения — выполняется поиск пути по крайним точкам пересечения по критерию минимальной суммарной длины ребер. Ребра учитываются в соответствии с признаком проходимости T_A.

10. Компоновка полученных отрезков путей в единый путь. Объединение производится за счет того, что берутся найденные пути для каждого графа и крайние точки этих путей соединяются прямой линией.

Описанный алгоритм, по крайней мере, не уступает в адекватности найденного пути базовому подходу, а в ряде случаев и превосходит его. Например, в усовершенствованном методе (рис. 10) путь по пространству свободному от препятствий проходит более естественно. Таким образом, расширенный алгоритм свободен от недостатка перемещения объектов по неестественной траектории, когда они следуют по строго заданным ребрам графа.

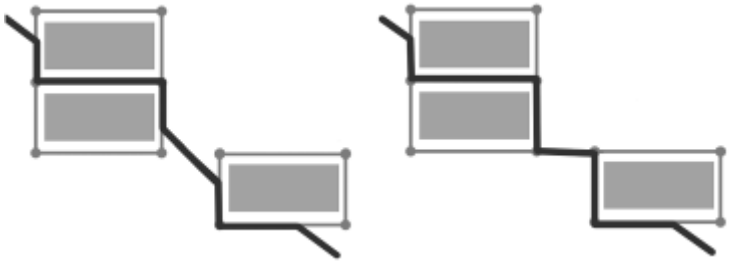

Рис. 10. Поиск пути в модифицированном (слева) и базовом методах (справа) навигационного графа.

Таким образом, был сформулирован алгоритм для поиска пути на базе метода навигационного графа. Алгоритм имеет свои преимущества и свободен от большей части недостатков базового метода.

4. Расширение метода для поиска пути с учетом динамических объектов

Динамическим объектом считается объект, который может изменить позицию и поворот, появиться или исчезнуть в любой момент времени. При поиске пути такие объекты учитываются также как и статические объекты. Рассмотрим методы, которые позволяют учитывать динамические объекты в рамках общего алгоритма поиска пути:

4.1. Столкновение в момент следования по пути

Так как динамические объекты могут изменить свою позицию и поворот, то необходим механизм разрешения коллизии в момент следования по пути. В таких случаях предлагается находить путь заново из точки столкновения и, таким образом, учитывать новое положение динамического объекта.

4.2. Изменение состояния проходимости ребра

Ребро навигационного графа, по которому идет найденный путь, может полностью

DOI: 10.7256/2305-6061.2014.1.11346

оказаться в непроходимом пространстве. Чтобы учесть этот случай, в момент коллизии ребро отмечается как непроходимое с помощью признака проходимости ТА и при новом поиске пути не учитывается (рис. 11). Когда у динамического объекта изменяется позиция или поворот, все его ребра отмечаются как проходимые.

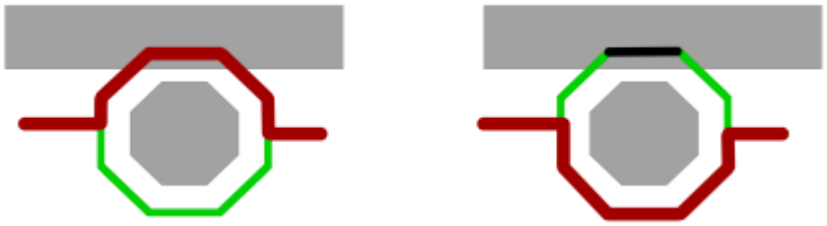

Рис. 11. Найденный путь обхода динамического объекта при первой попытке поиска пути (слева) и второй попытке (справа). Черным цветом выделено ребро, отмеченное как непроходимое после первой попытки.

4.3. Учет динамических объектов, расположенных на ребре НГ статического объекта

Для каждого ребра статического навигационного графа задается контекстная информация TD – информация о проходимости пространства вокруг ребра: ребро можно обойти слева, справа, слева и справа, нельзя обойти совсем. Эта информация используется для учета динамических объектов, расположенных на ребре навигационного графа статического объекта. Возможны 2 случая:

- ребро нельзя обойти: оно помечается как непроходимое (рис.12).

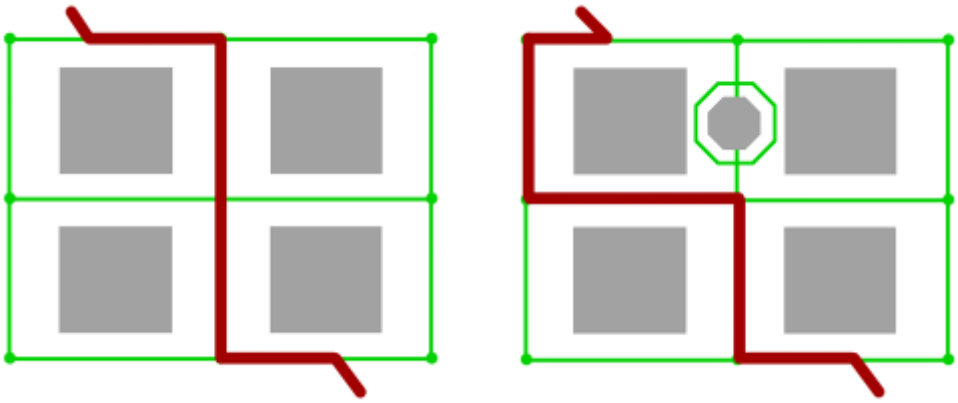

Рис. 12. Поиск пути с динамическим объектом на ребре без заданного признака проходимости статического навигационного графа (справа) и без него (слева).

- ребро можно обойти, тогда в п.10 алгоритма при составлении пути участок траектории по рассматриваемому ребру между первой и последней точкой пересечения с динамическим объектом заменяется обходным путем динамического объекта. Направление обхода соответствует информации о

DOI: 10.7256/2305-6061.2014.1.11346

проходимости (рис.13).

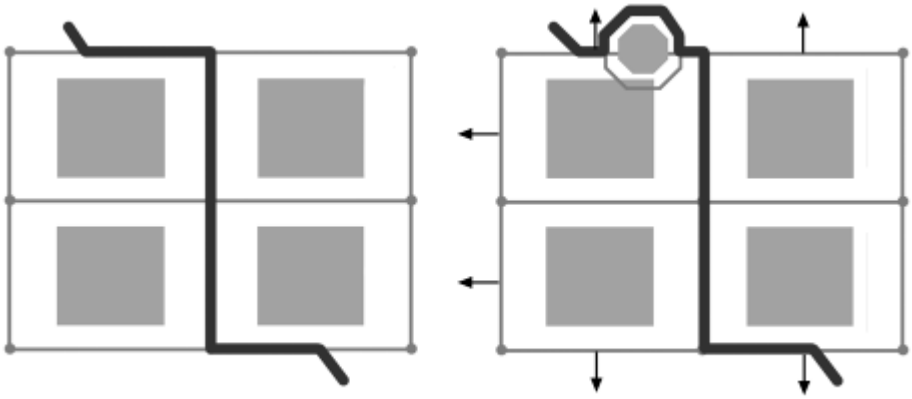

Рис. 13. Поиск пути с динамическим объектом на ребре с заданным признаком проходимости статического навигационного графа (справа) и без него (слева).

4.4. Слияние навигационных графов

Так как динамические объекты могут занимать произвольное положение в пространстве, то возможна ситуация, когда их навигационные графы пересекаются. Может оказаться, что ребро одного навигационного графа динамического объекта проходит через другой динамический объект, и это ребро входит в найденный путь, который окажется некорректным. Чтобы избежать такой ситуации, необходимо слияние графов, такое, что все ребра объединенного графа окажутся проходимыми (рис.14).

Алгоритм слияния навигационных графов:

1. поиск точек пересечения графов;
2. ребра, которым принадлежат точки пересечения, делятся ими на части. Эти части рассматриваются как новые ребра графа вместо исходного;
3. проверяем все ребра и отбрасываем те, которые находятся внутри одного из исходных графов.

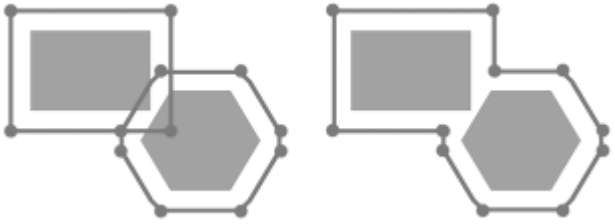

Рис. 14. Навигационный граф динамических объектов до слияния (слева) и после слияния (справа).

DOI: 10.7256/2305-6061.2014.1.11346

5. Оптимизация генерации навигационного графа и алгоритма поиска внутри него.

Модифицированный алгоритм также позволяет реализовать ряд возможностей для сокращения объема ручной работы при задании графа, что избавляет новый подход от одного из недостатков базового метода навигационного графа:

- для каждого объекта используется свой навигационный граф, пустое, гарантированно проходимое пространство не требует заполнения путями графа (рис.15);

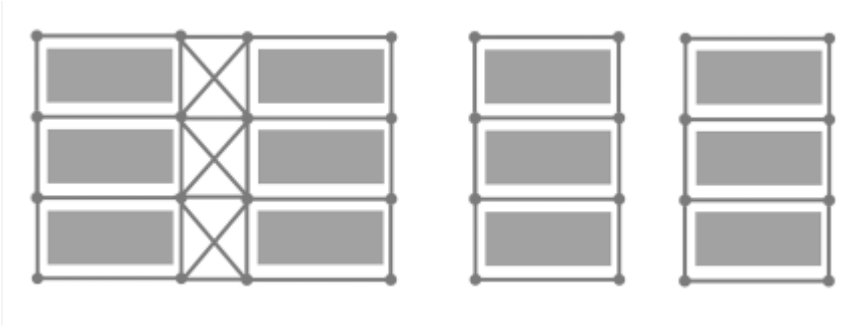

Рис. 15. Навигационный граф в базовом (слева) и модифицированном (справа) методах.

- для повторяющихся объектов навигационный граф может быть скопирован автоматически. На рис. 16 изображен объект с навигационным графом заданным вручную и копии этого объекта с автоматически скопированными путями – пунктирные линии;

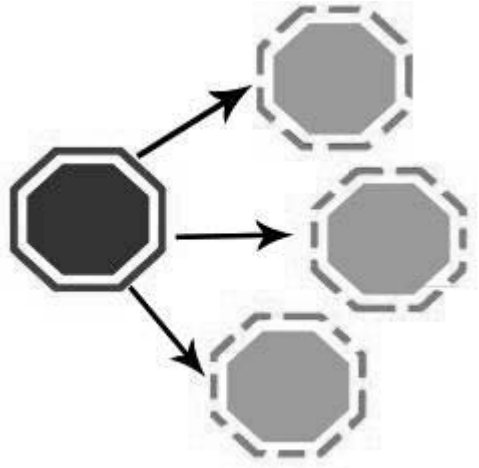

Рис. 16. Автоматическое копирование путей для повторяющихся объектов.

- для обходных путей простой геометрической формы может быть использована автогенерация пути на основе данных о bounding box объекта и его положении в пространстве.

DOI: 10.7256/2305-6061.2014.1.11346

За счет перечисленных особенностей метода значительно уменьшается объем ручной работы.

Для каждого навигационного графа рассчитывается AABB. В пункте 2 общего алгоритма поиска добавляется проверка пересечения отрезка пути с AABB графа. Только если эта проверка прошла успешно, проводится проверка пересечения отрезка с каждым сегментом графа. На рис. 2.6.7 приведен пример отображения AABB навигационного графа (обозначен синими линиями) и найденного пути (красные линии).

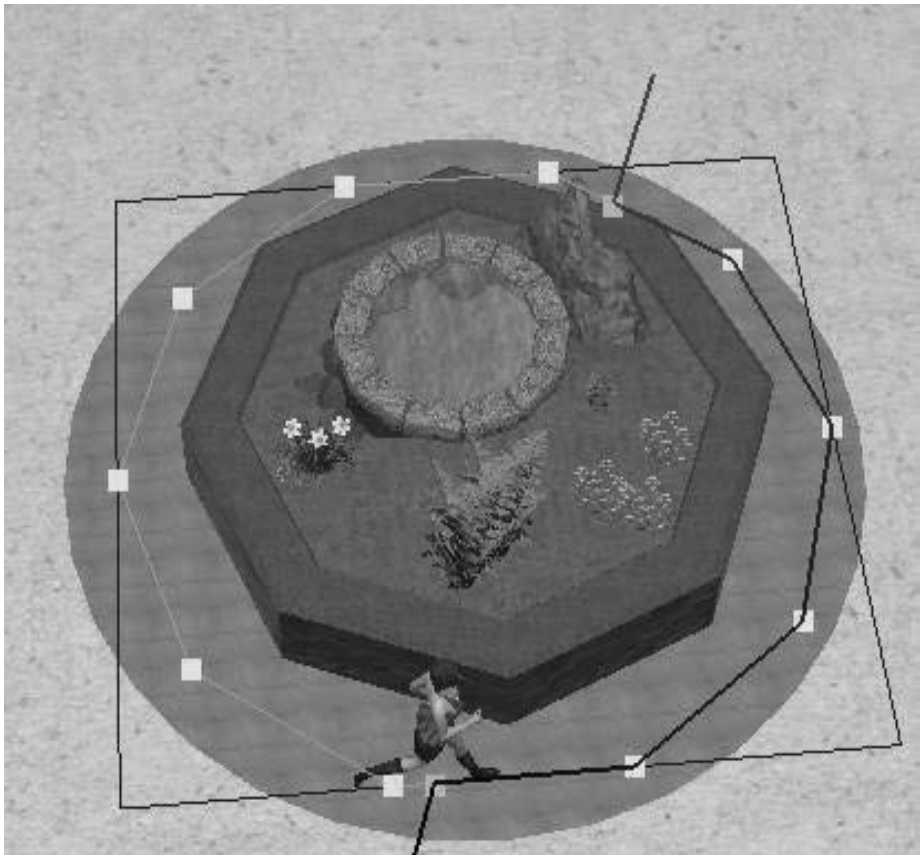

Рис. 2.6.7. Пример отображения AABB навигационного графа

Поиск пути внутри графа идет путем перебора возможных путей с использованием очереди с отсечением возможных путей, чья длина превысила длину наиболее оптимального из уже найденных путей. Отсекаются также пути, длина которых до текущей вершины превышает длину ранее найденного пути для этой вершины

6. Результаты

Оценим сложность алгоритма для статических объектов (табл. 1), будем рассматривать только пункты алгоритма, требующие значительных вычислительных затрат.

DOI: 10.7256/2305-6061.2014.1.11346

Табл. 1. Оценка сложности базового и расширенного алгоритмов

№ шага	Базовый алгоритм	Расширенный алгоритм (РА)	Результат
5	$O(N_s)$, где N_s – кол-во ребер	O(N), где N – общее кол-во объектов	РА оптимальнее, т.к. Ns > N
6	$O(N_s)$	$O(N_i)$ для каждого i-го НГ	РА оптимальнее, т.к. Ns > суммы Ni
7	O(K log K), K – общее кол-во ТП	$O(K_i \log K_i)$, K_i – кол-во ТП в i-ом НГ	РА оптимальнее, т.к. K log K > суммы $K_i \log K_i$
10	$O(N_s^2)$	$O(N_i^2)$	РА оптимальнее, т.к. N_s^2 > суммы N_i^2

В п. 10. Алгоритма сложность зависит от метода поиска на графе. Если M –кол-во вершин, а N – ребер, то для алгоритма Дейкстры сложность составит O(M2 + N) или O(N2), т.к. N сравнимо с M в нашем случае.

Также можно оценить среднее время ПП. Была взята выборка из 10000 реальных случаев ПП. Замеры производились на компьютере с ЦП Intel Core 2 Duo 2.7 ГГц (использовалось 1 ядро).

Среднее время поиска – 24 мкс, минимальное время поиска – 1 мкс, максимальное время поиска - 110 мкс. Полученные результаты говорят о возможности применения алгоритма в реальном времени.

Чтобы получить полное представление об эффективности предложенного подхода, требуется оценить сложность учета динамических объектов.

Табл. 2: Оценка сложности учета динамических объектов

Шаг	Описание	Сложность
1	Поиск динамических объектов, у которых изменился поворот или позиция	O(N), где N - число динамических объектов
2	Пересчет координат вершин НГ найденных динамических объектов	$O(\sum_{i=1}^{k} len(V_i))$, где k - число найденных динамических объектов, len - количество элементов множества
3	Поиск пересекающихся динамических объектов	O(k*N)

DOI: 10.7256/2305-6061.2014.1.11346

Исходя из оценки сложности (табл. 2), можно сделать вывод, что алгоритм подходит для применения в реальном времени. При этом расширенный алгоритм свободен от 3 из 4 недостатков базового метода, что делает его более перспективным.

7. Заключение

Была разработана система поиска пути в 3D-пространстве на основе расширенного метода навигационного графа. Модифицированный алгоритм свободен от значительной части недостатков базового метода и подходит для применения в реальном времени. Система была успешно апробирована в образовательной 3D-среде «vAcademia» [9].

Библиография :

1. А.Ю. Сморкалов. Математическая и программная модели генерации текстур на графических потоковых процессорах // Программные системы и вычислительные методы. – 2013. – № 1. – С. 104-107. DOI: 10.7256/2305-6061.2013.01.10
2. Yap, P. Grid-Based Path-Finding / P.Yap // AI '02 Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence-2002. – 44-55 pp
3. Cui, X., Shi, H. A*-based Pathfinding in Modern Computer Games/ X. Cui, H. Shi // IJCSNS International Journal of Computer Science and Network Security-Vol.11 No.1, January 2011.
4. Tozour, P. Fixing Pathfinding Once and For All [Electronic resource] / Game AI.-Electronic data.-Mode acess: http://www.ai-blog.net/archives/000152.html. free.
5. Mika, M., Charla, C. Simple,Cheap Pathfinding / M. Mika, C. Charla // AI Game Programming Wisdom-2002.
6. O'Neill, J. C. Efficient Navigation Mesh Implementation / J. C. O'Neill // Journal of Game Development-Vol. 1 No. 1, 2004.-71-90 pp.
7. Akbar, A. Raycast Navigation Mesh Generation and Path Finding System [Electronic resource]-Electronic data.-Mode acess: http://syntheticarc.com/?q=node/6, free.
8. Botea, A., Muller, M., Schaeffer, J. Near Optimal Hierarchical Path-finding / A. Botea, M. Muller, J. Schaeffer // Journal of Game Development-Vol. 1 Issue 1, 2004.
9. И.А.Садовин, А.Ю. Сморкалов. Система переноса лекций в виртуальный мир vAcademia с использованием возможностей Microsoft Kinect и потоковых процессоров // Программные системы и вычислительные методы. – 2013. – № 4. – С. 90-94. DOI: 10.7256/2305-6061.2013.04.10.

References:

1. A.Yu. Smorkalov. Matematicheskaya i programmnaya modeli generatsii tekstur na graficheskikh potokovykh protsessorakh // Programmnye sistemy i vychislitel'nye metody. – 2013. – № 1. – S. 104-107. DOI: 10.7256/2305-6061.2013.01.10.
2. Yap, P. Grid-Based Path-Finding / P.Yap // AI '02 Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence-2002. – 44-55 pp.

3. Cui, X., Shi, H. A*-based Pathfinding in Modern Computer Games/ X. Cui, H. Shi // IJCSNS International Journal of Computer Science and Network Security-Vol.11 No.1, January 2011.
4. Tozour, P. Fixing Pathfinding Once and For All [Electronic resource] / Game AI.-Electronic data.-Mode acess: http://www.ai-blog.net/archives/000152.html. free.
5. Mika, M., Charla, C. Simple,Cheap Pathfinding / M. Mika, C. Charla // AI Game Programming Wisdom-2002.
6. O'Neill, J. S. Efficient Navigation Mesh Implementation / J. C. O'Neill // Journal of Game Development-Vol. 1 No. 1, 2004.-71-90 pp.
7. Akbar, A. Raycast Navigation Mesh Generation and Path Finding System [Electronic resource]-Electronic data.-Mode acess: http://syntheticarc.com/?q=node/6, free.
8. Botea, A., Muller, M., Schaeffer, J. Near Optimal Hierarchical Path-finding / A. Botea, M. Muller, J. Schaeffer // Journal of Game Development-Vol. 1 Issue 1, 2004.
9. I.A.Sadovin, A.Yu. Smorkalov. Sistema perenosa lektsii v virtual'nyi mir vAcademia s ispol'zovaniem vozmozhnostei Microsoft Kinect i potokovykh protsessorov // Programmnye sistemy i vychislitel'nye metody. – 2013. – № 4. – S. 90-94. DOI: 10.7256/2305-6061.2013.04.10.

DOI: 10.7256/2305-6061.2014.1.11346