УДК 004.946, 004.031.42.

Д. А. Булгаков – магистрант кафедры вычислительных систем и сетей
Н. Н. Решетникова (канд. техн. наук, доц.) – научный руководитель

ПРИМЕНЕНИЕ АЛГОРИТМА ASTAR (A*) ДЛЯ ЗАДАЧ АДАПТИВНОЙ НАВИГАЦИИ В ВИРТУАЛЬНОМ ТРЕХМЕРНОМ ПРОСТРАНСТВЕ

Первоначально задача навигации определялась как процесс перемещения через виртуальную среду. В дальнейшем в это определение включили процесс нахождения пути перемещения. Наконец, его расширили еще и средствами и указателями для успешной навигации. Таким образом, навигация – процесс, при котором пользователи управляют своим движением, используя окружающие указатели и искусственные средства типа карт, так, чтобы они могли достигнуть своих целей без возможности потеряться.

Таким образом, задача адаптивной навигации в виртуальной среде заключается в том, чтобы помочь пользователям найти свой путь, в частности – в трехмерной среде с шестью степенями свободы.

Существует три метода адаптивной навигации:

– прямое руководство. При прямом руководстве система анализирует маршрут от начальной точки до конечной и в явном виде отображает его пользователю;

– пошаговая инструкция. При этом методе весь путь разбивается на некоторое количество отдельных контрольных точек. Начиная с первой точки, система дает пользователю подсказки для достижения следующей. Подсказки могут быть визуальными (маркер) или голосовыми;

– сокрытие. При методе сокрытия система скрывает полностью или частично те объекты окружающей среды, которые мешают прямой видимости конечной точки маршрута.

A* – алгоритм поиска по первому наилучшему совпадению на графе, который находит маршрут с наименьшей стоимостью от одного узла (начального) к другому (целевому, конечному).

Алгоритм A* использует термин узел для определения сегмента исследуемого пути. Таким образом, путь будет состоять из стартового узла, конечного и списка узлов, которые формируют лучший путь между ними.

Порядок обхода узлов определяется эвристической функцией «расстояние + стоимость», которая обычно обозначается как $f(x)$. Эта функция является суммой двух других функций: функции стоимости достижения рассматриваемого узла из начального (обозначается как $g(x)$) и эвристической оценкой расстояния от рассматриваемого узла к конечному (обозначается как $h(x)$).

Рассмотрим процесс работы алгоритма A* на конкретном примере.

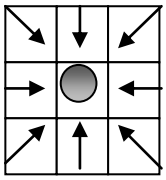

Рис. 1. Начало поиска

14	10	14
10		10
14	10	14

Рис. 2 Стоимости соседних узлов

На рисунке 1 круг в центре – стартовый узел. Все соседние узлы имеют указатель, направленный на центральный. Выберем один из соседних узлов. Тот, у которого стоимость меньше. Для оценки стоимости воспользуемся формулой $F = G + H$.

Путь генерируется путем повторного прохода через открытый список и выбора узла с наименьшей стоимостью F. Рассмотрим вычисление стоимости F.

Присвоим вертикальным и горизонтальным перемещениям стоимость 10, а диагональным – 14. Хотя правильнее было бы использовать значения 1 и $\sqrt{2}$, при таких числах алгоритм будет вычислять-

ся значительно дольше. Поскольку при значениях 10 и 14 соотношение сохраняется, то предпочтительнее использовать именно такие целые числа (рисунок 2).

Следующая составляющая стоимости – предполагаемая стоимость перемещения от выбранного узла до конечного. Эта составляющая делается с помощью эвристики (гипотезы или предположения), которая представляет собой формулу или алгоритм. Одна из самых простых эвристик в этом случае – определить расстояние между этими двумя узлами с помощью теоремы Пифагора:

$$S = \sqrt{x^2 + y^2},$$

где S – расстояние между узлами; x – расстояние между столбцами в которых находятся узлы; y – расстояние между строками в которых находятся узлы.

Расстояние S и есть H из формулы полной стоимости. Вычислим её; результат на рисунке 3.

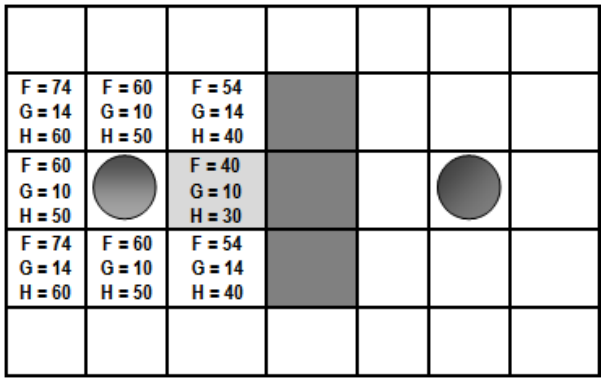

Рисунок 3 – полная стоимость для каждого узла

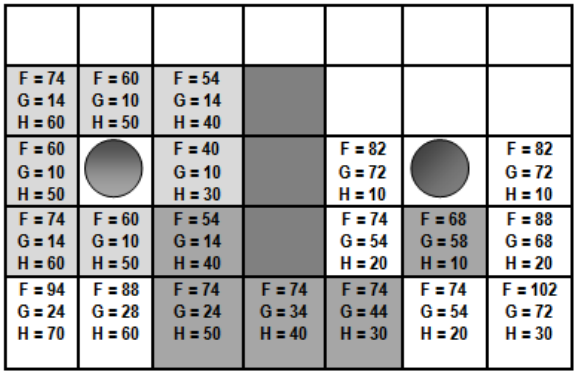

Рисунок 4 – конечный этап поиска пути

Опишем пошаговый алгоритм A*.

1. Добавляем стартовый узел в открытый список.
2. Повторяем следующее:
   а. Поиск в открытом списке узла с наименьшей стоимостью F. Делаем его текущим.
   б. Помещаем его в закрытый список. (И удаляем из открытого).
   в. Для каждого из соседних восьми узлов:
      – если узел непроходим или находится в закрытом списке, игнорируем. В противном случае делаем следующее;
      – если узел еще не в открытом списке, то добавляем его туда. Делаем текущий узел родительским для этого узла. Рассчитываем стоимости F, G и H узла;
      – если узел уже в открытом списке, то проверяем, не дешевле ли будет путь через этот узел. Для сравнения используем стоимость G. Более низкая стоимость G указывает на то, что путь будет дешевле. Если это так, то меняем родителя узла на текущий узел и пересчитываем для него стоимости G и F.
   г. Останавливаемся если:
      – добавили целевой узел в открытый список, в этом случае путь найден;
      – открытый список пуст и мы не дошли до целевой клетки. В этом случае путь отсутствует.
3. Сохраняем путь. Двигаясь назад от целевого узла, проходя от каждого узла к его родителю до тех пор, пока не дойдем до стартового узла. Это и будет путь.

Особенность алгоритма A* в том, что он может дать оптимальный путь независимо от того, какая эвристика используется. И этих оптимальных путей может быть несколько, т.е. стоимость их будет одинакова, а сами пути будут содержать разные узлы.

Таким образом, использование «лучшей» эвристики не значит, что она вернет кратчайший путь. Но некоторые эвристики работают быстрее, чем другие. Наибольшее различие между ними это количество узлов проверяемых при поиске пути каждой эвристикой.

Рассмотрим реализацию алгоритма в виртуальном пространстве на основе движка Virtools.

Движок Virtools изначально поддерживает алгоритм A* с двумя эвристическими методами его реализации: Евклида и Манхэттенским.

Создадим в пространстве сетку (grid). Сетка (Grid) является стандартным объектом Virtools. Сетка представляет собой трехмерный объект, состоящий из прямоугольных секторов размер, положение и цвет которых можно изменять. Цвет сектора является его основным параметром. За счет изменения цвета можно задавать различные условия, при выполнении или невыполнении которых будет генерироваться событие.

На рисунке 5 изображена сетка, цветные сектора которой будут являться условными препятствиями. Положение персонажа отмечено квадратом, цели – кругом. Линиями показан итоговый путь персонажа. Верхняя линия – с применением эвристики Эвклида, нижняя – с применением эвристики Манхэттена.

Вслед за сеткой создается действующее лицо – персонаж. Как правило, персонаж экспортируется из программы 3D моделирования, т.к. сам движок Virtools не позволяет придавать свойства персонажа обычному 3D объекту. Далее для персонажа создается сценарий (скрипт) в котором будет размещен алгоритм, показанный на рисунке 6.

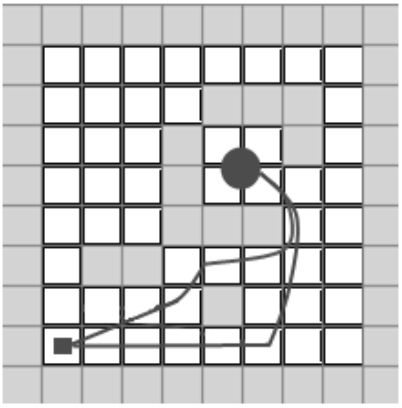

Рис. 5. Вид сетки в редакторе

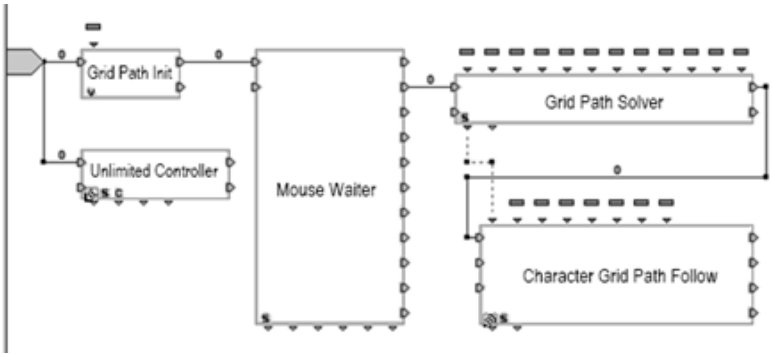

Рис. 6. Оновной алгоритм

Основным функциональным блоком является блок Grid Path Solver. Он активирует алгоритм поиска пути, учитывая заданные условия. Его строение показано на рисунке 7.

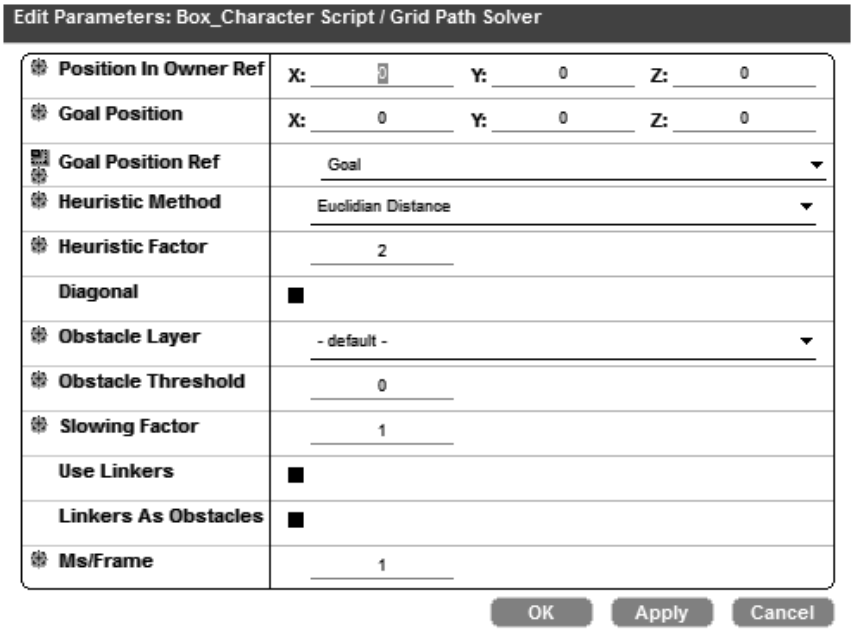

Рис. 7.Содержание блока Grid Path Solver

Рассмотрим некоторые наиболее значимые пункты:

– Goal Position Ref – ссылка на положение цели. Указав в данном пункте какой-либо объект, система будет постоянно отслеживать изменения его положения и корректировать алгоритм. При указании позиции строго, через предыдущий пункт, система просчитает путь один раз до конкретно заданной точки.

– Heuristic Method – используемый эвристический метод. Всего доступно два метода: Евклидовый и Манхэттенский. Евклидовый доступен в трех разновидностях: простой, квадратичный и оптимизированный.

– Heuristic Factor – определяет важность эвристики при расчетах.

Для реализации метода пошаговой инструкции создадим граф из 3D-фреймов – невидимых трехмерных объектов.

После создания достаточного количества фреймов и расположения их в сцене, необходимо объединить их в группу и создать для этой группы сценарий. В этот сценарий помещается один единственный блок – Create Nodal Path (создать путь по узлам).

Данный блок не имеет входов, выходов и не использует никакие переменные. Он является своего рода конструктором сетки путей. Его строение показано на рисунке 8. Фреймы обозначены белыми кругами.

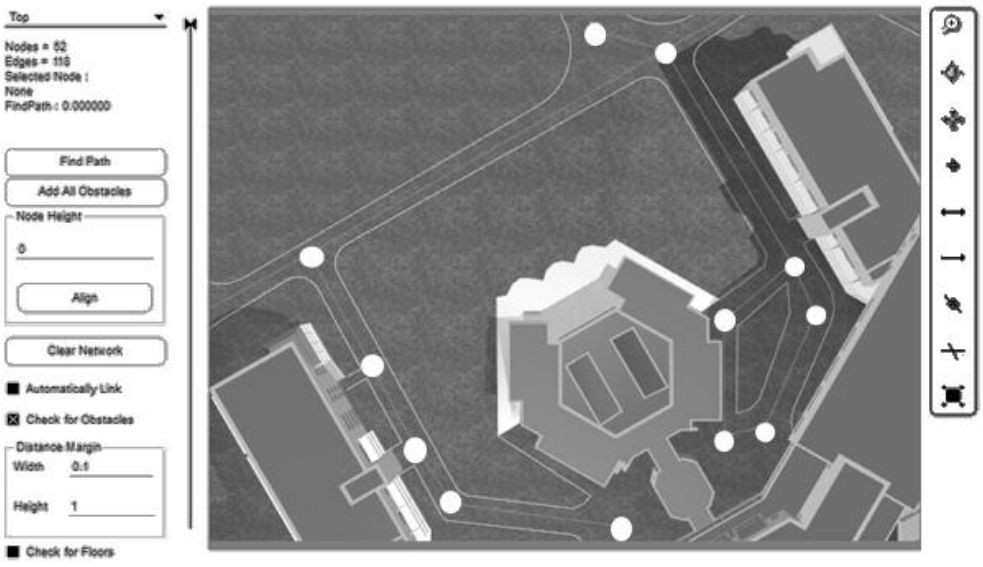

Рис. 8. Строение блока Create Nodal Path

На панели инструментов можно масштабировать, и перемещать вид, а также добавлять, удалять узлы, связи между узлами и препятствия. Связь между двумя узлами может быть однонаправленной и двунаправленной.

После создания необходимых связей в сцену добавляется персонаж и для него создается скрипт. Опишем последовательно принцип его работы:

1. Фиксируется произведенный клик левой кнопкой мыши в пространстве сцены.
2. Определяется местоположение курсора в момент клика.
3. Если клик произведен вне области сцены, то возвращаемся в начальное состояние, иначе определяется ближайший к месту клика узел. Этот узел считается концом пути.
4. Определяется ближайший узел к текущему положению персонажа.
5. На блок поиска пути поступает информация о двух узлах:
   – стартовом (начальное положение персонажа);
   – конечном (ближайший к точке клика).
6. Находится промежуточный узел, если таковой имеется, между начальным и конечным.

7. При наличии промежуточных узлов, на позицию ближайшего узла к персонажу ставится визуальный маркер.

8. После прохождения персонажем первого промежуточного узла, метка перемещается на следующий. Так происходит до тех пор, пока персонаж не достигнет конечного узла.

Алгоритм A* появился в 1968 году, но до сих пор остается актуальным и востребованным алгоритмом. Он применяется во многих играх и обучающих приложениях.

Движок Virtools изначально поддерживает данный алгоритм с двумя эвристическими методами работы, что позволяет эффективно решать задачи поиска пути и адаптивной навигации в виртуальной среде.

Библиографический список

1. Игнатьев М.Б., Никитин А.В., Войскунский А.Е. “Архитектура виртуальных миров” Санкт-Петербург, 2009г.
2. Русскоязычный форум Virtools (электронный ресурс). Режим доступа:- http://virtools.ve-group.ru/, свободный.
3. Королев Сергей "Основы Virtools" 2-я редакция, 2008г.
4. Кейт Петерс “AdvancED ActionScript 3.0 Animation”, 2008г.
5. Брайан Стоут “Smart Moves: Intelligent Pathfinding” (статья), 1997г.
6. Патрик Лестер: “Алгоритм A* для новичков” (статья), 2004г.