РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
АВТОМЕТРИЯ
2003, том 39, № 2

УДК 681.3.069

С. А. Кузиковский, М. М. Лаврентьев, И. В. Белаго
(Новосибирск)

МОДЕЛИРОВАНИЕ ДВИЖЕНИЯ АВТОМОБИЛЕЙ НА СЕТИ ДОРОГ*

Рассмотрены некоторые задачи, возникающие при построении компьютерных автомобильных симуляторов (АС) на базе персональных компьютеров. Приведены примеры возможных применений АС. Перечислены компоненты типичного АС.

Введение. В последние годы наблюдается повышенный интерес к автомобильным симуляторам (АС) на базе персональных компьютеров [1, 2]. Это связано в первую очередь с ростом производительности персональных компьютеров и графических акселераторов для них, что позволяет производить как адекватное моделирование динамики автомобиля [3], так и графическое отображение моделируемого окружения с высоким качеством [4]. Ранее сравнимые параметры имели только дорогие специализированные системы со значительными вычислительными ресурсами. Стандартизация графических акселераторов для персональных компьютеров позволяет существенно расширить пользовательскую базу такого рода приложений.

Автомобильный симулятор может использоваться для обучения вождению, а также как основная часть компьютерных игр соответствующего (автомобильного) направления. В научных целях симуляторы используются при изучении влияния алкоголя и наркотиков на время реакции водителя или при проектировании транспортных средств, пригодных для управления инвалидами.

Типичный симулятор предполагает наличие визуальной модели местности с набором свойств, визуальной и поведенческой моделей автомобиля, наличие автомобиля, управляемого пользователем, а также других автомобилей на местности, которые управляются компьютером. Возможен многопользовательский вариант симулятора, в котором несколько пользователей управляют своими автомобилями внутри одной модели и могут взаимодействовать друг с другом. В зависимости от задач симулятора могут потребоваться реализации моделирования времени суток, погодных условий, внештатных ситуаций и другие.

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 00-15-99092).

Симулятор является системой реального времени, иначе говоря, спецификой работы симулятора в отличие от других вычислительных задач являются требования малого времени отклика и высокой кадровой частоты графического отображения. Минимальная кадровая частота, в свою очередь, определяется временем реакции человека и физиологией зрения человека и не может быть уменьшена. Эти требования связаны и выражаются в том, что цикл работы симулятора, в течение которого производятся очередной шаг моделирования и отображение очередного кадра, должен быть не более нескольких десятков миллисекунд, а это накладывает серьезные ограничения на используемые алгоритмы: время, занимаемое алгоритмом в цикле, ограничено сверху, причем каждый из алгоритмов может занимать только небольшую отведенную ему часть цикла. Алгоритмы, не влияющие напрямую на время отклика, можно разделить на несколько циклов или реализовать отдельным потоком; в противном случае результат зависит только от качественной разработки и эффективной реализации алгоритма. Таким образом, эффективность используемых алгоритмов непосредственно влияет на основные параметры симулятора.

Симулятор имеет фазу загрузки, во время которой идет подготовка данных, и время выполнения операций не столь критично, как при дальнейшей работе симулятора. Для обеспечения качественной работы симулятора важно как можно большую часть расчетов перенести на фазу загрузки.

В первом приближении модель поведения автомобилей в симуляторе имеет два масштаба. Вокруг наблюдателя для моделирования поведения автомобилей используются более точные и дорогие алгоритмы. Более дешевые алгоритмы работают всегда и для всех автомобилей на всей моделируемой территории.

Автомобильный симулятор (и не только автомобильный), как правило, состоит из нескольких взаимодействующих подсистем. Основные подсистемы – это система отображения, отвечающая за графическое отображение обстановки, окружающей кабину водителя (и, возможно, части самой кабины) [4], подсистема динамики, которая моделирует динамику автомобиля (автомобилей) [3], и, наконец, подсистема, которая отвечает за сценарий поведения автомобилей на моделируемой территории.

В ведении последней из подсистем находятся такие важные характеристики, как траектории автомобилей, их скорости, выбор оптимальной трассы, внутренняя логика моделируемого мира и другие. При реализации такой подсистемы возникает ряд типичных задач, которые и рассмотрены в данной работе.

В разд. 1 рассмотрена задача нахождения кратчайшего пути между двумя произвольными точками на сети дорог В разд. 2 рассматриваются задачи над параметрическими полиномиальными кривыми, при помощи которых заданы участки дорог Это нахождение точки на кривой, ближайшей к заданной, и длины участка кривой. В разд. 3 коротко рассмотрен алгоритм обеспечения непротиворечивости положений автомобилей на дороге.

1. **Граф дорог.** Сеть дорог на местности рассматривается нами как граф (рис. 1). Перекрестки, развилки и другие элементы являются узлами этого графа, а сами дороги – ребрами. Например, узлом является место для парковки автомобиля, место на заправочной станции и т. д. Узлы графа имеют набор атрибутов, управляющих свойствами узлов. Ребра графа также имеют атрибуты: тип дорожного покрытия, количество полос, ширину проезжей части и

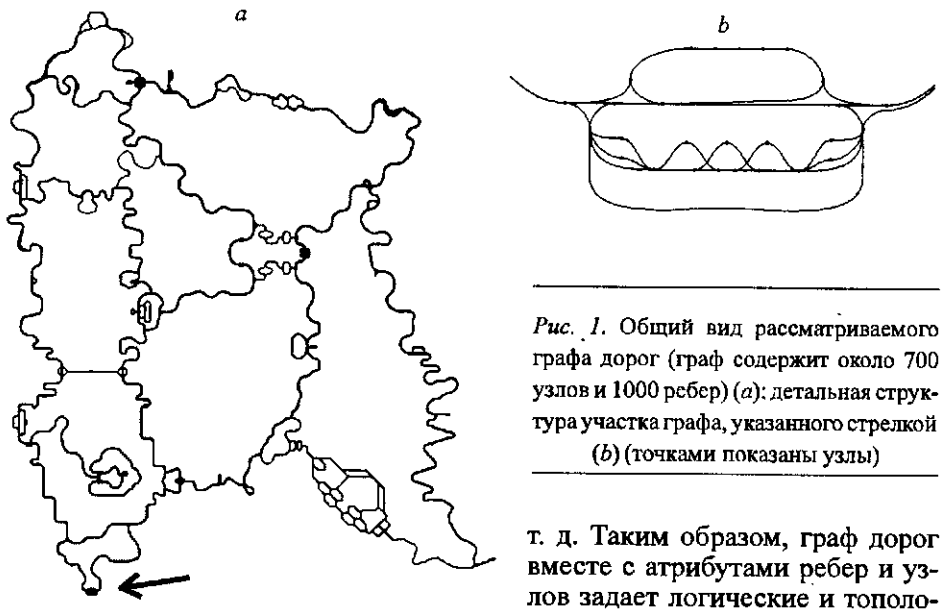

Рис. 1. Общий вид рассматриваемого графа дорог (граф содержит около 700 узлов и 1000 ребер) (*a*): детальная структура участка графа, указанного стрелкой (*b*) (точками показаны узлы)

т. д. Таким образом, граф дорог вместе с атрибутами ребер и узлов задает логические и топологические свойства моделируемой местности.

Автомобили обладают траекториями на графе; траектория – это последовательность узлов и ребер графа (развилок и дорог). Управляемые компьютером автомобили движутся вдоль этой траектории, а управляемые пользователем автомобили траектории в этом смысле не имеют, поскольку только пользователь знает, по какой именно траектории он собирается ехать. Однако если известен пункт назначения для автомобиля, управляемого пользователем, то может быть определена оптимальная траектория и использована для навигационной системы.

Граф дорог является основной структурой данных для рассматриваемой в работе подсистемы. В ведении этой подсистемы находится, в частности, назначение автомобилям их траекторий на графе.

Вся моделируемая территория геометрически поделена на участки, реализованные в виде отдельных файлов, которые загружаются по мере необходимости и могут быть добавлены или убраны из модели. Такой модульный принцип отчасти выполнен и для графа дорог (части графа определены в разных файлах по одному на модуль), который собирается на фазе загрузки. Это удобно при разработке модели территории, поскольку каждому модулю территории однозначно соответствует файл (модуль) графа дорог, и граф легко наращивать и модифицировать по модулям.

Задачи на графе дорог. В симуляторе необходимо устанавливать траектории автомобилей, замерять расстояния на графе (например, расстояние от пункта *A* до пункта *B*), моделировать навигационную систему автомобиля. В связи с этим возникает известная задача нахождения кратчайшего пути (или расстояния) на графе между двумя узлами. Эта задача решается известным алгоритмом Дейкстры [5, 6].

Весами ребер графа являются длины дорог с учетом типа дорожного покрытия, наличия препятствий на дорогах и прочих подобных условий.

Задача усложняется двумя факторами:

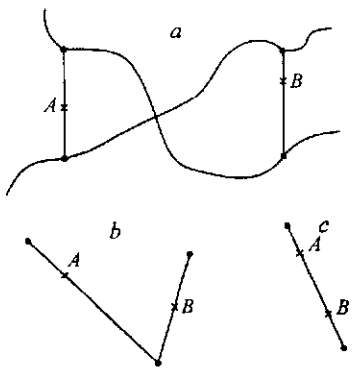

Рис. 2. Общая задача нахождения кратчайшего пути между произвольными точками (*A* и *B*) на графе (*a*): вырожденные случаи (*b*, *c*)

1) решать задачу приходится в реальном времени и настолько часто (а граф настолько большой), что классический алгоритм Дейкстры в чистом виде работает непозволительно долго;

2) полностью задача в нашем случае формулируется как нахождение кратчайшего пути между двумя произвольными точками (узлами или точками на ребрах), а не просто узлами (рис. 2).

Поэтому классический алгоритм был модифицирован.

Первая проблема решается введением кэша, т.е. памяти, в которой запоминаются результаты трассировки в приближении, что веса ребер и топология графа не меняются со временем. Топология графа в нашем случае, действительно, постоянна, а изменение весов ребер все-таки имеет место и диктуется изменением обстановки на дорогах, например ремонтными работами или образованием дорожного затора. Изменение весов происходит редко, например 1 раз в несколько минут, после чего кэш сбрасывается. Сброс кэша приводит к кратковременному ухудшению временных характеристик симулятора, поэтому такое изменение – относительно дорогая операция.

Кэш устроен следующим образом. В каждом узле для каждого другого узла назначения хранится номер следуюшего узла траектории. Это требует объема памяти, пропорционального квадрату количества узлов, но в нашем случае это оправдано. Фактически при работе алгоритма кэшируется не только путь до интересующей нас вершины, но и путь до промежуточных вершин, поэтому кэш заполняется быстро.

Вторая проблема решается применением алгоритма 4 раза взаимно попарно для соседних с интересующими нас точками узлов. Особые случаи (см. рис. 2) включают в себя ситуации, когда интересующие нас точки находятся на смежных ребрах или на одном и том же ребре. Все особые случаи корректно обрабатываются и, вообще говоря, дешевле в вычислительном смысле, чем полная задача. При этом подразумевается, что веса двух «псевдоребер», на которые ребро делится конечной (начальной) точкой, пропорциональны их длинам и в сумме дают полный вес ребра, иначе говоря, вес распределен по ребру равномерно. Применительно к нашей задаче это означает, что вдоль ребра дорожные условия не меняются.

Рассматривалась возможность введения двух псевдоузлов для интересующих нас точек с последующим выполнением алгоритма один раз вместо четырех. Это особенно усложняло реализацию кэша и увеличивало время работы алгоритма.

Таким образом, был разработан алгоритм, позволяющий находить кратчайшее расстояние между двумя произвольными точками на графе, при условии редко изменяющихся весов ребер графа.

2. **Параметрические полиномиальные кривые.** 2.1. *Общие подходы.* Для представления гладких участков дорог были выбраны параметрические

полиномиальные кривые (сплайны) [7, 8] третьей степени:

$$P(t) = \mathbf{A}t^3 + \mathbf{B}t^2 + \mathbf{C}t + \mathbf{D}, \quad t \in [0,1],$$

что позволяет более точно аппроксимировать изгибы дорог и не влияет отрицательно на динамику автомобиля. Использование ломаных не исключено для представления прямолинейных участков с изломами.

Параметрическая полиномиальная кривая представляется полиномом действительной переменной с векторными коэффициентами (или тремя полиномами с действительными коэффициентами). Переменная полинома называется параметром. Параметр кривой имеет область определения [0, 1] (начало области определения соответствует началу кривой, конец области – концу кривой). Выбранные границы области определения не принципиальны, это вопрос соглашения. Перевод из одной области определения в другую производится путем репараметризации.

Для вычисления значения полинома нужно всего лишь N сложений и N умножений, где N – степень полинома (для полинома третьей степени):

$$P(t) = (((\mathbf{A}t + \mathbf{B})t + \mathbf{C})t + \mathbf{D}).$$

Поскольку полином векторный, то и операции векторные. Это позволяет вычислить значение полинома в реальном времени.

Для оптимизации работы с полиномиальными кривыми нами создана специальная библиотека, позволяющая производить аналитические операции над полиномами, такие, как дифференцирование, интегрирование, комбинирование (репараметризация), сложение, вычитание, умножение и другие.

Поскольку полиномы позволяют аналитически решить ряд возникающих задач и тем самым оптимизировать работу алгоритма в целом, все вычисления были сведены к полиномам. Это само по себе представляет задачу, методы решения которой описаны далее. Как можно большая часть задачи решается на этапе загрузки аналитически (для этого и была построена библиотека аналитических операций), в результате чего получается решение в виде полинома (в той же области представления). В реальном времени полином вычисляется для разных значений параметра (свободной переменной). Поскольку вычисление значения полинома – сравнительно быстрая операция, такой подход позволил существенно ускорить работу алгоритмов.

В компьютерной графике широко используются рациональные полиномиальные кривые, являющиеся расширением обычных полиномиальных кривых. Если обычные полиномиальные кривые представлены полиномом, то рациональные – отношением полиномов. Пример таких кривых – NURBS (non-uniform rational B-splines). Рациональные кривые являются более широким классом кривых, в частности, они позволяют абсолютно точно представлять как дуги окружностей, так и произвольные конические сечения.

Алгоритмы на полиномиальных кривых, рассматриваемые в данной работе, легко расширяются на рациональные кривые. Например, дифференцирование отношения полиномов сводится к формуле дифференцирования дроби и в итоге может быть сделано аналитически. После дифференцирования опять получается отношение полиномов. То же самое относится к большинству других операций.

2.2. *Нахождение ближайшей точки.* Часто возникает задача привязки автомобиля к определенной точке на дороге (на графе дорог). Так, автомобиль может двигаться в произвольном направлении (съехать с дороги), но его положение на графе должно быть всегда определено для правильной работы многих алгоритмов: определения нужного направления; обеспечения возможности автомобилю, управляемому компьютером, вернуться на дорогу; замеров расстояний и т. д. Иначе говоря, входными данными для вышеописанных задач на графах является положение автомобиля на графе, поэтому оно должно быть всегда определено. Кроме того, по этому положению вычисляются направление движения, а также тип покрытия и все атрибуты участка дороги.

Задача сводится к нахождению минимума квадрата расстояния от точки X пространства до данной кривой $P(t)$. Это эквивалентно минимизации квадрата разности векторов, понимаемого как сумма квадратов координат разности:

$$\min_t (P(t) - X)^2, \quad t \in [0,1].$$

Задача решается на отрезке путем сравнения значений расстояний до критических точек, определяемых из уравнения

$$\frac{d}{dt}(P(t) - X)^2 = 0,$$

и крайних точек отрезка.

Заметим, что из полинома можно вычесть константу (она является полиномом нулевой степени), поэтому из кривой вычитаем X (сдвигаем точку X в начало координат), после чего аналитически вычисляем сумму квадратов компонент кривой и аналитически ее дифференцируем.

В итоге задача свелась к полиномиальному уравнению N-го порядка. Решается такое уравнение методом Ньютона. Метод Ньютона имеет недостатки, которые в данном случае легко обходятся. Самым главным недостатком является то, что метод при неудачном выборе начального приближения расходится. Кроме того, корням уравнения могут соответствовать несколько экстремумов. Нам же требуется абсолютный минимум на отрезке, а не любой локальный экстремум.

Поэтому на каждой итерации метода Ньютона подсчитываем значение минимизируемой функции и в итоге берем минимум по всем итерациям. Итерации делаем для нескольких начальных приближений, равномерно распределенных по отрезку. Независимо от того, как поведет себя метод Ньютона, берем наилучшее решение по всем итерациям при всех начальных приближениях. Кроме того, на каждой итерации при выходе решения за область определения полинома [0, 1] приводим его к диапазону путем простого присваивания соответствующего значения границы (0 или 1) в зависимости от того, за какую границу решение ушло. При такой стратегии в случае расхождения метода получаем точность не менее шага начальных приближений. Оценка сверху необходимого количества начальных приближений не проводилась, количество выбиралось эмпирически. Критерием служило поведение алгоритма на наборе кривых из конкретного симулятора, ошибка нахождения ближайшей точки не должна быть больше нескольких сантиметров.

Алгоритм легко расширяется на рациональные полиномиальные кривые (например, NURBS).

Решать эту задачу для всех кривых графа дорог непозволительно дорого, поиск ближайшей точки производится для кривых в окрестности исходной точки. Эта окрестность определяется при создании модели. Вся модель поделена на области, и при поиске принимаются во внимание только дороги, проходящие через область, в которой находится исходная точка. Кроме того, каждая кривая имеет габаритную сферу (круг, если рассматривать плоскостную задачу), что позволяет исключить из рассмотрения кривую, если в процессе выполнения алгоритма уже была найдена более близкая точка, чем любая другая на этой кривой.

2.3. *Длина участка кривой*. Нахождение длины участка полиномиальной кривой – задача, возникающая в симуляторе очень часто. Эта длина появляется как составляющая веса дуги при трассировке, а также необходима для обеспечения равномерного движения автомобилей по траектории.

Как хорошо известно, длина участка полиномиальной кривой определяется по формуле

$$L = \int_{t_0}^{t_1} \sqrt{(\dot{P}(t))^2}\, dt.$$

К сожалению, этот интеграл не вычисляется в квадратурах. Применение стандартных процедур численного интегрирования требует, в частности, вычисления значений подынтегральной функции (своей для каждой операции интегрирования) на разбиении отрезка и в итоге приводит к нарушению требования реального времени (является непозволительно дорогим). Поэтому подынтегральное выражение интерполируется методом Чебышева [9], после чего получившийся полином аналитически интегрируется и используется в реальном времени для вычисления собственно длины участка кривой.

Оценка сверху ошибки такого приближения не производилась, но экспериментальным методом было установлено, что восьмой порядок полиномов Чебышева дает на данных конкретного симулятора ошибку не более 10^{-5}, что достаточно для рассматриваемого приложения.

После введения этого алгоритма доля процессорного времени, которое тратилось на вычисление длин кривых, сократилась в десятки раз и составила доли процента от всего цикла симулятора. Алгоритм легко расширяется на рациональные полиномиальные кривые.

Таким образом, был разработан алгоритм, позволяющий приближенно вычислять длину участка полиномиальной кривой с высокой точностью в реальном времени.

3. **Непротиворечивость положения автомобиля.** Автомобили, управляемые компьютером, в симуляторе наделены ролями. Это может быть автомобиль-статист, создающий движение на дороге, дорожный полицейский или автомобиль с миссией. Автомобили с миссией нужны для моделирования различных ситуаций. Самая тривиальная миссия – это доставка груза из пункта A в пункт B. Кроме того, возможны миссии преследования (когда один автомобиль следует за другим), случайного блуждания и т. д. В зависимости от роли автомобиль следует разным сценариям поведения. Так, статисты порождаются и уничтожаются вокруг наблюдателя на некотором расстоянии от него, создавая видимость плотного движения. Автомобиль может,

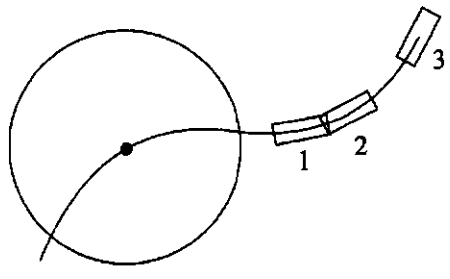

Рис. 3. Иллюстрация к алгоритму корректировки скоростей автомобилей для обеспечения визуальной непротиворечивости положения

например, просто стоять на стоянке. Автомобили должны останавливаться перед светофорами, объезжать препятствия и т. д., но описание этих механизмов выходит за рамки данной работы.

Автомобиль становится виден при переходе в непосредственную близость к наблюдателю, и поэтому для него включается более детальный и сложный алгоритм моделирования поведения. Для правильной работы этого алгоритма должны быть выполнены некоторые условия, например, автомобили должны быть на не менее чем определенном расстоянии друг от друга. Так, на рис. 3 автомобили 1 и 2 перекрываются, а 2 и 3 нет.

Нами разработан алгоритм, который корректирует скорости автомобилей, чтобы перед входом в зону видимости по мере приближения к наблюдателю автомобили располагались правильным образом. Алгоритм итеративный и выполняется по одной итерации на кадр. На каждой итерации автомобили, находящиеся в недопустимой близости друг к другу, раздвигаются соответствующим образом вдоль своих траекторий. Так, автомобиль 1 на рис. 3 продвигается влево, а автомобиль 2 – вправо. Имеющаяся реализация алгоритма учитывает полосы движения, на которых находятся автомобили, возможность начального перекрытия произвольного количества автомобилей (а не только двух), а также произвольную топологию дорог, на которых они при этом находятся.

Заключение. Разработан набор специализированных алгоритмов на графах и алгоритмов на полиномиальных кривых, позволяющих эффективно моделировать поведение большого количества автомобилей на сети дорог. Алгоритмы учитывают специфические требования систем реального времени. Для оптимизации полиномиальных алгоритмов была разработана библиотека аналитических операций над полиномами.

СПИСОК ЛИТЕРАТУРЫ

1. **Adzima J.** AI Madness: Using AI to Bring Open-City Racing to Life. http://www.gamasutra.com/features/20010124/adzima_01.htm
2. **Белаго И. В., Бартош В. С., Некрасов Ю. Ю., Роговой И. В.** Специфика разработки программной системы обучения вождению // 3-й Сибирский конгресс по прикладной и индустриальной математике (INPRIM98), Новосибирск, 22–27 июня, 1998. Новосибирск: Изд-во ИМ СО РАН, 1998.

3. **Бартош В. С., Лаврентьев М. М.** Динамическая модель автомобиля в реальном времени // Автометрия. 2000. № 4. С. 108.

4. **Белаго И. В., Некрасов Ю. Ю., Кузиковский С. А.** Возможности применения современных персональных ЭВМ для разработки систем трехмерной аудиовизуальной имитации // Ежегодный научно-технический семинар «Технические средства и технологии для построения тренажеров», Москва, Звездный городок, октябрь 1998.

5. **Dijkstra E. W.** A note on two problems in connection with graphs // Numer. Math. 1959. **1**. P. 269.

6. **Липский В.** Комбинаторика для программистов. М.: Мир, 1988.

7. **Boehm W., Farin G., Kahmann J.** A survey of curve and surface methods in CAGD // Comput. Aided Geom. Des. 1984. **1**. N 1.

8. **Yamaguchi F.** Curves and Surfaces in Computer Aided Geometric Design. Berlin: Springer-Verlag, 1988.

9. **Корн Г., Корн Т.** Справочник по математике. М.: Наука, 1984.

Институт автоматики и электрометрии СО РАН,
E-mail: stas@sl.iae.nsk.su

Поступила в редакцию
10 сентября 2002 г.