С.А.Кузиковский, М.М.Лаврентьев, И.В.Белаго, В.С. Бартош

«СофтЛаб-НСК», Новосибирск

Проблемы Управления Движением Виртуальных Автомобилей На Сети Дорог

1. АННОТАЦИЯ[1]

В докладе описаны некоторые задачи, возникающие при построении компьютерных автомобильных симуляторов (АС) на базе персональных компьютеров. Приведены возможные применения АС. Перечислены компоненты типичного АС. В первой части рассмотрена задача, связанная с сетью дорог – нахождение кратчайшего пути между двумя произвольными точками на этой сети. Во второй части рассматриваются задачи над параметрическими полиноминальными кривыми, при помощи которых заданы участки дорог. Это нахождение точки на кривой, ближайшей к заданной и нахождение длины участка кривой. В третьей части коротко описан алгоритм обеспечения непротиворечивости положений автомобилей на дороге.

Ключевые слова: Автомобильный симулятор, алгоритм Дейкстры, полиноминальные кривые.

2. ВВЕДЕНИЕ

В последние годы наблюдается повышенный интерес к автомобильным симуляторам на базе персональных компьютеров (PC) [1]. Это связано, в первую очередь, с ростом производительности PC и графических акселераторов для них, что позволяет как производить адекватное моделирование динамики автомобиля [5], так и графическое отображение моделируемого окружения с высоким качеством [7]. Ранее сравнимые параметры имели только дорогие специализированные системы со значительными вычислительными ресурсами. Стандартизация графических акселераторов для PC позволяет существенно расширить пользовательскую базу такого рода приложений.

Автомобильный симулятор может использоваться для обучения вождению, в научных целях, а также как основная часть компьютерных игр соответствующего (автомобильного) направления. Научное использование симуляторов включает в себя те применения, где использование реальных автомобилей было бы дорогим и небезопасным, например 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 наличие 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 может потребоваться реализации моделирования времени суток, погодных условий, внештатных ситуаций и др.

Симулятор является системой реального времени, иначе говоря, спецификой работы симулятора, в отличие от других вычислительных задач, является требование малого времени отклика и требование высокой кадровой частоты графического отображения. Минимальная кадровая частота, в свою очередь, определяется временем реакции человека и физиологией человеческого зрения, и не может быть уменьшена. Эти требования связаны, и выражаются в том, что цикл работы симулятора, в течение которого должен быть произведен очередной шаг моделирования и отображение очередного кадра, должен быть не более нескольких десятков миллисекунд. 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 (thread); в противном случае всё 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 собственно при 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) [7], подсистема динамики, которая осуществляет моделирование динамики автомобиля (автомобилей) [1], и, наконец, подсистема, которая отвечает

[1] Работа выполнена при финансовой поддержке Российского Фонда Фундаментальных Исследований Грант No 00-15-99092

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 др. При реализации такой подсистемы возникает ряд типичных задач, которые и рассмотрены в данном докладе.

В первой части рассмотрена задача, связанная с сетью дорог – нахождение кратчайшего пути между двумя произвольными точками на этой сети. Во второй части рассматриваются задачи над параметрическими полиноминальными кривыми, при помощи которых заданы участки дорог. Это нахождение точки на кривой, ближайшей к заданной и нахождение длины участка кривой. В третьей части коротко описан алгоритм обеспечения непротиворечивости положений автомобилей на дороге.

3. ГРАФ ДОРОГ

Сеть дорог на местности нами рассматривается как граф (Рис. 1). Перекрестки, развилки и т.п. элементы являются узлами этого графа, а сами дороги – ребрами этого графа.

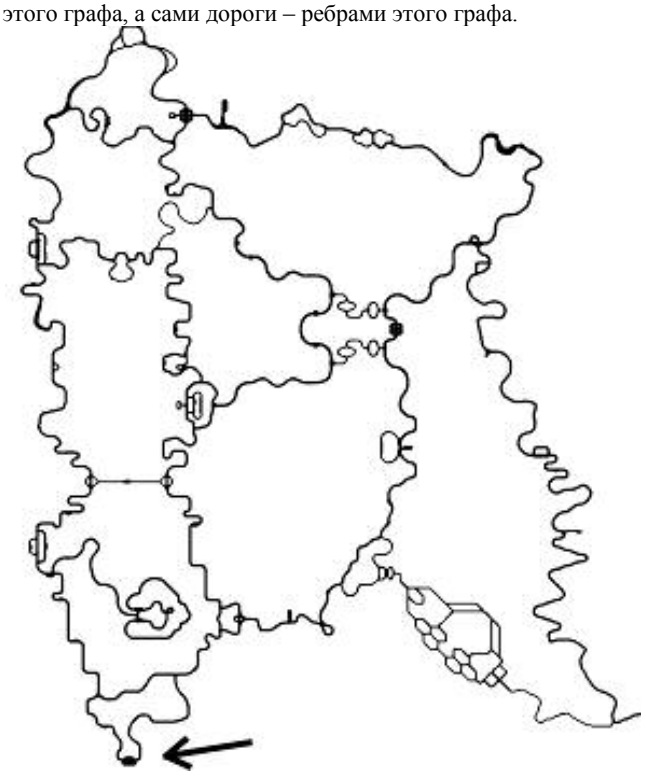

Рис. 1 – Общий вид рассматриваемого графа дорог. Граф содержит около 700 узлов и 1000 ребер. На этом рисунке не видны отдельные узлы. Детальная структура участка графа, указанного стрелкой показана на Рис. 2

Узлы графа имеют набор аттрибутов, управляющих свойствами узлов. Например, узлом является место для парковки автомобиля, место на заправочной станции и т. д.

Ребра графа также имеют аттрибуты, такие как тип дорожного покрытия, количество полос, ширина проезжей части и т. д.

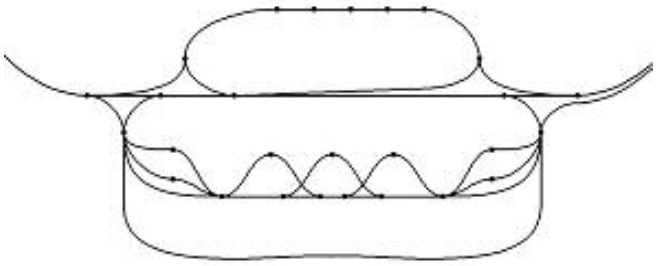

Рис. 2 – увеличенное изображение участка графа дорог (указанного стрелкой на Рис. 1). Точками показаны узлы.

Таким образом, граф дорог, вкупе с аттрибутами 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). Автомобили, управляемые компьютером, движутся вдоль этой траектории. Автомобили, управляемые пользователем, траектории в этом смысле не имеют, поскольку только пользователь знает, по какой именно траектории он собирается ехать. Однако, если известен пункт назначения для пользователя, то может быть определена оптимальная траектория, которая будет использована при имитации навигационной системы автомобиля.

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

Вся моделируемая территория геометрически была поделена на участки. Эти участки реализованы в виде отдельных файлов, которые загружаются по мере необходимости, могут быть добавлены или убраны из модели. Этот модульный принцип отчасти выполнен и для графа дорог – части графа определены в разных файлах (по одному на модуль) и собственно граф собирается на фазе загрузки.

Это удобно при разработке, тем, что каждому модулю территории однозначно соответствует файл (модуль) графа дорог, и тем, что граф легко наращивать и модифицировать по модулям.

Задачи на графе дорог

В симуляторе необходимо устанавливать траектории автомобилей, замерять расстояния на графе, например для ответа на вопрос о расстоянии от пункта А в пункт В, моделировать навигационную систему автомобиля. В связи с этим возникает известная задача о нахождении кратчайшего

пути (или нахождения расстояния) на графе между двумя узлами.

Эта задача решается известным алгоритмом Дейкстры [3] [10].

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

Задача усложняется двумя факторами:

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. 3).

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 один раз в несколько минут; после этого 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 следующего узла траектории. Это требует памяти объема, 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 и до изменения весов ребер просто используется.

Вторая проблема решается применением алгоритма четыре раза, взаимно попарно для соседних с интересущими нас точками узлов. Особые случаи (Рис. 3) 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. В применении к нашей задаче это означает, что вдоль ребра дорожные условия не меняются.

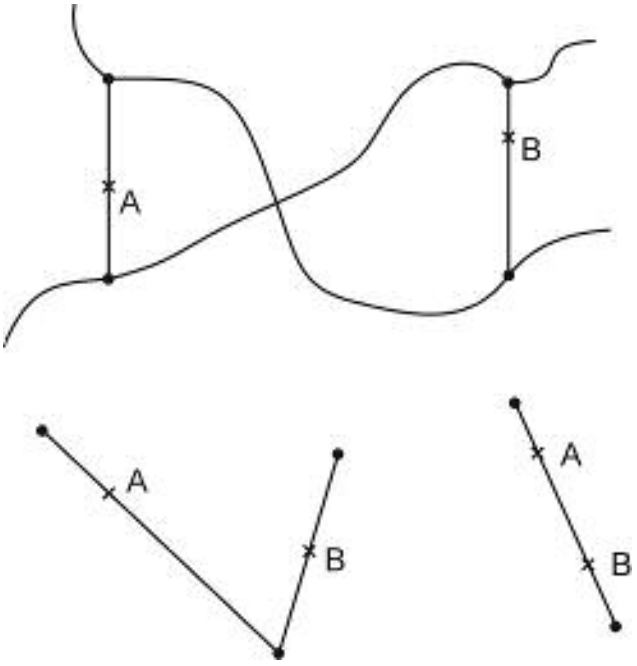

Рис. 3 – общая задача нахождения кратчайшего пути между произвольными точками на графе (A и B). В нижней части рисунка показаны вырожденные случаи.

Рассматривалась возможность введения двух псевдоузлов для интересующих нас точек, с последующим выполнением алгоритма один раз вместо четырех, этот алгоритм был реализован, но оказался сложнее, особенно сильно усложнял реализацию кэша, и, главное, работал дольше.

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

4. ПАРАМЕТРИЧЕСКИЕ ПОЛИНОМИНАЛЬНЫЕ КРИВЫЕ

Общие подходы

Для представления гладких участков дорог были выбраны параметрические полиноминальные кривые (сплайны) [1][4] третьей степени.

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

По сравнению с ломаными это обеспечивает компактность представления, позволяет более точно аппроксимировать изгибы дорог и не имеет отрицательных влияний на динамику автомобиля. Это не исключает использование ломаных там, где это удобно для представления прямолинейных участков с изломами.

Параметрическая полиноминальная 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 – 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, это видно из следующей записи (для полинома третьей степени):

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

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

Для оптимизации работы с полиноминальными кривыми была написана специальная библиотека, позволяющая делать аналитические операции над полиномами, такие как дифференцирование, интегрирование, комбинирование (репараметризация), сложение, вычитание, умножение и др.

Поскольку полиномы позволяют аналитически решать 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 (т.е. в той же области представления). В реальном времени полином вычисляется для разных значений параметра (свободной переменной). Поскольку вычисление значения полинома – сравнительно быстрая операция, указанный подход позволил добиться существенного ускорения работы алгоритмов.

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

Алгоритмы на полиноминальных кривых, рассматриваемые в данном докладе, 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.

Нахождение ближайшей точки

Часто возникает задача привязки автомобиля к определенной точке на дороге (графе дорог). Так, автомобиль может двигаться в произвольном направлении (съехать с дороги), но его положение на графе должно быть всегда определено. Это нужно для правильной работы многих алгоритмов – для определения нужного направления; для обеспечения возможности автомобилю, управлямому компьютером, вернуться на дорогу; для замеров расстояний и т. д. Иначе говоря, входными данным для вышеописанных задач на графах являтся положение автомобиля на графе, поэтому оно и должно быть всегда определено. Кроме того, по этому положению вычисляются направление движения, а также тип покрытия и все аттрибуты участка дороги, о которых уже было сказано выше.

Эта 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.

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

Задача решается на отрезке путем сравнения значений расстояния до кричитеских точек, определяемых из уравнения

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

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, после чего аналитически ее дифференцируем.

В итоге задача свелась к полиноминальному 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 мы подсчитываем значение минимизируемой функции и в итоге берем минимум по всем итерациям. Итерации делаем для нескольких начальных приближений, равномерно распределенных по отрезку. Вне зависимости от того, как поведет себя метод Ньютона, мы берем наилучшее решения по всем итерациям при всех начальных приближениях. Кроме того, на каждой итерации, при выходе решения за область определения полинома [0..1] мы 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, мы получаем точность не менее шага начальных приближений. Оценка сверху необходимого количества начальных приближений не проводилась, количество это выбиралось эмпирически. Критерием служило поведение алгоритма на наборе кривых из конкретного симулятора; ошибка нахождения ближайшей точки не должна была быть больше нескольких сантиметров.

Алгоритм легко расширяется на рациональные полиноминальные кривые (такие как 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 набор областей и при поиске принимаются во внимание только те дороги, которые проходят 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, нежели любая другая на этой кривой.

Длина участка кривой

Нахождение длины участка полиноминальной 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 полиноминальной кривой определяется по формуле

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

К сожалению, этот интеграл вообще говоря не вычисляется в квадратурах. Применение стандартных процедур численного интегрирования требует, в частности, вычисления значений подинтегральной функции (своей для каждой операции интегрирования) на разбиении отрезка, и, в конечном итоге, приводит к нарушению требования реального времени (является непозволительно дорогим).

Поэтому подинтегральное выражение интерполируется методом Чебышева [9], после чего получившийся полином аналитически же интегрируется. Получившийся в результате интегрирования полином затем 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 даёт на данных конкретного симулятора ошибку не более 10e-5, что более, чем достаточно для рассматриваемого приложения.

После введения этого алгоритма, доля процессорного времени, которое тратилось на вычисление длин кривых сократилось в десятки раз, и составило доли процента от всего цикла симулятора.

Алгоритм легко расширяется на рациональные полиноминальные кривые.

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

5. АВТОМОБИЛИ – НЕПРОТИВОРЕЧИВОСТЬ ПОЛОЖЕНИЯ

Автомобили, управляемые компьютером, в симуляторе обладают 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 А в пункт В. 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 создаются и уничтожаются вокруг наблюдателя, на некотором расстоянии от него, создавая видимость плотного движения. Автомобиль может, например, просто стоять на стоянке. Автомобили должны останавливаться на светофорах, объезжать препятствия и прочее.

Наличие сложных миссий, таких как доставка груза из одного пункта в другой, диктует необходимость моделирования поведения каждого такого автомобиля на всей территории, вне зависимости от того, как далеко от наблюдателя он находится. Однако, в случае, если автомобиль находится вне области непосредственной видимости наблюдателя, для моделирования его поведения используются упрощенные алгоритмы, позволяющие экономить вычислительные ресурсы. Так, например, вне области видимости может не учитываться взаимное расположение автомобилей и не производиться детальная имитация их динамики.

При переходе автомобиля в непосредственную близость к наблюдателю он становится виден, 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. 4 автомобили 1 и 2 – перекрываются, а 2 и 3 – нет.

Существует алгоритм, который корректирует скорости автомобилей таким образом, 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 от друга, раздвигаются соответствующим образом вдоль своих траекторий. Так, автомобиль 1 на Рис. 4 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.

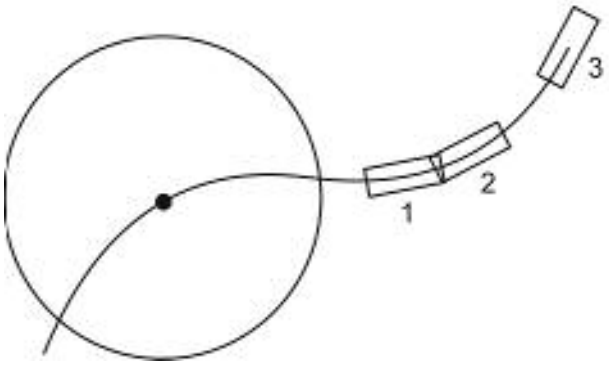

Рис. 4 – иллюстрация к алгоритму корректировки скоростей автомобилей для обеспечения визуальной непротиворечивости положения.

6. ВЫВОДЫ

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

7. ЛИТЕРАТУРА:

[1] Adzima, Joe: “AI Madness: Using AI to Bring Open-City Racing to Life”, *Gamasutra*, January 24, 2001, URL: http://www.gamasutra.com/features/20010124/adzima_01.htm

[2] Boehm, W., Farin, G., Kahmann J.: “A survey of curve and surface methods in CAGD” *in Computer Aided Geometric Design*, Volume 1, Number 1, July 1984, North-Holland – Amsterdam

[3] Dijkstra E. W.: “A note on two problems in connexion with graphs”, Numer. Math., 1959, 1, p 269-171

[4] Yamaguchi F.: “*Curves and Surfaces in Computer Aided Geometric Design*”, Springer-Verlag, 1988

[5] Бартош, В.С., Лаврентьев М.М.: “Динамическая модель автомобиля в реальном времени”, Новосибирск, *Автометрия*, 2000, 4, стр 108-115

[6] Белаго И.В., Бартош В.С., Некрасов Ю.Ю., Роговой И.В.: “Специфика разработки программной системы обучения вождению”, *Третий сибирский конгресс по прикладной и индустриальной математике (INPRIM98)*, Новосибирск, 22-27 июня, 1998г

[7] Белаго И.В., Некрасов Ю.Ю., Кузиковский С.А.: “Возможности применения современных персональных ЭВМ для разработки систем трехмерной аудио визуальной имитации” *Ежегодный научно-технический семинар “Технические средства и технологии для построения тренажеров”,* Москва, Звездный городок, октябрь 1998г.

[8] Гилой, В.: “Интерактивная машинная графика”, Москва, “Мир”, 1981

[9] Корн, Г., Корн, Т.: “*Справочник по математике*” Москва, “Наука”, 1984

[10] Липский В.: “*Комбинаторика для программистов*”, Москва, “Мир”, 1988

Об авторах:

Все авторы работают в компании Софтлаб-НСК, Новосибирск.

e-mail:

С.А.Кузиковский – stas@softlab-nsk.com
М.М.Лаврентьев – lavr@softlab-nsk.com
И.В.Белаго – bel@softlab-nsk.com
В.С. Бартош – vas@softlab-nsk.com