Реализация контроллера следования по пути из системы 3D Studio MAX[1]

В.М. Никифоров, М.А. Торгашев

Аннотация. В статье рассмотрена задача реконструкции контроллера следования по пути из системы трехмерного моделирования 3D Studio Max и реализации найденных алгоритмов в системе визуализации. Решение поставленной задачи включает поиск и реализацию алгоритмов вычисления положения и ориентации объектов в промежуточных кадрах с учётом всех параметров, установленных для контроллера следования по пути в системе моделирования.

Введение

В системе трехмерного моделирования 3D Studio MAX для задания анимационных треков объектов используются так называемые контроллеры – функции интерполяции анимируемых параметров. Эти контроллеры реализуют технологию ключевой анимации, суть которой состоит в том, что значение параметра устанавливается в нескольких ключевых кадрах - «ключах», в промежуточных же кадрах для вычисления значения параметра проводится интерполяция между соседними ключами. В качестве возможных контроллеров можно назвать контроллер интерполяции по кривой Безье, контроллер линейной интерполяции и т.д. К их числу относится также контроллер следования по пути, который позволяет задать движение объекта по заранее созданной траектории. Путь задается в виде кривой Безье и состоит из нескольких сегментов. Дополнительные параметры позволяют тонко регулировать перемещение объекта на кривой в процессе анимации. Важными достоинствами контроллера следования по пути является простота и удобство создания нужной траектории, а также высокая компактность описания анимации – довольно сложные траектории описываются лишь небольшим набором вершин кривой и несколькими дополнительными параметрами.

Для синтеза в системах визуализации виртуальных сцен, созданных в системах трехмерного моделирования, в частности 3D Studio MAX, необходимо воссоздать как внешний вид: геометрию и раскраску объектов виртуальной сцены, так и их анимацию, для чего необходимо в точности реализовать методы анимации, используемые системой моделирования. Точный алгоритм и детали реализации того или иного метода в системе моделирования часто бывают неизвестны и/или скрыты намеренно для защиты интеллектуальной собственности, поэтому для их реализации требуется восстановление, реконструкция алгоритмов по конечным результатам, а также по отдельным, часто обрывочным и неполным, описаниям в системах справки и по исходным кодам в пакете разработки (SDK), входящем в состав системы трехмерного моделирования. В статье рассмотрена задача реконструкции контроллера следования по пути в системе визуализации, что подразумевает реализацию алгоритма интерполяции в промежуточных кадрах с учётом всех параметров, установленных для контроллера следования по пути в системе моделирования.

[1] Работа выполняется при поддержке РФФИ (грант № 07-07-00161-а)

1. Параметры контроллера следования по пути

Рассмотрим все параметры рассматриваемого контроллера в системе трехмерного моделирования.

К параметрам контроллера относится собственно сплайн, являющийся траекторией движения объекта, и дополнительные параметры и флаги, описывающие, как именно будет двигаться объект по сплайну.

Сплайн состоит из множества вершин, соединённых между собой сегментами. Сегменты являются кубическими кривыми Безье и строятся по формуле:

$$P(t) = t^3 * P_{i-1} + 3 * t^2 * (t-1) * P_{i-1,1} + \\ + 3 * t * (t-1)^2 * P_{i,2} + (t-1)^3 * P_i,$$

где P_i - i-я вершина кривой, $P_{i,1}$ - выходящий вектор из i-ой вершины, $P_{i,2}$ - входящий вектор в i-ую вершину.

На Рис. 1 показан пример сплайна, состоящего из трех вершин.

Положение объекта на сплайне в текущем кадре определяется с использованием кривой процента. Кривая процента может быть задана с помощью произвольного контроллера, например, контроллера Безье [1], контроллера TCB (разновидность эрмитовой кривой, управляемой параметрами Tension – натяжение, Continuity - непрерывность, Bias - сдвиг), линейного контроллера.

Кривая процента, показанная на Рис. 2, построена с помощью линейного контроллера. На кривой установлены 2 ключевых кадра (в 5-ом и 50-ом кадрах). В 5-ом кадре процент равен 0%, в 50-ом кадре - 100 %. В промежуточных кадрах процент вычисляется с помощью линейной интерполяции.

Процент, полученный на основе кривой процента в текущем кадре, может интерпретироваться двумя способами. Если у контроллера следования по пути установлен флаг *ConstVel*, то процент будет соответствовать пройденной длине сплайна от его начала. Если флаг не установлен, то процент будет соответствовать локальному времени на сплайне. В таком случае локальное время на сплайне равномерно распределяется между его сегментами, вне зависимости от их длины, как это показано на Рис. 3. Значения процента *П* в каждой вершине сплайна соответствуют локальному времени на сплайне. Каждому сегменту в примере соответствует 25% от локального времени всего сплайна.

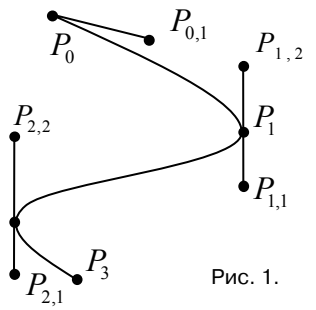

Рис. 1.

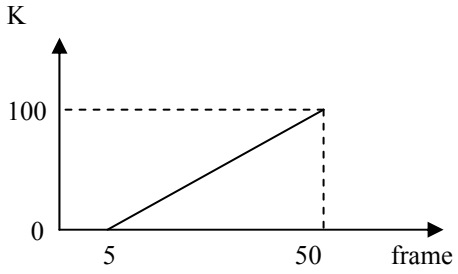

Рис. 2.

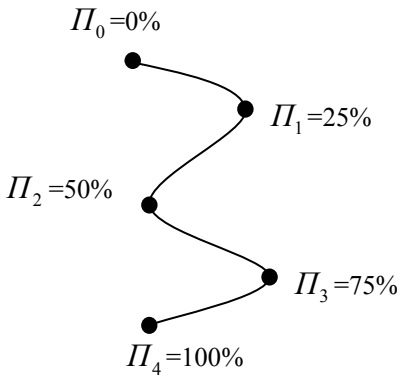

Рис. 3.

Процент, полученный из процентной кривой в текущем кадре, может иметь произвольное значение, однако величина процента на сплайне имеет смысл только в диапазоне от 0% до 100%, поэтому после получения значения процента необходимо привести его к этому диапазону. Для этого возможны два способа. Если установлен флаг *Loop*, то при достижении

конца сплайна объект продолжает своё движение из начала сплайна. Если флаг не установлен, то при значениях процента, меньших 0% или больших 100%, объект будет оставаться в начале или конце сплайна соответственно.

Объект, к которому применяется контроллер движения по сплайну, можно заставить следовать кривой не только по положению, но и по ориентации, для чего в системе 3D Studio Max задается флаг *Follow*. В таком случае одна из осей его координатной системы будет ориентирована по касательной к сплайну. Конкретная ось привязки объекта задается с помощью параметра *Axis*. Дополнительный флаг *Flip* дает возможность зеркального отражения координатных осей объектной системы.

По умолчанию для вычисления оси Y объекта, при установленном флаге *Follow*, берётся векторное произведение касательной к сплайну и оси Z мировой системы координат (WCS). Но если касательная к сплайну и ось Z WCS сонаправлены, то их векторное произведение равно нулю, вследствие чего при вычислении координатной оси Y объекта возникает ошибка деления на ноль. Чтобы избежать этой проблемы, в системе моделирования предусмотрен флаг *AllowUpsideDown*, при установке которого ось Y будет рассчитываться как векторное произведение касательной к сплайну и усредненной нормали к сплайну.

При прохождении изгибов сплайна объект может наклоняться вдоль оси движения для правильного распределения центробежной силы так, как это делает самолёт при поворотах. В системе 3D Studio Max для реализации этого эффекта предусмотрен флаг *Bank*. Величина поворота регулируется значениями параметров *Smoothness* и *BankAmount*. Эти значения, как и значения процента нахождения на кривой, устанавливаются с помощью отдельных контроллеров и имеют свои кривые значений.

При использовании контроллера можно задать еще один флаг - *Relative*, который позволяет задать не абсолютное, а относительное движение по сплайну, в таком случае траекторией объекта будет тот же сплайн, сдвинутый в точку первоначального расположения объекта. Для упрощения выкладок реализация этого параметра в статье не приведена.

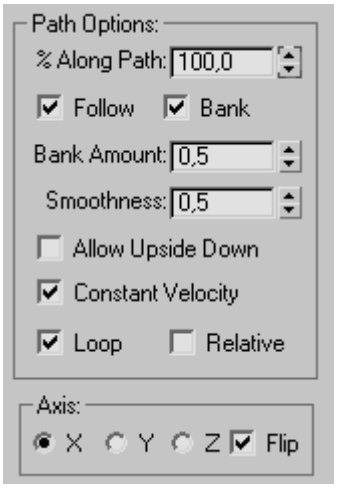

Рис. 4.

На Рис. 4 приведен вид панели свойств контроллера следования по пути из системы 3D Studio Max.

Итак, приведем еще раз список параметров контроллера следования по пути в системе моделирования 3D Studio Max, их краткое описание и исходные значения:

Сплайн $(P_i, P_{i,1}, P_{i,2})$	- кривая Безье, P_i - *i*-я вершина кривой, $P_{i,1}$ - выходящий вектор из *i*-й вершины, $P_{i,2}$ - входящий вектор в *i*-ю вершину, $0 \le i \le (N-1)$;
Контроллер процента PercentCurve	- задается в виде набора ключей и функции интерполяции, определяет кривую процента;
Флаг *ConstVel*	- флаг постоянной скорости, определяет как интерпретируется значение процента, значение по умолчанию *ConstVel* =1;
Флаг *Loop*	- флаг зацикленного движения по кривой, значение по умолчанию *Loop* =0;
Флаг *Follow*	- флаг следования осей объекта за кривой, значение по умолчанию *Follow* = 0;

Ось *Axis*	- ось привязки объекта к касательной на кривой (осям X,Y,Z – соответствуют значения 0,1,2), имеет смысл только в случае *Follow* = 1, значение по умолчанию *Axis* = 0;
Флаг *Flip*	- флаг зеркального переворота осей объекта при его следовании по пути, имеет смысл только в случае *Follow* = 1, значение по умолчанию *Flip*=0;
Флаг *AllowUpsideDown*	- флаг, регулирующий вычисление ориентации объекта при его следовании по пути, имеет смысл только в случае *Follow* =1, значение по умолчанию *AllowUpsideDown* = 0;
Флаг *Bank*	- флаг поворота объекта по крену в зависимости от кривизны кривой, имеет смысл только в случае *Follow* =1, значение по умолчанию *Bank* = 0;
Контроллер *SmoothnessCurve*	- задает кривую для параметра *Smoothness,* регулирующего степень крена объекта, имеет смысл только в случае *Follow* =1 и *Bank* = 1;
Контроллер *BankAmountCurve*	- задает кривую для параметра *BankAmount,* регулирующего степень крена объекта, имеет смысл только в случае *Follow* = 1 и *Bank* = 1;

2. Реализация контроллера следования по пути

Реализация контроллера следования по пути заключается в поиске функции для вычисления значений положения и ориентации объекта в произвольном кадре. Для решения этой задачи был проведен большой объем исследований с поиском любой существующей информации по реализации рассматриваемого контроллера: в системе помощи пакета 3D Studio Max, во входящем в его состав пакете для разработчиков (MAX SDK), а также в специализированных форумах для разработчиков во всемирной сети Интернет [4]. После этого были проведены тщательное тестирование, проверка и отладка алгоритмов. Рассмотрим отдельно две независимые подзадачи - вычисление положения объекта в виде вектора и вычисление ориентации объекта в виде матрицы поворота.

2.1. Вычисление положения объекта

Для определения положения объекта в заданный момент времени на кривой первоначально необходимо вычислить значение процента. Как уже было отмечено, у процента есть собственный контроллер, задающий кривую процента. Для текущего кадра анимации *frame* необходимо вычислить значение процента с помощью вызова функции контроллера:

$$k = PercentCurve(frame).$$

При параметрах, установленных в системе трехмерного моделирования по умолчанию, объект движется по сплайну равномерно (флаг *ConstVel* = 1). Это означает, что процент нахождения на сплайне в таком случае интерпретируется как пройденная длина кривой от её начала.

По умолчанию также принимается, что если процент выходит за пределы от 0% до 100%, то объект продолжает движение с начала или конца сплайна соответственно (флаг *Loop* = 1). Рассмотрим алгоритм приведения значения процента *k* к диапазону от 0% до 100%.

Алгоритм приведения процента к диапазону [0,100]

- $k_1 = |k| \% 100$;
- Если $k_1 = 0$, то:
  - Если $k \le 0$, то: $P = P_0$;
  - Иначе: $P = P_{N-1}$;
- Иначе
  - Если $k < 0$, то:
  - $k = 100 - k_1$

Для вычисления положения (точки *P*) объекта на сплайне сначала необходимо найти сегмент *i*, в который мы попали, и длину *dl*, пройденную на этом сегменте.

Сплайн, как известно, состоит из нескольких сегментов. Длины всех сегментов l_i нам известны. Найдём такой сегмент *i*, в котором длина dl пройденного пути в *i*-ом сегменте будет меньше или равна длине сегмента:

$$dl = L - \sum_{s=0}^{i-1} l_s < l_i,$$

где $L = S * k$; S - длина сплайна, k - процент в текущем кадре.

Для того чтобы найти точку *P*, отстающую на dl от начала сегмента *i*, сегмент заменяется ломаной *H*. Вершинами G_i ломаной *H* являются точки кривой Безье этого сегмента, взятые с локальным временем $j*\frac{1}{STEPS}$, где j – номер вершины, а *STEPS* – число сегментов ломаной (опытным путем было определено, что в системе 3D Studio Max используется значение этого параметра, равное 100). Путь dl_j, пройденный в *j*-ом сегменте ломаной *H,* равен:

$$dl_j = dl - \sum_{s=0}^{j-1} g_s,$$

где g_j - длина *j*-го сегмента ломаной H.

Тогда локальное время t на *j*-ом сегменте ломаной H вычисляется следующим образом:

$$t = \frac{dl_j}{g_j}.$$

Результирующую точку P можно найти линейной интерполяцией по времени t между вершинами G_j и G_{j+1}:

$$P(t) = t * G_j + (t-1) * G_{j+1}.$$

Далее рассмотрим алгоритм вычисления точки *P* при других значениях флагов *Loop* и *ConstVel*.

Если флаг *Loop* не установлен (*Loop* = 0), то при значении процента, большем 100%, процент принимается равным 100% и в качестве результирующего вектора положения берется конечная вершина сплайна. При значении, меньшем 0%, процент берется равным 0% и, соответственно, используется начальная вершина кривой Безье.

Если флаг *ConstVel* не установлен (*ConstVel* = 0), то процент соответствует локальному времени на сплайне. Каждой вершине P_i сплайна в таком случае соответствует процент Π_i на сплайне (локальное время сплайна):

$$\Pi_i = i * \frac{1}{N-1} * 100 \text{ , где } 0 \le i \le (N-1).$$

Поскольку значения этих параметров не зависят от времени, их вычисление можно и нужно вынести на этап предварительных вычислений.

Для текущего кадра необходимо найти, между какими двумя вершинами Π_{i-1}, Π_i находится текущее значение процента.

После этого можно вычислить локальное время t на сегменте $[\Pi_{i-1}, \Pi_i]$ по следующей формуле.

$$t = \frac{k - \Pi_{i-1}}{\Pi_i - \Pi_{i-1}}.$$

Далее по локальному времени *t* мы находим точку *P* на сегменте, подставив найденное время в формулу кривой Безье на соответствующем сегменте $[\Pi_{i-1}, \Pi_i]$:

$$P(t) = t^3 * P_{i-1} + 3 * t^2 * (t-1) * P_{i-1,1} + \\ + 3 * t * (t-1)^2 * P_{i,2} + (t-1)^3 * P_i.$$

Таким образом, полностью определена функция вычисления точки расположения на кривой в зависимости от значения процента k. Для удобства обозначим ее как:

$$P(k) = Curve(k).$$

2.2. Вычисление ориентации объекта

Как известно [3], для описания ориентации объекта существует несколько вариантов: ортонормированная матрица 3x3, нормированный кватернион либо тройка углов Эйлера-Крылова. Для данного случая наиболее удобным является представление поворота в виде матрицы 3x3. Известно, что в данном случае строки этой матрицы будут содержать координаты осей X, Y, Z объектной системы координат OCS в мировой системе WCS.

При параметрах, установленных по умолчанию, матрица поворота будет единичной, т.е. ориентация объекта будет совпадать с расположением мировой системы координат (*Follow* = 0).

Если флаг *Follow* установлен, то ось *Axis* объекта должна быть направлена по касательной к кривой. Рассмотрим для начала алгоритм нахождения матрицы, если остальные флаги контроллера, отвечающие за поворот объекта, установлены в значения по умолчанию.

Создадим матрицу *Q[3x3]*, строками которой будут новые значения осей объектной системы координат в мировой системе WCS. *Q[0]* – ось X, *Q[1]* – ось Y, *Q[2]* – ось Z. Нулевая строка матрицы будет равна нормированной касательной к сплайну в проценте *k*.

Было выяснено, что в системе трехмерного моделирования касательная вычисляется как вектор между точками P_1 и P_2 со значениями процента *k-dt* и *k+dt* соответственно (было определено, что в системе 3D Studio Max используется значение *dt = 0.001*).

$$P_1 = Curve(k - dt);\ P_2 = Curve(k + dt);$$

$$Q[0] = P_2 - P_1;$$

$$Q[0] = \frac{Q[0]}{\|Q[0]\|}.$$

Как было отмечено ранее, флаг *Flip* используется для зеркального отображения координатных осей объекта. Для применения флага на данном этапе достаточно поменять направление оси, направленной по касательной к сплайну, то есть, если флаг установлен, то:

$$Q[0] = -Q[0].$$

Остальные две строки матрицы (соответствующие осям Y и Z) вычисляются по формулам, обеспечивающим построение корректных векторов координатной тройки:

$$Q[1] = (0,0,1) \times Q[0] = (-Q[0]_y, Q[0]_x, 0),\ Q[1] = \frac{Q[1]}{\|Q[1]\|}.$$

$$Q[2] = Q[0] \times Q[1], \qquad Q[2] = \frac{Q[2]}{\|Q[2]\|}.$$

Если касательная к сплайну *Q[0]* и ось Z *(0,0,1)* сонаправленны, то при нормализации вектора Q[1] возникает ошибка деления на ноль, поэтому в таких случаях выбирается не ось Z, а глобальная ось Y (0,1,0).

Еще одним вариантом является использование флага *AllowUpsideDown*, при этом вместо глобальных осей Z или Y мировой системы координат используется усредненная нормаль к сплайну. Нормаль к сплайну постоянна и может быть вычислена на этапе предварительных вычислений.

Рассмотрим найденный в результате исследований алгоритм вычисления нормали к сплайну. Обозначим нормаль к кривой как N_c. Для вычисления нормали сначала находится средняя точка *c* сплайна по некоторому количеству элементов, после чего нормаль сплайна определяется усреднением нормалей треугольников, образованных вершинами *(v_{j-1} , v_j, c)*. Количество элементов является параметром алгоритма, было установлено, что в системе 3D Studio Max сплайн делится на 21 элемент, взятые через интервалы в 5%.

Алгоритм вычисления нормали к сплайну

- $c = 0$;
- Цикл по *j* от 0 до 20:
  - $v_j = Curve(5 \cdot j)$;
  - $c = c + v_j$;
- Конец цикла.
- Усреднение *c*:

$$c = \frac{c}{21};$$

- Цикл по j от 1 до 20
  - $n = (v_j - c) \times (v_{j-1} - c)$;
  - $N_c = N_c + \frac{n}{\|n\|}$;
- Конец цикла.
- Нормализация N_c:

$$N_c = \frac{N_c}{\|N_c\|}.$$

В итоге, для случая использования флага *AllowUpsideDown,* формулы вычисления 2-х последних строк матрицы *Q[1]* и *Q[2]* выглядят следующим образом:

$$Q[2] = Q[0] \times N_c; \qquad Q[2] = \frac{Q[2]}{\|Q[2]\|};$$

$$Q[1] = Q[2] \times Q[0]; \qquad Q[1] = \frac{Q[1]}{\|Q[1]\|}.$$

При изменении параметра *Axis*, задающего ту из осей объекта, которая следует по касательной к кривой, и имеющего по умолчанию значение 0 (ось X), нам потребуется поменять местами несколько строк матрицы Q.

Если *Axis* = 1 (выбрана ось Y, Рис. 5):
Vec = Q[0];
Q[0] = -Q[1];
Q[1] = Vec;

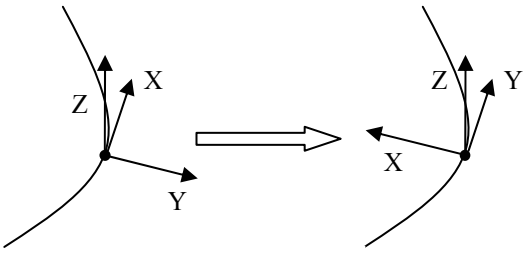

Рис.5.

Если *Axis* = 2 (выбрана ось Z, Рис. 6):
Vec = Q[0];
Q[0] = -Q[2];
Q[2] = Vec;

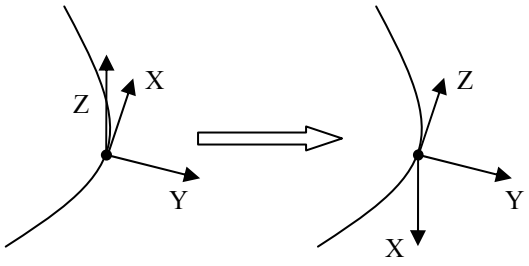

Рис.6.

Теперь рассмотрим действие флага *Bank*. При установке флага *Bank* объект начинает «следить» за изменением кривизны сплайна, наклоняясь в сторону поворота. Величина наклона определяется двумя специальными параметрами - *Smooth* и *BankAmount*. Величина *Smooth* определяет, за какое расстояние до начала поворота объект начнёт реагировать на поворот. *BankAmount* определяет, насколько сильно объект будет наклоняться при повороте. Значения данных параметров в текущем кадре *frame* необходимо вычислить по соответствующим контроллерам:

Smooth = SmoothCurve(*frame*);
BankAmount = BankAmountCurve(*frame*).

В точке P сплайна с процентом *k* необходимо получить угол *cv*, на который надо будет повернуть объект. Величина угла *cv* будет зависеть от локальной кривизны сплайна.

Угол наклона вычисляется суммированием значений, зависящих от локальной кривизны

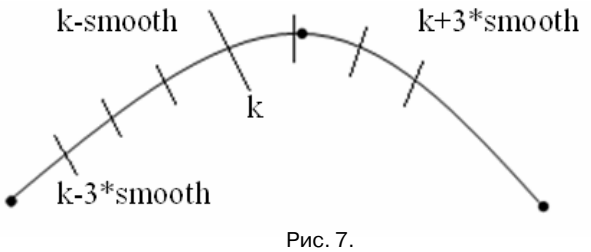

Рис. 7.

сплайна, по нескольким сегментам в окрестности точки P. По найденным исходным кодам из пакета для разработчиков (MAX SDK) было установлено, что используется деление на 6 сегментов, 3 сегмента до текущего значения процента *k* (от значения *k-3*smooth*) и 3 после (до значения *k+3*smooth*), как это показано на Рис.7. Длинной чертой отмечена точка с исходным значением процента *k*. Рассмотрим найденный алгоритм вычисления эффекта крена.

Алгоритм вычисления и применения матрицы крена

- Если *k+3*smooth* > *100%*, то:
  - *k = 100 – 6*smooth*;
- Иначе:
  - *k = k-3*smooth;*
- Если *k* < 0, то: *k* = 0;

- Цикл от 1 до 5 с шагом 1:
  - Возьмем 3 точки P_0, P_1, P_2 со значениями процента *k, k+smooth, k+2*smooth* соответственно:

$$P_0 = Curve(k); \qquad P_1 = Curve(k + smooth); \qquad P_2 = Curve(k + 2 \cdot smooth);$$

- Вычислим вектора $V_0 = \dfrac{P_1 - P_0}{\|P_1 - P_0\|}$ и $V_1 = \dfrac{P_2 - P_1}{\|P_2 - P_1\|}$;
- Вычислим их проекцию на плоскость XY, т.е. $V_{0,z}$ =0 и $V_{1,z}$ =0;
- Угол *cv* вычисляется по следующей формуле:

$$cv = cv + \frac{(V_0 \times V_1)_z * BankAmount}{\|P_1 - P_0\|};$$

- Увеличиваем *k на* величину *smooth: k = k + smooth;*
- Если *k > 100%*, то *k = 100%*;

- Конец цикла.
- Усредняем значение угла *cv*:

$$cv = \frac{cv}{5};$$

- Если был установлен флаг Flip, то:
  cv = - cv;
- Вычислим матрицу поворота $M(cv, Axis)$ относительно оси *Axis* на угол *cv*:
  - Если *Axis* = 0, то: $M(cv,0) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(cv) & -\sin(cv) \\ 0 & \sin(cv) & \cos(cv) \end{pmatrix}$;
  - Если *Axis* = 1, то: $M(cv,1) = \begin{pmatrix} \cos(cv) & 0 & \sin(cv) \\ 0 & 1 & 0 \\ -\sin(cv) & 0 & \cos(cv) \end{pmatrix}$;
  - Если *Axis* = 2, то: $M(cv,2) = \begin{pmatrix} \cos(cv) & -\sin(cv) & 0 \\ \sin(cv) & \cos(cv) & 0 \\ 0 & 0 & 1 \end{pmatrix}$;
- Умножим матрицу $M(cv, Axis)$ поворота вокруг оси *Axis* на угол *cv* и ранее вычисленную матрицу *Q:*
  $Q = Q \cdot M(cv, Axis)$.

Заключение

В результате проведенных исследований были восстановлены все алгоритмы вычисления как положения, так и ориентации объекта для контроллера следования по пути, применяемого в системе трехмерного моделирования 3D Studio Max. Тщательная проверка и тестирование показали полную идентичность результатов применения найденных алгоритмов и результатов анимации в самой системе трехмерного моделирования. Было также выяснено, что вычислительная сложность алгоритмов является величиной на порядок меньшей, нежели сложность трехмерной визуализации полигональных моделей, и не оказывает существенного влияния на время построения кадра в системе визуализации. Использование данного контроллера в системе визуализации позволяет реализовать очень сложные траектории движе-

ния объекта, важными преимуществами этого подхода являются простота получения анимационных траекторий и высокая компактность их описания.

Литература

1. Михайлюк М.В., Никифоров В.М. Контроллеры Безье в анимационных треках 3D Max. Сборник докладов научной конференции, посвященной 45-летию выхода человека в космос. – М.: Центр визуализации и спутниковых информационных технологий ИМВС РАН, 2006, с. 85-89.
2. Михайлюк М.В. Компьютерная графика в системах визуализации имитационно-тренажерных комплексов. // Программные продукты и системы – 2003, № 3. – С.7-15.
3. Михайлюк М.В. Основы компьютерной графики. Учебное пособие. М. 2001.
4. Веб-конференция по 3D Studio Max SDK: http://sparks.discreet.com/webboard/wbpx.dll/~maxsdk.

Никифоров Владимир Михайлович. Родился в1985 году. Является студентом 4-го курса Московского института радиотехники, электроники и автоматики (МИРЭА), Область научных интересов – алгоритмы трехмерной графики. Программист Центра визуализации и спутниковых информационных технологий НИИСИ РАН.

Торгашев Михаил Александрович. Родился в1974 году. Окончил Самарский государственный аэрокосмический университет. Кандидат физико-математических наук с 2001 года. Специалист в области трехмерной графики и моделирования сложных динамических систем. Имеет 15 публикаций. Ведущий научный сотрудник Центра визуализации и спутниковых информационных технологий НИИСИ РАН.