УДК 519.21:519.72

Оптимизация преобразований для скелетной анимации

© С.А. Ивлиев, А.А. Павельев, Н.Ю. Рязанова

МГТУ им. Н.Э. Баумана, Москва, 105005, Россия

В статье рассматриваются вопросы воспроизведения движения объектов графической сцены, заданных в виде скелетных моделей, в реальном масштабе времени. Предлагается подход к оптимизации временных затрат на формирование кадра изображения, заключающийся в реализации комбинированного решения: задавать преобразования набором — кватернион, коэффициенты масштабирования и перенос, а хранение и преобразование выполнять в виде матрицы перехода.

***Ключевые слова*:** *анимация, скелетная анимация, скелет, матричные преобразования, кватернион.*

Для генерации движения (анимации) реалистических изображений трехмерных графических объектов в реальном масштабе времени используются различные методы, выбор которых обусловлен сферой применения. Наиболее жесткие требования к изображению движения на экране предъявляются в 3D-играх. При разработке соответствующего программного обеспечения возникают следующие вопросы: как реализовать анимацию, задать ее и редактировать, как оптимизировать преобразования, связанные с движением для получения реального времени, как экспортировать в систему визуализации, какие форматы входных и выходных файлов использовать и т. п. В настоящее время скелетная анимация стала стандартом в анимации персонажей или механических объектов не только в игровой индустрии, но и в кино.

В основе скелетной анимации лежит скелетная структура объекта, т. е. набор иерархически упорядоченных костей, к которым уже крепится часть визуального представления объекта. Скелет удобно рассматривать как иерархическую структуру костей в отношении предок — потомок. Корневая кость формирует точку опоры для всей скелетной структуры. Все остальные кости скелета крепятся к корневой, как потомки. Для каждой кости задается трехмерное преобразование (позиция, размер, ориентация) относительно родительской кости. Каждая кость скелета задается как вектор в системе координат скелета, где опорная точка имеет координату 0. Для каждой кости скелета существует связанный с ней набор точек. Эти точки хранятся в собственном пространстве скелета и перед прорисовкой должны быть преобразованы в глобальное пространство с помощью преобразования, связанного с данной костью.

Преобразования задаются матрицами размерности 4×3 или 4×4. Полное преобразование кости-потомка — произведение преобразо-

вания родителя с ее собственным преобразованием. Соответственно, при изменении положения кости-родителя все ее кости-потомки также изменят свое положение.

Для моделирования более реалистических движений в скелетную модель вводится понятие веса. Вес определяет вклад той или иной кости в задание конечного положения. Введение веса позволяет формировать более реалистические изображения, но увеличивает объем требуемых вычислений. Для генерации движения все преобразования необходимо выполнять как минимум 24 раза в секунду.

В настоящее время известно два основных подхода к заданию преобразований: матрицей переходов и при помощи набора — кватернион, коэффициенты масштабирования, перенос. Первый подход не позволяет изменять только один из параметров преобразования, что исключает возможность интерполировать переход от одного положения к другому. Однако, поскольку матричные преобразования являются часто выполняемыми преобразованиями в компьютерной графике, фирма Intel ввела расширения набора команд, такие как AMD 3D Now и SSE, позволяющие ускорить выполнение преобразований. Второй подход помогает легко выполнять интерполяцию, однако требует большого количества вычислений, которые аппаратно не поддерживаются.

Как было отмечено, матричные преобразования выполняются часто, поэтому фирма Intel уже в набор команд процессора Pentium 3 включила набор инструкций SSE (англ. Streaming SIMD Extensions — потоковое SIMD-расширение) — это SIMD (англ. Single Instruction, Multiple Data — «одна инструкция — множество данных»), аппаратно поддерживающий вычисления с плавающей точкой. SSE состоит из восьми 128-битных XMM-регистров (от xmm0 до xmm7). Каждый регистр определяет набор из четырех последовательных значений с плавающей точкой одинарной точности. SSE включает в себя инструкции, которые производят операции со скалярными и упакованными типами данных. SSE позволяет выполнять операции над четырьмя 32-битными числами с плавающей точкой параллельно. Улучшенное расширение SSE — SSE2 появилось в процессорах Pentium 4 и выполняет потоковые вычисления с вещественными числами двойной точности (2 числа по 64 бита в одном регистре SSE). Однако большинство графических приложений не нуждаются в дополнительной точности данных.

Для умножения вектора на матрицу и матрицы на матрицу используются следующие функции SSE: movups — пересылка выровненных по 16-байтовой границе 128-разрядных операндов с использованием XMM-регистров; xorps — параллельная операция «исключающего или», shufps — параллельное перемещение 32-разрядных упакованных операндов с использованием маски; addps — параллельное сложение

32-разрядных упакованных операндов; mulps — параллельное умножение 32-разрядных упакованных операндов.

Применяя перечисленные команды для умножения вектора на матрицу, можно воспользоваться кодом из [2], листинг которого показан на рис. 1.

```
void MatrixMultiply3(Matrix4f &m, Vector4f *vin, Vector4f *vout)
{
    // Получить указатель на элементы m
    float *row0 = m.Ref();
    __asm {
            mov                 esi, vin
            mov                 edi, vout
            // загрузить матрицу по столбцам в регистры xmm4-7
            mov                 edx, row0
            movups      xmm4, [edx]
            movups      xmm5, [edx+0x10]
            movups      xmm6, [edx+0x20]
            movups      xmm7, [edx+0x30]
            // загрузить v в xmm0.
            movups      xmm0, [esi]
            // конечный результат будет храниться в xmm2;
            // инициализируем его нулем
            xorps       xmm2, xmm2
            // записать x в xmm1, умножить его на первый
            // столбец матрицы (xmm4), и добавить его к результату
            movups      xmm1, xmm0
            shufps xmm1, xmm1, 0x00
            mulps       xmm1, xmm4
            addps       xmm2, xmm1
            // повторить для y, z и w
            movups      xmm1, xmm0
            shufps xmm1, xmm1, 0x55
            mulps       xmm1, xmm5
            addps       xmm2, xmm1
            movups      xmm1, xmm0
            shufps xmm1, xmm1, 0xAA
            mulps       xmm1, xmm6
            addps       xmm2, xmm1
            movups      xmm1, xmm0
            shufps xmm1, xmm1, 0xFF
            mulps       xmm1, xmm7
            addps       xmm2, xmm1
            // записать результат в vout
            movups      [edi], xmm2
    }
}
```

Рис. 1. Использование функций SSE для умножения вектора на матрицу

Для решения задачи скелетной анимации предлагается использовать комбинированное решение: задавать преобразования набором — кватернион, коэффициенты масштабирования и перенос, а хранение и преобразование выполнять в виде матрицы перехода.

Кватернион представляет собой пару $(a,\vec{u})$ где $\vec{u}$ — вектор трехмерного пространства, a — скаляр. Для кватернионов операции сложения и умножения определяются следующим образом:

$$(a,\vec{u})+(b,\vec{v})=(a+b,\vec{u}+\vec{v}),$$

$$(a,\vec{u})(b,\vec{v})=(ab-\vec{u}\bullet\vec{v},a\vec{v}+b\vec{u}+\vec{u}\times\vec{v}),$$

где $\bullet$ — обозначает скалярное, а $\times$ — векторное произведение.

Кватернионы можно также определить и как вещественные матрицы следующего вида:

$$\begin{pmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{pmatrix}.$$

При такой записи: сопряженному кватерниону соответствует транспонированная матрица $\overline{q}\to Q^T$.

Кватернион, обратный по умножению к q, вычисляется как: $q^{-1}=\frac{q}{|q|^2}$, где $|q|=\sqrt{q\overline{q}}=\sqrt{a^2+b^2+c^2+d^2}$, а кватернион $\overline{q}=a-bi-cj-dk$ называется сопряженным к q.

Допустим (w, x, y, z) — координаты вращения. Тогда кватернион $q=(w, x, y, z)$ можно определить как

$$q=\omega+xi+yj+zk=\omega+(x,y,z)=\cos\left(\frac{\alpha}{2}\right)+u\sin\left(\frac{\alpha}{2}\right),$$

где u — единичный вектор. Таким образом, произведение qvq^{-1} вращает вектор v на угол α вокруг оси u [3]. Вращение происходит по часовой стрелке, если рассматривать его по направлению вектора u. А само преобразование над $\vec{v}$ можно записать как

$$\vec{v}=qvq^{-1}=\left(\cos\frac{\alpha}{2}+u\sin\frac{\alpha}{2}\right)v\left(\cos\frac{\alpha}{2}-u\sin\frac{\alpha}{2}\right).$$

Кватернион описывается структурой:

```
struct Quaternion{
 float x,y,z; // Вектор
 float w; // Скаляр
};
```

Преобразование сферических координат в кватернион, например, выполняется следующей функцией [1] (рис. 2).

```
void SphericalToQuaternion(Quaternion * q, float latitude, float longitude, float angle)
{
 float sin_a = sin( angle / 2 );
 float cos_a = cos( angle / 2 );
 float sin_lat = sin( latitude );
 float cos_lat = cos( latitude );
 float sin_long = sin( longitude );
 float cos_long = cos( longitude );
 q->x = sin_a * cos_lat * sin_long;
 q->y = sin_a * sin_lat;
 q->z = sin_a * sin_lat * cos_long;
 q->w = cos_a;
}
```

Рис. 2. Преобразование сферических координат в кватернион

Для того чтобы воспользоваться аппаратно оптимизированными функциями расширения SSE, преобразования зададим в матричном виде. Для получения итогового положения костей в результате движения, которое разбивается на сдвиг, поворот и масштабирование, преобразования задаются при помощи матрицы аффинных преобразований для каждой из костей, участвующей в движении, начиная с родительской.

Смещение задается матрицей трехмерного переноса:

$$M(dx,dy,dz)=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ dx & dy & dz & 1\end{pmatrix},$$

где dx, dy, dz — перемещения вдоль осей x, y, z.

Масштабирование задается следующей матрицей:

$$M(cx,cy,cz)=\begin{pmatrix}cx & 0 & 0 & 0\\ 0 & cy & 0 & 0\\ 0 & 0 & cz & 0\\ 0 & 0 & 0 & 1\end{pmatrix},$$

где cx, cy, cz — масштабные коэффициенты по осям x, y, z.

Если поворот задан кватернионом $q = (w, x, y, z)$, то матрица поворота будет иметь следующий вид:

$$M(\alpha) = \begin{pmatrix} 1-2y^2-2z^2 & 2xy-2zw & 2xz+2yw & 0 \\ 2xy+2zw & 1-2x^2-2z^2 & 2yz-2xw & 0 \\ 2xz-2yw & 2yz+2xw & 1-2x^2-2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

В результате умножения матриц получим матрицу совмещенного преобразования, которую обозначим как

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} & 0 \\ m_{21} & m_{22} & m_{23} & 0 \\ m_{31} & m_{32} & m_{33} & 0 \\ m_{41} & m_{42} & m_{43} & 1 \end{pmatrix}.$$

Получив результирующую матрицу, выполним ее декомпозицию для получения кватерниона. Для этого сначала найдем модуль строк матрицы [3] и тем самым определим масштабные коэффициенты:

$$k_x = \sqrt{m_{11}^2 + m_{12}^2 + m_{13}^2}\ ,\ k_y = \sqrt{m_{21}^2 + m_{22}^2 + m_{23}^2}\ ,\ k_z = \sqrt{m_{31}^2 + m_{32}^2 + m_{33}^3}\ .$$

Затем для получения кватерниона $q = (W, X, Y, Z)$ выполним следующие действия:

– разделим каждую строку матрицы на соответствующий масштабный коэффициент и получим матрицу $M' = \{a_{ij}\}$;

– в преобразованной матрице вычисляется сумма элементов главной диагонали:

$$T = m'_{11} + m'_{22} + m'_{33} + 1;$$

если $T > 0$, тогда

$$S = \frac{0,5}{\sqrt{T}},\ W = \frac{0,25}{S},\ X = (m'_{32} - m'_{23})S,$$
$$Y = (m'_{13} + m'_{31})S,\ Z = (m'_{21} + m'_{12})S;$$

если $T \le 0$, тогда в главной диагонали необходимо найти максимальный элемент. В зависимости от того, в каком столбце матрицы он находится, выполняются следующие действия:

– максимум находится в нулевом столбце:

$$S = 2\sqrt{1 + m'_{11} - m'_{22} - m'_{33}}, X = \frac{0,5}{S},$$

$$Y = (m'_{12} + m'_{21}) / S, Z = (m'_{13} + m'_{31}) / S, W = (m'_{23} + m'_{32}) / S;$$

– максимум находится в первом столбце:

$$S = 2\sqrt{1 - m'_{11} + m'_{22} - m'_{33}}, X = \frac{m'_{12} + m'_{21}}{S}, Y = \frac{0,5}{S},$$

$$Z = (m'_{23} + m'_{32}) / S; W = (m'_{13} + m'_{31}) / S;$$

– максимум находится во втором столбце:

$$S = 2\sqrt{1 - m'_{11} - m'_{22} + m'_{33}}, X = (m'_{13} + m'_{31}) / S,$$

$$Y = (m'_{23} + m'_{32}) / S; Z = \frac{0,5}{S}, W = (m'_{12} + m'_{21}) / S.$$

Таким образом, матричные преобразования удалось свести к кватерниону.

Кроме использования комбинированного подхода к выполнению графических преобразований для уменьшения временных затрат при формировании кадра изображения, можно создать буфер для записи в него информации о пикселях изображения без отрисовки каждого пикселя. Как известно, при этом процессе постоянно происходит переключение в режим ядра, что существенно замедляет процесс воспроизведения изображения. Для этого в GDI+ имеются функции: LockBit() — предназначена для создания буфера, и UnLockBit() — копирует содержимое буфера в видеобуфер. В результате их использования переход в режим ядра происходит только при заполнении буфера.

ЛИТЕРАТУРА

[1] Ваткин С.Г. *Кватернионы в программировании игр*. URL: http://wat. gamedev.ru / article

[2] *Streaming SIMD Extensions — Matrix Multiplication*. URL: http:// download. intel. com/design/PentiumIII/sml/24504501.pdf , June 1999 Order Number: 245045-001

[3] Норель М. *Вращение и кватернионы. Сборник рецептов*. URL: http:// www. gamedev.ru/code/articles/?id=4215

Статья поступила в редакцию 10.06.2013

Ссылку на эту статью просим оформлять следующим образом:

Ивлиев С.А., Павельев А.А., Рязанова Н.Ю. Оптимизация преобразований для скелетной анимации. *Инженерный журнал: наука и инновации*, 2013, вып. 6. URL: http://engjournal.ru/catalog/it/hidden/783.html

Ивлиев Сергей Андреевич родился в 1987 г. Студент 6-го курса кафедры «Программное обеспечение ЭВМ и информационные технологии» МГТУ им. Н.Э. Баумана. Область научных интересов: обработка изображений, распознавание изображений, машинная графика.

Павельев Александр Анатольевич родился в 1961 г. Старший преподаватель кафедры «Программное обеспечение ЭВМ и информационные технологии» МГТУ им. Н.Э. Баумана. Область научных интересов: поиск информации, базы данных, обработка и преобразование изображений. e-mail: pavelyev@bmstu.ru

Рязанова Наталья Юрьевна родилась в 1951 г. Канд. техн. наук, доцент кафедры «Программное обеспечение ЭВМ и информационные технологии» МГТУ им. Н.Э. Баумана. Автор 36 печатных работ. Область научных интересов: разработка системного программного обеспечения, алгоритмы машинной графики. e-mail: ryaz_nu@mail.ru