Реализация карт отражения окружающей среды в реальном режиме времени[1]

А.В. Мальцев, М.В. Михайлюк, В.Н.Решетников

Аннотация. Предлагаются методы и алгоритмы повышения реалистичности объектов трехмерных виртуальных сцен при визуализации в реальном режиме времени с использованием сферических и кубических карт отражений окружающей среды. Для их реализации используются современные средства визуализации, включая шейдеры и р-буферы.

При визуализации объектов виртуальной сцены ставится задача максимального их соответствия реальным прототипам. Одним из свойств многих реальных объектов является их способность отражать окружающую обстановку. Поэтому появляется естественное желание реализовать это свойство и у виртуальных объектов сцены. Один из способов имитации отражения окружающей среды на визуализируемых объектах применение так называемых текстур (карт) отражения (reflection maps). При перемещении объекта с наложенной картой отражения по сцене мы будем наблюдать на нем изменение рисунка, в отличие от случая с обычной текстурой, которая остается жестко связанной с объектом при его движении. Такой метод реализации отражений используется в системе трехмерного моделирования 3D Studio Max. Однако в 3D Studio Max рендеринг сцены не происходит в реальном режиме времени, поддержка которого требуется во многих современных графических приложениях. В данной статье мы опишем, как создаются и применяются карты окружающей среды для реализации отражений в реальном режиме времени. Так как в системах визуализации виртуальные сцены должны выглядеть так же, как их задумал и нарисовал дизайнер в системе моделирования, то в алгоритмах необходимо учесть все параметры карт отражения, используемых в системе 3D Studio Max.

1. Сферические карты отражения окружающей среды

Для реализации отражения на зеркальном объекте необходимо в каждой вершине объекта вычислить отраженный луч (т.е. луч, симметричный относительно нормали к направлению из вершины в точку положения камеры). Затем по отраженному лучу определить его пересечение с окружающей средой и цвет точки пересечения добавить к цвету вершины зеркального объекта. Ясно, что результат зависит как от направления вектора отражения, так и от положения вершины на объекте. Однако, если объект небольшой и окружающая среда достаточно удалена от него, то положением вершин можно пренебречь и определять цвет отражения только в зависимости от направления вектора отражения. Эту зависимость можно хранить в двумерной текстурной карте. Способ отображения множества 3D векторов направлений на 2D текстурную карту называется параметризацией.

[1] Работа выполняется в рамках программы ОИТВС РАН «Фундаментальные основы информационных технологий и систем», проект 2.8

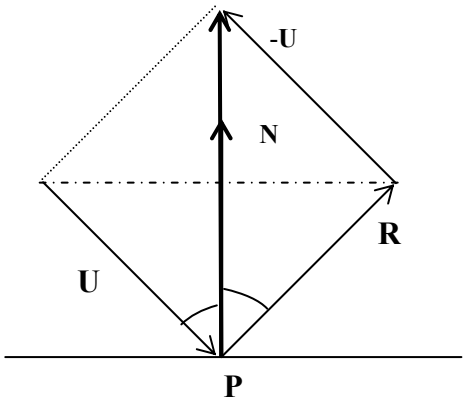


Рис. 1.

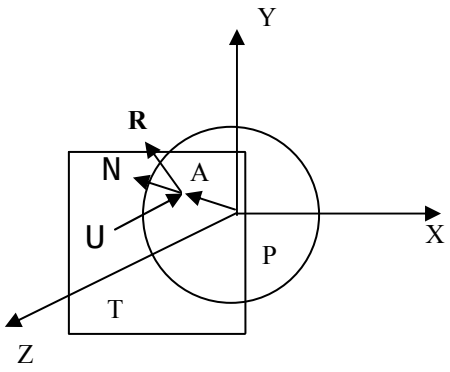


Рис. 2.

Рассмотрим сначала, как вычислить отраженный вектор R, если известна нормаль N к точке поверхности и падающий в точку вектор U. Пусть все эти векторы нормализованы. Так как угол падения равен углу отражения, можно написать (Рис. 1) следующее равенство:

$$2\,||U||\,\mathrm{Cos}\ \cdot N = R\quad U$$

Подставляя в него значения Cos = (N,-U) = -(N, U) и ||U|| = 1, получаем отраженный вектор

$$R = U - 2(N, U)N; \qquad (1)$$

Теперь определим параметризацию для сферической карты отражения. Пусть P – сфера единичного радиуса с центром в начале координат, T – единичный квадрат, расположенный параллельно плоскости XY, центр которого лежит на оси Z (Рис. 2). Рассмотрим круг радиуса 1, вписанный в этот квадрат, и произвольную точку (s,t) этого круга. Проецируя эту точку на сферу вдоль вектора U параллельного оси Z, получим точку A с координатами

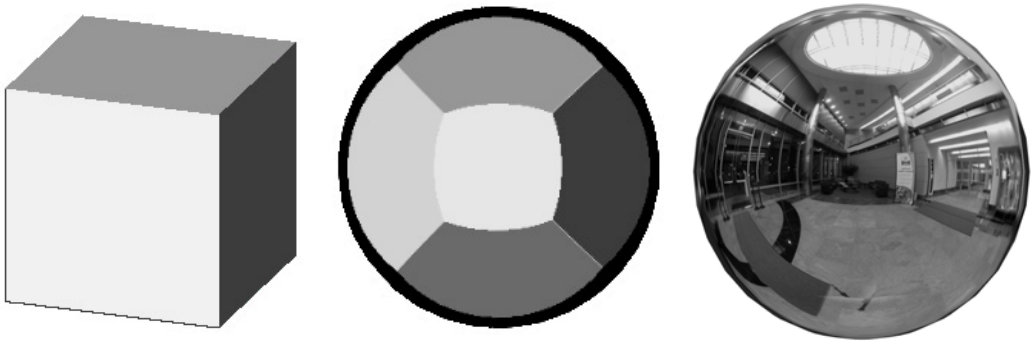
$$A_x = s;$$
$$A_y = t;$$
$$A_z = \sqrt{1 - s^2 - t^2};$$

Единичная нормаль N в точке A совпадает с радиусом-вектором сферы, т.е. N = (Ax, Ay, Az). Так как вектор U = (0,0,-1), то (N,U) = -Az и, расписывая формулу (1) по координатам, получаем:

$$R_x = 2A_z A_x;$$
$$R_y = 2A_z A_y; \qquad (2)$$
$$R_z = 2A_z^2 - 1;$$

Таким образом, зная s и t, можно вычислить 3D вектор направления R = (Rx,Ry,Rz). Заметим, что для точек (s,t), в которых $s^2 + t^2 = 1$, мы получим Az = 0, поэтому вектор направления R для всех этих точек будет одинаков и равен R = (0, 0, -1).

Для создания текстурной карты размером WxH нужно для каждого пиксела (i,j) вычислить соответствующие ему значения s и t по формулам s = 2([i/W]-0.5), t = 2([j/H]-0.5). Если $s^2 + t^2$ 1, то вычисляется вектор отражения в соответствии с формулами (2) и находится цвет окружающей среды по направлению этого вектора. Этот цвет и записывается в пиксел (i,j) создаваемой текстурной карты. Для остальных пикселов можно записать любое значение, например, черный цвет. Таким образом, текстурная карта будет содержать круг, в котором будет записана текстура окружающей обстановки. При этом во всех точках окружности будет записано одно и то же значение, соответствующее вектору направления R = (0, 0, -1). На Рис. 3б показана текстурная карта для окружающей среды, представляющей собой куб с гранями разного цвета (Рис. 3а). Видно, что передняя грань кубика отображается в центральную часть круга текстуры, а задняя грань



а) б) в)

Рис. 3

– в несколько искаженное кольцо по периметру круга. На Рис. 3в показана текстурная карта реальной окружающей среды.

Приближенно текстурную карту сферического отражения можно также получить, сфотографировав блестящий шар, отражающий окружающую среду.

В качестве исходных данных для реализации отражений с помощью сферической карты окружающей среды в системе визуализации «3D Viewer» используются следующие параметры, конвертируемые из 3D Studio Max:

- 2D текстура (карта) отражения;
- процент p_r применения карты отражения;
- степень k_s отражения зеркального освещения материалом.

Если I – интенсивность освещения (т.е. цвет) точки на поверхности объекта, вычисленный без учета отражения, а C_t – цвет текстуры отражения для этой точки, то полная интенсивность I' в этой точке вычисляется по формуле

$$I' = I + k_s p_r C_t; \qquad (3)$$

Рассмотрим теперь вопрос о нахождении цвета Ct для произвольной точки A на поверхности объекта, т.е. фактически о вычислении текстурных координат этой точки. Пусть U – вектор, направленный из начала координат видовой системы (VCS) в точку объекта A, N – нормаль в точке A, R – вектор отражения (Рис. 4). При этом U, N и R имеют единичную длину и заданы в координатах видовой системы (VCS). Теперь уже вектор U не параллелен оси Z, так как в системе визуализации обычно используется перспективное проецирование. Вектор R отражения находится в соответствии с формулой (1). Из формулы (2) получаем

$$\begin{aligned} A_z &= \sqrt{(R_z + 1)/2}; \\ s &= A_x = R_x/(2A_z); \\ t &= A_y = R_y/(2A_z); \end{aligned} \qquad (4)$$

Вычисленные таким образом значения s и t находятся в пределах от -1 до 1. Так как текстурные координаты s' и t' должны принадлежать отрезку [0,1], получаем

$$\begin{aligned} s' &= (s+1)/2, \\ t' &= (t+1)/2. \end{aligned} \qquad (5)$$

Фактически, для точки A при перспективном проецировании (с вектором отражения R) выбирается пиксел текстурной карты, соответствующий той точке B, в которой при ортографическом проецировании получается тот же вектор отражения R (Рис. 5).

Точку B легко найти, если заметить, что $B = N' = (R - U')/\|R - U'\|$. Как видно из формул (4), s и t совпадают с координатами x и y точки B.

Отметим, что при стандартном повершинном освещении OpenGL карта отражения накладывается на объект не путем задания текстурных координат для каждой вершины, а с помощью функции glTexGenf автоматической генерации сферических текстурных координат, при вызове которой текстурные координаты вычисляются по формулам (1), (4) и (5) автоматически. Мы же реализуем в настоящей работе попиксельный расчет освещения с помощью шейдеров, что позволяет существенно улуч-

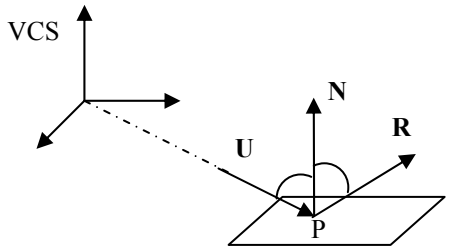


Рис. 4.

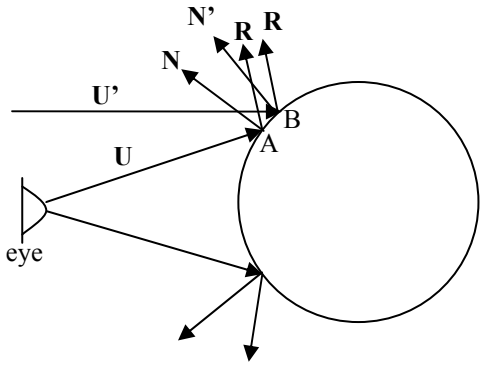


Рис. 5.

шить качество моделирования отражения и ускорить визуализацию сцен в реальном времени. В этом режиме механизм автогенерации не работает, поэтому фрагментная программа (пиксельный шейдер) должна содержать код, вычисляющий сферические текстурные координаты по формулам (1), (4) и (5). Рассмотрим, как получить все необходимые данные для этих формул.

Вектор U для пиксела рассчитываем следующим образом:

- в вершинном шейдере вычисляем векторы U_i (i = 1,2,3) в вершинах полигона, вычитая координаты вершин из координат наблюдателя (камеры) в объектной системе координат (OCS);

- переводим координаты каждого вектора U_i в видовую систему координат (VCS), умножив их на текущую модельно-видовую матрицу (также в вершинном шейдере);

- находим вектор U в пикселе интерполяцией векторов Ui для трех вершин полигона (для этого всего лишь нужно подать вектор U на выход вершинного шейдера в качестве текстурных координат, а на входе фрагментного шейдера мы уже будем иметь интерполированный вектор U);

- нормализовать вектор U в пиксельном шейдере.

Вектор N в пикселе получается интерполяцией нормалей (представленных в видовой системе координат VCS) для трех вершин полигона или берется непосредственно из карты нормалей в фрагментном шейдере и переводится в систему координат VCS.

Так как описанные вычисления являются довольно трудоемкими, а потребность в них, в общем случае, может появиться при шейдерной визуализации только некоторых объектов из всей большой сцены, то возникает вопрос: “Как избежать этих расчетов для остальных объектов сцены, если используемое расширение OpenGL не поддерживает организацию ветвлений в шейдере?” Одним из решений этой проблемы является описанный ниже (в разделе 3) метод статической оптимизации.

Сферические карты отражения окружающей среды зависят от положения наблюдателя (камеры) при их создании. В связи с этим при перемещении камеры отражение на объекте не будет меняться, поэтому сферические карты удобно использовать в статических сценах или в сценах, в которых объект и камера перемещаются незначительно.

2. Кубические карты окружающей среды

Для реализации отражений на объектах динамической сцены в реальном режиме времени более предпочтительными являются кубические карты окружающей среды (cube environment mapping).

Кубическая карта состоит из шести планарных карт среды, образующих грани куба, в центре которого располагается отражающий объект. Каждая карта является проекцией окружающей обстановки из центра куба на соответствующую его грань. При расчете отражения на объекте для каждой видимой точки на его поверхности проводится луч из центра объекта до пересечения с кубом и цвет точки пересечения берется в качестве цвета отражения ок-

ружающей среды. При фиксированной кубической карте объект и камера могут перемещаться, при этом отражения на объекте будут также меняться. Сами карты окружающей среды могут быть статичными и динамичными, т.е. они могут быть созданы заранее и могут создаваться на лету в процессе изменения в сцене самой окружающей среды.

В системе 3D Studio Max существуют три способа реализации отражения окружающей среды с помощью кубической карты:

- используя заранее подготовленную кубическую карту;
- генерируя для каждого отражающего объекта кубическую карту отражения окружающей среды только в первом кадре;
- генерируя для каждого отражающего объекта кубическую карту отражения окружающей среды в каждом N-м кадре.

Первый способ предполагает наличие шести заранее подготовленных двумерных текстур, образующих кубическую карту среды (Рис. 6). Аналогично сферической карте, для каждой видимой точки A на поверхности объекта в соответствии с равенством (1) в видовой системе координат (VCS) вычисляется вектор отражения $R = R_{VCS}$ на основе вектора U, направленного из начала видовой системы в точку A, и нормали N в точке A. Рассмотрим, как по вектору R вычислить текстурные координаты в кубической карте. В системе 3D Studio Max кубическая карта среды задается в системе координат OWCS, центр которой совпадает с центром объекта, а оси направлены так же, как в мировой системе (WCS). Так как видовая матрица MV текущей виртуальной камеры осуществляет преобразование из видовой в мировую систему, то вектор R в системе OWCS будет иметь координаты

$$R_{OWCS} = (MV)^{-1} \cdot R_{VCS} \qquad (6)$$

Поскольку векторы в 3D-аффинном пространстве имеют нулевую четвертую координату, то можно брать в качестве MV матрицу 3x3, осуществляющую только поворот системы координат и не учитывающую переноса. В таком случае $(MV)^{-1}$ совпадает с транспонированной матрицей $(MV)^T$. Если использовать аппаратное умножение векторов размерности 4, то к матрице MV можно добавить строку и столбец вида (0,0,0,1).

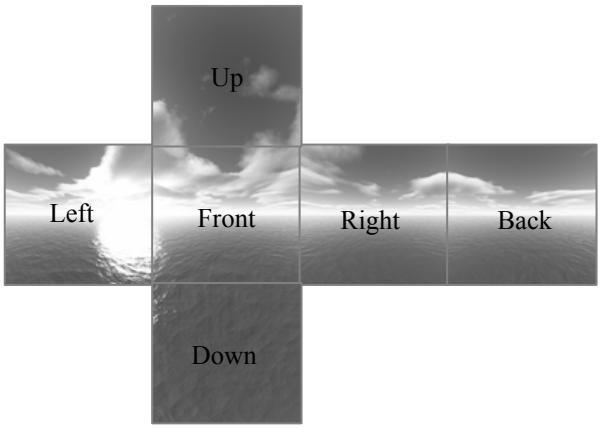


Рис. 6.

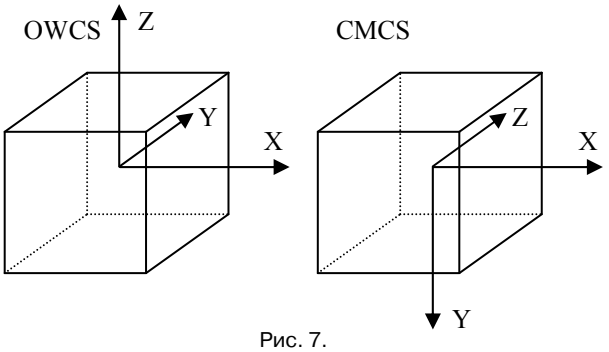


Рис. 7.

При реализации отражений с помощью графической библиотеки OpenGL следует иметь в виду, что в ней кубическая текстурная карта имеет свою систему координат (Cube Map Coordinate System, CMCS), повернутую относительно системы OWCS вокруг оси X на 90 градусов (Рис. 7). Матрица поворота имеет вид

$$M_{rot} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

а вектор R в системе CMCS будет иметь координаты

$$R_{CMCS} = M_{rot} \cdot R_{OWCS} = M_{rot} \cdot (MV)^{-1} \cdot R_{VCS}. \qquad (7)$$

Умножение вектора $R_{OWCS} = (x, y, z)$ на матрицу M_{rot} дает вектор (x, -z, y), поэтому в целях оптимизации возможна замена умножения на

матрицу M_{rot} перестановкой (с учетом знаков) координат y и z вектора R_{OWCS}.

Ориентация текстурных карт на гранях куба также отличается в системах OpenGL и 3DS MAX. Так, например, из Рис. 7 видно, что передняя грань куба (FRONT) расположенная в системе OWCS по направлению -Y, должна соответствовать передней грани в системе CMCS по направлению -Z, и иметь стандартное имя GL_TEXTURE_CUBE_MAP_NEGATIVE_Z_ARB. На Рис. 8 приведена полная таблица соответствия имен и ориентаций граней кубической карты в системах OWCS и CMCS.

По вектору R_{CMCS}, найденному из равенств (1) и (7), необходимо определить, какую грань куба следует взять и как на ней вычислить соответствующие текстурные координаты. В качестве текстурных координат для кубической карты среды выступают координаты x, y и z вектора R_{CMCS}. Они задают направление луча, выходящего из начала координат и пересекающего одну из граней единичного текстурного куба. Пересекаемая грань определяется текстурной координатой, имеющей наибольшее по модулю значение, знаком этой координаты. Например, если $|x|>|y|$, $|x|>|z|$ и $x > 0$, то выбирается правая грань текстурного куба (GL_TEXTURE_CUBE_MAP_POSITIVE_X_ARB). Координаты s и t выбранной двумерной текстурной карты вычисляются по оставшимся координатам y и z, с использованием формул

$$s = \frac{-z}{2|x|} + \frac{1}{2}; \qquad t = \frac{-y}{2|x|} + \frac{1}{2}.$$

Для остальных случаев выбор грани куба и расчет для нее текстурных координат производятся аналогично. Более подробная информация о работе с кубическими текстурными картами изложена в [5].

Отметим, что в случае визуализации без использования шейдеров можно не подсчитывать текстурные координаты для кубической карты среды самостоятельно, а воспользоваться стандартной функцией автоматической генерации текстурных координат glTexGeni, подставив в качестве параметра генерации GL_REFLECTION_MAP_ARB. При этом координаты будут рассчитываться на основе вектора R, но представленного в видовой системе координат. При расчете в системе CMCS, требуется выбрать текстурную матрицу с помощью glMatrixMode(GL_TEXTURE) и загрузить в нее (glLoadMatrixf) матрицу, равную $M_{rot}\,(MV)^{-1}$ (7). На Рис. 9 показан пример реализации отражения на основе кубической карты окружающей среды.

Второй способ реализуется аналогично первому, за исключением того, что вместо заранее подготовленной кубической карты среды, взятой из графических файлов, используется сгенерированная в первом кадре кубическая карта. Процесс генерации такой карты для объекта состоит из следующих пунктов:

1. Рендеринг сцены по шести взаимно-перпендикулярным направлениям из центра объекта с получением, соответственно, шести изображений;

2. Создание кубической карты из полученных шести изображений.

OWCS		CMCS	
Имя грани	Ориентация грани	Имя грани	Ориентация грани
FRONT	-Y	GL_TEXTURE_CUBE_MAP_NEGATIVE_Z_ARB	-Z
BACK	+Y	GL_TEXTURE_CUBE_MAP_POSITIVE_Z_ARB	+Z
UP	+Z	GL_TEXTURE_CUBE_MAP_NEGATIVE_Y_ARB	-Y
DOWN	-Z	GL_TEXTURE_CUBE_MAP_POSITIVE_Y_ARB	+Y
LEFT	-X	GL_TEXTURE_CUBE_MAP_NEGATIVE_X_ARB	-X
RIGHT	+X	GL_TEXTURE_CUBE_MAP_POSITIVE_X_ARB	+X

Рис. 8

Рис. 9.

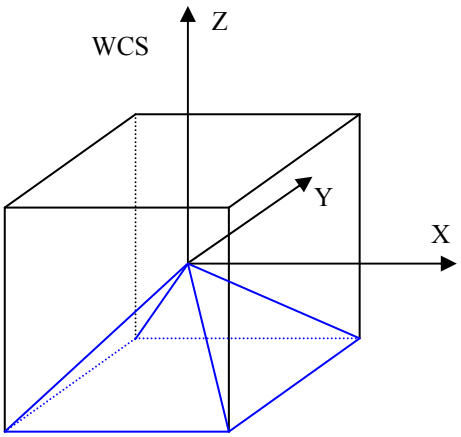


Рис. 10.

Пункт 1 предполагает введение в сцену шести виртуальных камер, расположенных в центре объекта и сонаправленных с осями +X и -X, +Y и -Y, +Z и –Z мировой системы координат (Рис.10). Угол раствора камер должен составлять 90°. Рассмотрим, например, вычисление видовой матрицы для камеры, снимающей по направлению –Z в системе OWCS.

Расположим камеру в точке, совпадающей с началом локальной системы координат OCS объекта. Для того, чтобы видовая система координат камеры совпадала с системой OWCS, необходимо задать видовую матрицу в следующем виде:

$$M_V = \begin{pmatrix} 1 & 0 & 0 & -m_{03} \\ 0 & 1 & 0 & -m_{13} \\ 0 & 0 & 1 & -m_{23} \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

где (m_{03}, m_{13}, m_{23}, 1) – последний столбец модельной матрицы MM объекта, то есть столбец, отвечающий за перенос. Действительно, единичная подматрица матрицы MV обеспечивает совпадение направления осей видовой системы с направлениями осей мировой системы координат, а последний столбец – совпадение начала координат видовой системы с началом локальной системы координат объекта. Отметим, что для формирования правильной карты среды в данном случае начало локальной системы координат должно находиться в геометрическом центре объекта. При построении объекта сцены в системе моделирования 3D Studio Max начало его локальной системы OCS в общем случае не совпадает с геометрическим центром. Однако с помощью специального параметра pivot систему OCS можно смещать относительно объекта. Поэтому при формировании сцены в 3D Studio Max дизайнер должен для каждого отражающего объекта задать точку pivot в центре этого объекта.

Так как направление взгляда камеры определяется осью –Z, то камера, заданная видовой матрицей M_V, и является искомой. Матрицы для остальных камер можно получить из M_V путем умножения слева на матрицу вращения в нужном направлении.

После визуализации окружающей среды каждой из шести определенных выше камер мы будем иметь шесть изображений, которые представляют стороны кубической карты окружения для рассматриваемого объекта. Применение этой карты полностью аналогично первому способу.

Для генерации кубической карты и визуализации сцены в реальном режиме времени целесообразно применение так называемого рендеринга в текстуру с использованием p-буфера. P-буфер – это пиксельный буфер, хранящийся непосредственно в видеопамяти, обладающий собственным размером и другими атрибутами и никак не связанный с основным буфером кадра. При этом содержимое p-буфера может быть непосредственно использовано в качестве текстуры, а рендеринг в p-буфер аппаратно ускорен [3]. Другими словами, отрисовав последовательно каждую из шести граней кубической

карты в p-буфер, мы получим готовую кубическую карту, находящуюся в видеопамяти и пригодную для использования в качестве карты отражения окружающей среды. Данный метод позволяет избежать лишнего копирования данных из видеопамяти в оперативную память и обратно, которое требовалось бы в других реализациях, что заметно повышает скорость визуализации в реальном режиме времени и дает возможность увеличить качество отражений. Для подключения p-буфера на платформе Windows и получения возможности использования его в качестве текстуры необходимо инициализировать расширения OpenGL WGL_ARB_pixel_format, WGL_ARB_pbuffer и WGL_ARB_render_texture.

Третий способ отличается от второго только тем, что через каждые N кадров необходимо повторять процедуру генерации кубической карты среды, а в остальных кадрах использовать кубическую карту, созданную при предыдущей генерации. Этот способ наиболее эффективен в виртуальных сценах, где объекты окружения находятся в динамике, а, следовательно, отражение в зеркальных объектах должно меняться во времени.

Литература

1. Paul Haeberli and Mark Segal. Texrure mapping as a fundamental drawing primitive. In Fourth Eurographics Workshop on Rendering, pages 259-266, June 1993.
2. Greene, “Environment Mapping and Other Applications of World Projections,” IEEE Computer Graphics and Applications, Vol. 6, No. 11 (1986), pp. 21-30.
3. Боресков А.В. Расширения OpenGL - СПб.: БХВ-Петербург, 2005.
4. M. J. Kilgard: Perfect Reflections and Specular Lighting Effects With Cube Environment Mapping, Technical Brief, nVidia Corp., 1999.
5. Мальцев А.В., Михайлюк М.В. Технология рельефного текстурирования в системах визуализации // Сборник докладов научной конференции, посвященной 45-летию выхода человека в космос. – М., 2006, с.59-75.
6. http://www.ixbt.com/video/creative-geforce256.html
7. http://www.developer.com/lang/other/article.php/10942_2169281_1.
8. http://en.wikipedia.org/wiki/Reflection_mapping.

Мальцев Андрей Валерьевич. Программист 1-ой категории Центра визуализации и спутниковых информационных технологий НИИСИ РАН. Окончил факультет вычислительных машин и систем Московского государственного института радиотехники, электроники и автоматики в 2008 году. Автор 6 научных публикаций. Область научных интересов: компьютерная графика, системы виртуальной реальности, информационные технологии, мультимедийные информационные системы. Специалист в области компьютерной графики.

Михайлюк Михаил Васильевич. Заведующий отделом программных средств визуализации НИИСИ РАН. Доктор физико-математических наук, профессор. Автор более 60 научных работ.

Решетников Валерий Николаевич. Директор Центра визуализации и спутниковых информационных технологий НИИСИ РАН. Окончил механико-математический факультет Московского государственного университета в 1965 году. Доктор физико-математических наук, профессор. Автор более 60 научных публикаций. Область научных интересов: компьютерная графика, информационные технологии, виртуальные студии, телекоммуникации. Специалист в области обработки информации в системах управления.