УДК 515.2:528.71
аспирант Мирошниченко Н.Ю.,
Национальная металлургическая академия Украины

ВИЗУАЛИЗАЦИЯ ТРЁХМЕРНЫХ СЦЕН В РЕАЛЬНОМ ВРЕМЕНИ

В данной статье рассмотрены и проанализированы основные принципы и алгоритмы тенеобразования в трехмерной графике. Рассмотренные алгоритмы были разделены на два явных действия:

- построение теневой маски;

- отрисовка сцены с использованием теневой маски;

и разделены на три стадии. Стадии детально изучены. Определена центральная проблема тенеобразования в трехмерных сценах в реальном времени.

Постановка проблемы. Визуализация — это завершающий и, безусловно, самый ответственный этап создания трехмерной сцены. Редактор трехмерной графики просчитывает изображение, учитывая геометрию объектов, свойства материалов, из которых они сделаны, расположение и параметры источников света и т.д. Неудачно выполненная визуализация может свести на нет все многодневные усилия по моделированию, освещению и текстурированию сцены. Создание сцены с реалистичным освещением — это еще одна задача, которую решают для того, чтобы придать конечному изображению большую реалистичность. В реальном мире световые лучи многократно отражаются и преломляются в объектах, в результате чего тени, отбрасываемые объектами, в основном, имеют нечеткие, размытые границы. За качество отображения теней, в основном, отвечает аппарат визуализации. К теням, отбрасываем в сцене, предъявляются отдельные требования. Отбрасываемая от объекта тень может сказать о многом — как высоко он находится над землей, какова структура поверхности, на которую падает тень, каким источником освещен объект и т.д. Кроме этого, тень может подчеркнуть контраст между передним и задним планом, а также «выдать» объект, который не попал в поле зрения объектива виртуальной камеры. В этом случае зрителю дается возможность самому домыслить окружающую обстановку сцены.

Анализ последних исследований. Основная проблема построения теневого проецирования состоит в том, что аппаратная часть для описания поверхности использует, как универсальную единицу треугольники. Треугольники являются элементарной «поверхностью», для построения которой используется минимальное число вершин. Описываемые алгоритмы построения поверхностей аппаратной частью выделяются в общую группу

триангулятивных (использующих в своей основе универсальные единицы треугольники). Современные графические ускорители работают на уровне не выше триангулятивных алгоритмов и ничего не знают о наличии сцены, поэтому тени ими напрямую не поддерживаются и вся подготовительная работа ложится на плечи программиста. Но, тем не менее, в наиболее распространённых алгоритмах практически все стадии построения тени реализуются именно через графический ускоритель. Кроме того, за последние годы были внесены значительные изменения как в объем, так и в составляющие динамических библиотек, openGL, Direct3D и т.д. в направлении описания тенеобразования. Улучшение алгоритмов построения теней привели к высоким результатам реалистичности трехмерной визуализации. Появление таких функций и понятий, как Decay, ShadowBias, ShadowType, Near/Far Attenuation и т.д., свидетельствуют об увеличении гибкости в управлении отбрасываемой объектами тенью.

Цель статьи. Статья посвящена вопросам выбора соответствующего алгоритма построения тени, который является важным этапом трехмерной визуализации сцены.

Основная часть. Тени бывают чёткие (*hard shadows*) и мягкие (*soft shadows*). Чёткие тени получаются, когда имеется точечный источник или источник направленного параллельного света. Согласно геометрической оптике Френеля, в случае наличия протяжённого источника света, от объекта получается не одна тень, а целая серия теней, которые накладываются друг на друга и образуют более или менее затемнённые области, которые и составляют основную тень (*umbra*) и область полутени (*penumbra*). Все существующие алгоритмы построения теней позволяют строить чёткие тени, но некоторые из них позволяют размыть полученную тень, создавая псевдо-мягкую тень.

Существует несколько основных подходов к построению теней:

- Преобразование модели «на землю» и отрисовка её как тени.
- Построение теневой маски объекта и проективное наложение её на другие объекты.
- Теневые объёмы.
- Использование информации о глубине (*depth buffer*).

1. Проецирование (Преобразование "на землю"). Фактически, это первый алгоритм построения тени, который был применён в играх. Он отличается простотой реализации и хорошим качеством получаемой тени. Этот алгоритм был впервые описан Д. Блинном [1], который описал уравнения для проектирования полигона «на землю», т.е. на плоскость z=0, в направлении от источника света.

Рассмотрим два случая:

1) Источник на бесконечности (рис. 1)
2) Локальный источник (точечный источник недалеко от объекта)

При этом используется геометрическое взаимоотношение источника света и полигона для вычисления проекции каждого полигона модели «на землю». «Теневые полигоны» должны быть рассчитаны для каждого источника света, т.е. если объект освещается *N* источниками света, то необходимо рассчитать *N* его «теневых проекций».

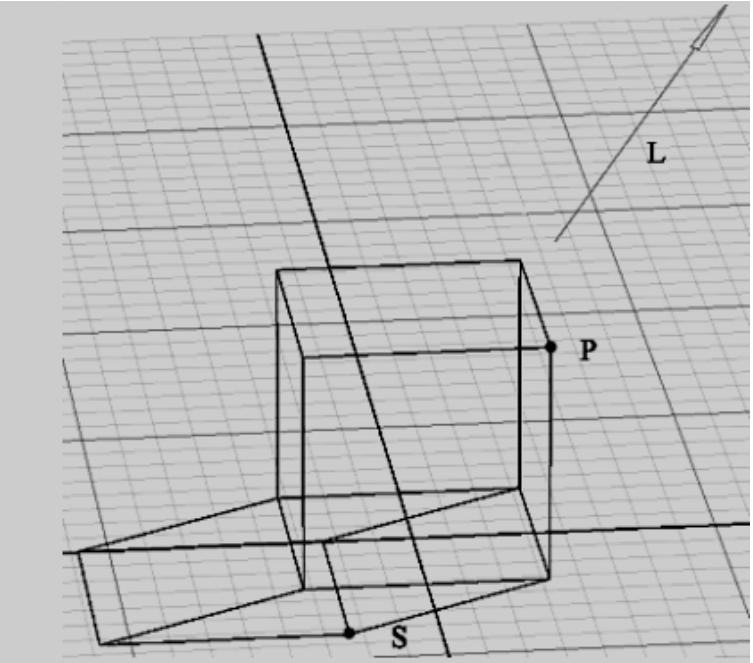

Рис. 1

*1)**Источник на бесконечности.*** В случае бесконечно удалённого источника света мы предполагаем, что лучи света, приходящие к объекту, полностью параллельны. Это позволит нам решить уравнение проекции только раз и применять полученное решение ко всем вершинам объекта.

Общая постановка задачи: имея точку источника света (x_l, y_l, z_l) и вершину объекта (x_p, y_p, z_p), мы хотим получить проекцию вершины объекта на плоскость z=0, т.е. точку тени (x_s, y_s, z_s) .

Из подобных треугольников получаем:

$$\frac{x_p - x_s}{z_p - z_y} = \frac{x_l a - x_p}{z_l - z_p}. \quad (1)$$

Решая это уравнение относительно x_s, получаем:

$$x_s = x_p - \left(z_p - z_s\right)\left(\frac{x_l - x_p}{z_l \quad z_p}\right). \quad (2)$$

Если принять, что *L* это вектор из точки *P* к источнику света, то точку *S* можно выразить как

$$S = P - \propto L. \quad (3)$$

Поскольку мы проецируем на плоскость z=0, то уравнение (3) можно переписать в следующем виде:

$$0 = z_p - \propto z_l \quad (4)$$

или

$$\propto = \frac{z_p}{z_l}. \tag{5}$$

Решая (3) относительно x_s и y_s, получаем:

$$x_s = x_p - \frac{z_p}{z_l} x_l \tag{6}$$
$$y_s = y_p - \frac{z_p}{z_l} y_l,$$

либо в матричной форме:

$$M_s = \begin{vmatrix} 1 & 0 & -x_l/z_l & 0 \\ 0 & 1 & -y_l/z_l & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}. \tag{7}$$

Теперь имея координаты точки P в мировом координатном пространстве, можно получить её проекцию на плоскость z=0 просто путём умножения на матрицу M_s:

$$S = M_s * P. \tag{8}$$

2)Локальный источник. Уравнение (6) для бесконечно удалённого источника света может быть обобщено для случая, когда источник света находится на конечном расстоянии от объекта. В этом случае нам понадобятся дополнительные вычисления на каждую вершину, т.к. каждая вершина имеет своё собственное направление на источник света. Тем не менее, в этом случае мы тоже можем перенести большую часть вычислений в матрицу M_s. Если L — точка расположения источника света, то (3) принимает вид:

$$S = P + \propto (P - L). \tag{9}$$

И снова нам необходимо произвести проекцию на плоскость z=0:

$$\propto = \frac{-z_p}{z_p - z_l} \tag{10}$$

и

$$x_s = \frac{x_l z_p - x_p z_l}{z_p - z_l} \tag{11}$$
$$y_s = \frac{y_l z_p - y_p z_l}{z_p - z_l}$$

Если использовать гомогенизацию после преобразования, то (11) можно записать в виде матрицы

$$M_{sh} = \begin{vmatrix} -z_l & 0 & x_l & 0 \\ 0 & z_l & 0 & 0 \\ 0 & 0 & y_l & 0 \\ 0 & 0 & 1 & -z_l \end{vmatrix}. \tag{12}$$

Тогда

$$S_h = M_{sh} * P\,, \tag{13}$$

после чего провести гомогенизацию точки S_h для получения проекции точки P на плоскость z=0.

Построение теневой маски и проективное наложение. Этот метод является логическим продолжением предыдущего, но, в отличие от него, обладает целым рядом преимуществ. Объект, который отбрасывает тень (*shadow caster*) тем или иным способом рисуется с точки зрения источника света чёрным цветом в белую текстуру. Размер текстуры зависит от того, насколько мелкие элементы объекта мы хотим видеть в тени. Представим себе плоскость, перпендикулярную направлению на источник света и расположенную сразу за объектом по лучу света — это и есть текстура, в которую производится отрисовка объекта. Для того, чтобы получить проекцию объекта на эту текстуру, применим элементы стандартного конвейера трансформации. Предположим, что текстура, в которую нам необходимо нарисовать объект, это экран. Тогда последовательность преобразований становится просто очевидной (стандартный конвейер преобразования): *пространство объекта => мировое пространство => пространство камеры* (у нас это источник света) => *проективное пространство => нормализованное проективное пространство => экранное пространство*. Для наших целей необходимо модифицировать матрицы переходов в *пространство камеры*, *проективное пространство* и в *экранное пространство*.

Пространство камеры. Практически, это обыкновенная матрица камеры, но только расположенная в точке нахождения источника света и направлена на *shadow caster*. В случае бесконечно удалённого источника света виртуальная камера может располагаться в любой точке на прямой между *shadow caster* и источником света (обычно её выбирают близко к *shadow caster*).

Проективное пространство. Эта матрица существенно зависит от того, является ли источник света локальным или бесконечно удалённым. Для бесконечно удалённого источника света это матрица имеет вид:

$$M_{pj} = \begin{matrix} \frac{2}{width} & 0 & 0 & 0 \\ 0 & \frac{2}{height} & 0 & 0 \\ 0 & 0 & -\frac{1}{depth} & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \qquad (14)$$

где *width*, *height* и *depth* — ширина, высота и глубина проективного объёма. В этом случае не требуется гомогенизация.

Для локального источника света необходима матрица перспективного преобразования:

$$M_{pj} = \begin{matrix} \cos(\varphi/2) & 0 & 0 & 0 \\ 0 & aspect * \cos(\varphi/2) & 0 & 0 \\ 0 & 0 & \frac{\sin(\varphi/2)*far}{far-near} & \sin(\varphi/2) \\ 0 & 0 & -\frac{\sin(\varphi/2)*far*near}{far-near} & 0 \end{matrix} \quad (15)$$

где φ — угол зрения, *aspect* — отношение ширины к высоте экрана (при отрисовке в квадратную текстуру *aspect=1*), *far* и *near* — дальняя и ближняя стенки проективного объёма соответственно. После перспективного преобразования необходима гомогенизация, как и в случае локального источника при преобразовании «на землю».

Экранное пространство. Модель из нормализованного проективного пространства должна лечь в квадратную текстуру с минимальными зазорами. После того, как *shadow caster* прорисован в текстуре, её необходимо наложить на затенённые объекты (*shadow receiver*). Для этого можно воспользоваться технологией проективного наложения текстур. Для избежания повторения теневой текстуры необходимо выставить *clamp* или *border color*. В результате мы получаем тень от одного объекта (*shadow caster*) на другом (*shadow receiver*). Далее можно сделать с этой тенью:

1. Размыть (получить псевдо-мягкую тень). Сделать это можно несколькими способами:

- «Вручную» путём наложения какого-нибудь blur-фильтра. Такой подход удобен при программной отрисовке самой теневой текстуры.
- Использовать «железо». Можно отрисовать текстуру саму в себя со смещением, либо сгенерировать mip-level.

2. Уводить тень в прозрачность на основе расстояния до источника света. К сожалению, реализовать это без vertex shader при помощи аппаратной части невозможно. Единственное, что можно здесь сделать для улучшения вида тени, так это сделать её не угольно чёрной, а равномерно полупрозрачной.

3. Минимизировать затраты на отрисовку тени. Это достигается при использовании технологии LOD.

Теневые объёмы. При освещении сцены в тени оказываются те объекты, которые попадают внутрь теневого объёма [2]. Для того, чтобы точка была затенена, луч света должен войти в теневые объёмы большее число раз, нежели выйти из них. Для упрощения задачи можно принять, что не луч света от источника должен проделать этот путь, а луч от наблюдателя. При таком

упрощении возникает несколько неприятных случаев, которые рассмотрим отдельно.

Алгоритм можно разделить на два явных действия: построение «*теневой маски*» (маски освещённых и затенённых областей 2-мерной картинки) и отрисовка сцены с использованием *теневой маски*. Предлагается реализация, которую можно разбить на три стадии.

Первая стадия. На этой стадии просто рисуем всю сцену таким образом, как будто она вся затенена. Все последующие стадии служат для добавления освещения в картинку. Участки картинки, которые так и не будут освещены в последующих проходах, останутся затенёнными.

Вторая стадия. На этой стадии строим теневую маску, используя информацию о трехмерных теневых объёмах. Для создания и хранения теневой маски нам понадобится *буфер шаблонов* (*stencil buffer*). Построение оптимальных теневых объёмов является само по себе достаточно сложной задачей, которая выходит за рамки данной работы. Описание одного из алгоритмов построения силуэта, который необходим для построения оптимального теневого объёма, можно найти, например, в [3]. Итак, нам необходимо подсчитать для каждой точки разность того, сколько раз луч от наблюдателя пересёк положительно ориентированные полигоны теневого объёма и отрицательно ориентированные. Это производится за два прохода. На первом проходе, рисуем все теневые объёмы с нормальным (для нашего приложения) отбрасыванием тени поверхности, алгоритм *back face culling*. В процессе выполнения алгоритмов тенеобразования, например, в z-buffer для ускорения и упрощения просчета модель хранится в триангулятивном виде. Поэтому часто идет речь об отбрасывании тени набора треугольных поверхностей. Каждая точка, которая проходит тест по глубине, увеличивает значение в буфере шаблонов на 1. На втором проходе меняем *back face culling* на противоположный — рисоваться будут задние стенки теневых объёмов. Теперь каждая точка, которая проходит тест по глубине, будет уменьшать значение в буфере шаблонов на 1. В итоге после этих двух проходов мы имеем в буфере шаблонов нулевые значения для освещённых областей и значения больше нуля для затенённых.

Третья стадия. На этой стадии уже проведены все предварительные приготовления - есть и затенённая сцена и теневая маска. Остаётся отрисовать ещё раз всю сцену освещённой, используя проверку по буферу шаблонов.

Использование информации о глубине. Метод базируется на идее о том, что затенённые точки — это те, которые «спрятаны» от источника света. Другими словами, это «hidden surfaces» с точки зрения источника света. Для того, чтобы определить невидимые поверхности с точки зрения источника

света, необходимо отрисовать всю сцену с точки зрения источника света в буфер глубины. Далее необходимо отрисовать всю сцену с точки зрения наблюдателя, проделывая следующую проверку: если глубина точки, переведённая в систему координат источника света, больше значения, которое мы сохранили из буфера глубины на предыдущей стадии, значит, точка затенена. В противном случае точка освещена.

Если «железо» не поддерживает специфических функций (сохранение и доступа к буферу глубины, сравнение глубины с сохранённым значением, и т.д.), то реализовать его «в лоб» практически невозможно. Но т.к. метод очень привлекательный, необходимо «обойти» аппаратную часть. Предположим, что имеющееся аппаратная часть не даёт доступа к буферу глубины, у него нет никаких возможностей сравнивать значения, кроме сравнения по альфа-каналу. Единственное, что требуется от аппаратной части, так это возможность смены цветового буфера для текущей отрисовки (*render target*) и 2-текстурный растеризатор. На рис. 2. изображена сцена с точки зрения источников света, а на рис. 3. — сцена с точки зрения наблюдателя с построенными тенями.

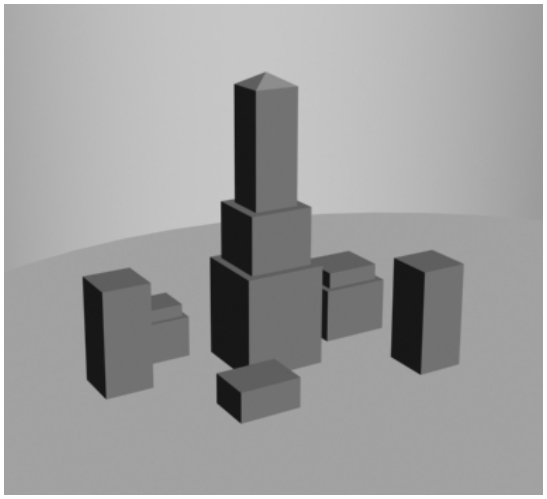

Рис. 2

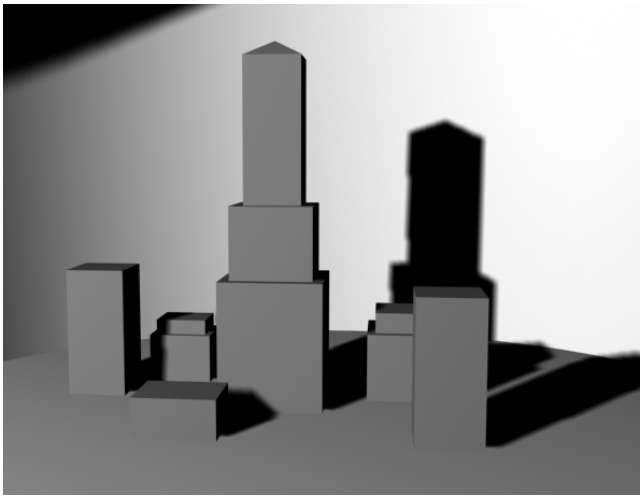

Рис. 3

Как и в предыдущем методе, алгоритм можно разделить на два явных действия: построение *теневой маски* и отрисовка сцены с использованием *теневой маски*.

Для получения глубины мы воспользуемся текстурой 256x1x32, в которой значения размещены от 0 до 255 в альфа-канале (цветовые значения нас не интересуют). Чтобы текстура легла необходимым образом, сгенерируем текстурные координаты с последующей трансформацией

$$S = \left(M_{pj} * M_{sel}\right) * P_{cam} \mid \text{нормализация} \qquad (16)$$

где M_{pj} — это матрица проективного преобразования, а M_{sel} — матрица выбора

$$M_{sel} = \begin{matrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{matrix} \quad (17)$$

После такой отрисовки, в текстуре мы имеем образ глубины сцены с дискретом в 1/256. Далее создадим теневую маску, базируясь на сравнении глубины. Для этого мы наложим первую текстуру (256x1x32) на объект способом, сходным с первой стадией, и наложим вторую текстуру, полученную на первой стадии, проективным методом. После чего, в каждой точке вычтем значения альфа-канала первой текстуры из второй и отрисуем в теневую маску с проверкой по альфа-каналу. Точки, которые пройдут эту проверку, запишем в теневую маску как 1, а которые не пройдут оставим в 0. После этого остаётся только отрисовать сцену, проставляя значения в буфер шаблонов (рис. 4).

Выводы и перспективы дальнейших исследований. Самая главная проблема при трехмерной визуализации сцены — это ошибочное самозатенение, которое возникает из-за недостаточной точности представления глубины через альфа-канал. Перспективы дальнейших исследований связаны с решением указанной проблемы.

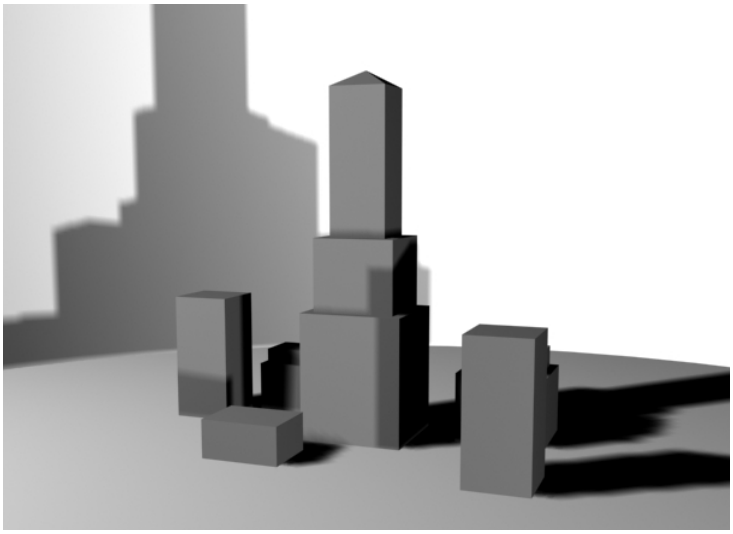

Рис.4

ЛИТЕРАТУРА

1. Blinn, James, "Me and my (fake) shadow", IEEE Computer Graphics and Applications, January 1988.
2. Franklin C. Crow. Shadow algorithms for computer graphics. In Computer Graphics (SIGGRAPH '77 Proceedings), pages 242-248, July 1977.
3. P. Sander, X. Gu, S. Gortler, H. Hoppe, J. Snyder. "Silhouette Clipping", Computer Graphics (SIGGRAPH 2000 Proceedings), pages 327-334.

Анотація.

У даній статті розглянуті та проаналізовані основні принципи і алгоритми побудови тіней в тривимірній графіці. Представлені алгоритми були розділені на дві явних дії:

- Побудова тіньової маски;
- Побудова сцени з використанням тіньової маски;

і розділені на три стадії. Стадії детально вивчені. Визначена центральна проблема побудови тіней в тривимірних сценах, в реальному часі

Abstract.

This article describes and analyzes the basic principles and algorithms of shadow creation in three-dimensional graphics. The above algorithms were divided into two distinct steps:

- Construction of the shadow mask;
- Rendering a scene using the shadow mask;

Also are divided into three stages. Stages studied in detail. Determined the central problem of shadow creation in three-dimensional scenes in real time.