# Визуализация реалистичных теней в системах компьютерной генерации изображения для аэрокосмических тренажеров

## Обзор

Данная работа описывает алгоритм генерации теней для трехмерных сцен, построенный на основе алгоритма Shadow Volumes. Главной особенностью алгоритма является построение геометрически правильных теней, как от выпуклых, так и от невыпуклых объектов, а также отсутствие зависимости от положения наблюдателя, которая существует в классическом алгоритме.

## 1. Введение

В сфере компьютерной трехмерной графики главным направлением деятельности на данный момент можно назвать задачу увеличения реалистичности изображения. Это проявляется в увеличении сложности и детализации трехмерных сцен, применении различных психологических факторов, создающих эффект присутствия, а также применении спецэффектов, таких, например, как туман, системы частиц или тени. Главным ограничением на этом пути является недостаточная производительность современных компьютеров.

Известно, что более 80% воспринимаемой информации человек получает через глаза. Одной из важных составляющих этого потока данных является восприятие тени. Во многом благодаря теням человеческий мозг получает информацию о взаимном расположении объектов в пространстве. Поэтому отображение теней является тем фактором, который может существенно улучшить реалистичность трехмерных сцен.

## 2. Классический алгоритм построения теней Shadow Volumes

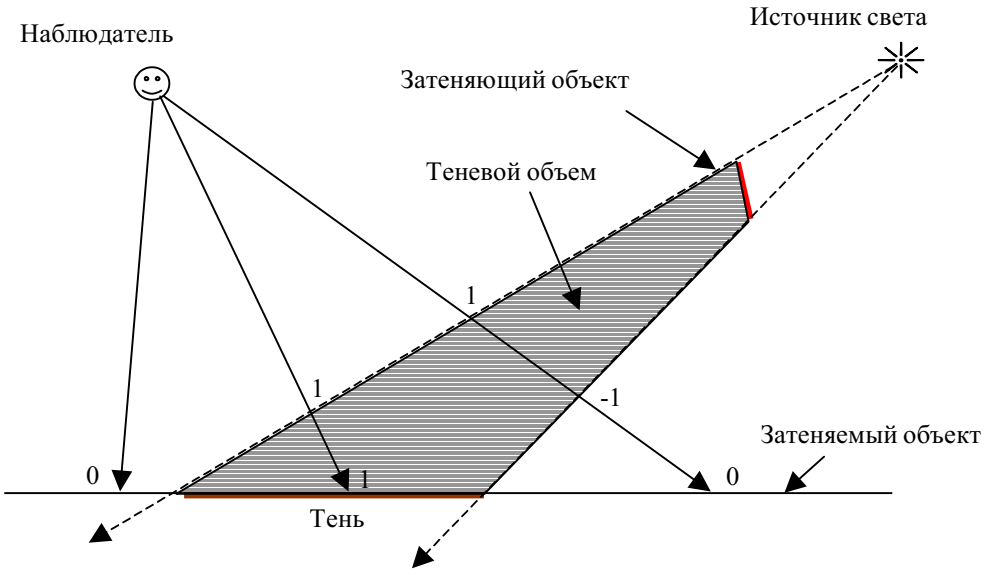

Рис. 1 “к понятию о теневом объеме”

Алгоритм Shadow Volumes [1,2] был впервые предложен Франклином Кроу (Franklin Crow) в 1977 году для генерации теней в трехмерном пространстве. Особенностями алгоритма можно назвать построение геометрически правильных теней с четкими краями, а также использование буфера шаблонов (stencil-буфера) в современных реализациях [4]. К достоинствам алгоритма можно отнести также то, что тени можно строить только от необходимых объектов, а не вообще от всех объектов сцены (хотя такая возможность есть).

Буфер шаблонов – это, по сути, дополнительная плоскость в буфере кадра (обычно 8 битов на точку), используемая для поточечного отсечения изображения [3]. Практически все современные графические ускорители поддерживают эти возможности отсечения (stencilling capabilities).

Алгоритм основан на использовании так называемых “теневых объемов” (shadow volumes). Теневой объем представляет собой область трехмерного пространства, полностью заполненную тенью (см. рис. 1). Теоретически, теневые объемы строятся по точкам силуэта объекта, отбрасывающего тень, и лучам, исходящим из этих точек по направлению от источника света. Однако практически, вместо лучей используются отрезки конечной длины, так как, во-первых, технически сложно работать с бесконечными величинами, а во-вторых, сцена всегда ограничена пирамидой видимости.

Чтобы определить, попадает ли данная конкретная точка экрана в тень, или нет, предлагается подсчитывать число пересечений луча, идущего от наблюдателя через эту точку с границами теневых объемов. Если результат нечетный – точка лежит в тени. Этот подход был впоследствии усовершенствован для использования с буфером шаблонов. Мы увеличиваем значение в буфере, соответствующее точке на экране, при пересечении лучом «видимых» граней (нормаль которых повёрнута к наблюдателю) теневых объемов и уменьшаем при пересечении «невидимых» граней (см. рис. 1). В результате мы получаем маску в буфере шаблонов, по которой впоследствии рисуем тень, причем делается это за два цикла обхода сцены, за один цикл рисуются освещенные участки сцены, а за другой – затененные. Более подробное описание дается в работах [1,2]

## 3. Тени от невыпуклых объектов

Классический алгоритм основывается на построении теневого объема по точкам силуэта объекта, отбрасывающего тень. Для выпуклых объектов достаточно построить проекцию объекта на плоскость, перпендикулярную направлению на источник света, и затем по полученным точкам построить огибающую ломаную. Точки, принадлежащие ломаной, являются точками силуэта.

В случае если объект невыпуклый, классический алгоритм не подходит по очевидным причинам. Наиболее общим решением в этом случае является алгоритм, когда для каждой «невидимой» источнику света грани объекта строится отдельный теневой объем. Однако этот вариант не является оптимальным, поскольку в этом случае будут построены лишние теневые объемы от внутренних граней.

Выход, однако, есть. Для того чтобы исключить внутренние грани, нами был адаптирован алгоритм [6]. В результате адаптации в алгоритм была внесена возможность построения силуэта от источников света, находящихся на конечном расстоянии от затеняющего объекта (перспективная теневая проекция).

```
ИщемСилуэт (набор граней объекта P, список ребер силуэта L)
{
    для каждой грани p из P
        для каждого ребра n из p
```

```
        {
            i = первая вершина n
            j = вторая вершина n
            если L[j] уже содержит i то удалить i из L[j]
            иначе добавить j в L[i]
        }
    вернем L
}
```

Таким образом, отбрасываются повторяющиеся более одного раза ребра из полного списка всех ребер всех граней, отбрасывающих тень.

## 4. Ограничение теневых объемов

Так как теневой объем состоит из граней, он, как и все другие объекты сцены, проходит геометрический конвейер. Часто бывает, что в пирамиду видимости попадает только его часть (см. рис. 2). Это приводит к некорректной работе классического алгоритма, например, когда наблюдатель находится внутри тени. Чтобы этого избежать, необходимо ограничить теневые объёмы ближней и дальней отсекающими плоскостями пирамиды видимости.

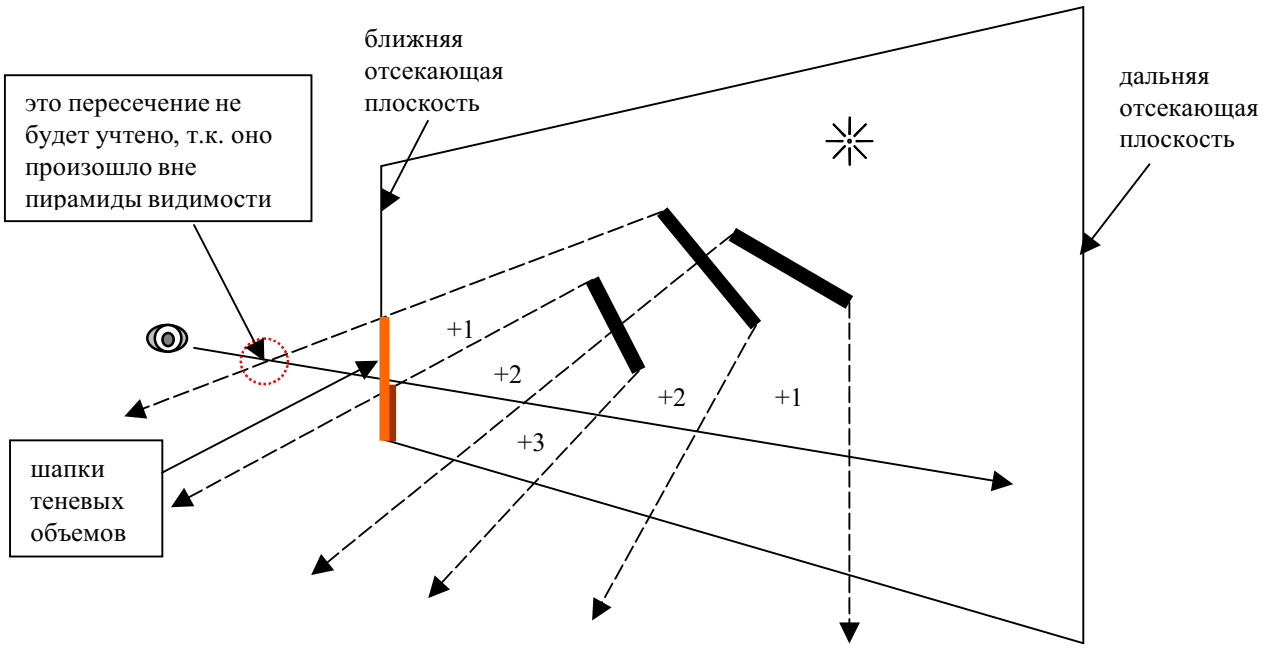


Рис. 2 “к понятию о шапках”

Шапка – это полигон, являющийся результатом пересечения теневого объема и отсекающей плоскости. Так как есть две отсекающих плоскости, различают “ближние” и “дальние” шапки. Нормаль шапки выбирается таким образом, чтобы теневой объем был всегда замкнутым.

Следует заметить, что построение шапок - сложная задача вычислительной геометрии, так как теневой объем представляет собой сложный многогранник, в общем случае невыпуклый и с дырами.

В случае построения “дальних” шапок был реализован простой способ решения. В сцену была добавлена плоскость, позади всех остальных объектов, развернутая от наблюдателя (с нормалью, направленной от наблюдателя). Это дало возможность не заботиться о “дальних” шапках, так как теневой объем в любой конфигурации замыкается этой плоскостью.

Для “ближних” шапок такой способ не подходит, потому что когда в z-буфере будет лежать подобная замыкающая плоскость, но впереди сцены, то все остальные грани сцены не смогут быть нарисованы и мы не увидим сцену. В результате мы приходим к необходимости точного подсчета шапок для каждого теневого объема.

Для того чтобы построить шапку тени, нам пришлось решить задачу нахождения пересечения теневого объема с произвольной плоскостью.

В общем случае, чтобы решить поставленную задачу для произвольного тела, нам потребовались бы сложные математические вычисления, параметризация теневого объема системой уравнений и затем решение полученной системы. Однако в нашем случае, теневой объем - не совсем произвольное тело.

Для построения ограничивающего многоугольника теневой объем можно разделить на усечённые пирамиды, каждая из которых образована:

1. Из грани (треугольника), отбрасывающей тень.
2. Из трех боковых граней (косых трапеций), образованных проецированием ребер верхней грани по направлению от источника света.
3. Из грани (треугольника), замыкающей теневой объем снизу.

Для каждой отдельной усечённой пирамиды можно достаточно просто найти пересечение с плоскостью. Решая задачу для теневого объема и ближней отсекающей плоскости, мы получим в результате набор граней, которые и составляют “ближнюю” теневую шапку.

## 5. Проблемы с точностью

В процессе работы были решены задачи, связанные с недостаточным уровнем точности вычислений - как на уровне алгоритма, так как размеры теневого объема значительны, так и на уровне используемого API (DirectX8). DirectX8 использует для внутреннего представления числа 4 байта, что эквивалентно максимальному десятичному числу $4.3 \times 10^9$. В некоторых случаях (в частности, для случая “космических” расстояний) этой точности оказывается недостаточно. В этом случае “невидимые” грани теневого объема могут “не сшиваться” по краям, оставляя после себя видимые артефакты, например, в виде линий из точек.

Для того чтобы максимально нивелировать артефакты, связанные с недостаточной точностью, нами были использованы приемы:

1. На уровне алгоритма реализован как ручной (через UI), так и автоматический выбор расстояния, на которое производится проекция тени. Это позволяет обеспечивать достаточно гибкий контроль размеров теневого объема.
2. На уровне API применяется преобразование формата списков граней из простого линейного списка (LIST) в список-полоску (STRIP). Это позволяет, во-первых, сократить объем используемой памяти, а во-вторых, практически избавиться от артефактов, связанных со сшиванием граней.

Еще одним примером, иллюстрирующим проблему с точностью, является рисование шапок, построенных на ближней отсекающей плоскости. Недостаточно просто отправить их в графический конвейер - из-за недостаточной вычислительной точности рисование граней, слишком близко расположенных к отсекающей плоскости, становится проблематичным. В этом случае нами был реализован перенос граней шапки на некоторое расстояние от

наблюдателя вглубь сцены и последующее масштабирование.

## 6. Производительность

Тема производительности является одной из самых важных, когда заходит разговор о генерации теней в реальном времени. Генерация теней является затратной операцией, так как каждый теневой объем имеет геометрическую сложность, сопоставимую со сложностью объекта, от которого он был образован. В самом худшем случае, когда все объекты сцены отбрасывают тень, нагрузка на API и лежащее за ним аппаратное обеспечение возрастает как минимум в два раза. Более того, так как теневые объемы считаются динамически, в классическом алгоритме их нужно строить отдельно для каждого нового кадра.

Предложенный алгоритм изначально разрабатывался с таким расчетом, чтобы получить максимально возможную производительность при достаточном качестве. В процессе работы над алгоритмом был разработан и реализован ряд оптимизаций, которые позволили существенно сократить время работы:

1. Прежде всего, это однопроходная реализация, в отличие от классического варианта. Вместо второго цикла обхода сцены для рисования тени применяется смешивание цветов, грубо говоря, уже сгенерированный буфер кадра смешивается с плоскостью цвета тени, по маске из буфера шаблонов.
2. Во-вторых, оптимизация уже заложена в самом алгоритме – мы можем выбирать те объекты сцены, от которых хотим получить тень, а все другие игнорировать.
3. Еще один способ оптимизации, как следствие уже существующих возможностей системы, заключается в использовании информации об ограничивающих сферах объектов сцены. Эта информация при подсчете “ближних” шапок позволяет сократить число исследуемых ребер на одну треть, так как если сам объект не пересекается с ближней отсекающей плоскостью, то и все его грани (по части из которых построен теневой объем) не пересекаются с этой плоскостью.
4. Известно, что теневой объем, как тело в пространстве, зависит от положения источника света, положения объекта отбрасывающего тень и не зависит от положения камеры. В приложениях, где полезная работа осуществляется за счет движения камеры, а сцена по большей части статична (таких, как большинство современных 3D стрелялок) это может играть существенную роль: теневой объем может просчитываться только один раз на все время жизни сцены (прегенерация). В общем случае, когда сцена не является статичной, подобная оптимизация тоже имеет право на жизнь, так как позволяет сократить вычисления (в худшем случае мы получаем ту же картину, как и без оптимизации).
5. Если же у нас неподвижна и камера и источник света и теневые объемы, это дает возможность не строить заново маску в буфере шаблонов, так как маска, сохранившаяся с предыдущего кадра, все еще валидна. Таким образом, время работы алгоритма сокращается практически до нуля. Этот случай имеет место, когда, например, движутся объекты, не отбрасывающие тень.

Кроме всего прочего нами был реализован управляющий интерфейс (через UI) который дает возможность максимально гибко настроить алгоритм в соотношении скорость/качество для каждой конкретной задачи. В число возможностей, среди прочего, входят:

1. отключение теней (и всех затрат на них со стороны системы)
2. изменение цвета и интенсивности тени
3. выбор параллельной или перспективной проекции теневых объемов
4. отключение генерации шапок, как дальних, так и ближних

## 7. Результаты

Все результаты получены на тестовом стенде с конфигурацией 256Mb RAM, GeForce3, модель: МКС (Международная космическая станция) ~ 20000 граней. Измерения проводились в полноэкранном режиме.

| **Число граней затеняющих объектов** | 0 | 4258 | 6787 | 9720 | 11710 | 16489 |
|---|---|---|---|---|---|---|
| **Время генерации кадра, мс (минимальное/среднее/максимальное)** | 15 | 15/20/31 | 15/24/35 | 15/27/41 | 15/29/45 | 15/31/62 |

## 8. Заключение

В отличие от классического варианта алгоритма [5], предложенный вариант обладает рядом существенных достоинств, в частности, реализованные оптимизации позволяют существенно сократить среднее время работы до 0% - 50% от времени работы классического алгоритма. В заключение следует отметить, что представленный в данной работе алгоритм был успешно реализован и нашел практическое применение в системе визуализации реального времени. Алгоритм успешно применяется в аэрокосмических тренажерах, которые используются в настоящий момент для подготовки космонавтов.

## 9. Список литературы


[1] Crow, F. (1977) “Shadow Algorithms for Computer Graphics”, *Computer Graphics* 11(2) 242-247
[2] Bergeron, P. (1986) “A General Version of Crow’s Shadow Volumes”, *IEEE CG&*A, 6(9), 17-28
[3] Роджерс, Д. Алгоритмические основы машинной графики, М.: Мир, 1990.
[4] McReynolds, T., Blythe, D., (1998) “Advanced Graphics Programming Techniques Using OpenGL”, *SIGGRAPH’98 Cours*e.
[5] Кочергин А.А., Каипов Н.Р., Михайлюк М.В., Торгашев М.А, Моделирование теней в системах видеотренажеров // труды четвертой международной научно-практической конференции «Пилотируемые полеты в космос» 21-22 марта 2000 года, Звездный городок.
[6] DirectX 8.1 SDK (2001) “Shadow Volume Sample”. Microsoft Corp., http://www.microsoft.com/DirectX