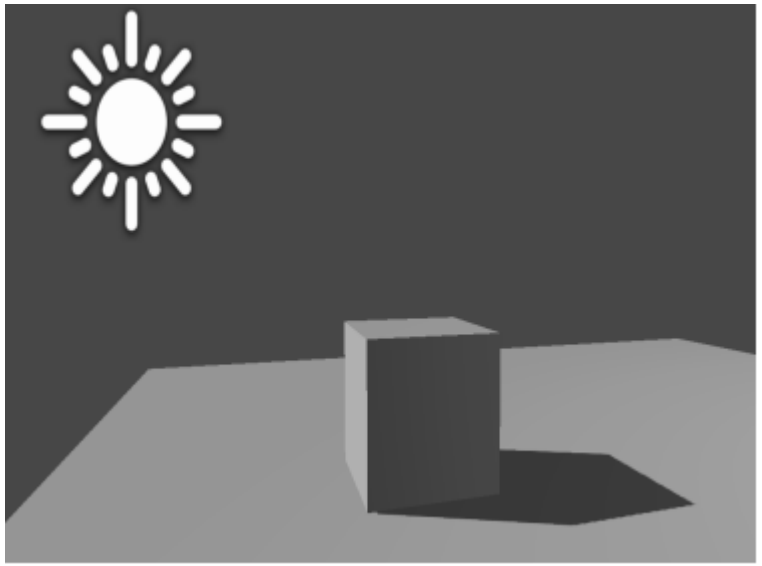
Наложение теней в реальном времени

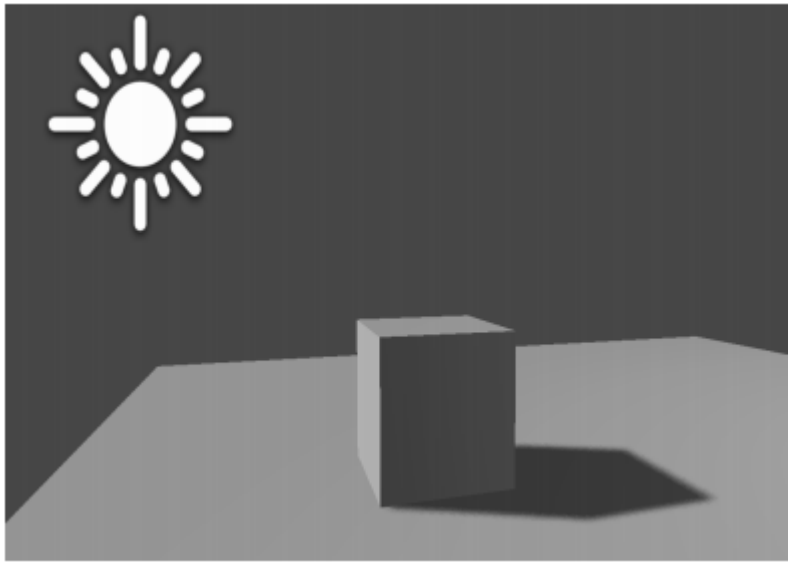
Компьютерная графика

Тени в жизни

Типы теней



Жёсткие тени (hard shadows)



Мягкие тени (soft shadows)

Жёсткие тени

Согласно геометрической оптике Френеля жёсткие тени образуются от точечных или направленных источников света

Мягкие тени

Протяжённый источник — набор большого числа точечных источников света.

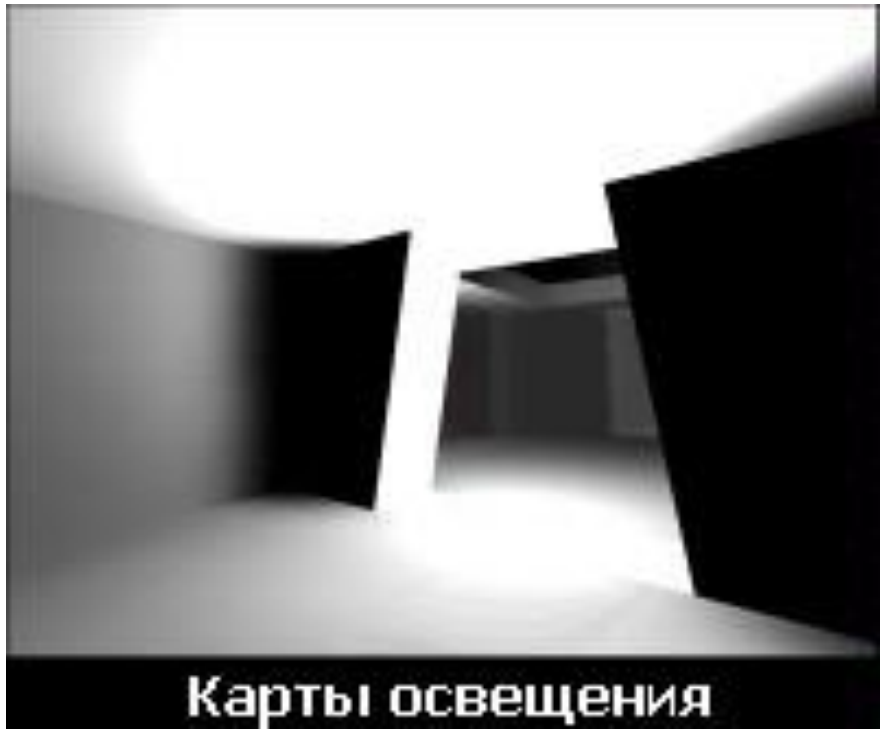
Получается не одна тень, а целая серия теней, накладываемых друг на друга, в результате чего образуются полутени.

Методы рендеринга теней

- **Карты освещения**
- Вершинные тени
- Проектируемая геометрия
- Теневые объемы
- Теневые карты
- Теневые буферы
- Буферы ObjectID / приоритетов

Карты освещения — lightmap

Метод освещения пространства, заключающийся в том, что создаётся текстура, содержащая информацию об освещённости трёхмерных моделей

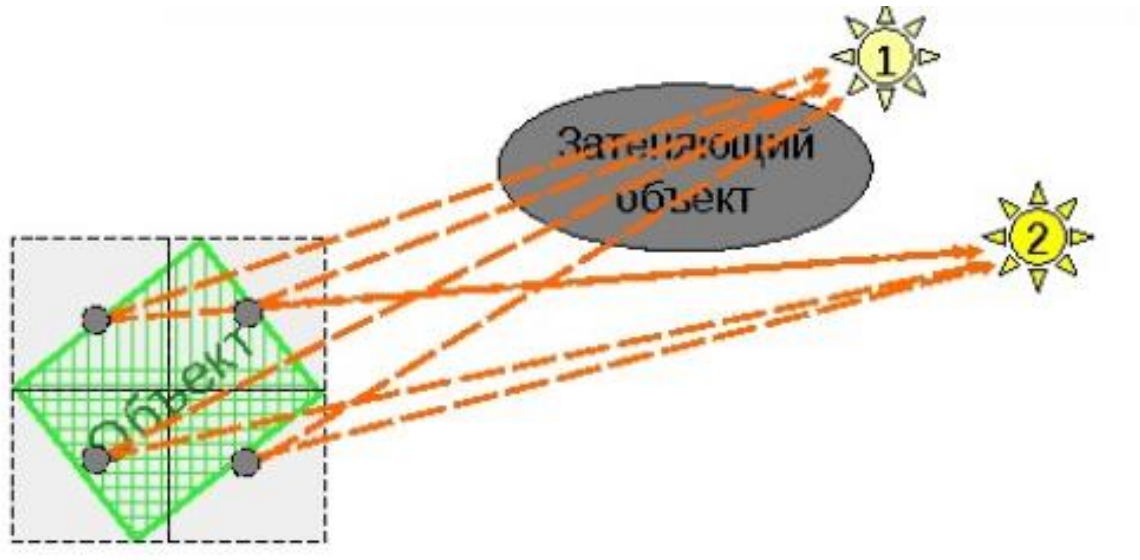


Поддерживается такими игровыми движками, как id Tech, Unreal Engine, Lithtech, GoldSrc, Xash3D, Source, X-Ray, Unity

Использование карт освещения

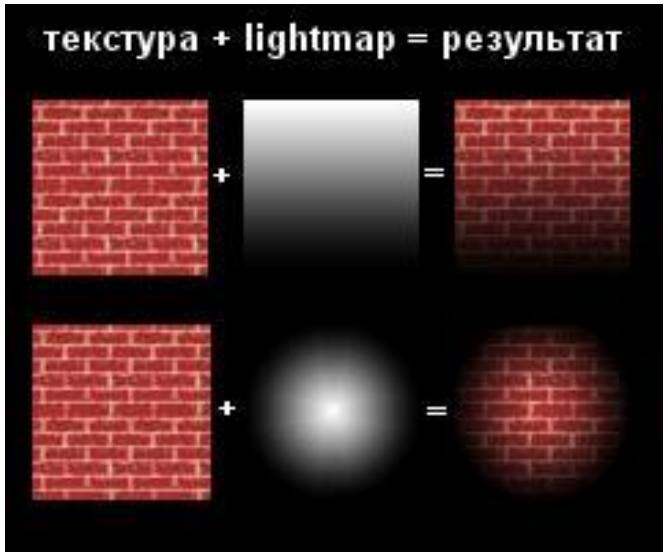
- Используются для расчёта освещения, в том числе теней
- Генерируется для статической геометрии до начала цикла рендеринга, и во время рендеринга в основном не изменяется
- Карты освещения для движущихся моделей создают только в случаях, если источник света присоединён к движущейся модели и результат её освещения не зависит от её перемещения и вращения.

Разрешение карты освещения



- Для каждого полигона рассчитывается освещённость для каждого тексела его карты
- Каждый тексел карты освещения соответствует 4-32 текселам текстуры

Фильтрация и наложение текстур



- В целях улучшения внешнего вида картинки, к картам освещения часто применяется билинейная фильтрация. Эти операции повторяются для каждого освещаемого полигона сцены.
- Во время рендеринга, карты освещения могут накладываться вторым проходом, с использованием альфа-блендинга. При наличии мультитекстурного оборудования можно накладывать текстуру и карту освещения за один проход.

LightMap: Плюсы и минусы

+ Самое высокое качество получаемого освещения

+ Не требует высокополигональных сцен

– Долгий расчет

– Занимает много видеопамяти

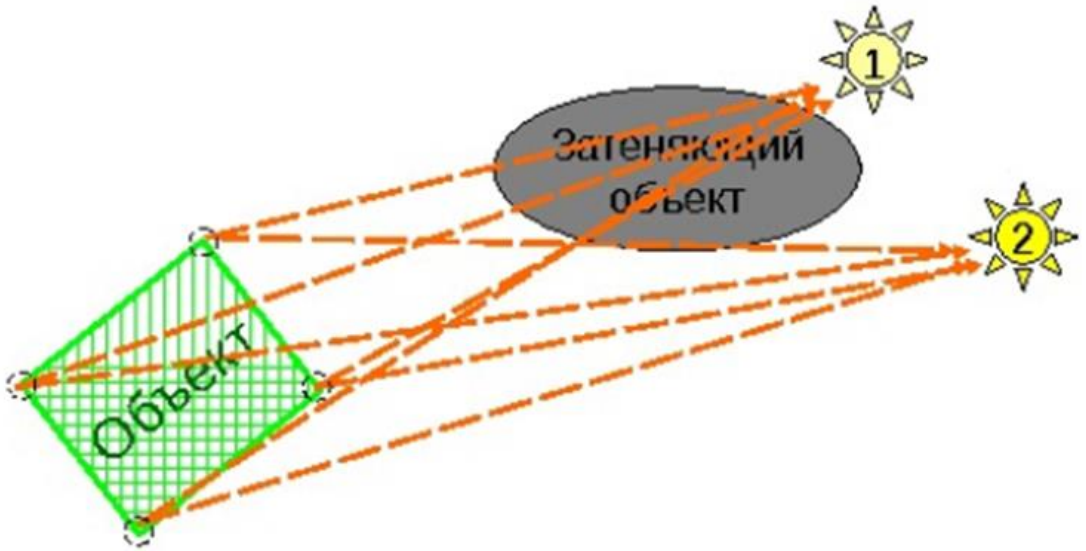
– Без упаковки LightMap'ов в одну текстуру из-за большого количества смен текстур долго

– При рендере занимает один текстурный юнит, который с успехом мог бы пойти на детализированную текстуру

Методы рендеринга теней

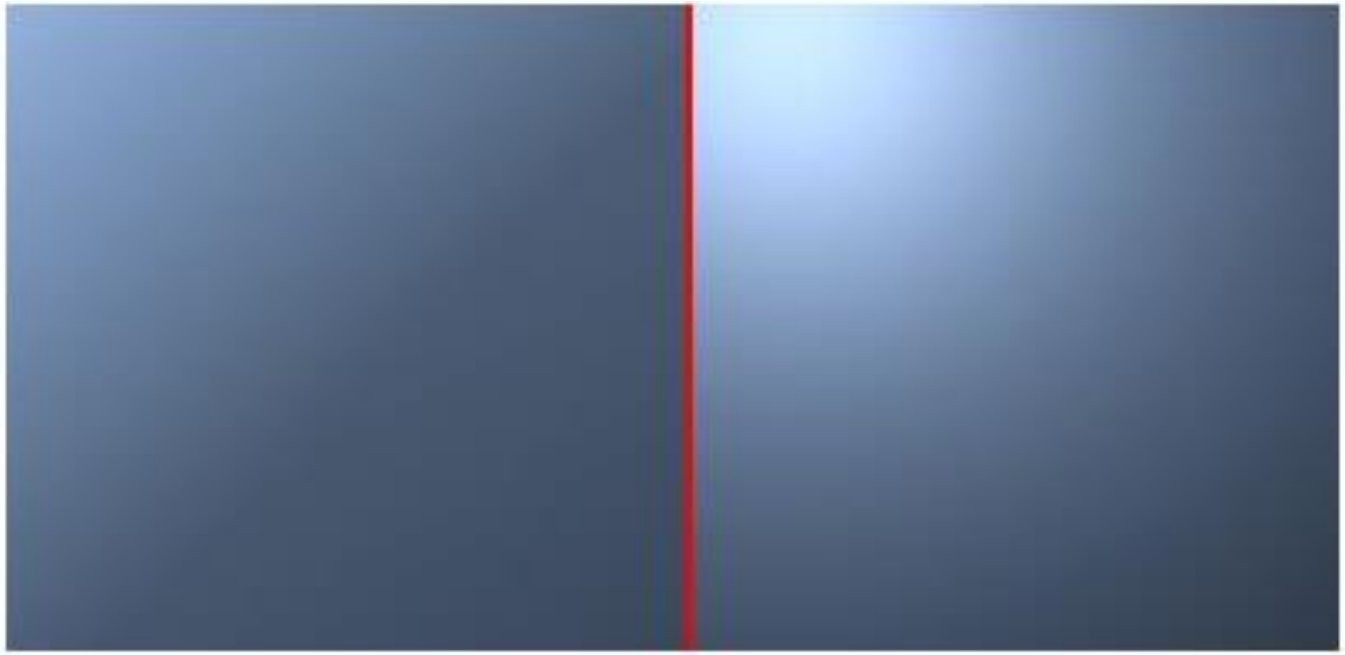
- Карты освещения
- **Вершинные тени**
- Проектируемая геометрия
- Теневые объемы
- Теневые карты
- Теневые буферы
- Буферы ObjectID / приоритетов

Вершинные тени

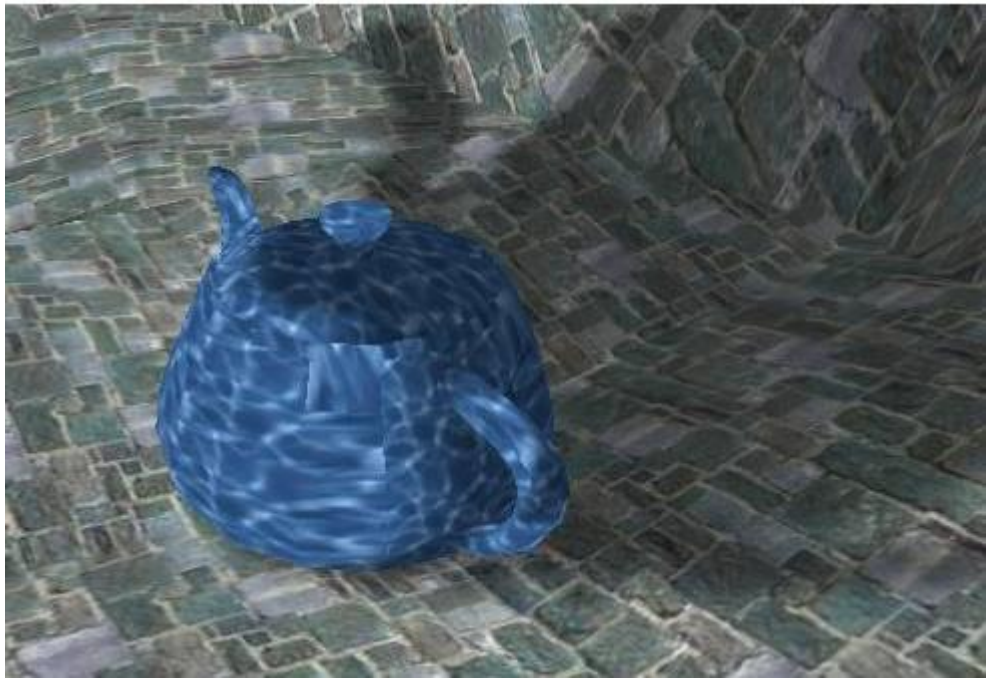


- Вершинное освещение, как правило, применяется для объектов, состоящих из большого количества маленьких полигонов
- Освещённость рассчитывается для каждой вершины полигональной сетки, и во время рендеринга интерполируется по полигону

Vertext Lighting vs Lightmapping



Вершинное освещение для высокополигональных моделей



VertexLight: Плюсы и минусы

+ Быстрый расчёт для одного объекта
+ Требует малое количество видеопамяти
+ Не занимает текстурный юнит
+ Работает быстрее, чем LightMap, так как нет необходимости менять текстуры
+ Может быть немного качественнее LightMap'ов из-за отсутствия текстурных фильтров

– Нужны высокополигональные объекты для качественного результата
– Тени в среднем все равно будут хуже, чем у LightMap'ов
– Расчёт может быть не таким уж и быстрым, только из-за того, что используются высокополигональные объекты

Где Vertext Lighting и где Lightmapping?

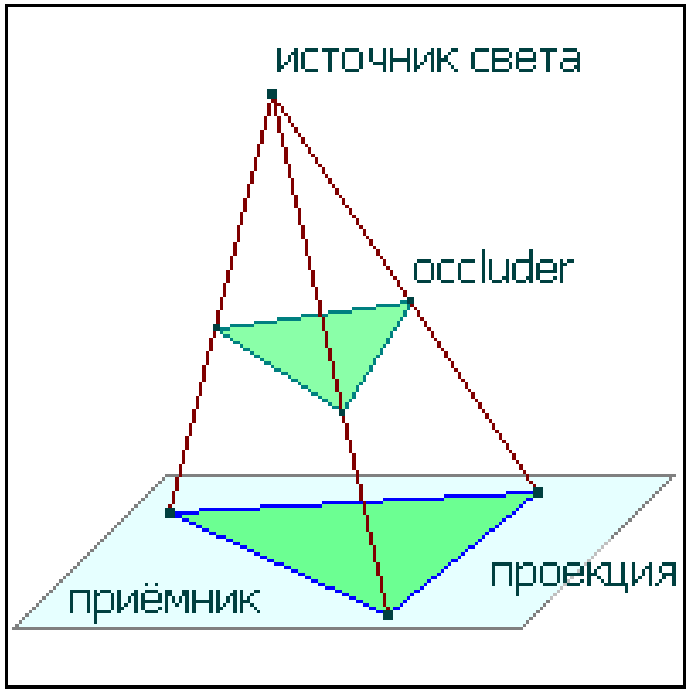
Где хорошо Vertext Lighting ?

Методы рендеринга теней

- Карты освещения
- Вершинные тени
- **Проектируемая геометрия**
- Теневые объемы
- Теневые карты
- Теневые буферы
- Буферы ObjectID / приоритетов

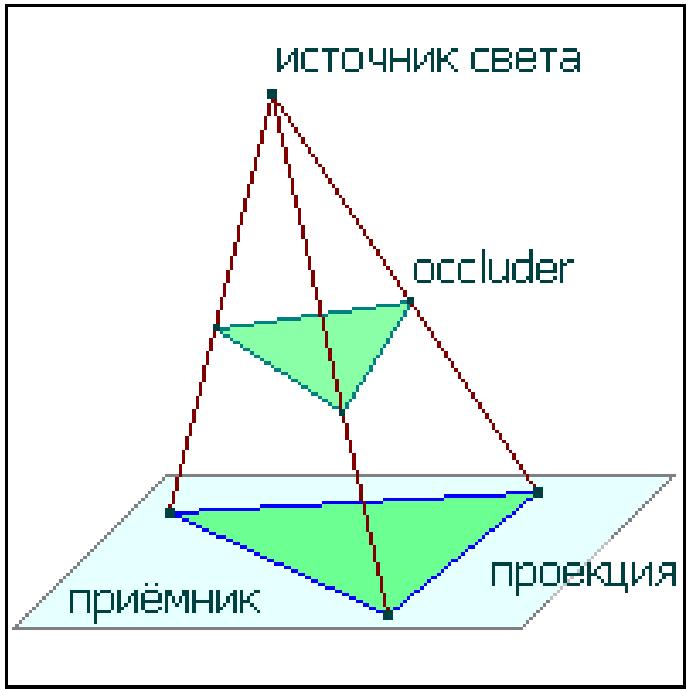
Проектируемая геометрия

- Алгоритм относится к аналитическим
- Предполагается, что множество небольших occluder'ов отбрасывает тени на малое количество больших плоских объектов
- С помощью этого алгоритма нельзя рисовать все тени сцены, лишь самые важные



Проектируемая геометрия: основная идея

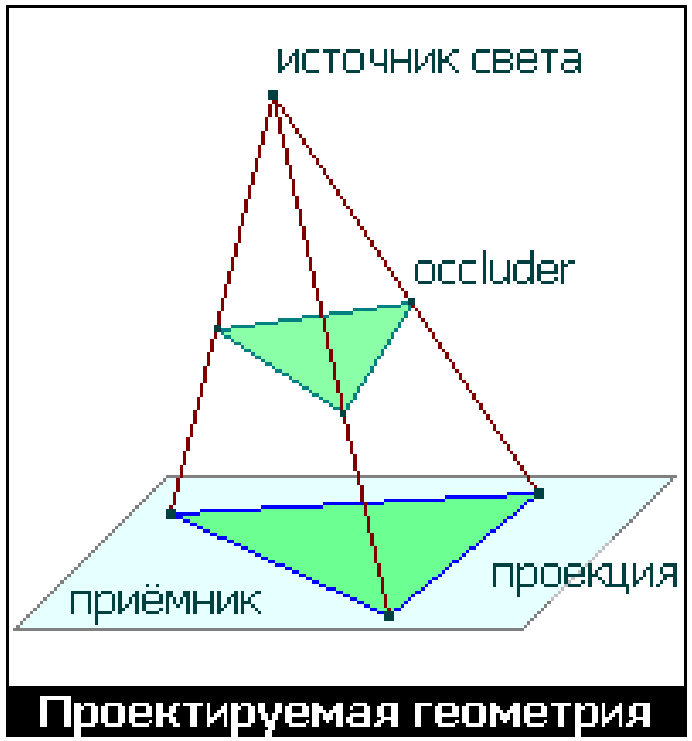
- Основная идея алгоритма – проецирование объекта из точки положения источника света на приёмник теней
- Проекция получается применением трансформации модель/вид к оригинальному объекту: S = V * M * P



Occluder рисуется дважды:

1. один раз с нормальными матрицами V и M и оригинальными цветами и текстурами,

2. а второй раз - с использованием полученной матрицы S, цветом тени



Особенности реализации

Проецирование осуществляется с помощью CPU, который должен также корректно обрабатывать случай выхода тени за границу приемника, т. е. отсекать тень по приёмнику

Вместо рисования проекции цветом тени, можно использовать alpha-blending для уменьшения яркости приёмника в областях тени, тогда сквозь тень видно текстуру приемника

В случае использования прозрачности возникает проблема двойного затенения: пересекающиеся спроецированные полигоны будут затенять приёмник 2 раза, что выглядит некорректно

Поэтому нужно искать минимальный силуэт объекта или избегать двойного наложения с помощью destination alpha или stencil-операций

Полигоны проекции нужно располагать не на самом приёмнике, а на некотором расстоянии от него, чтобы избежать ошибок точности z, либо опять использовать stencil-буфер

Проектируемая геометрия: плюсы и минусы

Метод хорош тем, что не имеет проблем со ступенчатостью тени и использует ресурсы процессора меньше, чем остальные алгоритмы.

Однако, предполагается, что в сцене существует не так много больших приёмников тени.

С ростом количества приёмников и количества/сложности объектов алгоритм быстро становится неприемлемым из-за интенсивного использования процессора и большого количества необходимых проходов рендеринга.

Также требуется 32-битный z-буфер (для stencil) или 32-битный framebuffer (для destination alpha).

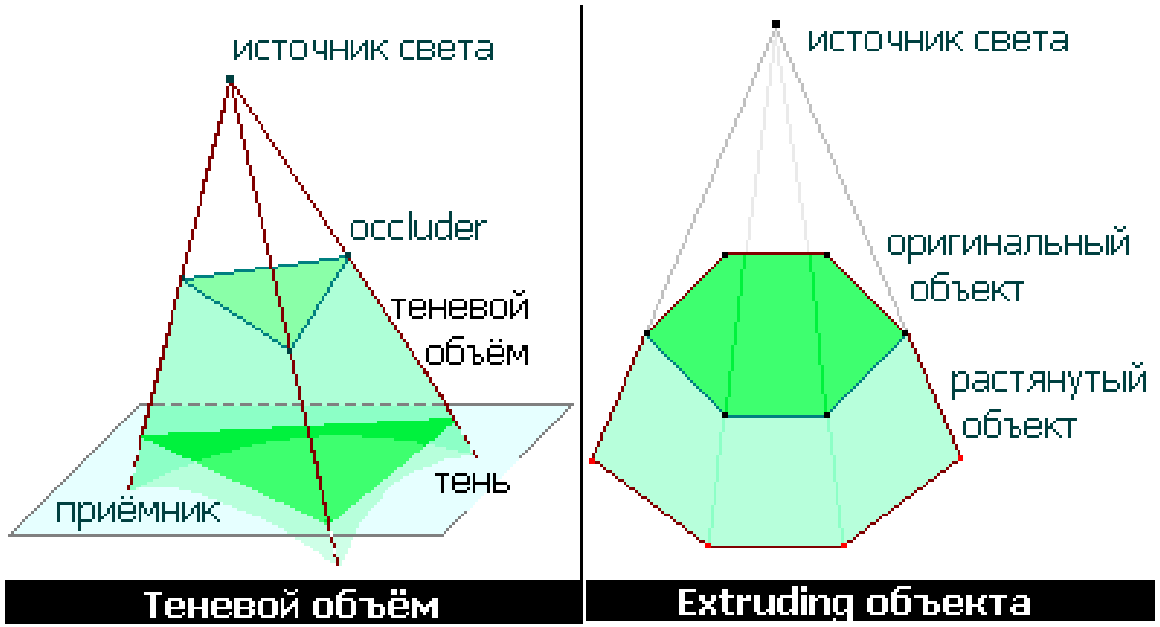
В результате получаются чёткие тени с острыми краями, что не подходит для большинства типов источников света.

Таким образом, этот метод вряд ли можно назвать общим решением проблемы теней, он подходит только для того, чтобы добавлять тени в отдельные области, где они наиболее заметны и необходимы.

Методы рендеринга теней

- Карты освещения
- Вершинные тени
- Проектируемая геометрия
- **Теневые объемы**
- Теневые карты
- Теневые буферы
- Буферы ObjectID / приоритетов

Теневые объемы



- Алгоритм также относится к аналитическим
- Основан на представлении затенённого пространства полигональным многогранником
- Многогранник генерируется на этапе препроцессинга путём проектирования каждого ребра минимального силуэта объекта вдаль от источника света.

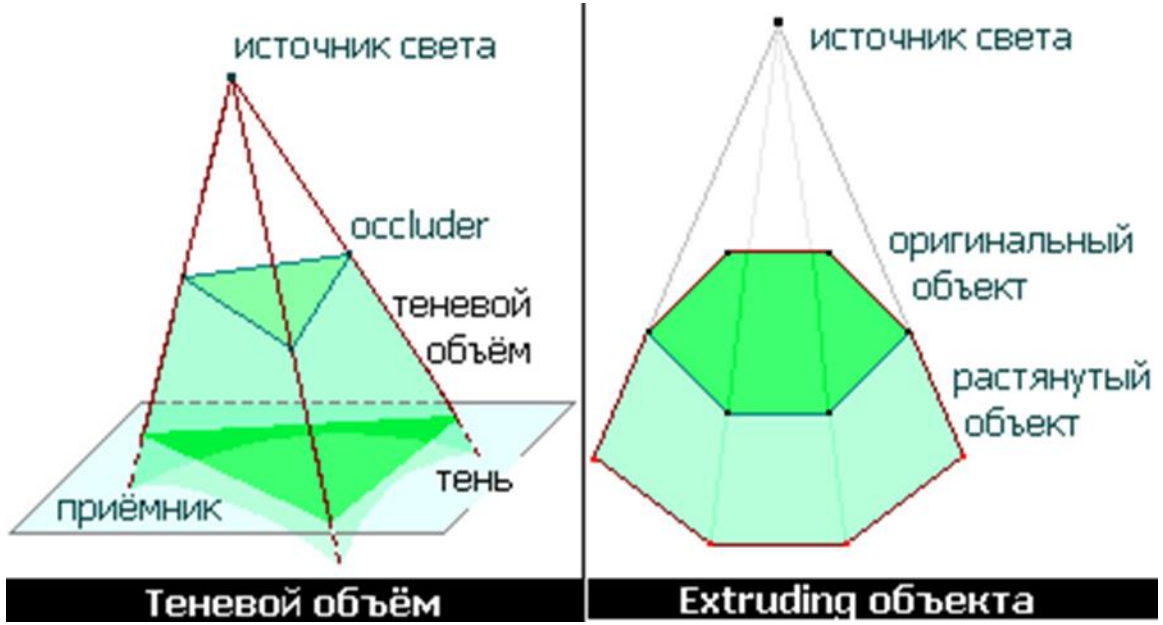
Методы генерации теневого объёма

Существует два основных метода генерации теневого объёма:

- проектирование ребер минимального силуэта объекта
- растягивание/выдавливание (Extruding) модели, с помощью перемещения задней части модели вдаль от источника света.

Первый работает всегда, на любых объектах, даже на невыпуклых, но реализуется только с помощью процессора.

Второй метод возможен только для «правильно» тесселированых выпуклых объектов, зато может быть реализован в Vertex-шейдере.



Теневые объемы: особенности

- Алгоритм работает достаточно эффективно только с использованием современного оборудования, поддерживающего z- и stencil-буферы
- До начала генерации теней сцена рисуется без учёта источников света, генерирующих тени
- Перед началом обработки каждого источника света stencil-буфер очищается

Теневые объемы: принцип действия

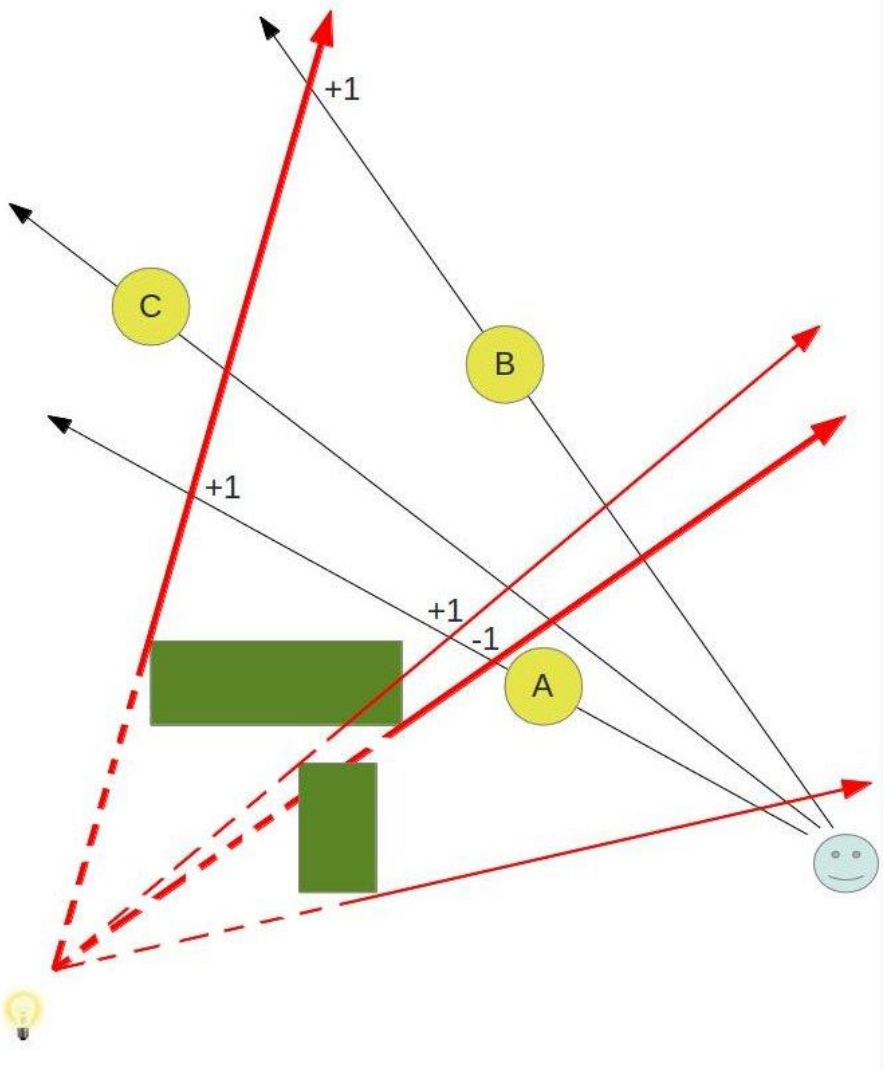
В начале все теневые объёмы данного источника рисуются без добавления информации о цвете (то есть, только в z-буфер), при этом каждый раз, когда пиксель теневого объёма проходит z-тест, соответствующая ячейка stencil-буфера увеличивается на единицу.

Потом режим отбрасывания нелицевых полигонов инвертируется, и теневой объём рисуется еще раз, уже уменьшая соответствующие ячейки stencil’а.

Другими словами: передняя часть объёма увеличивает значения в stencil-буфере, а задняя уменьшает, в результате большинство пикселей в stencil-буфере остаются нулями, а пиксели с положительными значениями образуют тень (это происходит потому, что части многогранника, представляющего объём, не проходят z-тест по уже нарисованной геометрии). Финальный проход добавляет освещение в незатенённые области.

Проверка на попадание в сложный теневой объем

a) Depth pass/Z-pass
Задние и передние грани теневого объёма рисуются в два прохода.
В каждом проходе различаются настройки рендера.
1. Отключим запись в буфер глубины и цвета
2. Включим отсечение задних граней. Установим операцию трафарета в инкрементирование при прохождении теста глубины. Нарисуем объём.
3. Включим отсечение передних граней. Установим операцию в уменьшение на единицу при прохождении теста глубины. Нарисуем объём.
Все освещённые точки будут иметь значение 0 в буфере трафарета.



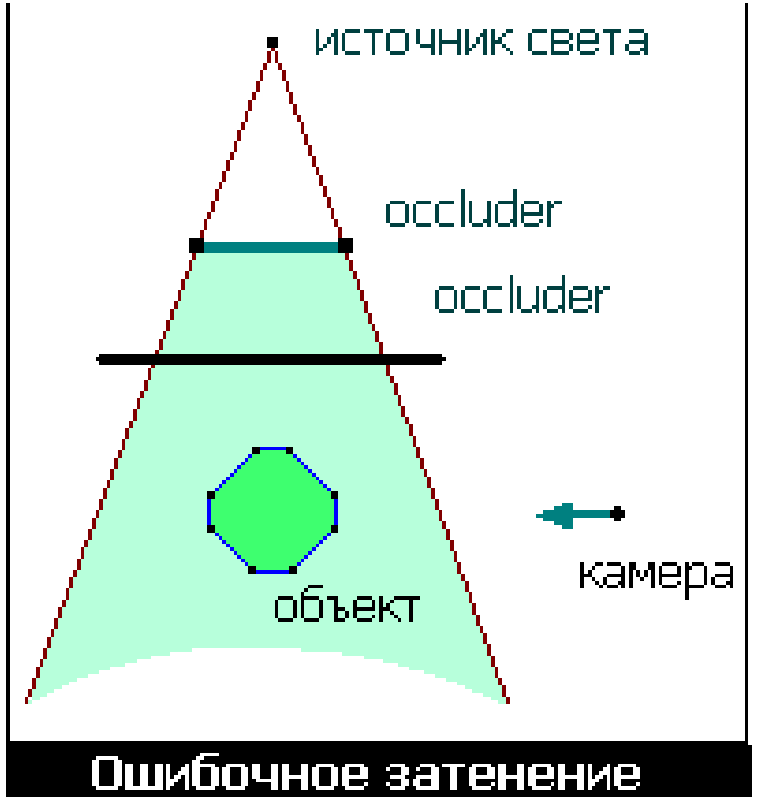
Теневой объём: проблемы

При больших размерах конуса тени видеокарта будет вынуждена его отсекать, что снижает производительность

Если метод используется для генерации лишь некоторых теней из всех возможных в сцене (что бывает чаще всего), то возникают нежелательные артефакты.

Первый occluder отбрасывает тень, генерируется теневой конус, проходящий сквозь второй occluder, и затеняет объект, который вообще не освещается этим источником света.

Этот артефакт может возникнуть только в том случае, если в финальной части алгоритма свет не добавляется в сцену, а вычитается из нее. Если данные stencil-буфера используются для маскирования освещения от данного источника света, то ничего страшного не произойдет: запрещается рисование на и так не освещенный регион; если же по stencil'у строится альфа-тень, то объект будет ошибочно затенен.



Теневой объем: Плюсы и минусы

+ Намного большая по сравнению с проектируемой геометрией универсальность – производительность больше не зависит от сложности и количества приёмников тени, тень может падать на сколь угодно сложную поверхность

– Алгоритм слишком требователен к ресурсам, чтобы обеспечить генерацию всех теней сцены в реальном времени, к тому же размеры структур данных для любой достаточно сложной сцены могут достигать сотен мегабайт

– Производит тени с острыми краями

– Невыпуклые объекты не будут отбрасывать тени сами на себя (рука не затеняет торс)

Методы рендеринга теней

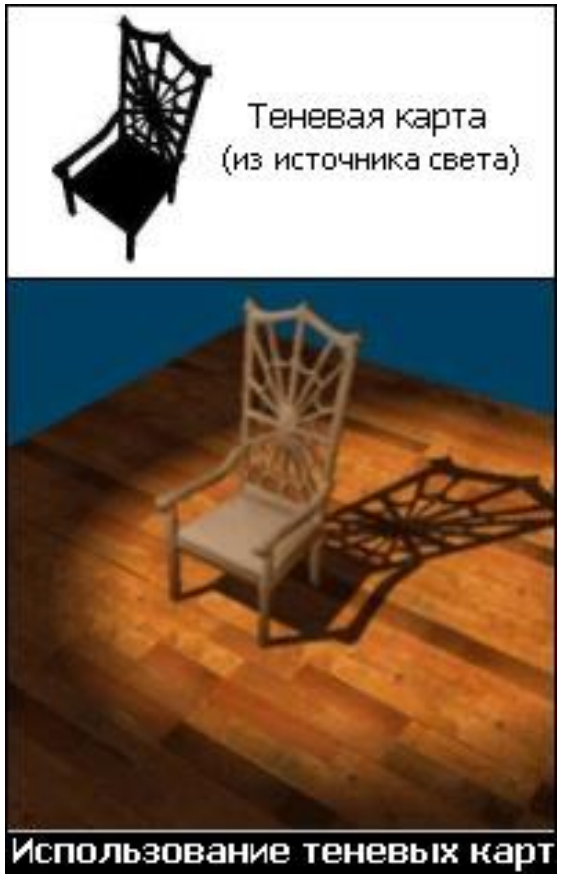
- Карты освещения
- Вершинные тени
- Проектируемая геометрия
- Теневые объемы
- **Теневые карты**
- Теневые буферы
- Буферы ObjectID / приоритетов

Теневые карты

Попиксельный (sample-based) алгоритм

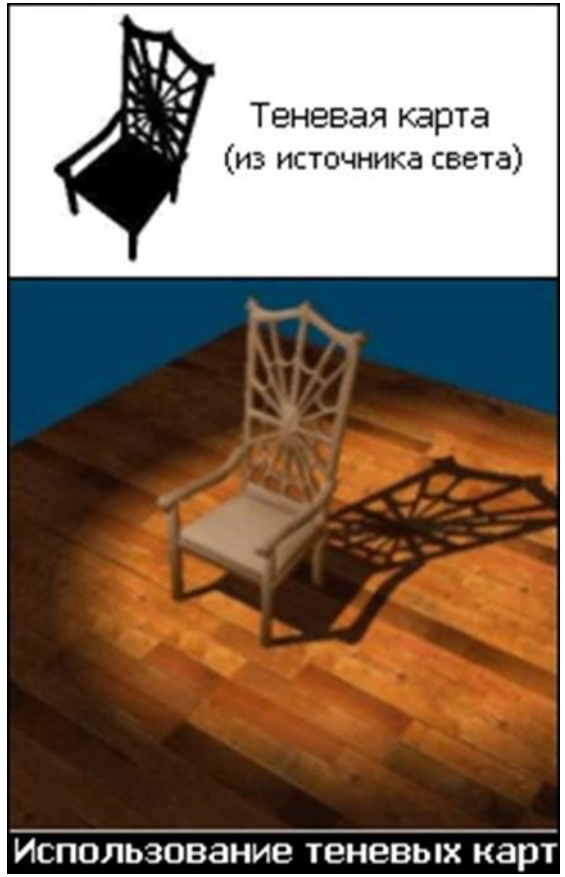
Алгоритм, основанный на использовании карт теней, можно считать более простым вариантом алгоритма теневых буферов

Для каждой пары «источник света – объект» создаётся теневая карта, которую необходимо спроектировать на все приёмники теней



Теневые карты: первый шаг алгоритма — генерация карт теней

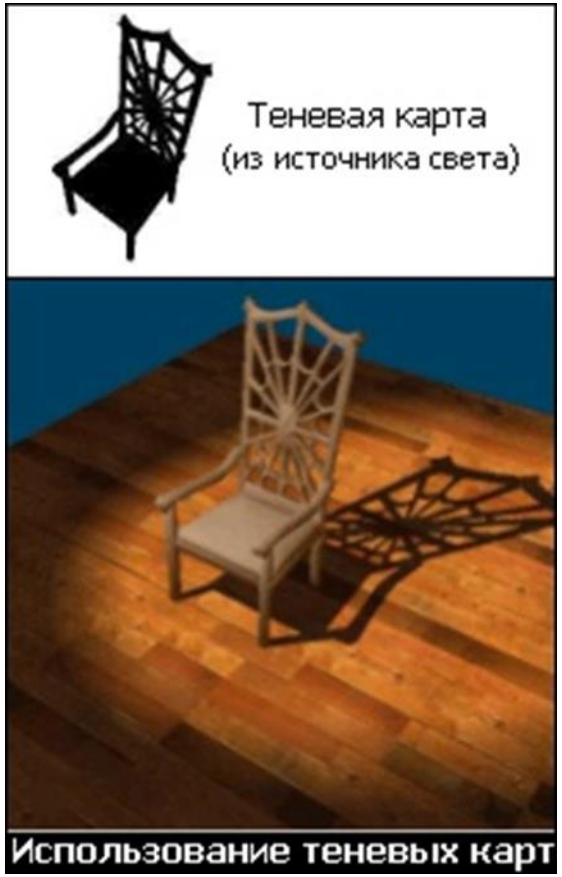
- Текстура очищается белым цветом, потом на нее из точки положения источника света рисуется чёрным shadow caster
- Для статических объектов и источников света это можно сделать перед началом цикла рендеринга, для движущихся придется делать в начале рисования каждого кадра



Теневые карты: выяснить, на какие объекты падает тень

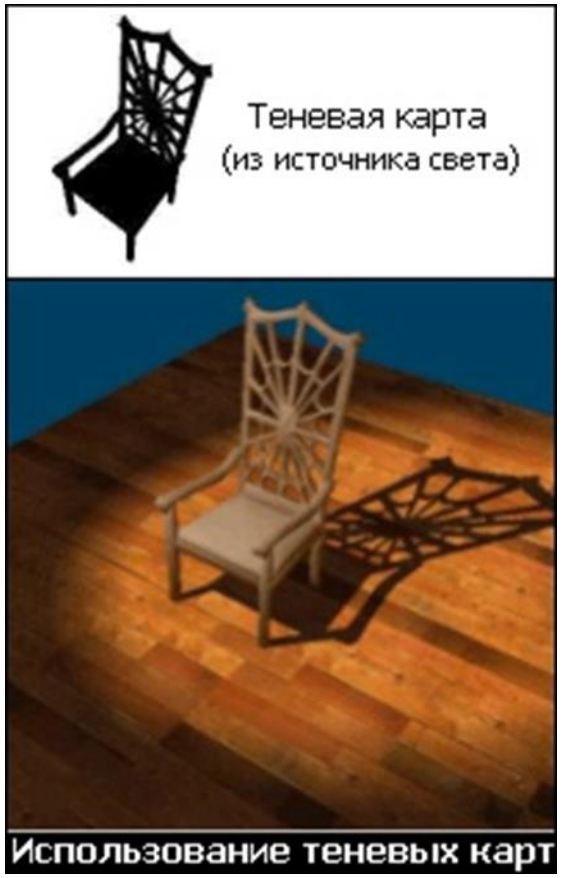
- Для этого строится усечённый конус (frustum), который соответствует матрице вида, использовавшейся для рендеринга теневой карты, ограниченный виртуальной плоскостью окончания действия источника света с одной стороны, и самим источником с другой.

- Все потенциальные приемники теней проверяются на попадание в найденный frustum. На затенённые объекты накладывается тень.



Теневые карты: принцип действия

- Сначала объекты рисуются с обычными текстурами и текстурными координатами
- Затем каждая теневая карта устанавливается как активная текстура для объекта, проективные текстурные координаты генерируются, используя матрицу вида источника света, и объект рисуется еще раз в соответствующем альфа-режиме
- На оборудовании, которое поддерживает мультитекстурирование, можно рисовать объект и накладывать несколько теней за один проход



Теневые карты: Плюсы и минусы

+ Алгоритм очень простой
+ Сравнительно быстрая генерация теней для небольшого количества объектов
+ Необходима только одна дополнительная прорисовка для каждого shadow caster'a
+ Алгоритм имеет проблемы с aliasing'ом, но допускает простое и дешевое их решение путем билинейной фильтрации

– Shadow caster'ы не отбрасывают тени сами на себя – если рассматривать объекты как приёмники теней, то они будут полностью затеняться своей же теневой картой. Поэтому на объекты нельзя проектировать их тени. Эта проблема существенна только для сложных, невыпуклых объектов
– Для каждой пары «объект - источник света» необходим свой render-to-texture, что ограничивает возможное количество таких пар

Методы рендеринга теней

- Карты освещения
- Вершинные тени
- Проектируемая геометрия
- Теневые объемы
- Теневые карты
- **Теневые буферы**
- Буферы ObjectID / приоритетов

Теневые буферы

- Алгоритм базируется на том, что область, находящаяся в тени относительно источника света, не видна из точки его положения
- Для начала сцена рисуется из источника света во внеэкранный буфер
- В z-буфере (теневом буфере) записаны глубины всех ближайших к источнику света точек. Любая точка, находящаяся дальше соответствующей точки в теневом буфере, будет в тени

Теневые буферы

Аппаратная многопроходная реализация метода была предложена на SIGGRAPH'92 Марком Сигалом.

Первый шаг — получение теневой карты путем рендеринга сцены из точки источника света.

Матрица вида равна матрице «look at» источника света, а матрица проектирования зависит от типа источника света. Так, для конических источников угол конуса равен FOV, для направленных используется ортографическая проекция, для точечных используется cubemap и каждая грань куба рисуется с FOV=90.

Для статических сцен это можно сделать до начала цикла рендеринга, для динамических придется делать в начале каждого кадра.

При рендеринге теневой карты необходимо отодвигать все полигоны на небольшое расстояние для учета ошибок точности и предотвращения самозатенения.

Теневые буферы и точечные источники

То, что теневые буферы лучше всего работают с коническими и направленными источниками света, не исключает возможности их применения к точечным источникам.

В этом случае необходимо произвести рендеринг на шесть поверхностей кубической карты, чтобы получить глубину каждой точки.

Эти буферы нельзя использовать, как в случае направленных или конических источников света.

Для сравнения расстояний нужно применять функции глубины типа 1-sqtr(x^2+y^2+z^2), а таблицы значений функции удобно хранить в двумерных текстурах.

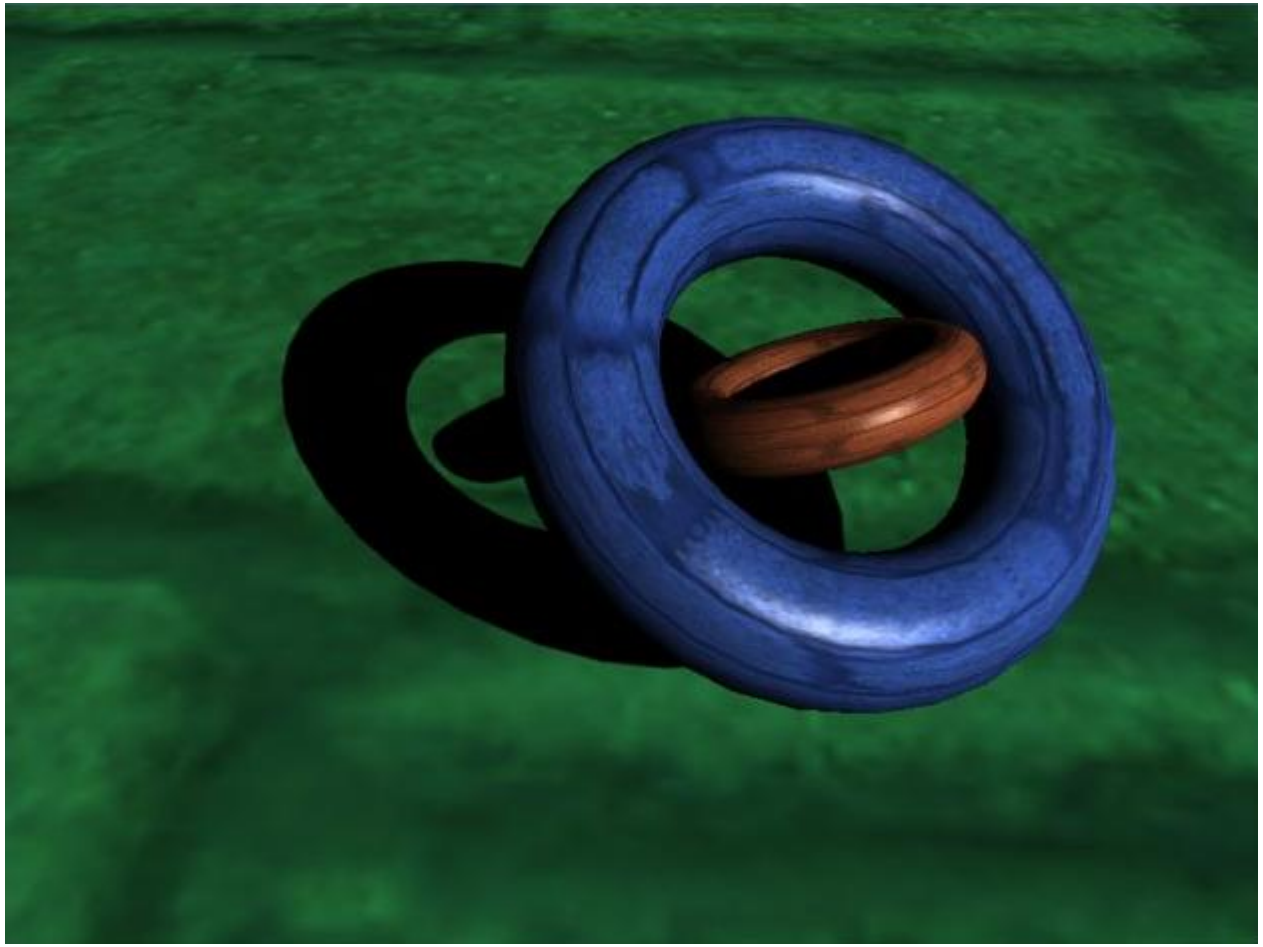
В любом случае, реализация теней для точечных источников с помощью этого алгоритма слишком ресурсоемка: требуется 5 лишних render-to-texture.

Теневые буферы

+ Алгоритм не зависит от сложности геометрии сцены
+ Поддерживает самозатенение

– Алгоритм подвержен aliasing’y
– Для каждого источника света требуется отдельный рендеринг сцены
– Плохо работает для точечных источников
– Требует аппаратной поддержки или много дополнительных проходов рендеринга

Построение теней при помощи теневых буферов



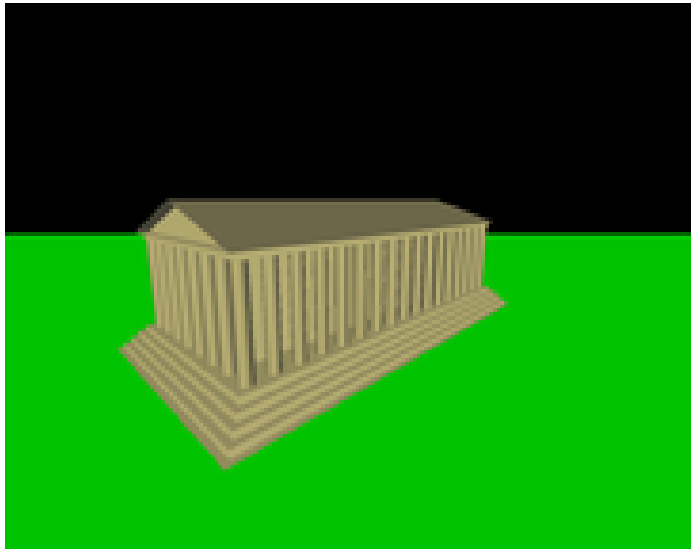
От идей к реализации

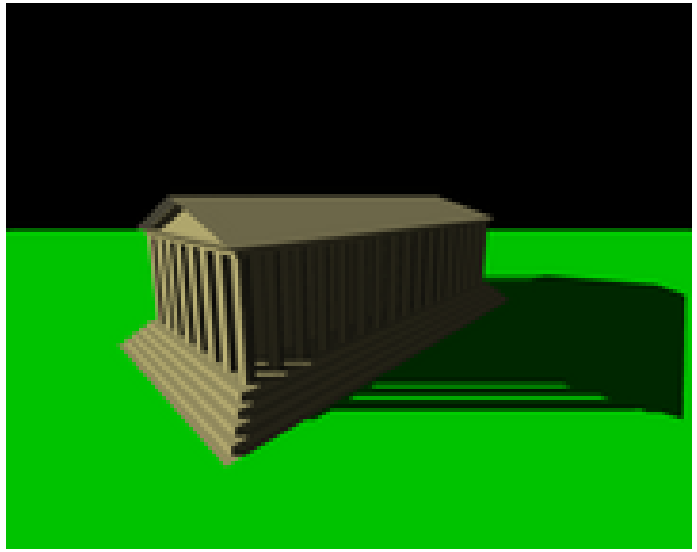
- В основе этого метода лежит крайне простая идея, что освещенные фрагменты — это те фрагменты, которые видны из положения источника света.
- Поэтому, если мы расположим камеру в положении источника света и отрендерим сцену при помощи этой камеры, то мы получим все освещённые фрагменты.
- Видимость фрагмента источнику света в терминах буфера глубины означает, что данный фрагмент успешно проходит тест глубины при рендеринге из положения источника света.

От буфера глубины к текстуре

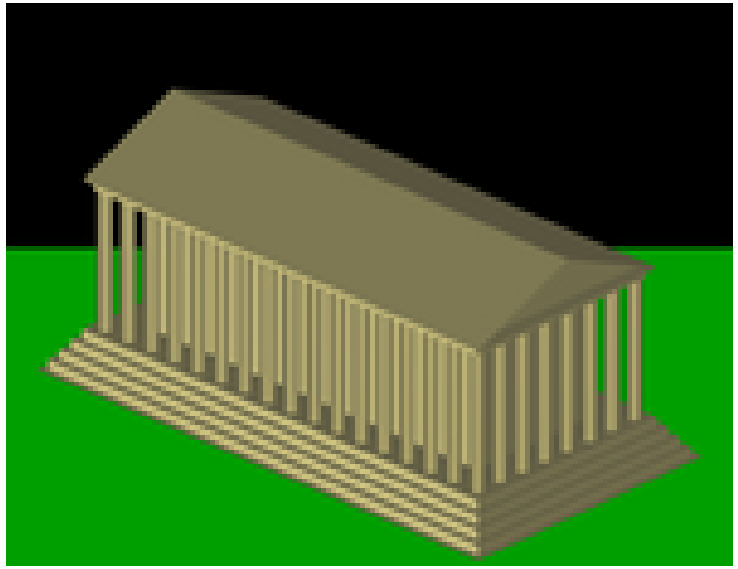
- Буфер глубины поместить в специальную текстуру, которая будет использоваться для проверки видимости фрагментов из положения источника света
- Вычисление текстурных координат (s,t,r), при котором (s,t) определяют проекцию соответствующего фрагмента на картинную плоскость при рендеринге из положения источника света, а третья компонента r является глубиной фрагмента относительно источника света
- При этом текстурные координаты (s,t,r) фактически являются координатами фрагмента в системе координат камеры, использованной для рендеринга из положения источника света
- Тогда для определения видимости источником света данного фрагмента достаточно просто сравнить значение глубины depthMap(s,t) и r

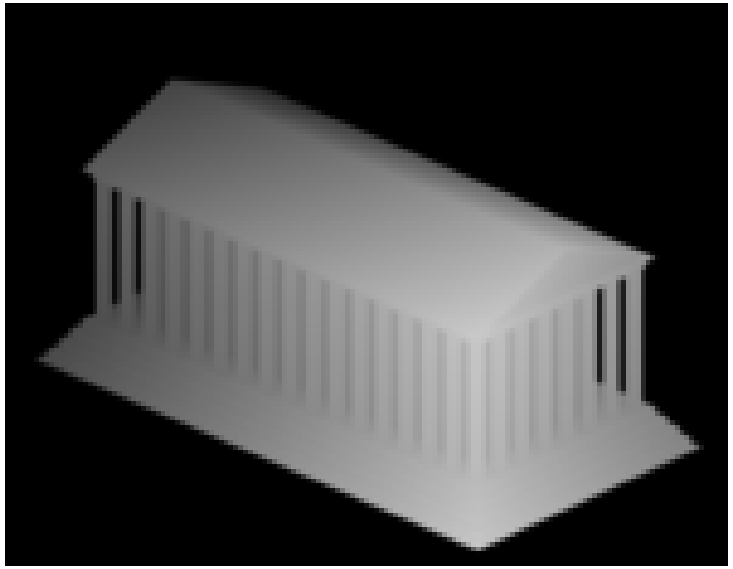
Эффект наложения тени



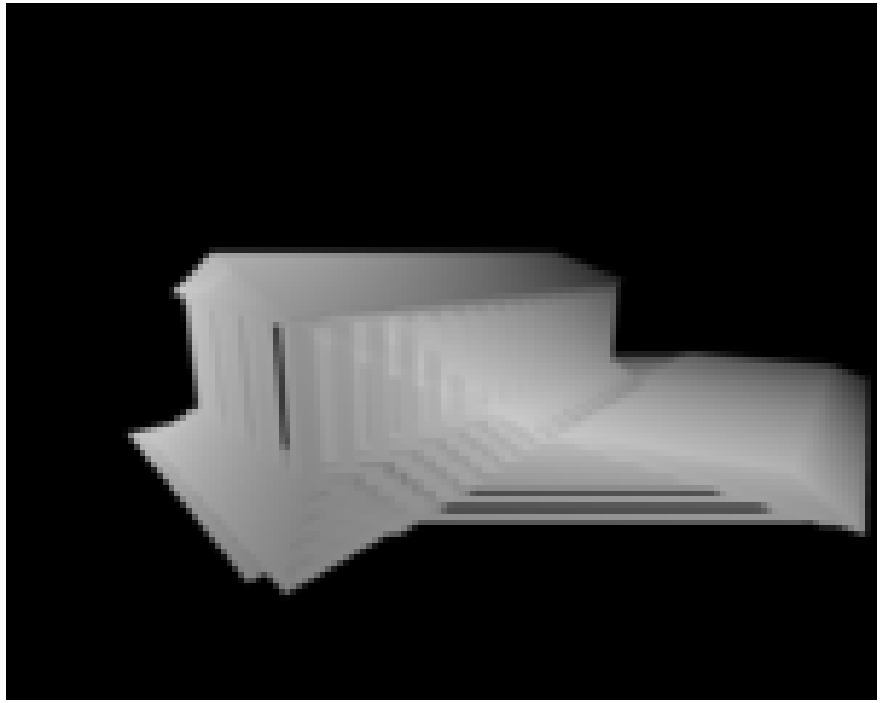


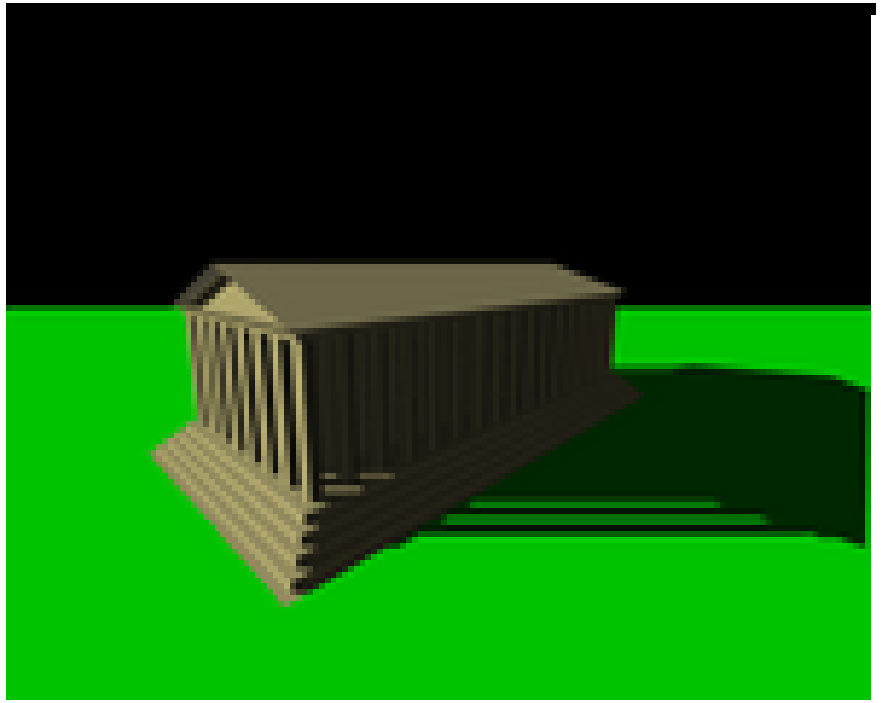
Создание теневого буфера





Наложение теневого буфера





Что нужно?

- Построить две матрицы:
  - матрицу вида (view matrix)
  - матрицу проекции (projection matrix)
- Для направленного источника освещения используется ортогональная матрица проекции
- Для точечного источника освещения используется перспективная матрица проекции
- На каждый источник освещения необходим дополнительный проход рендера сцены

Алгоритм

Запрещается запись в буфер кадра, нужно только содержимое буфера глубины

В положении источника света размещается виртуальная камера и производится рендеринг всей сцены при помощи этой камеры

Данные из буфера глубины копируются в специальную текстуру, называемую теневым буфером

Проводится обычный рендеринг сцены из положения наблюдателя

Для каждого выводимого фрагмента вычисляются специальным образом текстурные координаты (s,t,r) для доступа к теневой карте

Этапы реализации

- Создать текстуру для хранения буфера глубины
- Создать Framebuffer Object (FBO) и привязать к нему текстуру
- Настроить камеру и выполнить рендер сцены в созданный FBO

Требования

- Специальный формат текстуры для хранения карты глубины (обладающий достаточной точностью)
- Быстрый способ вычисления depthMap(s,t)<=r

Формат текстуры

//создание текстуры под depth данные

glTexImage2D(GL_TEXTURE_2D, 0, **GL_DEPTH_COMPONENT**, width, height, 0, GL_DEPTH_COMPONENT,
GL_FLOAT,NULL);

Быстрый способ вычисления

// Установить новый режим текстурирования для использования **depthMap(s,t)<=r**

// pname GL_TEXTURE_COMPARE_MODE
// param GL_COMPARE_REF_TO_TEXTURE

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_MODE, **GL_COMPARE_REF_TO_TEXTURE**);

// Вернуться к нормальному режиму текстурирования

glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_COMPARE_MODE, **GL_NONE**);

Методы рендеринга теней

- Карты освещения
- Вершинные тени
- Проектируемая геометрия
- Теневые объемы
- Теневые карты
- Теневые буферы
- **Буферы ObjectID / приоритетов**

Буферы ObjectID/приоритетов

Буферы ObjectID или приоритетов — ещё один перспективный sample-based метод генерации теней в реальном времени.

Аппаратно буферы приоритетов реализованы на чипах ATI, начиная с R100.

Этот алгоритм похож на алгоритм теневых буферов.

Буферы ObjectID/приоритетов: основные принципы

Перед рендерингом сцены из источника света каждому объекту присваивается уникальный номер (ObjectID), который зависит от расстояния от объекта до источника света.

Под объектом здесь понимается геометрия, которая отбрасывает тени на другие объекты, но не на себя. То есть объектом может быть единичный полигон, выпуклый объект или его часть.

При рендеринге сцены из источника света, в текстуру записываются приоритеты объектов. В результате буфер содержит номера ближайших к источнику света объектов.

Во время основного рендеринга осуществляется сравнение, похожее на сравнение в алгоритме теневых буферов. Приоритет текущего объекта сравнивается с соответствующим значением в буфере, и если значения совпадают, то пиксель освещен, иначе - существует объект, расположенный ближе к источнику света, и пиксель затенен.

Буферы ObjectID/приоритетов

Буферы приоритетов работают с точечными источниками лучше, чем теневые буферы, так как сравниваются не значения глубины пикселей, а их приоритеты.

Недостатки алгоритма: невыпуклые объекты не самозатеняются; как и у теневых буферов, существуют проблемы с aliasing'ом. В результате получаются четкие, ступенчатые тени.

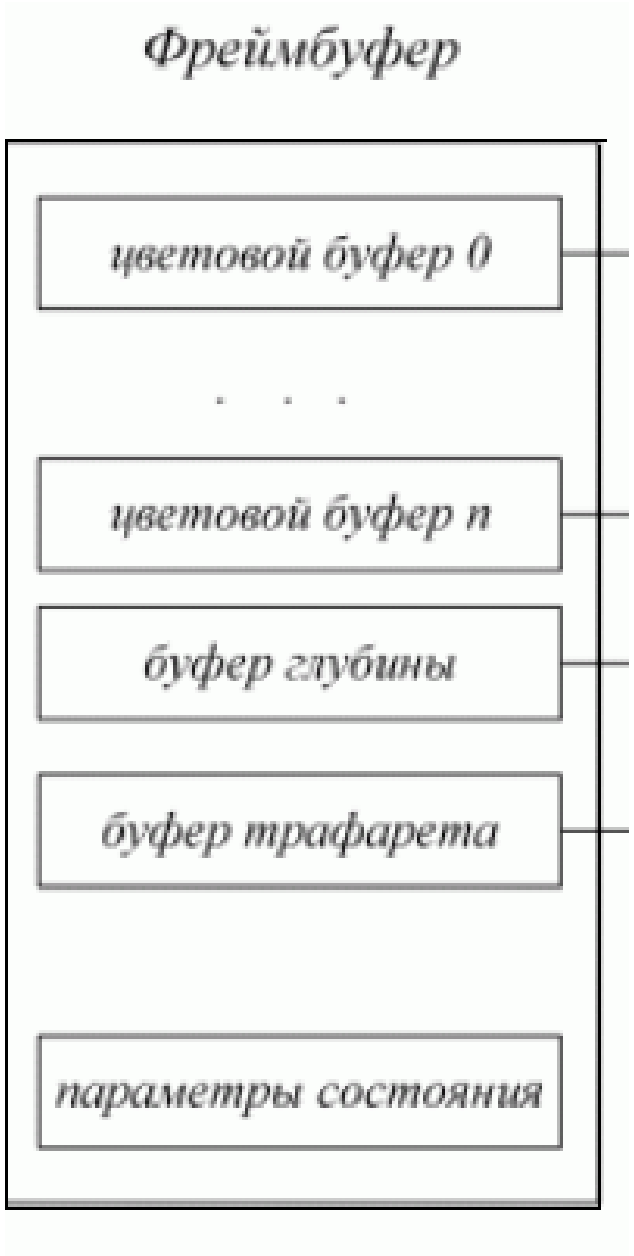
Типы экранных буферов

- Буфер **цвета**, в котором хранятся значения цвета фрагментов
- Буфер **глубины**, хранящий информацию о глубине фрагментов
- Буфер **трафарета**, позволяющий отбросить часть фрагментов согласно определенному условию.

Комбинация этих трех буферов зовется **кадровым буфером** (фреймбуфером) и хранится в определенной области памяти.

Фреймбуфер, сам по себе это только точка привязки, к нему привязываются так называемые рендер-текстуры.

Рендер-текстуры уже будут хранить непосредственно цветовую информацию.



Кадровый буфер

- Все операции отрисовки, что мы выполняли до сих пор, исполнялись в рамках буферов, прикрепленных к базовому кадровому буферу.
- Базовый буфер кадра создается и настраивается в момент создания окна приложения.
- Создавая собственный кадровый буфер мы получаем дополнительное пространство куда можно направить рендер.

Применение собственных кадровых буферов

- Тени (теневой буфер)
- Screen Space Ambient Occlusion (SSAO) – эффект самозатенения
- Эффекты зеркал
- Screen Space Reflections (SSR) – алгоритм который позволяет получить эффект отражения для плоских поверхностей
- Программная генерация текстур
- Постобработка
- Deferred Shading – отложенное освещение и затенение
- Depth of Field (DoF) – глубина резкости
- High Dynamic Range Compression (HDR) – технология расширения диапазона яркости кадра
- Bloom – своеобразный эффект свечения, заключается в размытии границ ярких объектов.
- Screen Space Subsurface Scattering (SSSS) – подповерхностное рассеивание