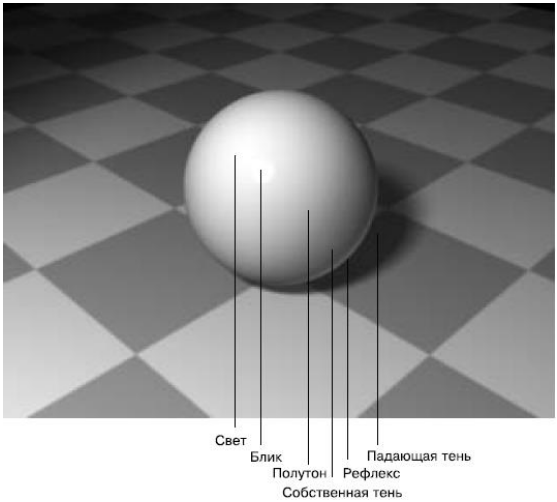
Лекция 3.2. Генерация тени

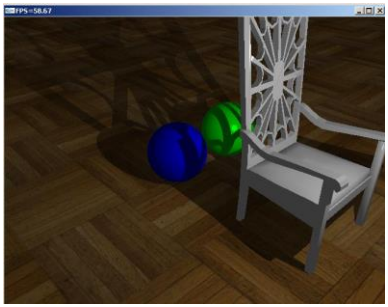
Быстрое и в то же время точное, физически корректное вычисление тени при генерации искусственных изображений по сей день является сложной и до конца не решенной задачей. На данный момент существует большой перечень алгоритмов, способных с той или иной степенью успешностью решить ее, однако тот факт, что все они, прежде всего, ориентированы на качество и игнорируют такие аспекты синтеза изображений, как скорость вычислений и легкость реализации, делает их практически непригодными для нужд графики реального времени.

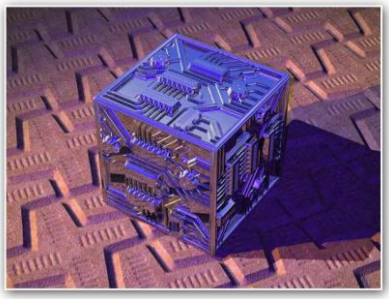
Перечислим три основные требования, которым должны удовлетворять алгоритмы генерации тени

Во-первых, генерируемая нами тень должна быть физически корректна, ведь задачей синтеза изображений всегда являлось имитация свойств реального мира. Однако подчас выполнение данного требования влечет за собой неприемлемые накладные расходы, что выражается в снижении общей производительности, вследствие чего порой нам придется пойти на преднамеренное упрощение физической модели.



Во-вторых, процесс генерации тени затрагивает большое число аспектов взаимодействия множества объектов обрабатываемой сцены, будь то объекты или источники света. Подобное предположение накладывает на алгоритмы требование универсальности, и единственной причиной тому будет являться тот факт, что изначально мы попросту не можем иметь какого- либо представления о конечной конфигурации сцены.





В-третьих, рассматриваемые нами алгоритмы по возможности должны быть поддерживаемы сегодняшними графическими системами, то есть поддерживаться аппаратно. Подобное ограничение находит свое выражение в проблеме следующего характера: графический конвейер, рассчитанный на параллельную обработку множества однородных объектов, будь то вершины полигональной модели или пиксели изображения, попросту не приспособлен анализировать множественные связи, возникающие между объектами в процессе генерации тени. Так или иначе, выполнение данного требования было бы весьма для нас желательным по той простой причине, что перенесение части вычислений на плечи эффективного графического конвейера без сомнения привело бы к повышению производительности.

Первые графические работы были весьма примитивны по сегодняшним меркам и оперировали сеточными примитивами, по большей части, в контексте моделирования тех или иных объектов. В 60-х годах 20-го века появились первые алгоритмы для отображения

твердотельных объектов в условиях освещенности. Данные алгоритмы были так же весьма просты. Они основывались на простейших моделях освещения и не учитывали всей специфики взаимодействия света со множеством объектов. Однако в конце 60-х годов Артуром Аппелем была представлена работа, впервые предлагавшая отображать твердотелые объекты в условиях их множественного взаимодействия со светом, что приводило к появлению теней на финальном изображении. Несмотря на то, что представленный им алгоритм был весьма груб и низко производителен, его работа открыла путь целому семейству новых более совершенных алгоритмов.

В те далекие времена генерация изображения могла занимать достаточно продолжительное время, начиная от нескольких минут, и кончая несколькими днями даже для весьма тривиальных сцен. Однако, с ростом производительности графических систем, появлением систем визуализации реального времени, в настоящий момент стало возможным синтезировать изображения на лету, что привело к выделению двух больших семейств алгоритмов генерации тени.

Первое семейство алгоритмов — это так называемые **алгоритмы препроцессинга**. Основным их достоинством является реалистичность. Данные алгоритмы по своей природе способны учитывать огромнейшее количество всевозможных аспектов взаимодействия света со сценой. Однако, требуя высоких производительных затрат, данные алгоритмы оказываются полностью неспособны сгенерировать тень «на лету», то есть в реальном времени.

Второе семейство алгоритмов генерации тени, получивших распространение с появлением систем аппаратной визуализации - **алгоритмы реального времени**. Основным их достоинством является скорость и относительная физическая корректность. Не налагая больших требований на производственные мощности, они позволяют генерировать финальное изображение за доли секунды, в то же время, сильно снижая его качество. Одним из неприятных последствий использования подобных алгоритмов является целое семейство графических артефактов, заметных невооруженным взглядом и порой

сильно влияющих на впечатление, создаваемое финальным изображением. Исследования, позволившие бы избавиться от столь неприятных побочных эффектов, проводятся по сей день.

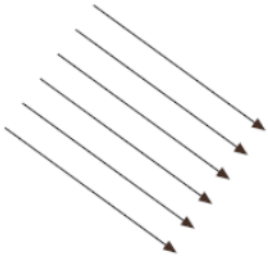
Несмотря на четкое разделение между вышеперечисленными подходами к вопросу о генерации тени, основной задачей на сегодняшний день является поиск компромисса, позволившего бы совместить реалистичность методов препроцессинга и реального времени. К счастью, много шагов в этом направлении уже было предпринято разработчиками индустрии видеоигр, по иронии судьбы, являющимися на сегодняшний день двигателями прогресса в этой области.

Источники света

Любая тень по сути своей является результатом взаимодействия одного или более объектов со световыми лучами. Следовательно, любая **тень** является вторичным продуктом, когда сам свет является первичным, ведь именно свет приводит к появлению тени, а не наоборот. Принимая этот факт во внимание, изучение вопросов генерации тени следовало бы начать с рассмотрения природы света, а еще лучше - с его источников.

Источники света являются неотъемлемой частью процесса синтеза изображения, так как именно они являются в реальном мире производителями световой энергии, благодаря которой мы вообще способны видеть. И несмотря на то, что в природе любой источник света обладает весьма сложной структурой, в прикладных задачах синтеза изображений зачастую можно позволить себе некоторое упрощение модели в целях повышения общей производительности.

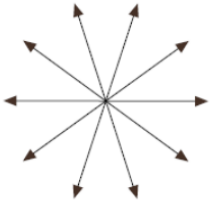
Первым из наиболее часто используемых источников света является **свет *направленный.*** Как показано на рисунке все лучи, испускаемые данным источником математически параллельны друг другу. За такой источник обычно принимается **математическая точка, удаленная в бесконечность**, и единственными его характеристиками являются его направление и интенсивность. Данный вид источника наилучшим образом подходит для моделирования солнечного света.



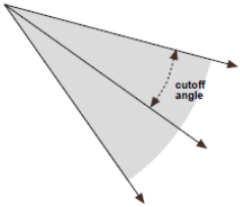
а. направленный

Для моделирования источника света, располагающегося на более близкой позиции к некоторому объекту, часто используют ***точечный источник***. При этом полагается, что свет излучается радиально во всех направлениях с одной и той же интенсивностью, а лучи зарождаются в некоторой математической точке, обладающей некоторыми координатами. Пример такого источника изображен на рисунке. **Основными его характеристиками являются точка приложения (координаты источника) и его интенсивность.** Так же для придания освещению реалистичности часто вводят дополнительную характеристику для точечных источников - **фактор затухания**. Основная причина для этого заключается в том, что природные источники света, которые обычно имитируются данным видом источников, обладают сравнительно небольшой силой и интенсивность, в следствии чего, затухают на расстоянии (в отличие от солнечного света, обладающего очень большой интенсивностью).

Данная модель источников так же может быть дополнена, если в качестве ее характеристик ввести еще две: **основное направление и максимальный угол.** Результирующий источник мог бы имитировать такой природный источник света, как фонарик или **прожектор**, пример которого изображен на рисунке.

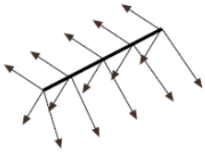


б. точечный

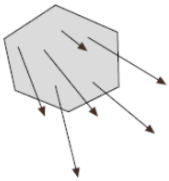


в. прожектор

Вышеперечисленные источники обычно классифицируют как **простые**, в то время как в реальном мире зачастую приходится иметь дело с более **сложными источниками**, которые в дальнейшем будем называть **расширенными**. Основное отличие между двумя этими классами источников заключается в том, какую именно тень они способны генерировать. В то время как простые источники обычно способны выделять лишь **области тени**, расширенные источники так же способны выделять области, известные как п**олутень**.



г. линейный



д. площадной

е. объемный

Первым из расширенных источников является ***линейный***. В реальном мире такому источнику мог бы соответствовать **свет галогенной лампы**. По сути своей данный вид источников света излучает свет радиально и равномерно во всех направлениях из каждой точки некоторого отрезка. Основная сложность, связанная с реализацией заключена в том, что количество математических точек, излучающих свет, вообще говоря, не ограничено, что следует из математического взгляда на отрезок.

Второй из расширенных источников света - ***площадной***. Площадным источником света может быть признан любой из

источников, обладающих некоторой площадью. В реальном мире примером может являться **площадь окна, рассеивающая свет по комнате.**

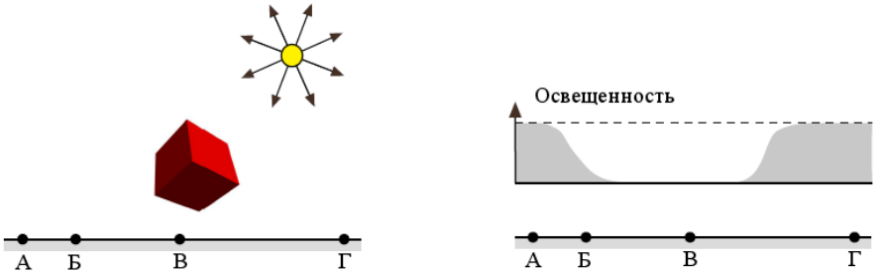
И последний вид расширенных источников - ***объемный***. Как следует из названия, данные источники света характеризуются целым объемом математических точек, способных радиально излучать свет. В реальном мире, данный источник света может **являться более приближенным к реальности точечным источником**, так как последний предполагает лишь наличие некоторой математической точки, что в нашем мире не совсем корректно.

Все вышеперечисленные источники света **характеризуются равномерностью распространения лучей за исключением прожектора**. В то же время, достаточно часто в интерактивных приложениях и видеоиграх **используется вариация площадного источника, при которой интенсивность излучения может варьироваться по некоторой поверхности.** Обычно подобная вариация задается при помощи текстуры в градациях серого, каждый тексель которой тем ярче, чем ярче светится соответствующая ему точка поверхности. При помощи такого подхода достаточно просто можно имитировать разнообразного рода прожекторы, лампы и фонари, имеющие дефект проникновения света на своей поверхности, либо объекты, частично экранирующие светящуюся поверхность, но при этом являющиеся частью самого источника.

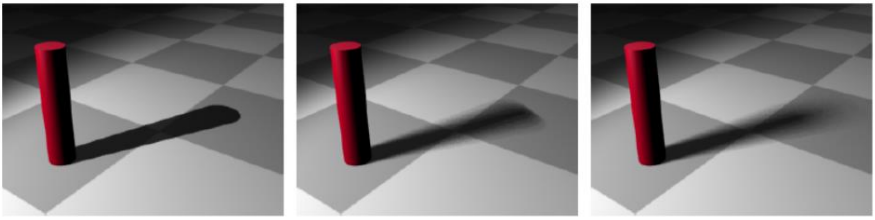
В свете вышеописанного логично появление следующего вопроса: для каких нужд стоило бы выделять расширенные источники света, если достаточное количество эффектов освещенности можно было бы достигнуть с применением только простых источников. Ответ на этот вопрос кроется в том, насколько разным способом будет генерироваться тень для первых и для вторых источников. Для этого рассмотрим пример из реальной жизни.

На рисунке изображен некоторый объект, находящийся в условиях освещенности солнечными лучами. Из реальной жизни мы помним, что кроме прямого хода, световые лучи могут создавать эффекты рассеивания и дифракции, а так же тот факт, что источник света может

обладать площадью. **Попытаемся определить освещенности четырех точек на некоторой поверхности в условиях возможного экранирования света объектом.** При этом, мы придем к заключению, что точка А будет освещена всеми лучами, потенциально достигающими ее, часть лучей в точке Б будет экранирована объектом (помним о рассеивании и площади), все лучи в точке В будут экранированы, а в точку Г все лучи вновь дойдут, как и в случае точки А. Теперь, если составить более подробную картину освещенности на интересующем нас отрезке, появится гладкая кривая. Как не трудно заметить, в контексте данной поверхности, освещенность не является логической функцией и наряду с наличием *тени,* присутствует и ***полутень***.



Имея некоторую математическую точку, излучающую свет, мы имеем лишь единственный путь для нее, благодаря которому она может достичь конкретной точки некоторой поверхности. Данный тезис, разумеется, будет иметь место только в том
случае, если из нашего уравнения будет полностью вычеркнуто рассеивание. Однако **что будет в том случае, если точек, излучающих свет, будет больше, чем одна**? Простые и расширенные источники генерируют разные тени, и примером того может быть рисунок ниже.



точечный линейный площадной

Простейшее объяснение подобного эффекта может быть найдено в геометрической природе экранирования, которое представлено на рисунке. **Здесь рассматривается тень, полученная от двух источников: точечного и линейного.** Так как во втором случае мы имеем дело с более сложной моделью распространения света, то и для реализации тени при такой модели, необходим и более сложный алгоритм.

Классификация алгоритмов

Object-space и Image-space

С точки зрения аппаратного ускорения визуализации алгоритмы генерации тени **классифицируются по пространству**, в котором они оперируют. Один из классов алгоритмов, называемых **object-space алгоритмами, оперирует в пространстве геометрии.** При этом принимается во внимание геометрическая ценность информации об экранировании участков поверхности сцены. Так как эти алгоритмы требуют принятия во внимания множественного взаимодействия между множеством объектов, представляется крайне затруднительным

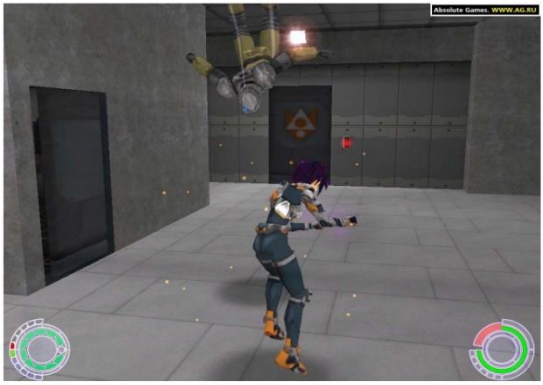
или, как минимум, неочевидным аппаратное решение данной проблемы. И хотя подобное решение все же возможно (доказательством чему является целое семейство по сей день существующих в реальном времени алгоритмов), само по себе оно будет допускать ряд артефактов, а также и ряд ограничений. Причиной тому является специфичность существующего на настоящий момент графического конвейера, ориентированного на обработку независимых друг от друга объектов, а не взаимодействия между ними.

Вторым классом алгоритмов являются так называемые **image-space** алгоритмы, оперирующие **в пространстве изображений**. При использовании алгоритмов данного семейства, информация об экранировании помещается в некоторое изображение (текстуру), данные из которой считываются графическим конвейером. Так как последний был весьма неплохо приспособлен к работе с изображениями, данное семейство алгоритмов наилучшим образом подходит для **аппаратной реализации**. Однако стоит оговориться, что практически любой алгоритм, ориентированный на изображения будет в той или иной степени страдать от дискретности, возникающей из-за дискретной природы растра, что в свою очередь может приводить к разнообразным артефактам.

Существует так же и **третье семейство алгоритмов, когда результирующая тень заменяется достаточно простым геометрическим объектом, некоторым образом взаимодействующим со сценой.** Примером тому может являться персонаж, расположенный над некоторой плоской поверхностью, тень от которого представляется в виде размытого круга. К таким приемам часто прибегали в старых видеоиграх, когда полный расчет экранирования света был попросту не возможен по причине крайне низкой производительности.

Примером такого подхода является изображение на рисунке, на котором тени от двух персонажей в видеоигре Oni заменяются такими размытыми кругами. Несмотря на то, что данный способ весьма эффективен, не трудно заметить, что он вряд ли отражает истинную

природу тени. В последствие алгоритмы подобного класса мы будем называть *обманками.*

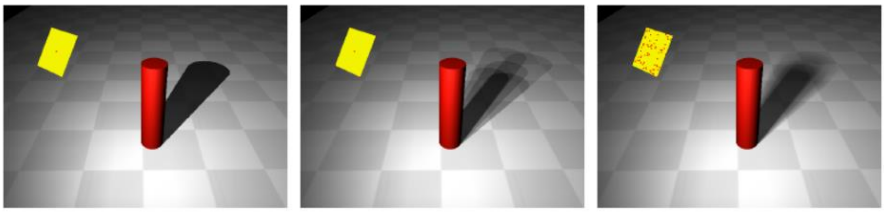


Четкие и мягкие тени

Так же немаловажным аспектом является **метод генерации тени, а точнее способ геометрического ее расчета от источника.** Как уже было замечено, расширенные источники света создают весьма сложную теневую картину, в то время, когда простые источники создают простые теневые объемы. С этой точки зрения конечным алгоритмам было бы весьма неплохо фокусироваться на типе светового источника, тем самым поднимая планку эффективности. Как известно, алгоритмы, ориентированные на простые источники, сравнительно легко справляются даже с очень сложными сценами за требуемые промежутки времени, в то время как расширенные источники до сих пор используются крайне редко по причине большой сложности реализации.

Вообще говоря, отделить алгоритмы, генерирующие мягкие тени, от алгоритмов, генерирующие четкие не настолько просто, как object-space от image-space. Ведь в теории, любой из расширенных источников может быть отмасштабирован таким образом, что его мягкая тень перейдет в четкую. При этом реализация расширенных источников уже задача не тривиальная.

Наиболее интересен обратный подход. Ведь, по сути, **мягкую тень расширенного источника можно представить как совокупность теней от множества простых источников**, то есть попросту с поверхности расширенного источника можно брать несколько семплов. На рисунке представлен такой подход. Здесь красной точкой на поверхности расширенного площадного источника берется некоторое количество семплов, каждый из которых генерирует свою тень. Не трудно заметить, что чем больше семплов мы берем, тем больше конечная тень походит на мягкую.



1 семпл 5 семплов 64 семпла

Аппаратная и программная генерация тени

Данный вид классификации так же является крайне важным для задач генерации тени. Как известно, **аппаратная реализация** графического конвейера накладывает достаточно много ограничений как на обрабатываемые данные, так и на алгоритмы, что не может не влиять на методы, используемы в вычислениях, связанных с экранированием. В то время, когда чисто **программные средства** могут извлечь полную выгоду от механизмов управления памятью, процедур и гибкости программирования на ЦПУ, аппаратно реализуемым алгоритмам приходится рассчитывать на крайне бедный набор доступных средств.

Современные **видеоадаптеры ориентированы** на локальную обработку данных, к примеру, вычисление цвета фрагмента, преобразование координат, семплирование текстуры, то есть обработку некоторого объекта в контексте его независимости от множества других объектов. Построение же тени есть более

глобальный процесс, и это противоречие делает разработку алгоритмов генерации тени трудно решаемой задачей.

С другой стороны, возможность **переложить некоторые методы на графический конвейер** дает возможность использовать эти методы в реальном времени, в то время, когда чисто **программные алгоритмы зачастую вынуждены существовать только в условиях препроцессинга.**

Алгоритмы генерации тени

Трассировка

Вероятно, самый простой с точки зрения разработки, но в то же время и один из самых сложных с точки зрения производительности алгоритмов. **Подобный подход логично вытекает из природы света и тени.** Ведь что такое тень по сути - ситуация, когда световой луч экранируется некоторым объектом.

Что есть, по сути, финальное изображение, как не набор фрагментов различных объектов, полученных в результате растеризации. Каждый фрагмент проходит через множество операций - тест на видимость, обработка моделью освещения и так далее. Каждый фрагмент обладает одной важной характеристикой - местоположением в пространстве, а проще говоря - координатами. При этом он будет освещен тогда и только тогда, когда хотя бы один из лучей, испущенных источником, света попадет на него. В свете вышесказанного, **затененность конкретного фрагмента определяется простым тестом:** из его координат проводится отрезок до источника света. Если на пути отрезка (трассе луча) возникнет некоторый объект, пересечение с которым можно определить геометрически, данный объект не получит света. В противном случае, если ни один из объектов сцены не воспрепятствует трассе, можно говорить, что данный фрагмент освещен, и применять к нему некоторую модель освещения.

Не смотря на всю простоту алгоритм трассировки лучейон обладает **двумя фундаментальными недостатками**. **Первый** из них заключается в том, что для определения освещенности конкретного фрагмента необходимо протестировать трассу на предмет пересечения

со множеством геометрических объектов. Фрагменты генерируются на этапе растеризации, которая может производиться аппаратно. С другой стороны, графический конвейер не дает возможности проводить для каждого фрагмента тесты трассы, так как это вовлекает операции с другими объектами, что в свою очередь нарушает принцип локальности графического конвейера. Другими словами, если мы захотим использовать трассировку для генерации тени, нам придется исполнять программно не только ее, но и растеризацию, что переносит данный алгоритм в область чисто программных и неприемлемых для нужд реального времени.

Второй яркий недостаток алгоритма заключен в том, что, отслеживая трассу некоторого фрагмента, мы можем придти только к двум возможным выводам: фрагмент либо экранирован, либо нет. С точки зрения природы света он будет либо освещен, либо в тени. Откуда же при данном подходе взяться полутени? Данный недостаток можно обойти методами множественного семплирования расширенных источников, но, если мы имеем дело с простыми источниками, решение становится нетривиальным.

Геометрический анализ

Подобный подход может являться второй попыткой решить задачу в лоб. Его суть заключается в том, что методами анализа производится **разбиение всех объектов сцены на затененные, полузатененные и освещенные области.** Несмотря на то, что конечным результатом может явиться действительно физически корректная картина экранирования, сложность получения подобной картины поддается какой-либо оценки с большим трудом, особенно в условиях существования динамических объектов, изменяющих состояние сцены на каждом кадре. Вдобавок, подобный подход ни коем образом не может быть реализован аппаратно, что классифицирует алгоритм в категорию чисто программных решений.

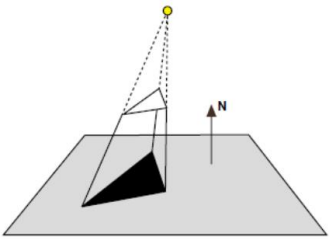
Проецируемая геометрия (Projective Geometry)

Зачастую вопросы генерации тени не всегда должны вовлекать сложный анализ взаимодействия множества объектов. Примером таких ситуаций могут являться видеоигры, в которых тень как таковая является скорее декоративным элементом, нежели жизненно-важным атрибутом. В следствии, часто можно сократить объем вычислений, связанных с генерации тени, сведя их лишь к специфическим случаям.

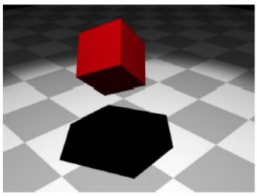
Ярким примером таких случаев является проецируемая геометрия, использующаяся только тогда, когда **есть некоторый геометрический объект, который должен отбрасывать тень на достаточно большую плоскую поверхность**. В видеоиграх, такие поверхности встречаются достаточно часто из-за ограничений на детализацию объектов. Обычно стены и пол, являющиеся получателями теней как раз представляются такими плоскостями, что позволяет сократить задачу генерации тени до создания теней только на такого рода поверхностях.

Задача сводится к тому, чтобы к уже существующему изображению растеризованной геометрии **добавить некоторый плоский объект, имеющий очертания тени**. По сути, эта тень будет являться той же полигональной сеткой, что и объект, ее порождающий, за тем исключением, что все его полигоны будут находиться в одной плоскости - плоскости приемника. То есть **необходимо попросту спроецировать все вершины объекта в одну плоскость**, что является одним из простейших геометрических преобразований, после чего нарисовать объект поверх приемника, используя некоторую

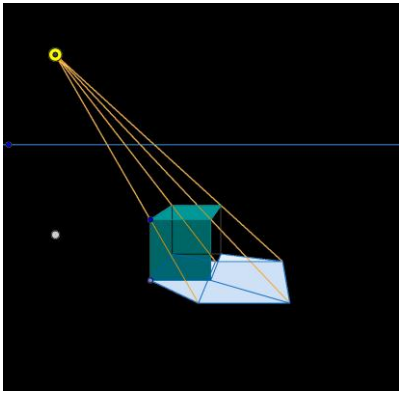
операцию, скажем, модуляции, которая заставит выглядеть поверхность темнее. Пример такого подхода продемонстрирован на следующем рисунке. Здесь клетчатая поверхность является приемником, куб - объектом, отбрасывающим тень, а силуэт тени - ничем иным, как тем же кубом, спроецированным на приемник с использованием операции модуляции.



проецирование полигона



результат



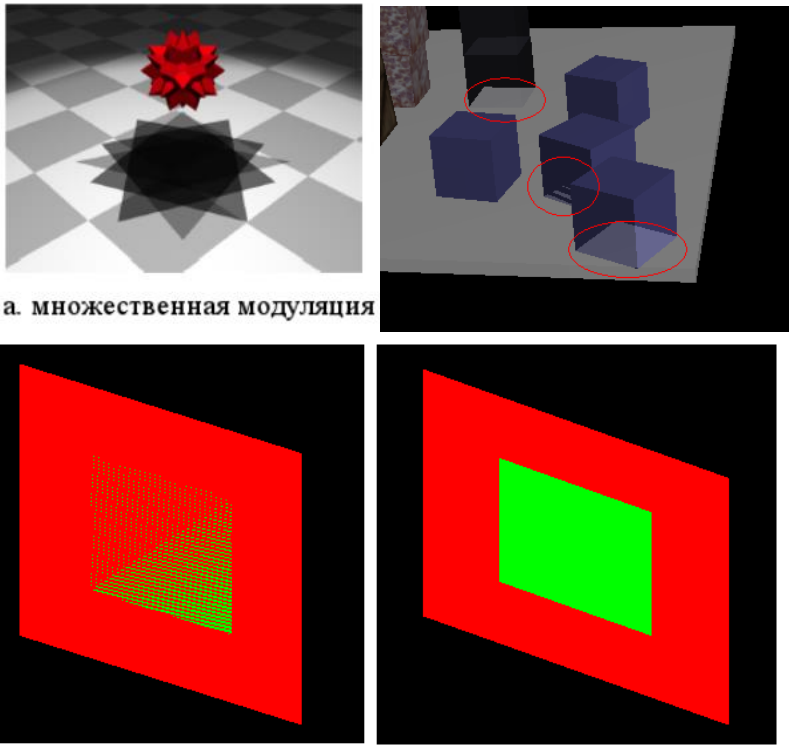
Однако, не смотря на всю простоту приведенного метода, он явно будет порождать ряд весьма неприятных артефактов.

Первый из них - это так называемый ***z-fighting*** - ситуация, возникающая при конфликте различных фрагментов за глубину. Возникает она тогда, когда два объекта находятся в одной плоскости, но в то же время из-за математической погрешности расчетов фрагменты объекта А и объекта Б, которые должны иметь одну и ту же глубину, будут иметь глубины разные. В качестве результата,

«осколки» тени могут выпадать из общей картины, что выглядеть будет весьма неприятно.

Второй артефакт - **множественная модуляция**. Для того чтобы понять суть проблемы, необходимо вспомнить, что конечная тень, как и объект ее порождающий является сложной структурой. В следствии, на различные фрагменты тени могут оказывать влияние различное количество полигонов.

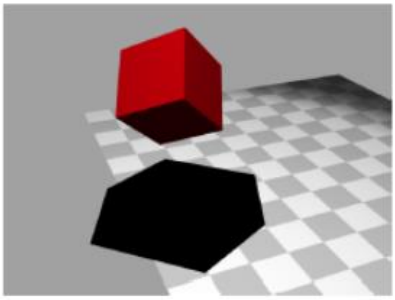
Однако при разном количестве полигонов, модулируемых на различные фрагменты, количество операций модуляции будет различным, а чем больше операций модуляции на фрагмент - тем он темнее. В результате возникает эффект неравномерной плотности тени, продемонстрированный на рисунке.



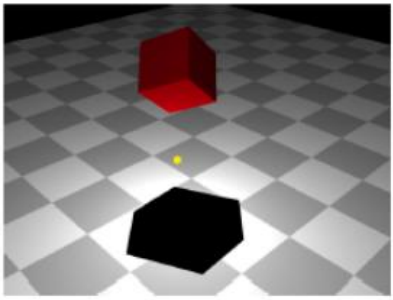
а. множественная модуляция

Третий из возможных артефактов - **выход за границы приемника**. В этом случае часть геометрии тени, спроецированная на виртуальную плоскость, может оказаться «нависающей над пустотой», что в реальной жизни, разумеется, не встретишь, так как тень, по сути, не является объектом. Разрешение подобной ситуации - геометрический анализ тени с последующим ее отсечением, что может являться задачей весьма трудоемкой. Пример ситуации выхода за границы приемника проиллюстрирован на рисунке.

Еще один возможный артефакт — это **ложная тень**. Не смотря на строгое наличие объекта, отбрасывающего тень и самого объекта, сам объект может находиться над источником света, следовательно не давать тени. Подобная ситуация разрешается простым анализом взаимного положения приемника, объекта и источника света. Однако решение в ситуации, когда часть объекта будет находиться над источником, а часть под ним - нетривиально и, вообще говоря, неочевидно. Пример подобного артефакта проиллюстрирован на рисунке.



б. выход за границы приемника



в. ложная тень

Так же не стоит забывать о еще одной проблеме - **данный алгоритм в классическом подходе не позволяет получать полутень**. И даже не смотря на то, что существуют решения, позволяющие в его более усложненной версии ее получать, все они работают скорее для частных случаев, нежели общих, да и в добавок, приводят к необходимости

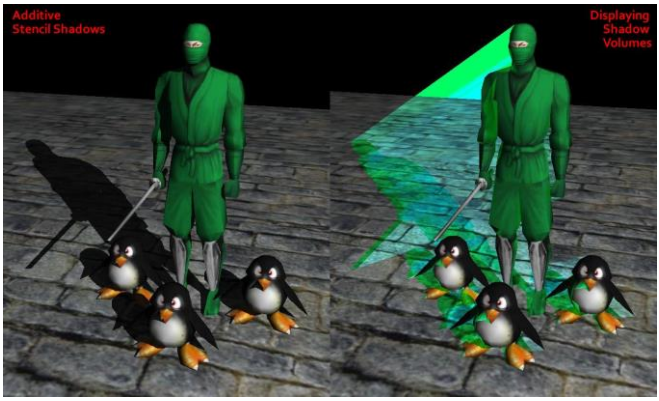
проведения сложного геометрического анализа, буквально являющимся «врагом» графики реального времени.

Алгоритм 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 приемников тени. С ростом количества приемников и количества/сложности объектов алгоритм быстро становится неприемлемым** из-за интенсивного использования процессора и большого количества необходимых проходов визуализации. В результате получаются четкие 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.

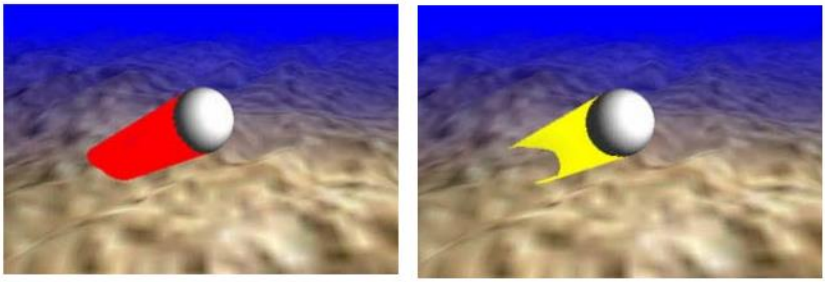
Теневые объемы (Shadow volumes)

Следующий рассматриваемый алгоритм основан на **представлении затененного пространства полигональным многогранником.** Многогранник генерируется на этапе препроцессинга путем проектирования каждого ребра минимального силуэта объекта вдаль от источника света.

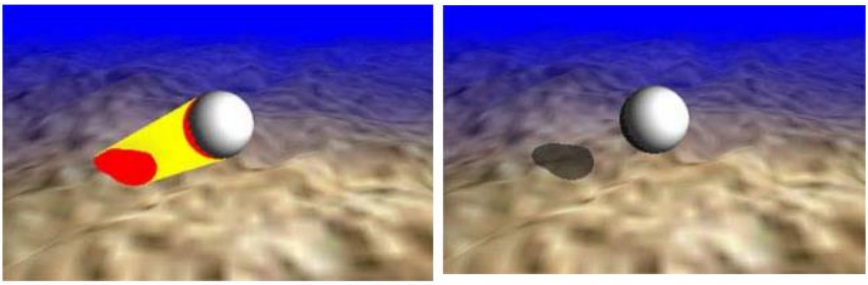
Алгоритм работает достаточно эффективно только с использованием современного оборудования, поддерживающего **z- и stencil-буферы**. До начала генерации теней сцена рисуется без учета источников света, генерирующих тени. После того, как изображение сцены готово, **задача распадается на две части**: сгенерировать теневые области в stencil-буфере и затем воссоздать по ним затенение сцены.



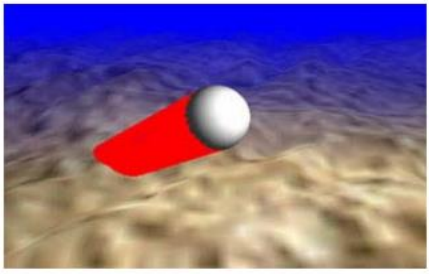
Stencil-буфер представляет из себя матрицу целочисленных значений, с размерами, соответствующими экранной области. Для каждого растеризуемого фрагмента геометрии можно ввести простые операции увеличения и уменьшения на единицу. То есть, если некоторый фрагмент проходит z-тест, он автоматически будет либо увеличивать, либо уменьшать значение в соответствующей ячейке stencil-буфера. Так как очертания тени на первом этапе остаются только в stencil-буфере, выводить на экран прошедшие тест фрагменты нет необходимости. Теперь все, что необходимо — это поочередно нарисовать сначала лицевые грани теневого объема, применяя к ним операцию увеличения значения в stencil-буфере, а затем - не лицевые грани, уменьшая значение. Область, сформированная ненулевыми значениями stencil-буфера, даст нам очертание тени.



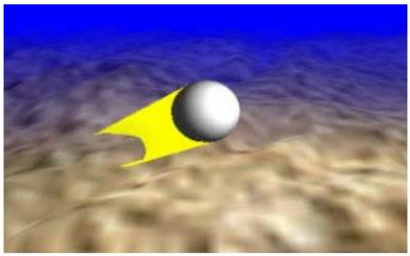
Область, сформированная ненулевыми значениями stencil-буфера, даст нам очертание тени. Теперь для того, чтобы вывести на экран теневую область, соответствующую очертанию, необходимо нарисовать поверх всего изображения большой серый четырехугольник с использованием операции модуляции и stencil-тестом. В данном случае под последним мы будем понимать следующее: на экран будут выведены лишь те фрагменты, соответствующее значение stencil-буфера которых больше нуля (будут соответствовать красным областям на рисунке. Результат подобного подхода проиллюстрирован на правом рисунке.



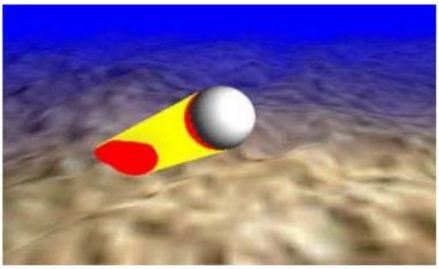
Проецирование лицевых граней. Красным цветом обозначен силуэт области в stencil-буфере, в которой значения будут увеличены на единицу.



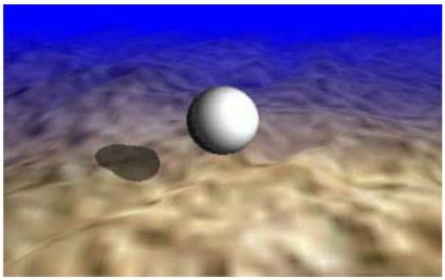
Проецирование не лицевых граней. Желтым цветом обозначен силуэт области в stencil-буфере, в которой значения будут уменьшены на единицу.



Вычисление очертания тени. Красным цветом обозначены ненулевые области в stencil-буфере.



Объемная тень



Для применения алгоритма на практике предстоит решить множество проблем. Так, нужно уточнить детали нахождения

минимального силуэта *occluder'a* (объекта, отбрасывающего тень), генерации теневого объема, построения самой тени. К тому же нужно корректно обрабатывать случай, когда камера находится внутри теневого объема или ближняя плоскость отсечения пересекает полигоны теневого объема.

Для поиска теневого объема предлагается следующее решение: для каждого объекта, который будет отбрасывать тени, до начала визуализации считаются данные о связности полигонов. Эти данные могут быть представлены как списки соседних полигонов для каждого треугольника. Когда эти данные доступны, приложение может легко найти список всех граничных ребер. Для каждого полигона объекта, который повернут к источнику света, необходимо проверить всех соседей. На границе с соседями, которые не повернуты к источнику света, будут находиться необходимые ребра. После получения списка ребер, принадлежащих теневому объему, можно генерировать полигоны объема. В самом простом случае нужно создать второй набор вершин в направлении *от* источника света на достаточно большом расстоянии и построить полигональную пирамиду по имеющимся данным (то есть каждое ребро трансформируется в два треугольника теневого объема).

Однако подобный алгоритм может порождать весьма неприятный артефакт. Дело в том, что весьма вероятен случай, когда камера находится внутри одного или нескольких теневых объемов. Не трудно заметить, что с такой ситуации мы лишаемся части передних граней, которые должны дать прирост значений в stencil-буфере. В этом случае общий алгоритм несколько изменяется: теперь stencil-буфер очищается не нулями, а числом, которое равняется количеству теневых объемов, в которых находится камера. Таким образом компенсируется отсечение передних полигонов теневых конусов ближней плоскостью проекции. Может случиться, что ближняя плоскость проекции пересекает полигоны теневого объема в пределах экрана. В этом случае необходимо применить методы геометрического анализа и модифицировать отсекаемую часть теневого объема так, чтобы она стала параллельна плоскости проекции.

К преимуществам этого алгоритма следует отнести намного большую по сравнению с *проектируемой геометрией* универсальность - производительность больше не зависит от сложности и количества приемников тени, тень может падать на сколь угодно сложную поверхность. Как и у всех аналитических алгоритмов, у *теневых объемов* нет проблем со ступенчатостью.

Однако, этот 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. Так же на сегодняшний день не существует его какой-либо версии, адаптированной для создания мягких теней.

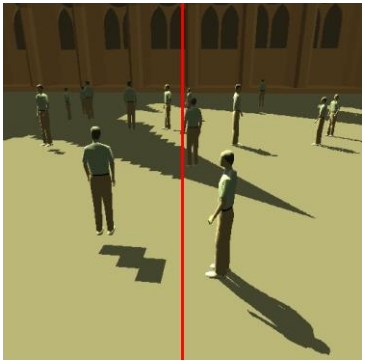
Теневые карты

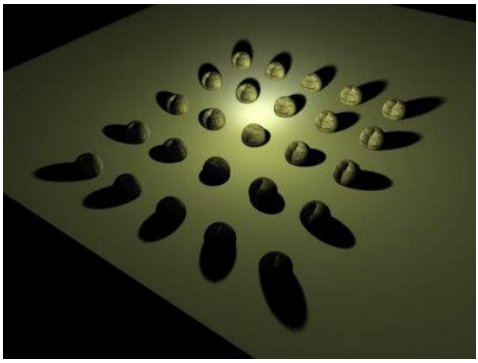
Рассмотрим теперь алгоритмы, **ориентированные на изображение**. Самый простой и первый из появившихся алгоритмов данного семейства - это алгоритм теневых карт, по сути являющийся некоторым компромиссом между использовавшимися на ранней стадией существования видеоигр обманками и проецируемой геометрией.

Этот метод использует текстуру для генерации карты затененности некоторой области, представленной виртуальным

четырехугольником. **Для каждой пары "источник света - объект" создается текстура.** Сначала текстура очищается белым цветом, потом на нее из точки положения источника света рисуется объект черным цветом. В результате получается теневая карта, которую необходимо спроектировать на все приемники теней.

Суть алгоритма проста: получить маску освещенности виртуального четырехугольника с точки зрения источника света, а затем наложить данный четырехугольник на все плоскости, на которых мы хотели бы видеть данную тень. Данный алгоритм может быть рассмотрен как потомок проецируемой геометрии. С одной стороны, он снижает количество полигонов, которые необходимо проецировать в плоскость приемника всего до двух треугольников, с другой стороны - позволяет добиться эффектов мягких теней. Последнего достичь очень просто. Так как мы имеем дело с текстурой (то есть некоторым изображением) мы можем попросту размыть растр и использовать конечную картинку.





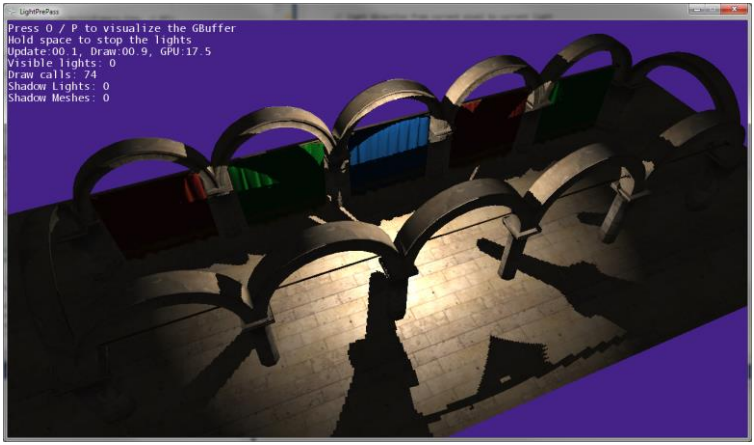
Преимущества: очень простой алгоритм, сравнительно быстрая генерация теней для небольшого количества объектов, необходима только одна дополнительная прорисовка для каждой пары. Этот алгоритм имеет проблемы со ступенчатостью, но допускает простое и дешевое их решение путем билинейной фильтрации теневой карты.

Недостатки: объекты не отбрасывают тени сами на себя - если рассматривать объекты как приемники 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.

Другой недостаток заключается в том, что для каждой пары "объект - источник света" необходима своя текстура, что ограничивает возможное количество таких пар. Так же не стоит забывать о проблемах, порожденных предком данного алгоритма - проецируемой геометрией, которые могут найти свое отражение в артефактах выхода за пределы приемника и ложных тенях.

Метод годится для генерации теней лишь для нескольких объектов сцены, и является хорошей альтернативой ранее рассмотренным алгоритмам.



Теневые буферы

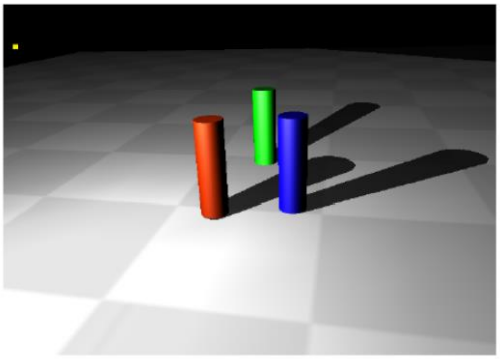
Алгоритм впервые был представлен Лансом Вильямсом в 1978 году. Потенциально метод теневых буферов - **самый универсальный метод построения теней в реальном времени**, хотя его реализация до недавнего времени была затруднена отсутствием необходимого оборудования. В настоящий момент, данный алгоритм занимает лидирующее положение при разработке видеоигр и единственной достойной ему альтернативой являются лишь теневые объемы.

Алгоритм базируется на том факте, что область, находящаяся в тени относительно источника света, не видна из точки источника света его положения. Для начала сцена рисуется из источника света во внеэкранный буфер. Нас интересуют только z-значения точек буфера. Картина глубин, сохраненная в z-буфере (теневом буфере), будет использоваться для теста на принадлежность точек тени. В буфере записаны глубины всех ближайших к источнику света точек. Любая точка, находящаяся дальше соответствующей точки в теневом буфере, будет в тени.

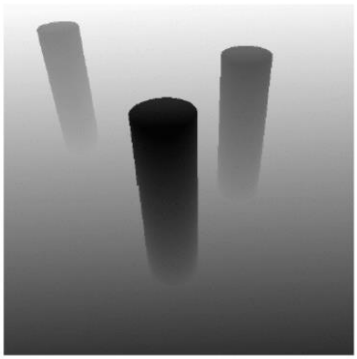
Во время основного рендеринга сцены каждый фрагмент, видимый из камеры, должен быть спроектирован на виртуальную плоскость теневого буфера. Если расстояние от точки до источника света равно (либо меньше) соответствующему z-значению буфера, то пиксель освещен. Если точка дальше, то пиксель в тени.

Не смотря на кажущуюся простоту и универсальность данного алгоритма, он **порождает ряд неприятных артефактов**, решение которых по сей день является задачей весьма неочевидной.

Первый из них - **ступенчатость (aliasing).** Возникновение данного артефакта связано с природой растра. По сути, каждый тексель теневого буфера будет являться частью своего рода маски. Если представить, что эту маску мы будем использовать для некоторой плоскости и сделать срез буфера этой плоскостью, то мы увидим, что фрагменты плоскости могут оказаться только в двух состояниях: затенены либо освещены. При недостаточном разрешении теневого буфера проявится дискретная структура маски.



конечное изображение

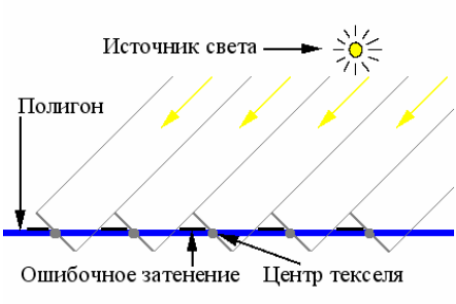


содержимое теневого буфера

Второй из таких артефактов - **перспективный алиасинг.** Он возникает по той причине, что математическая точка, чаще всего являющаяся источником света порождает конус видимости, то есть последовательность плоскостей проекции через этот конус будет с отдалением обладать все большей и большей площадью. При этом разрешение растра теневого буфера неизменно. Таким образом с удалением от источника света некоторого объекта исследования на него будет приходится все меньше и меньше текселей буфера, что в конце концов приведет к сильной ступенчатости тени для отдаленных объектов. Единственный выход - увеличивать математическое разрешение растра до тех пор, пока мы не придем к удовлетворительным результатам.

Третий артефакт и самый неприятный из всех - **алиасинг глубины**. Данный артефакт возникает в результате расхождения представления о глубине, записанной в теневой карте, со значением реальной глубины (расстоянием до источника света) освещаемой геометрии, что приводит к ошибочному затенению (или "незатенению").

Происхождение алиасинга глубины



Как видно на рисунке, правильное значение глубины будет только для пикселей полигона, на которые приходятся центры текселей теневой карты. В остальных ситуациях глубина будет либо больше положенного значения (ошибочное затенение), либо меньше.

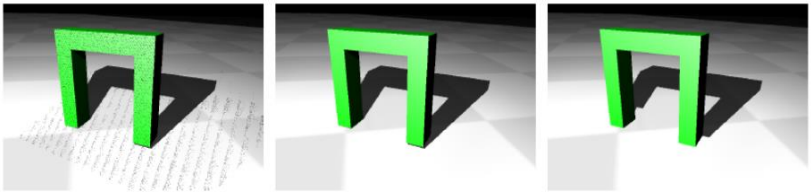
Один из эффективных способов борьбы с данной проблемой - применение наклонно-масштабируемого смещения глубины (slope-scale depth bias). То есть чем сильнее поверхность наклонена по отношению к источнику света, тем на большее значение будет увеличена глубина. К этому смещению добавляется небольшая константа для компенсации погрешностей при вычислениях и представлении данных.

Применение bias (сдвига) помогает уменьшить проблемы ошибочного самозатенения, но создает другие проблемы - ошибочное "незатенение", обычно выражающееся в том, что тень начинает отставать от объекта.

Wang и Molnar предложили метод "Second-depth shadow mapping", в котором не используется bias. Идея заключается в том, что в теневой текстуре сохраняется не ближайшая глубина, а следующая после ближайшей. На практике - это не лицевые грани объекта. Метод предъявляет очень жесткие требования к геометрии - она должна быть не просто замкнутой, но еще и выпуклой, в противном случае возможны артефакты. Также метод плохо работает для тонких объектов.

Другую идею предложил Woo - "midpoint shadow mapping", где предлагается брать глубину как среднее значение между ближайшим и вторым после ближайшего значением глубины. Позже на основе этих двух идей Weiskoph и Etrl предложили "dual shadow maps" (метод лучше работает для невыпуклых объектов). Недостатком данных техник являются высокие вычислительные затраты. По сути, для получения двух ближайших значений глубины необходимо визуализировать сцену два раза. Для этих методов также желательна замкнутость геометрии, в противном случае алгоритм усложняется. И опять же, есть проблемы с тонкими объектами.

Были и более экзотические предложения. Например, комбинирование проверки глубины с проверкой на равенство. Впрочем, и это позволяет лишь уменьшить артефакты, но не устранить их полностью.



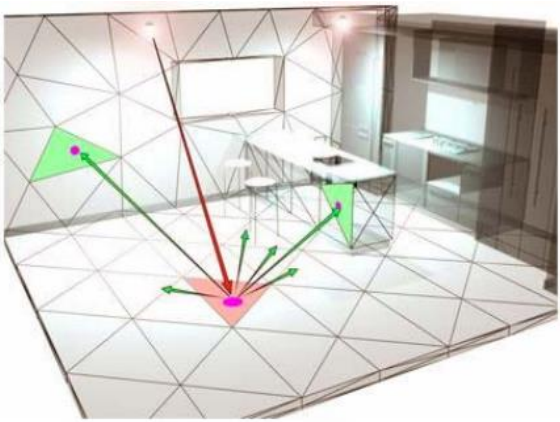
сдвиг слишком мал | то, что надо | слишком велик

Излучательность (Radiosity)

Один из методов глобального освещения (global illumination, GI).

Суть его состоит в том, что **все поверхности сцены разбиваются на небольшие фрагменты - патчи (patches)**, каждый из которых наделён свойствами излучать, поглощать и отражать свет.

Процесс вычисления освещения по алгоритму radiosity состоит из набора итераций, каждая из которых уточняет результат расчёта (radiosity solution). Для каждого патча на сцене подсчитывается полученная им от других патчей энергия, а также доля этой энергии, которая будет излучена патчем на следующей итерации. На первом этапе свет излучают только источники, на остальных - источники света игнорируются.



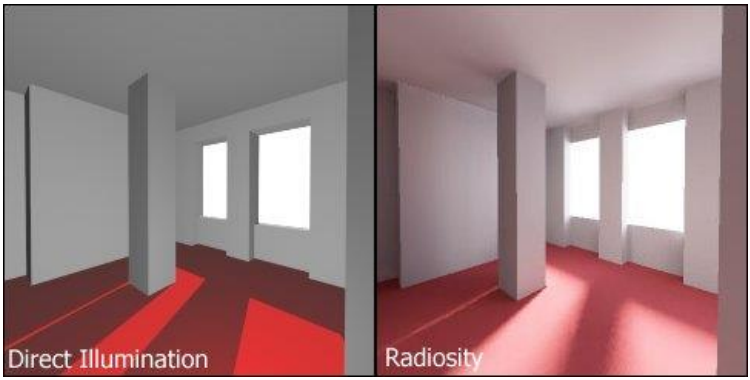
Луч света, попадая на полигон, отражается на другие

В результате алгоритм radiosity позволяет получать реалистичные эффекты вторичных отражений, неточечных источников света, мягких теней и т.д.

Изначально данный алгоритм является очень ресурсоёмким, пригодным только для предрассчитанного освещения (precalculated lightning). Однако существует ряд оптимизаций, позволяющих значительно снизить количество вычислений, и ускорить их (в частности, использование GPU для расчёта). Несмотря на это, **алгоритм относится к чисто программным.**

В современных видеоиграх данный алгоритм используется до сих пор на этапе препроцессинга. Патчи, принадлежащие одной плоскости группируются на некоторой текстуре, задающей глобальную освещенность всех полигонов сцены, а каждому из полигонов назначаются текстурные координаты на этой карте.

Эта текстура используется в дополнение к диффузным картам при визуализации статичной геометрии. Так как данный алгоритм позволяет создавать реалистичную картину распространения света в пространстве, его использование придает современным видеоиграм высокую степень реализма визуализации.


Direct Illumination
Radiosity


Current FPS: 27.0531
Average FPS: 29.99
Worst FPS: n/a
Triangle Count: 115766