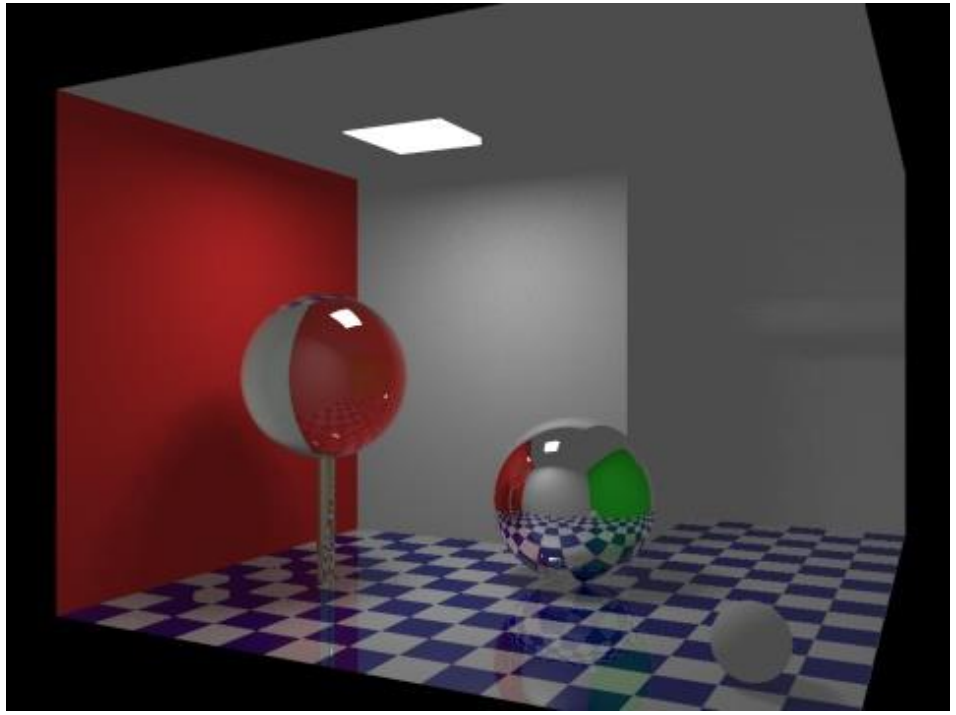
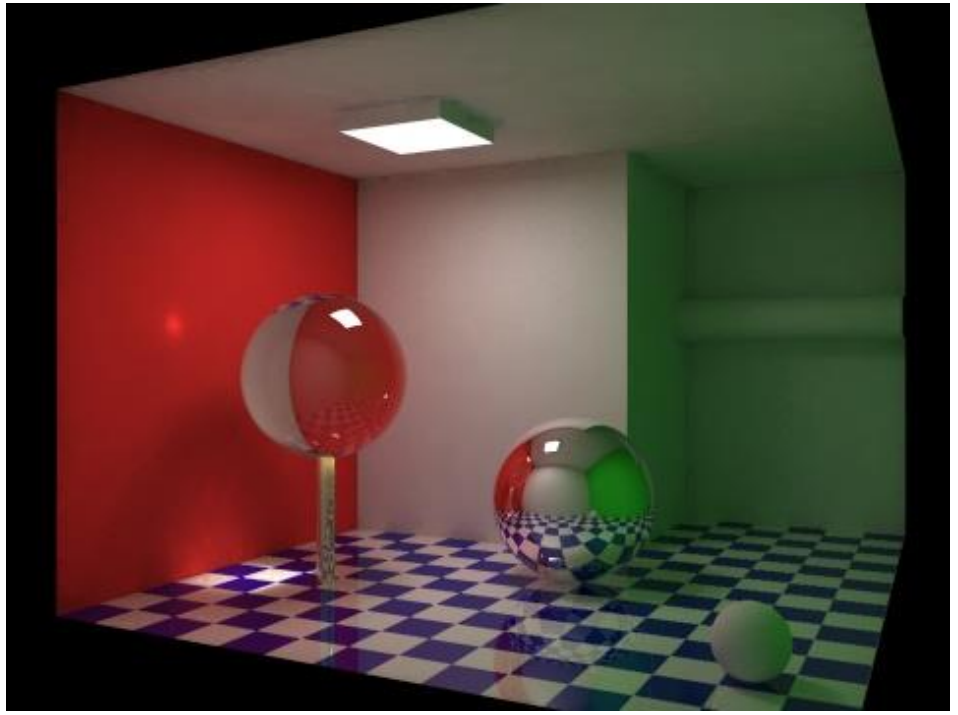
Реалистичный рендеринг (realistic rendering)

Компьютерная графика

Рендеринг. Локальное и глобальное освещение



Изображение, обработанное только алгоритмами локального (прямого) освещения



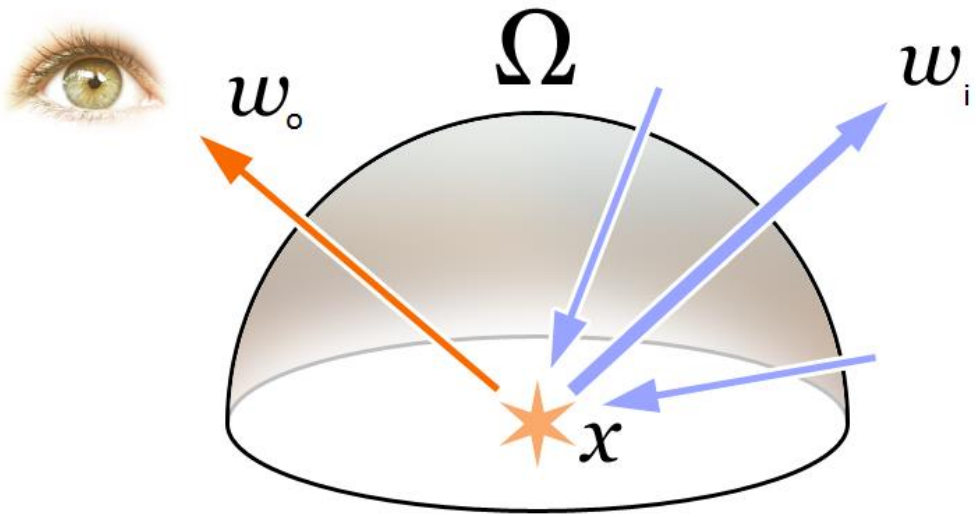
Изображение, обработанное алгоритмами глобального освещения

Источник: Wikipedia. Корнуэльская комната

Уравнение рендеринга

- интегральное уравнение, которое определяет количество светового излучения в определённом направлении как сумму собственного и отражённого излучения, учитывая функцию входящего излучения и двунаправленную функцию распределения отражения

wiki



Уравнение рендеринга

$$L_o(\mathbf{x}, \omega_o, \lambda, t) = L_e(\mathbf{x}, \omega_o, \lambda, t) + \int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o, \lambda, t)\, L_i(\mathbf{x}, \omega_i, \lambda, t)\, (\omega_i \cdot \mathbf{n})\, \mathrm{d}\omega_i$$

λ — длина волны света

t — время

L_o — общее спектральное излучение

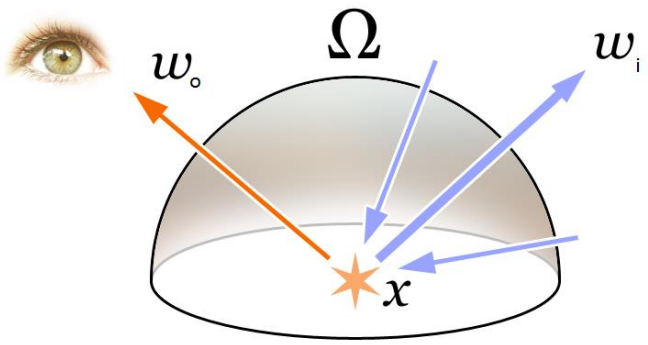
L_e — собственный излучённый свет

$\int_\Omega$ — интеграл берётся по полусфере входящих направлений

f_r — **д**вулучевая **ф**ункция **о**тражения / **b**idirectional **r**eflectance **d**istribution **f**unction

L_i — излучение по входящему направлению

$w_i \bullet n$ — cos θ — фактор ослабления внешнего излучения за счет угла падения



Уравнение рендеринга

- Впервые было опубликовано в работах David Immel [1] и James Kajiya [2] в 1986 году.
- Хотя уравнение является очень общим, оно не отражает все аспекты отражения света.

1. Immel, David S.; Cohen, Michael F. & Greenberg, Donald P. (1986), "A radiosity method for non-diffuse environments", Siggraph 1986: 133
2. Kajiya, James T. (1986), "The rendering equation", Siggraph 1986: 143, doi:10.1145/15922.15902

Некоторые отсутствующие аспекты

- Transmission – Прохождение света через поверхность, например, через стеклянный объект или поверхность воды
- Subsurface scattering – Подповерхностное рассеивание. Поверхности, отображаемые без его учёта, могут казаться непрозрачными
- Polarization – Поляризация, например, когда свет отражается от поверхности воды
- Phosphorescence – Фосфоресценция, которая возникает, когда свет или другое электромагнитное излучение поглощается в один момент времени и испускается в более поздний момент времени, как правило, с более длинной длиной волны
- Interference – Интерференция, где проявляются волновые свойства света
- Fluorescence – Флуоресценция, где поглощенный и испускаемый свет зависит от длины волны

Ещё некоторые отсутствующие аспекты

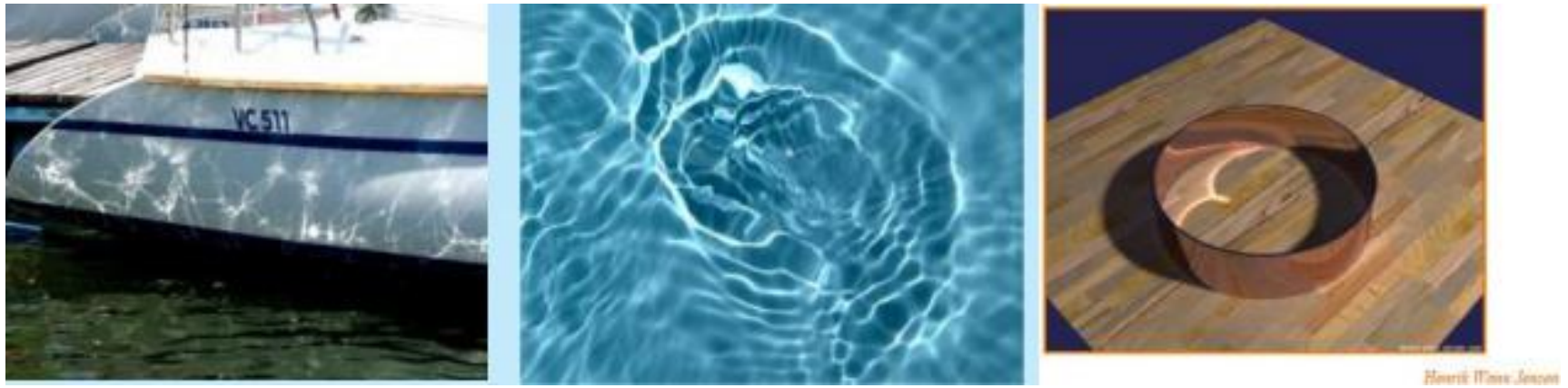
- Каустики – Световые узоры, которые появляются при преломлении или отражении световых лучей от поверхности

- Non-linear effects – Нелинейные эффекты, при которых очень интенсивный свет может повысить энергетический уровень электрона с большей энергией, чем у одного фотона (это может произойти, если электрон ударяется одновременно двумя фотонами). И излучение света с большей частотой, чем частота света, попадающего на поверхность, неожиданно становится возможным

- Relativistic Doppler effect – Релятивистский эффект Доплера – Изменение частоты и, соответственно, длины волны излучения, воспринимаемое наблюдателем, вследствие движения источника излучения и/или движения наблюдателя

Световые эффекты. Проблемы моделирования. Глобальное освещение

Каустики



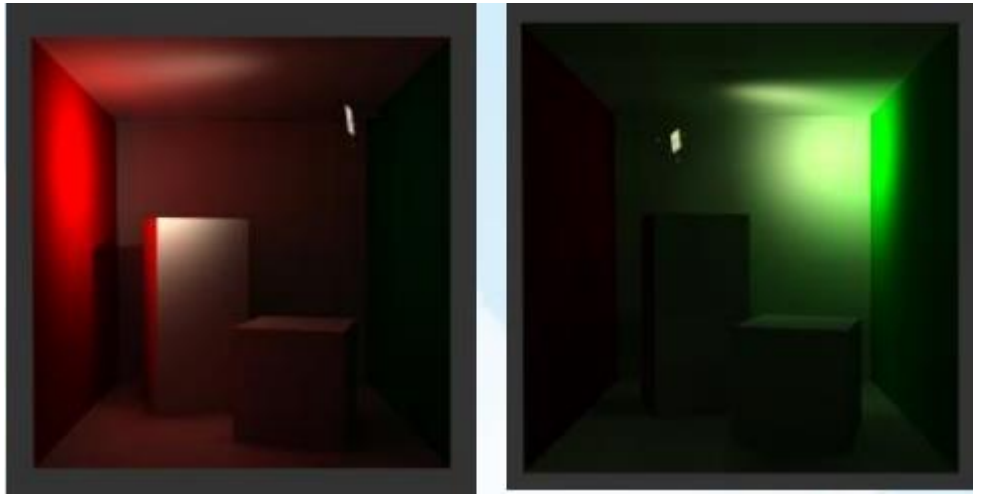
Рассеивание света

Световые эффекты. Проблемы моделирования. Глобальное освещение

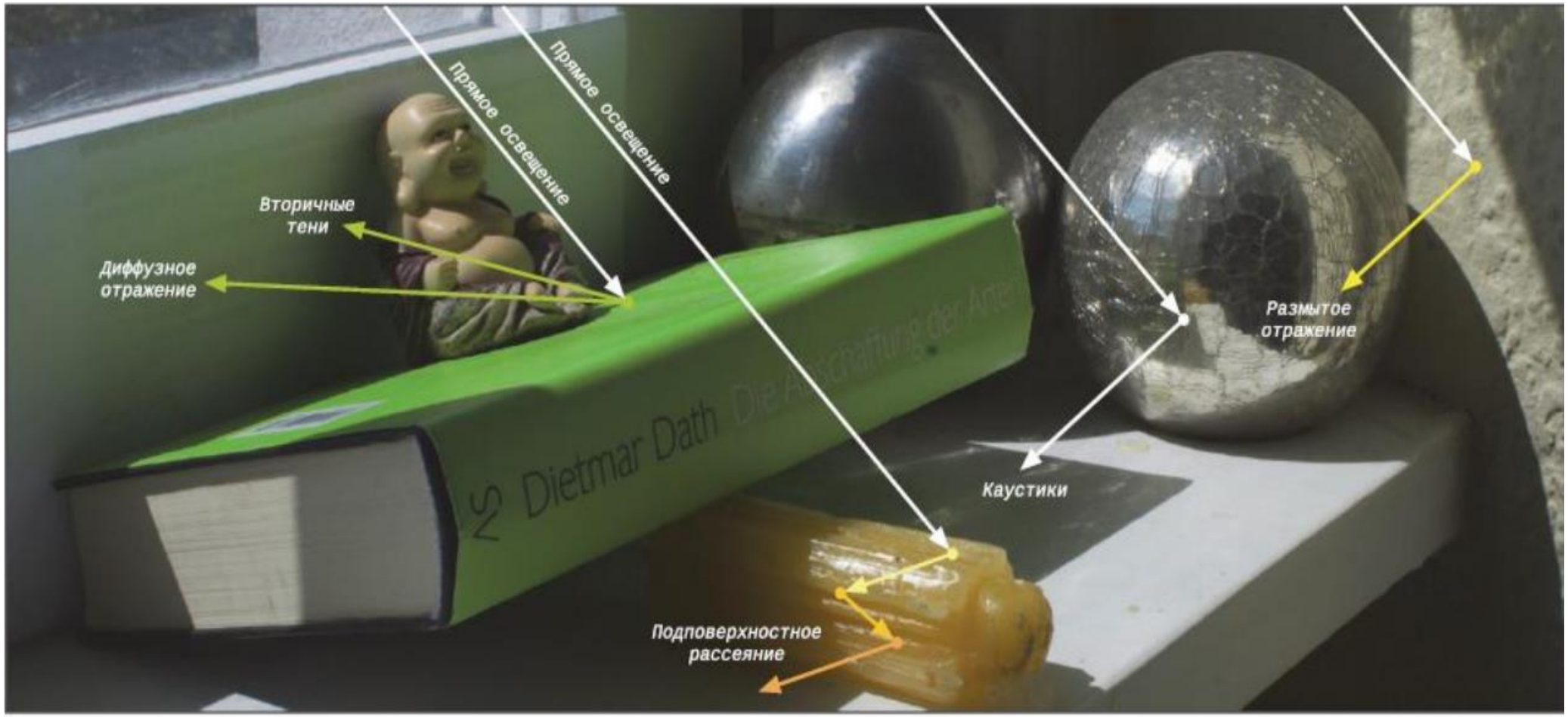
Радуга: рефракция и полное внутреннее отражение в капле

HDR

Освещение отраженным светом



Световые эффекты. Проблемы моделирования. Глобальное освещение



Примеры эффектов из публикации Ritschel T., Dachsbacher C., Grosch T., Kautz J. The state of the Art in Interactive Glabal Illumination // Computer Graphics Forum. – 2012. –February. –Vol.31, no.1. –Pp.160-188

Уравнение рендеринга

- **Решение уравнения** рендеринга для любой конкретной сцены является **главной задачей в реалистичном рендеринге** (realistic rendering)
- Т.к. это уравнение **не имеет аналитического** решения, на практике применяются различные **подходы к** его **аппроксимации**.

Различные алгоритмы КГ решают это основное уравнение

- Один из подходов к решению уравнения основан на методах конечных элементов, приводящих к алгоритму излучения – radiosity.
- Другой подход, использующий методы Монте-Карло, привёл, в частности, к созданию множества различных алгоритмов, включая:
  - трассировку путей – path tracing,
  - отображение фотонов – photon mapping,
  - легкий транспорт Метрополиса – Metropolis light transport
- Кроме того:
  - beam tracing,
  - cone tracing,
  - ambient occlusion,
  - image based lighting.

Наиболее известны

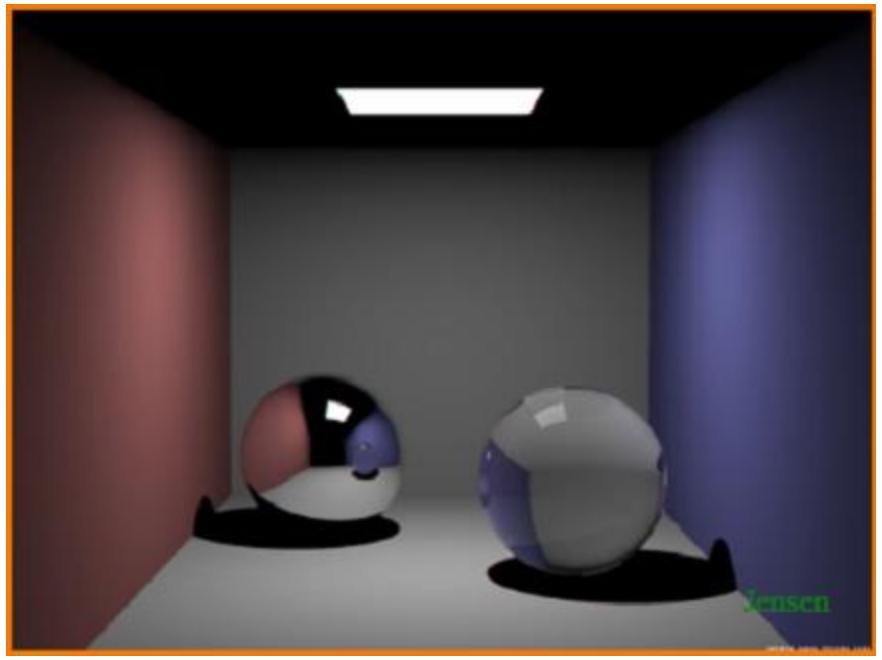
- Метод трассировки лучей (Ray tracing) (1968)
- Метод трассировки путей – (Path tracing) (1986)
- Radiosity (Michael F. Cohen, John R. Wallace,1995)
- Photon Mapping (Henrik Wann Jensen,1996)

Трассировка лучей — как общая технология

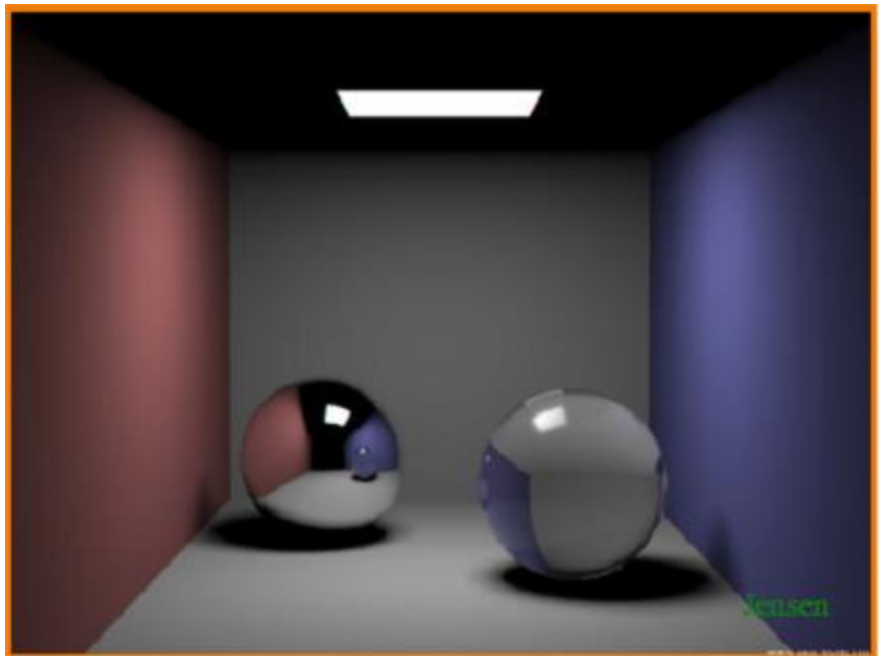
Трассировка лучей — как **общая технология**, включает **все методы** просчета движения, отражения и преломления лучей света, исходящих от камеры и прослеживающих путь к источнику света для создания физически корректной картинки.

Ray tracing – первый метод расчёта глобального освещения, рассматривающий освещение, расчёт тени, многократные отражения и преломления.

Подходы к упрощению основного уравнения освещённости



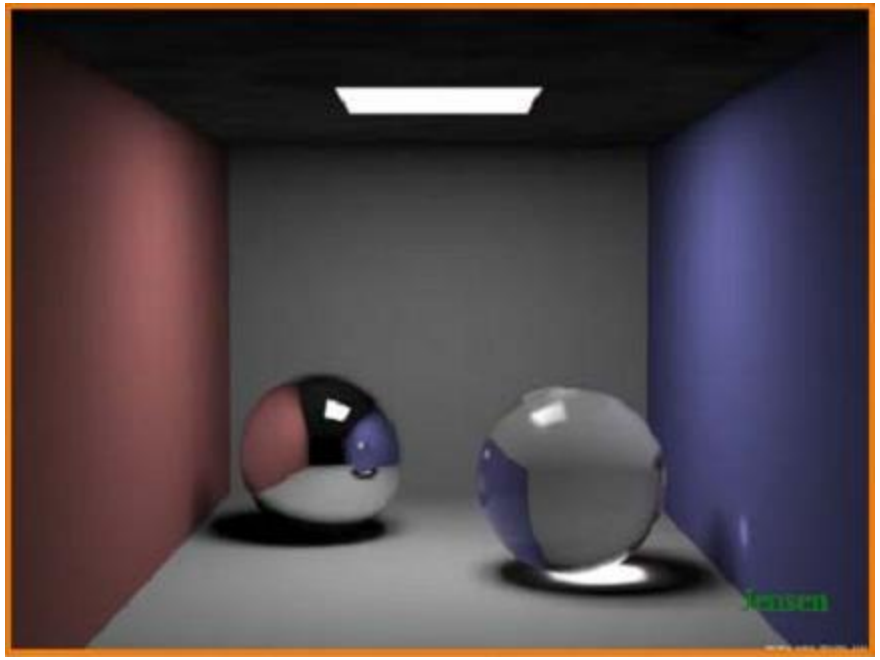
Ray Tracing

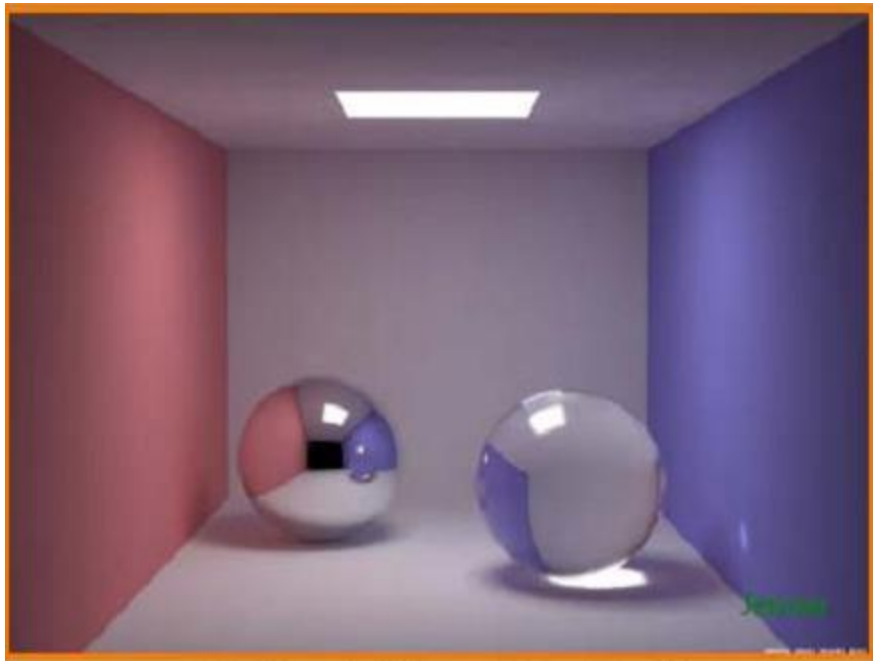


Henrik Wann Jensen

+ soft shadows

Подходы к упрощению основного уравнения освещённости





Henrik Wann Jensen

+caustics

+ indirect diffuse illumination

Классический ray tracing (метод трассировки лучей)

- предложен Артуром Аппелем (Arthur Appel, который в **1968** году описал технику прослеживания лучей между поверхностью и источником света для создания освещения с тенями — эту работу нередко упоминают, как **первое применение трассировки** лучей в рендеринге.

- дополнен алгоритмом общей рекурсии, разработанным Whitted в 1980 году

- понадобилось почти 12 лет эволюции вычислительных систем, прежде чем этот алгоритм стал доступен для широкого применения в практических приложениях

Вклад Тернера Уиттеда

Технология трассировки лучей для визуализации впервые была представлена **Тернером Уиттедом** в работе **1979** года под названием "Улучшенная модель освещения для затененного отображения".

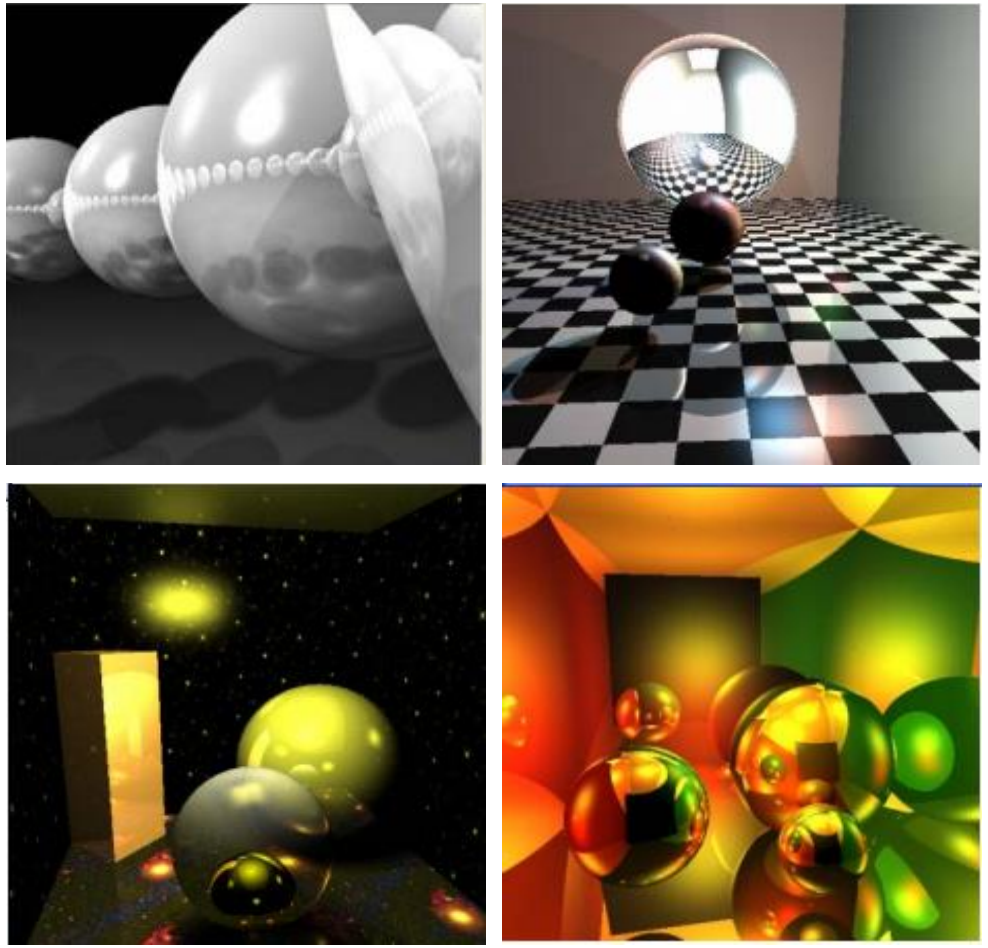
Публикация Уиттеда включала не только трассировку лучей для **генерирования теней**, но и для **вторичного влияния света** от **отражений** и преломлений.

Такой метод значительно **повысил реализм**, но был **лишен вторичного рассеянного** освещения.

Суть метода трассировки лучей

- **Суть метода**: отслеживание траекторий лучей и расчёта взаимодействий с лежащими на траекториях объектами, от момента испускания лучей источником света до момента попадания в камеру.

- Под взаимодействием луча с объектами понимаются процессы:
  - диффузного отражения (в смысле модели локальной освещённости),
  - многократного зеркального отражения от их поверхности
  - прохождение лучей сквозь прозрачные объекты



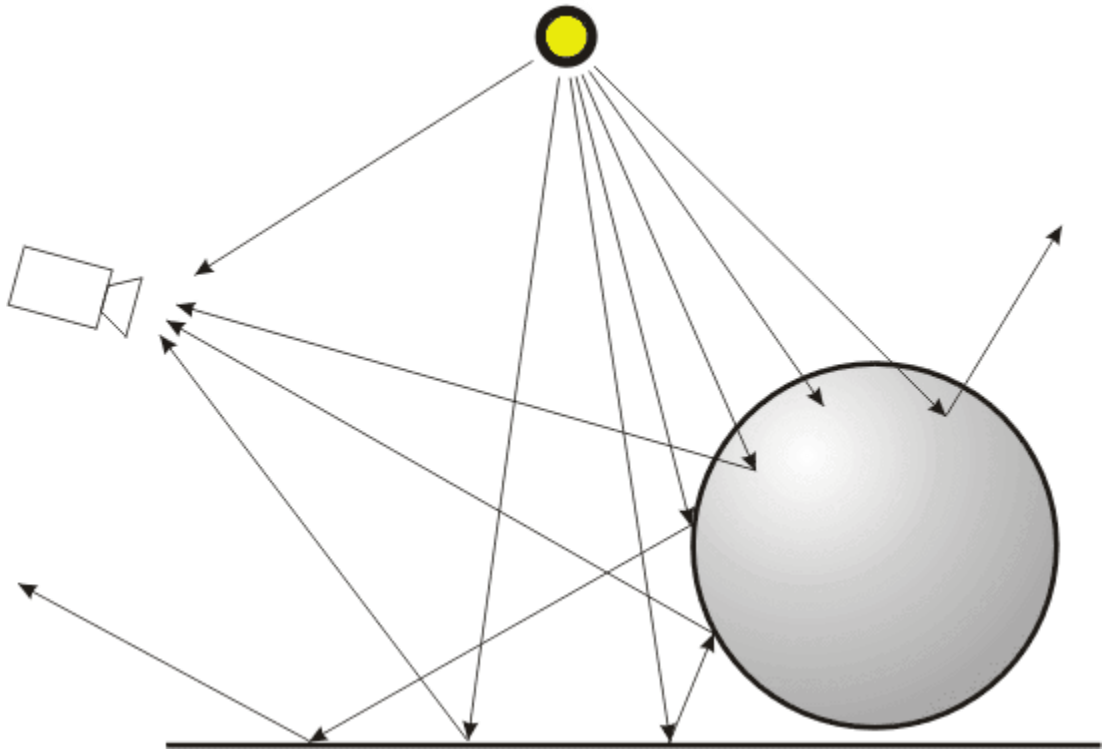
Результаты трассировки лучей на процессоре Core 2 Duo, полученные стажёрами Зимней школы 2008 по компьютерной графике (Intel-ННГУ). 512x512pix. Время – порядка 1 мин.

Трассировка лучей (Ray tracing). Прямая и обратная трассировка

- Различают два подхода к трассировке лучей:
  - метод прямой трассировки – forward ray tracing
  - метод обратной трассировки – backward ray tracing

- Ray tracing не использует модели шейдинга
- Расчёт освещённости «честно» выполняется во всех точках пересечения лучей и объектов

Прямая трассировка лучей. (Forward ray tracing)

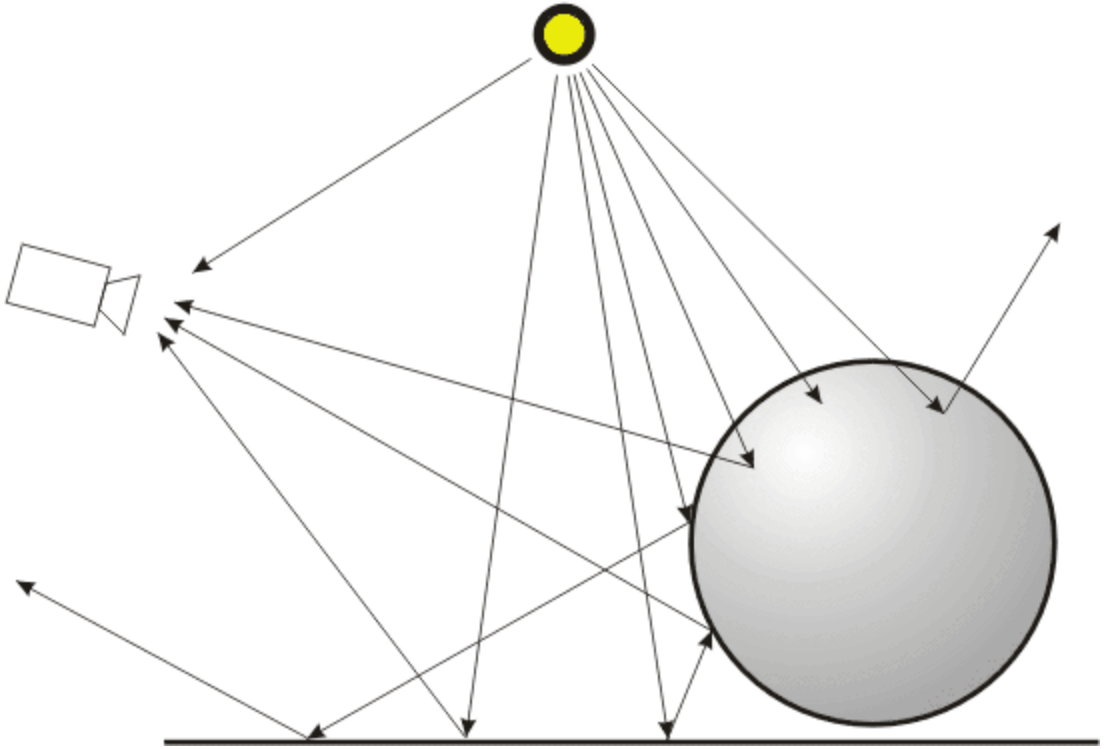


Лучи должны быть построены от каждого источника освещения ко всем точкам поверхностей всех объектов сцены и отслежены в соответствии с законами отражения и преломления.

Три исхода для лучей

- Луч выходит за пределы видимой из камеры области сцены – расчёты отбрасываются
- Луч попадает в камеру – формируется цвет
- Луч встречает на своем пути новый объект – рекурсивно продолжается расчёт

Давайте посчитаем

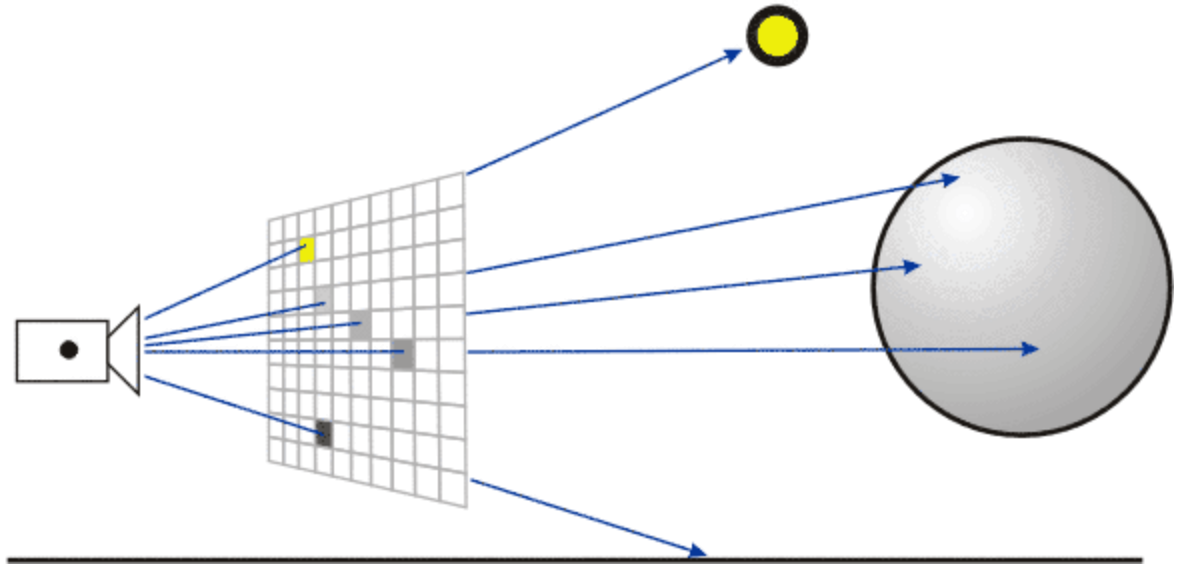


Лучи должны быть построены от каждого источника освещения ко всем точкам поверхностей всех объектов сцены и отслежены в соответствии с законами отражения и преломления.

«Грузить» процессор :))

- Помните? «Луч выходит за пределы видимой из камеры области сцены – расчёты отбрасываются»
- И зачем мы всё это рассматривали?

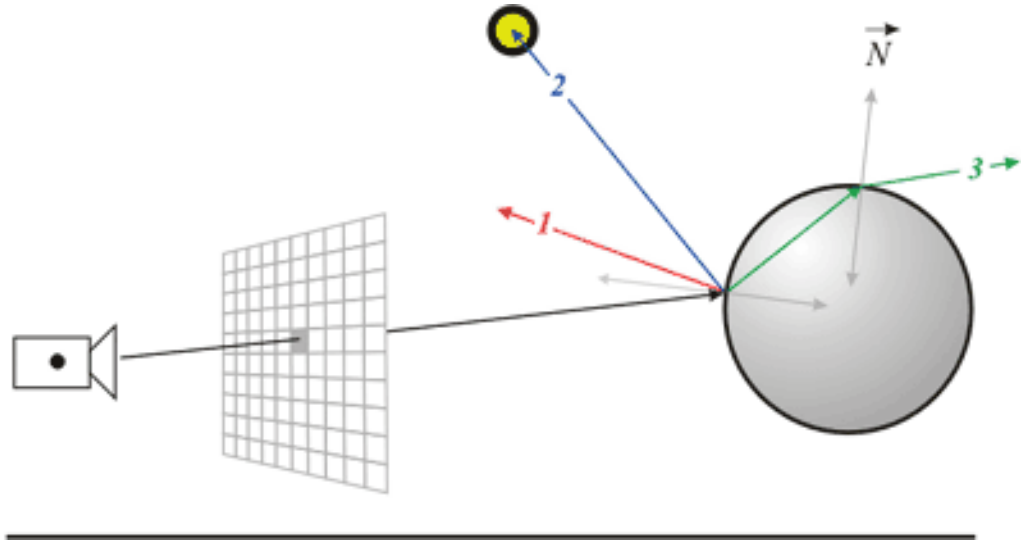
Обратная трассировка лучей (backward ray tracing)



- Количество первичных лучей известно – это общее количество пикселей видового окна

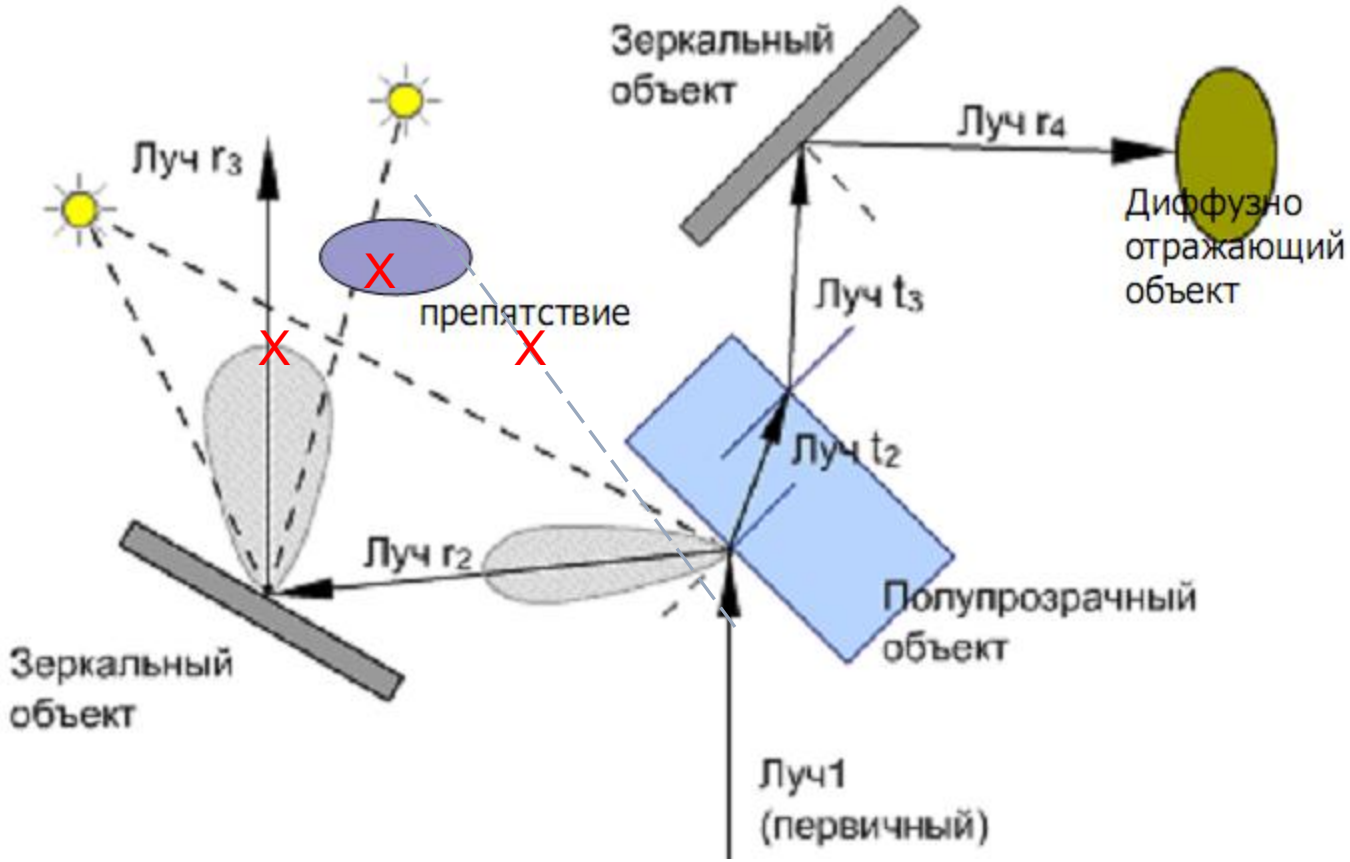
Обратная трассировка лучей

Разработана в 80-х годах. Основополагающими считают работы Уиттеда и Кея, ссылки на них можно найти в книге Дж.Фоли и А. ван Дэма (1985)



В точке пересечения луча с объектом строится три вторичных луча – один в направлении отражения (**1**), второй – в направлении источника света (**2**), третий в направлении преломления прозрачной поверхностью (**3**).

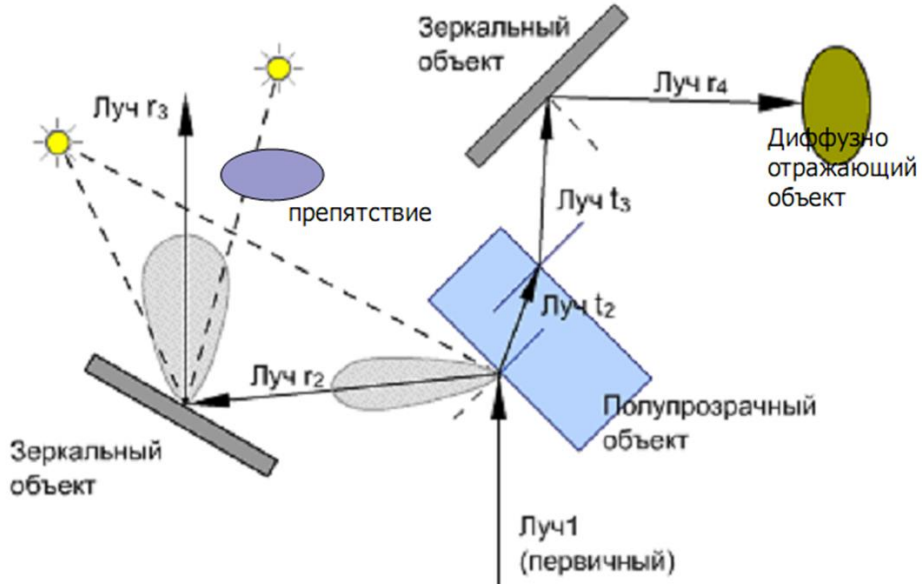
Обратная трассировка лучей. Пример



Направление от наблюдателя

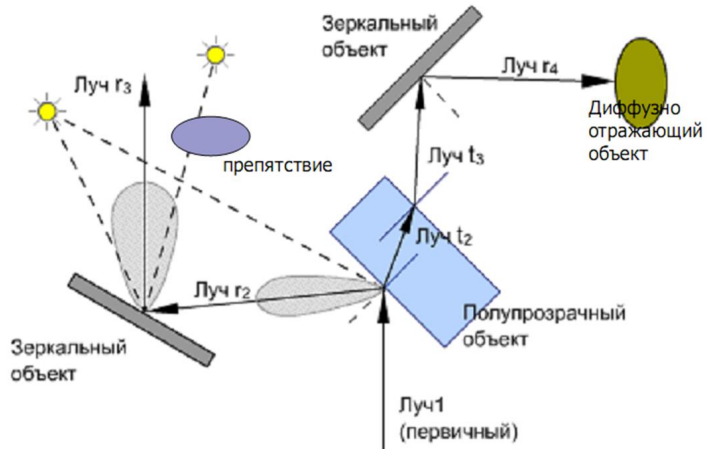
Луч в направлении источника света

- Основное назначение:
  - определить ориентацию точки (обращена точка к источнику или от него),
  - наличие объектов, закрывающих точку от источника света,
  - расстояние до источника света.



Алгоритм

- Если точка обращена в противоположную сторону от источника света или закрыта другим непрозрачным объектом, освещённость от такого источника не рассчитывается, точка находится в тени. Если точка закрыта от освещения всеми источниками сцены, ей присваивается фоновый ambient цвет.
- В случае затеняющего прозрачного объекта интенсивность освещения уменьшается в соответствии со степенью прозрачности.
- В противном случае точка освещена, интенсивность и цвет освещения рассчитываются при помощи локальной модели освещённости (diffuse + specular) как сумма освещённостей от всех источников, для которых эта точка не закрыта другими объектами.

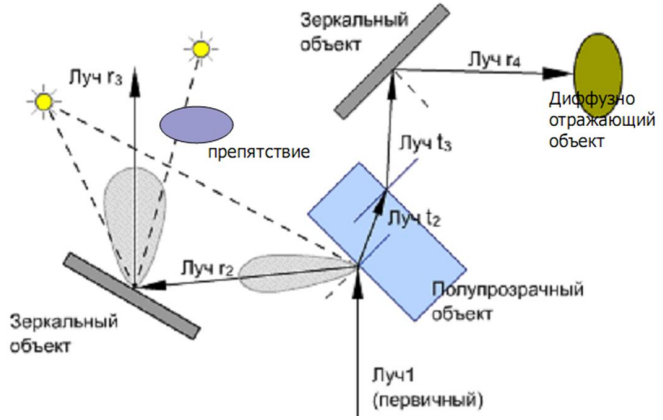


Теневой луч

- Этот тип луча получил название shadow ray (иногда его ещё называют illumination ray) – теневой луч.

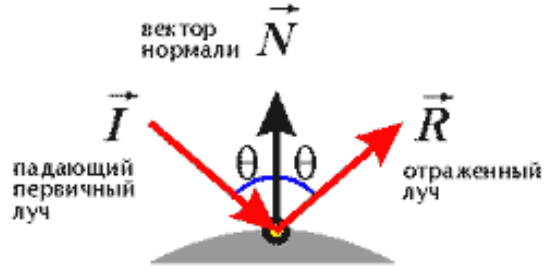
- Если поверхность объекта не является отражающей и непрозрачна, теневой луч – единственный тип лучей который строится, траектория первичного луча обрывается (заканчивается), и дальнейшие расчёты не выполняются.

- Рассчитанный цвет (освещённости или тени) присваивается тому пикселю видового окна, через который проходит соответствующий первичный луч.



Луч отражения (reflection ray)

Направление отраженного луча определяется по закону:



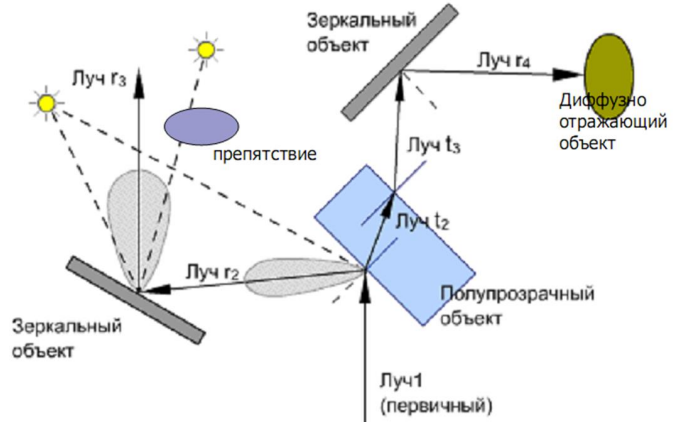
$$\vec{R} = \vec{I} - 2\vec{N}(\vec{N}\cdot\vec{I}),$$

где $(\vec{N}\cdot\vec{I})$ - скалярное произведение векторов

Для отраженного луча проверяется возможность пересечения с другими объектами сцены.

Если пересечений нет, то интенсивность и цвет отраженного луча равна интенсивности и цвету фона.

Если пересечение есть, то в новой точке снова строится три типа лучей – теневые, отражения и преломления.

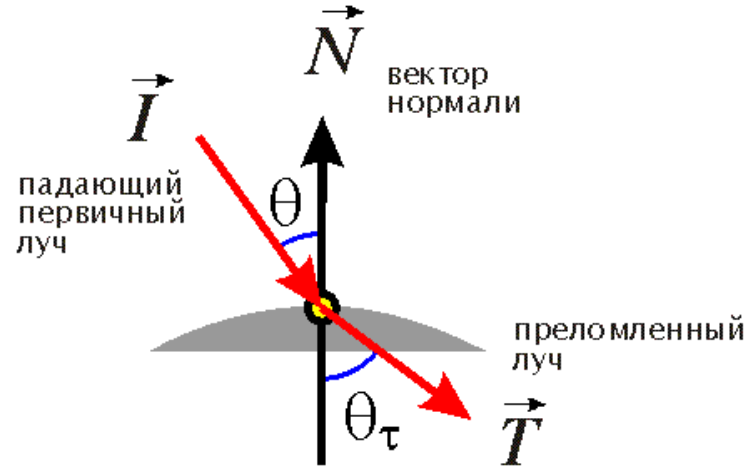


Луч прозрачности (transparency ray)

Направление для преломлённого луча определяется следующим образом:

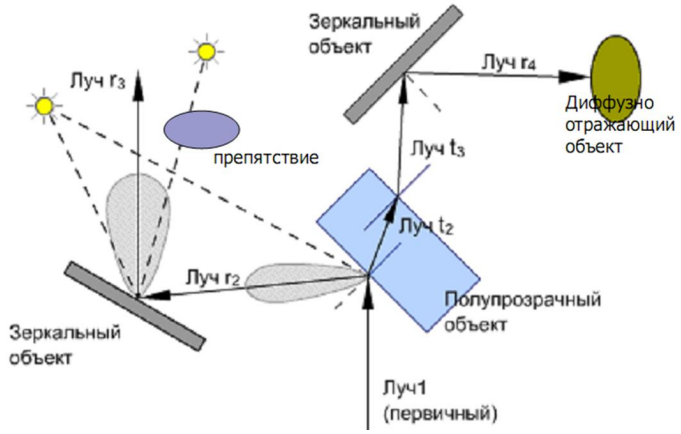
n1 и n2 – коэффициенты рефракции для первой среды (в которой распространяется первичный луч) и второй среды прозрачного объекта.

Как и в предыдущем случае, проверяется пересечение вновь построенного луча с объектами, и, если они есть, в новой точке строятся три луча, если нет – используется интенсивность и цвет фона.

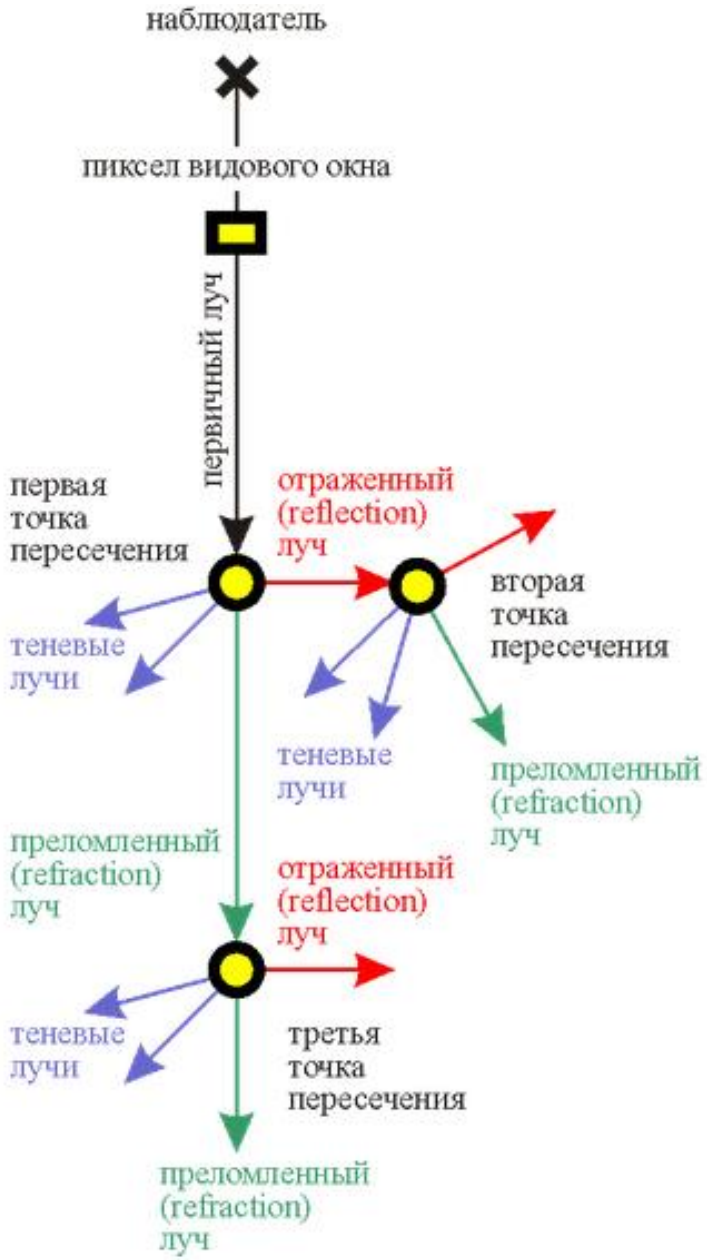


$$\vec{T} = \frac{n_1}{n_2}\vec{I} - \left[\cos\theta_\tau + \frac{n_1}{n_2}(\vec{N}\cdot\vec{I})\right]\vec{N},$$

$$\cos\theta_\tau = \sqrt{1 - \left(\frac{n_1}{n_2}\right)^2(1 - (\vec{N}\cdot\vec{I})^2)}$$

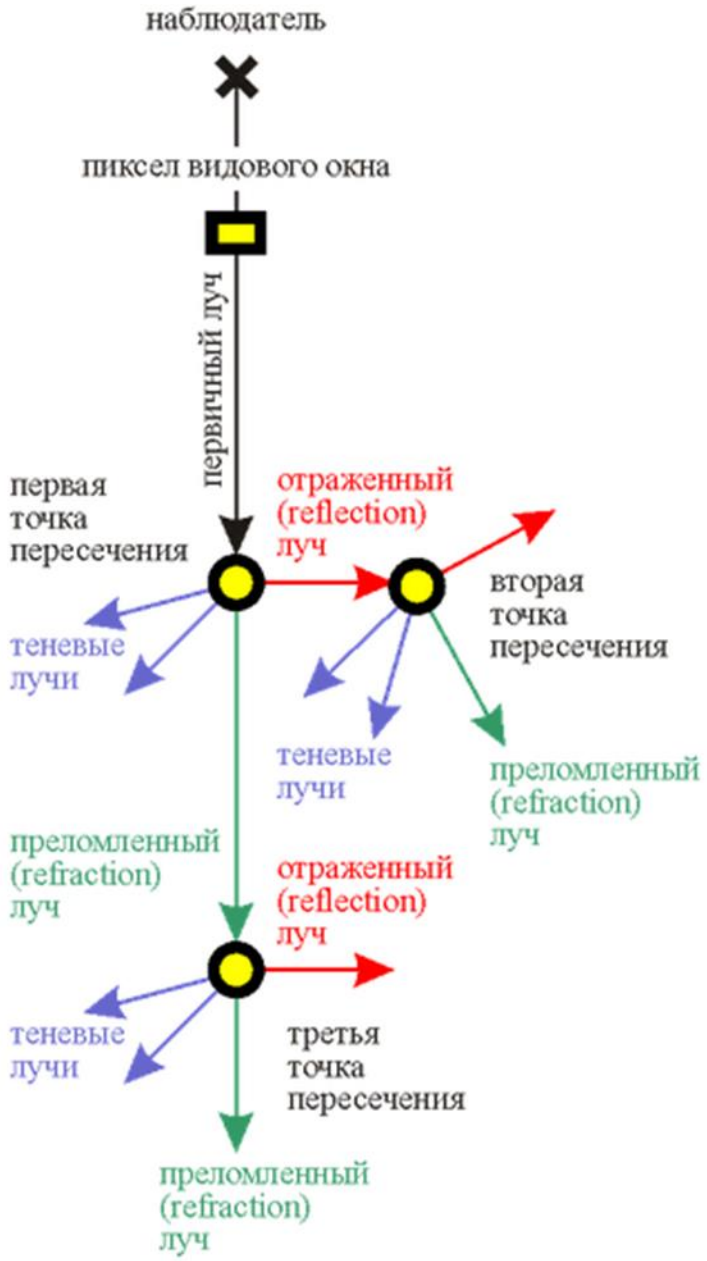


Для каждого первичного луча можно построить древовидную структуру следующего вида



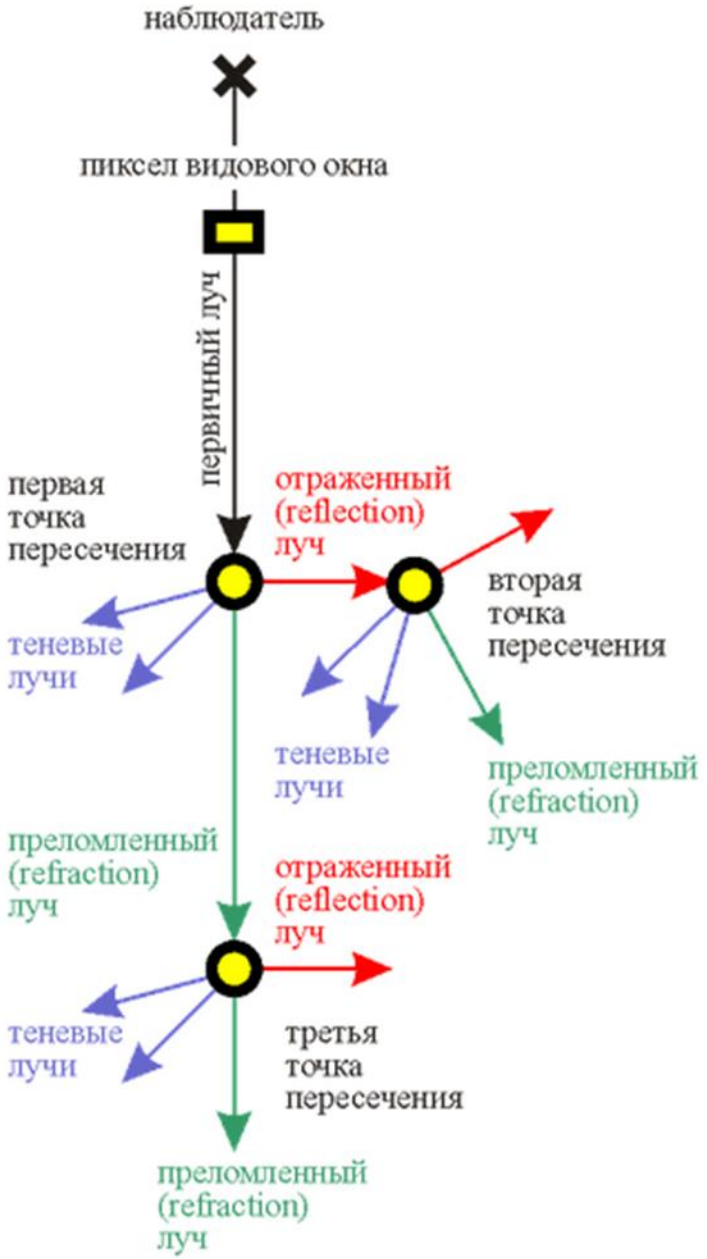
Построение дерева

- В **каждой** точке пересечения **теневые** лучи строятся **всегда**.
- **Преломлённый** и **отражённый** лучи строятся **при условии**, что поверхность пересечения обладает свойствами отражения или преломления.
- Если объект такими свойствами не обладает, в точке пересечения **вычисляется освещённость (diffuse и specular)**, и луч обрывается (считается, что он заканчивается на источнике света).
- В идеале, процесс продолжается до тех пор, пока все лучи либо не будут рассеяны на чисто диффузных поверхностях, либо не выйдут за пределы видимой области.



Расчёт освещённости по дереву

- Для каждой ветви дерева спускаемся вдоль древовидной структуры к последнему пересечению вторичного луча и поверхности (будем дальше называть их узлами).
- Поскольку это последний узел в цепи, вкладов от преломлений и отражений нет.
- Поэтому освещённость узла вычисляется при помощи локальной модели освещения с учетом видимости источников света для данного узла.
- Затем, вычисленная освещённость передается вверх по ветви к следующему ближайшему узлу.



Освещённость в нелистовом узле будет вычисляться по формуле:

$$\mathbf{I}_n = \mathbf{I}_{local} + \mathbf{k}_{reflection} * \mathbf{I}_{reflection} + \mathbf{k}_{refraction} * \mathbf{I}_{refraction}$$

где

I_n – полная освещённость в узле *n*;

I_{local} – локальная освещённость в узле *n*, вычисленная от источников освещения с помощью локальной модели освещённости;

$k_{reflection}$ – коэффициент, определяющий отражающие свойства поверхности;

$I_{reflection}$ – освещённость предыдущего узла, переданная вдоль ветки отражения;

$k_{refraction}$ – коэффициент, определяющий преломляющие свойства поверхности;

$I_{refraction}$ – освещённость предыдущего узла, переданная вдоль ветки преломления.

Когда изображение считается построенным?

Когда изображение считается построенным?

- Результаты освещённости накапливаются при подъёме от самого последнего узла через все узлы ветви.

- Изображение можно считать построенным, когда цвета всех пикселов видового окна (все соответствующие деревья для каждого первичного луча) вычислены.

Насколько глубоким может быть дерево узлов?

Как далеко прослеживаются вторичные лучи?

Насколько глубоким может быть дерево узлов? Как далеко прослеживаются вторичные лучи?

- Естественным завершением трассировки лучей является выход всех испущенных вторичных лучей за пределы видимой области и их рассеяние на чисто диффузных объектах.

- Результат вычислений будет наиболее точным.

Насколько глубоким может быть дерево узлов? Как далеко прослеживаются вторичные лучи?

- Но, если сцена достаточно сложна, такой расчёт будет очень медленным, а в некоторых случаях и невозможным по причине ограниченности аппаратных ресурсов.
- Вклад освещённости от каждого нового вторичного луча очень быстро уменьшается, т.к. коэффициенты свойств отражения и преломления <1.
- Например, если коэффициент отражения одинаков для всех поверхностей и равен 0.3, то вклад 4-го отражения уменьшится, будучи умноженным на 0.0081 (0.3*0.3*0.3*0.3), а общее количество лучей значительно увеличится.
- Поэтому часто трассировку лучей прекращают, когда вклад от следующего узла ветви становится меньше заданной величины.
- Это также достаточно точный метод расчётов, который может быть использован для получения качественных результатов.

Насколько глубоким может быть дерево узлов? Как далеко прослеживаются вторичные лучи?

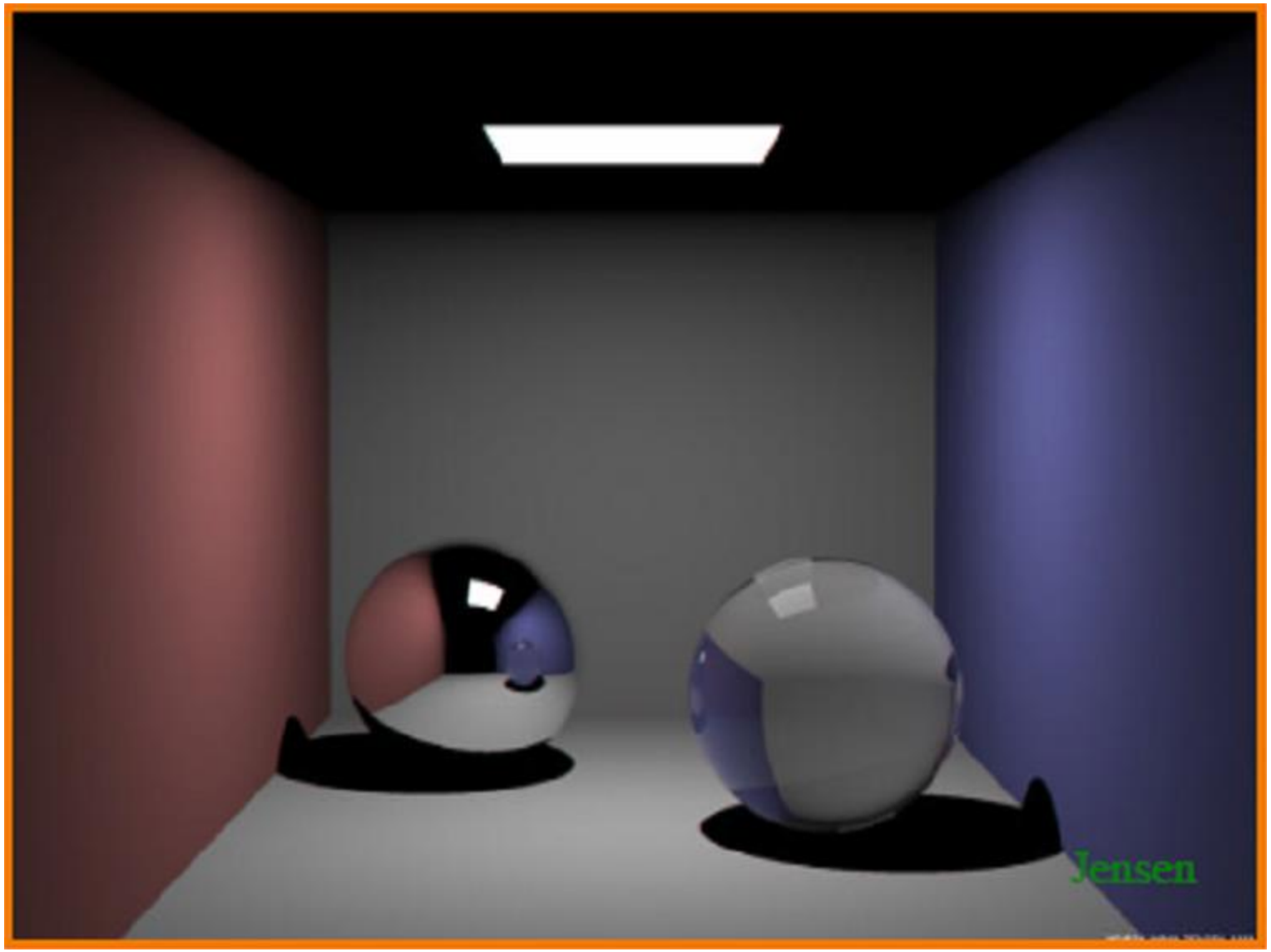
- Наконец, для получения оценочного расчёта можно оборвать трассировку лучей после выполнения заданного количества итераций, это самый быстрый и наименее точный расчет.

Резюме

- Алгоритм обратной трассировки лучей стал основным способом расчётов освещённости методом трассировки лучей.

- Метод трассировки лучей — первый метод расчёта глобальной освещённости, учитывающий взаимное влияние объектов сцены друг на друга.

Трассировка лучей (Ray tracing)



Плюсы

Основные достоинства рекурсивного метода обратной трассировки лучей, значительно повысивших степень реалистичности компьютерных рендеров:

- расчёт теней,
- расчёт многократных отражений и преломлений.

Ограничения метода обратной трассировки

1. Выделяют источники света. Они могут только излучать, но не отражать и преломлять. Ограничивают многообразие источников точечными.
2. Свойства отражающих поверхностей задаются суммой диффузной и зеркальной компонент. Зеркальность описывается составляющими reflection и specular. Reflection учитывает отражение других объектов, не являющихся источниками. Строится только 1 зеркально отражённый луч r для дальнейшей трассировки.
3. Specular даёт световые блики от источников. Для этого определяются углы α отражённого луча обратной трассировки со всеми источниками.
4. При диффузном отражении учитываются только лучи от источников света (от зеркал игнорируются).

Ограничения метода обратной трассировки

5. Для прозрачных (transparent) объектов обычно не учитываются зависимость преломления от λ. Иногда прозрачность моделируется без преломления.
6. Для учёта освещённости объектов светом, рассеиваемым другими объектами, вводится фоновая составляющая (ambient)
7. Низкая скорость и высокая вычислительная стоимость расчётов – в классическом рейтресинге необходимо проверять на пересечение каждый луч со всеми объектами сцены. В результате от 70% до 95% всего времени расчётов тратится на вычисление пересечений.
8. Резкие границы цветовых переходов тени/подсветок/прозрачности.
9. Aliasing – "зазубренность" линий и т.д.
10. Дискретность определяющих цвет пиксела первичных лучей – одного первичного луча недостаточно для корректного определения цвета пиксела, формирующего изображение.

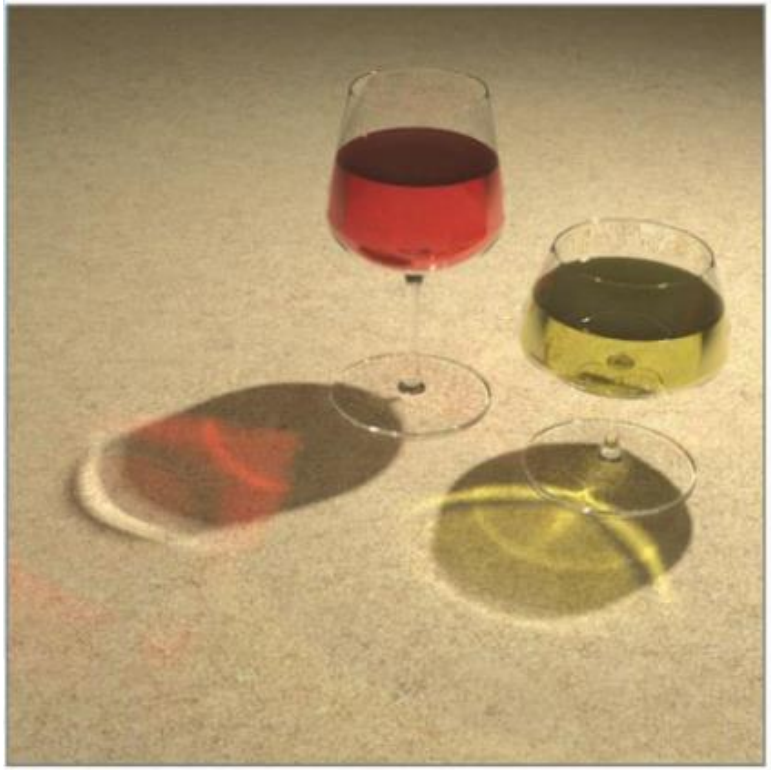
Оптимизация — использование оболочек

- В данном алгоритме следует при каждом вызове проверять (вычислять) наличие пересечения с гранями объектов → перебор всех граней.
- Для ускорения процесса применяется метод оболочек – для отбрасывания заведомо неприемлемых кандидатов.
- Оболочки могут образовывать древовидную структуру.
- Это позволяет существенно ускорить перебор и сделать его теоретически пропорциональным логарифму от числа граней.

Алгоритм с учетом отимизаций

Базовая операция обратной трассировки – вычисление интенсивности для трассируемого луча:

ЛУЧ(номер итерации ind, тип луча, направление луча dir, номер объекта no) {
Находим точку пересечения луча с ближайшим объектом (гранью); **Если точка найдена, то**
{ no=номер пересекаемого объекта; // использовать метод оболочек
Вычисляем нормаль к видимой стороне пересекаемой грани;
Если ($k_d > 0$), то // задано свойство диффузного отражения
{L_d=Сумма интенсивности диффузного отражения для всех источников;}
Если ($k_s > 0$), то // зеркальные блики от источников света
{Определяем направление отраженного луча dirR;
L_s=Интенсивность зеркального блика с учетом α для всех источников;}
Если ($k_r > 0$), то // зеркальное отражение других объектов
{Определяем направление отраженного луча dirR;
L_r=ЛУЧ(ind+1, отраженный, dirR, no);} // рекурсия
Если ($k_t > 0$), то // объект полупрозрачный
{Определяем направление преломленного луча dirT;
L_t=ЛУЧ(ind+1, преломленный, dirT, no);} // рекурсия
return $k_a \cdot L_a \cdot C + k_d \cdot L_d \cdot C + k_s \cdot L_s + k_r \cdot L_r + k_t \cdot L_t$; // k,L и C - вектора (для RGB)
}
иначе {Луч уходит в свободное пространство; return Значение по умолчанию (цвет фона)}}

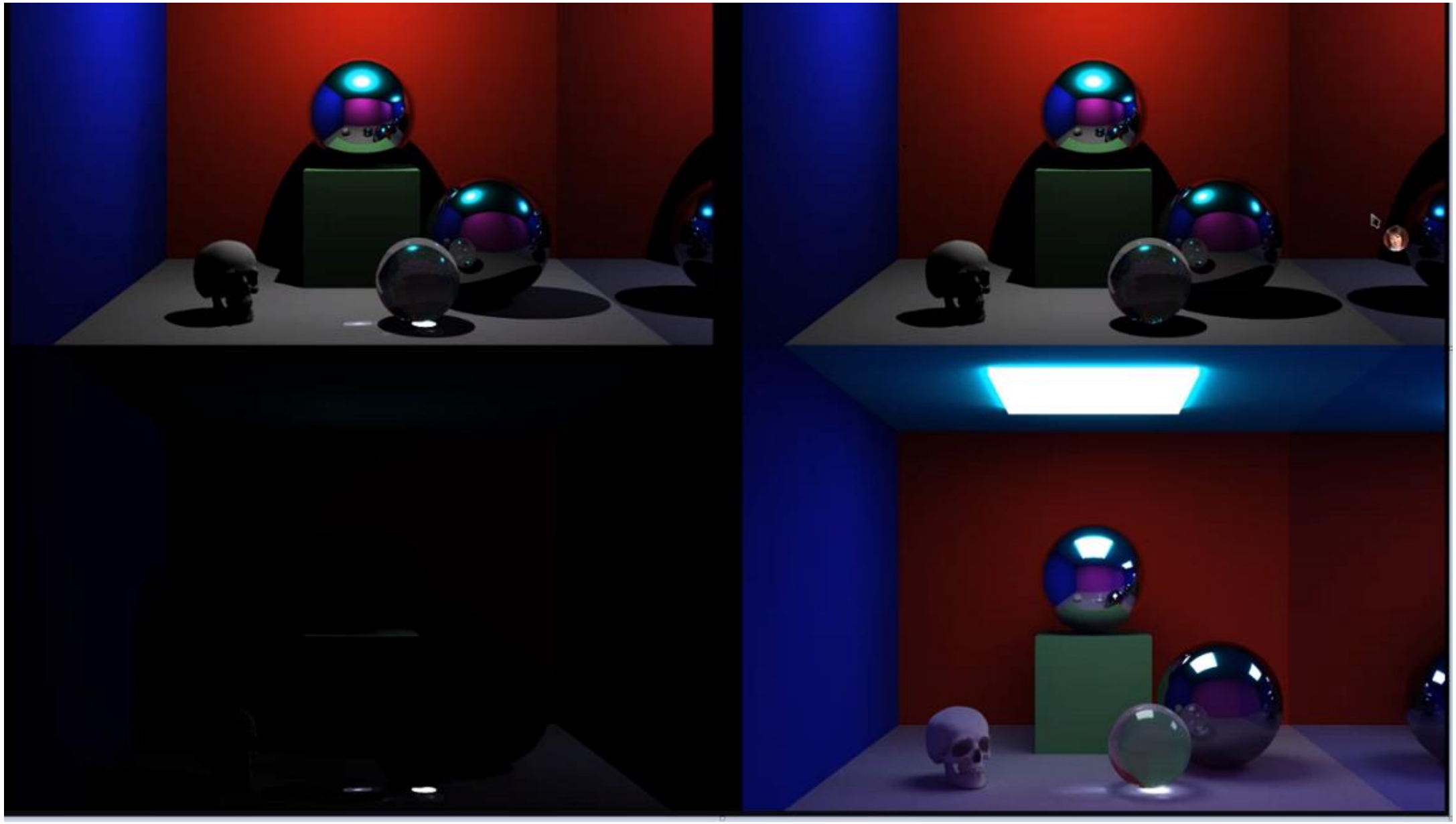


Результат суммирования

Результат прямой трассировки света

Результат трассировки пути от камеры

Реализация - Сергей Дуюнов - 2020



Достоинства

В **1984** году была опубликована работа **Роберта Кука** «Распредёленная трассировка лучей», в которой была открыта важная **концепция случайного распределения лучей** во времени и пространстве для достижения **размытия в движении**, **полуглянцевых отражений**, **освещения по площади и глубины резкости**.

И все это при помощи тех же лучей, которые необходимо генерировать для работы со сглаживанием.

Недостатки

Но, весь процесс трассировки лучей по своей сути крайне неэффективен, даже если он эффективен в сравнении с иными методами физического рендеринга.

Несмотря на оптимизацию метода в работе Кука, экспоненциальная характеристика трассировки оставалась фундаментальной преградой для применения технологии.

Трассировка путей

В 1986 году Джеймс Кажийя опубликовал «Уравнение рендеринга» и предложил **новую, скоростную форму трассировки лучей, которую назвал «трассировка путей».**

Трассировка путей предоставляет **решение проблемы экспоненциального роста** количества лучей. Вместо постоянно расширяющегося древа путей, трассировка путей ведет себя **подобно распределенной трассировке лучей, но выделяя лишь один луч на «отскок»**.

Рандомизируя тип луча и направление на каждый «отскок», вместо экспоненциальной генерации лучей, значительно снижается время просчёта и необходимость вычислительной мощности, позволяя работать со всеми источниками освещения, в том числе с глобальным освещением.

Трассировка путей: недостатки

В современных движках рендеринга с трассировкой лучей разработчики обнаружили, что трассировка путей включает свои недостатки.

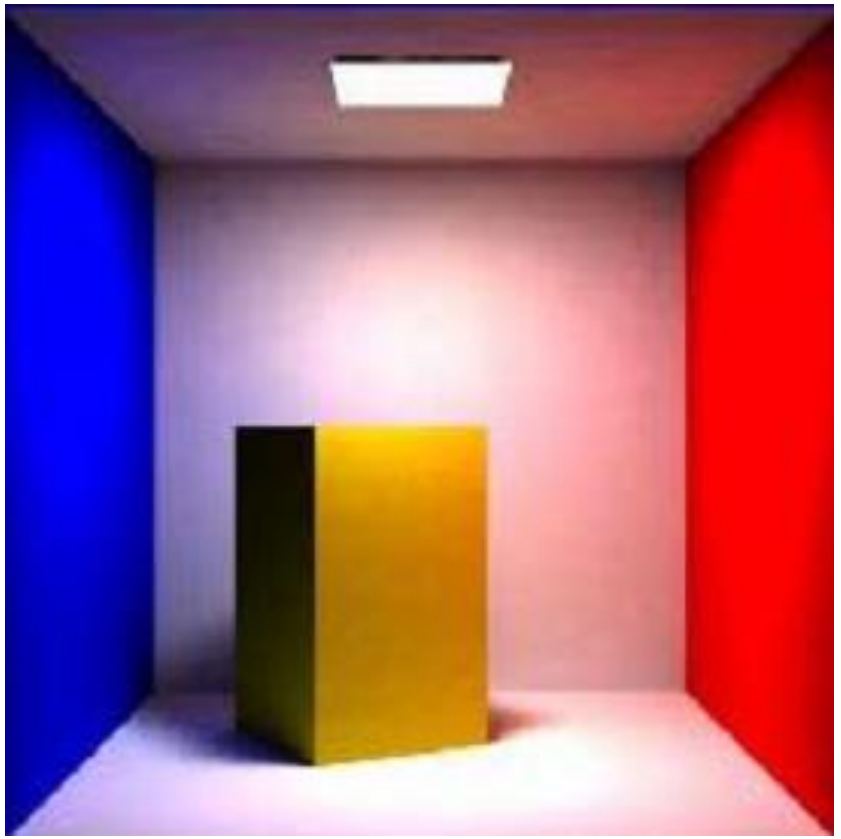
Особенно это касается поздних «отскоков», которые начинают требовать гораздо больше лучей для просчета.

В связи с этим разработчики постепенно оптимизируют эффективность рендеринга с использованием традиционной ветвящейся системы трассировки.

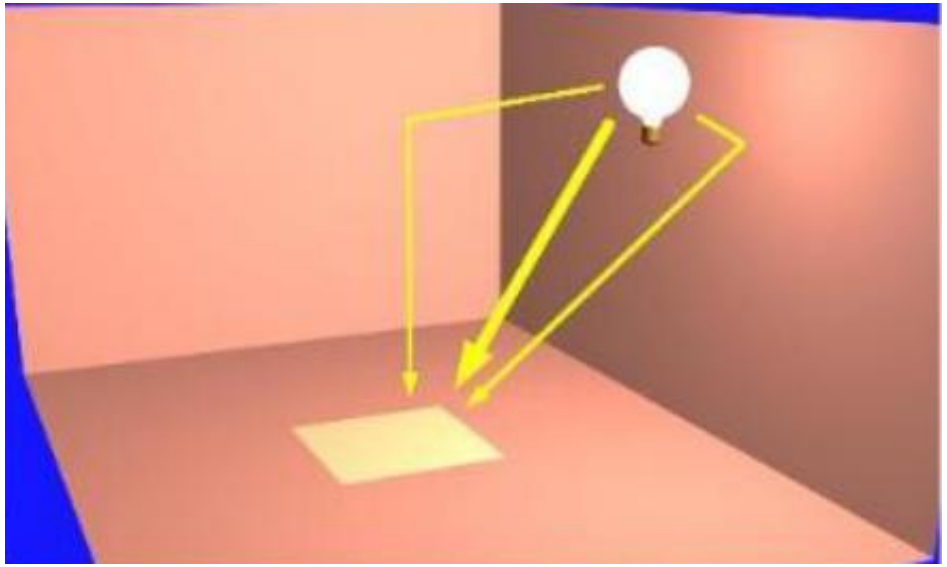
Метод Radiosity

- Рейтресинг рассчитывает диффузную освещённость трёхмерных поверхностей только от прямых источников света в соответствии с законом Ламберта.
- Учитывая, что диффузная освещённость несёт основную визуальную информацию о геометрии и цвете объектов, совершенно очевидно, что только закона Ламберта для достижения фотореалистичных результатов расчёта освещённости недостаточно.
- В частности, при расчёте диффузного освещения данной поверхности необходимо учесть не только её освещение прямыми источниками света, но и так называемое вторичное (или отраженное) диффузное освещение – диффузное рассеяние света окружающими объектами.
- Метод расчётов, позволяющих **учесть вторичное диффузное освещение**, получил название **radiosity**, или **метод излучательности**.

Radiosity



Метод Radiosity



- В ранних 1960-х инженеры разработали методы моделирования перемещения теплового излучения между поверхностями в печах или двигателях.

- В середине 1980-х → компьютерная графика.

- Метод Radiosity вычисляет интенсивность (цвет) всех поверхностей в окружении, что предпочтительнее, чем вычислять цвет каждой точки экрана.

Основные идеи radiosity

- *Потоком* энергии (вообще – тепловой, но в нашем случае – световой) называют количество энергии, проходящей через некоторую площадь в единицу времени.
- *Плотностью* потока энергии называют поток энергии, проходящий через единицу площади.
- Другими словами, плотность потока энергии – это количество энергии переносимой через единицу площади в единицу времени.
- Под собственной энергией поверхности можем понимать диффузное рассеяние света от прямого источника, а под приносимой энергией – освещение, отражённое диффузно всеми окружающими объектами.
- Сама величина radiosity определяется как плотность потока энергии, испускаемой патчем (покидающей патч).
- Таким образом, radiosity – это поток энергии, исходящей от единицы площади поверхности в единицу времени.

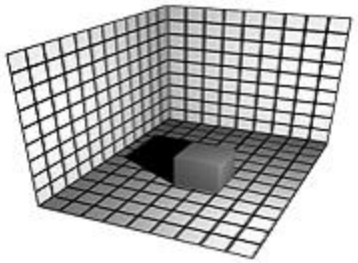
Метод Radiosity

- В основе аппроксимации основного уравнения визуализации методом Radiosity лежит предположение о том, что все поверхности сцены являются идеальными диффузными поверхностями, т.е. рассеивают падающий свет во все стороны с одинаковой интенсивностью (закон Ламберта).
- После принятия такого упрощения мы можем вынести функцию BRDF из-под знака интеграла, т.к. она будет постоянной. В результате имеем основное уравнение Radiosity:

$$B(x) = E(x) + \rho(x) \int_{x'} B(x') \frac{V(x,x')G(x,x')}{\pi} dx'$$

Здесь B(x) – это энергия рассеиваемая элементом.

Допущения



- Для решения уравнения Radiosity, мы должны **разбить все поверхности** нашей сцены на дискретные элементы конечной величины, так чтобы это позволило нам перейти от интеграла к сумме.

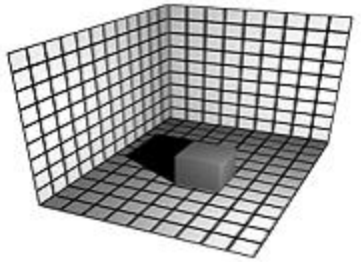
- **Карта освещенности** (lightmap). Под дискретными элементами конечной величины подразумеваются точки лайтмэпов.

- Также мы полагаем, что **светопередающие характеристики** (отражающая и рассеивающая способность) этих элементов **одинаковы**.

Подробнее о допущениях

- Поверхности всех объектов трёхмерной сцены **разбиваются на плоские небольшие участки – патчи** (patch).
- В некотором смысле они напоминают полигоны, с помощью которых описываются поверхности, хотя на самом деле таковыми не являются. То есть, они могут быть больше или меньше размера реальных полигонов и по сути **являются модельным инструментом radiosity, а не средством точного описания поверхности**.
- **Размер** каждого патча должен быть **настолько мал**, что **плотность распределения интенсивности** световой энергии в его пределах можно считать **постоянной величиной** (константой).
- И наконец, считают, что **диффузное рассеяние света не зависит от угла**, то есть свет рассеивается равномерно по всем направлениям.
- Такой подход для получения и решения уравнений освещённости radiosity получил название **finite element method – метод ограниченных(конечных) элементов**.

Процедура Radiosity



- После разбиения поверхностей 3D сцены на дискретные элементы, в один проход по стандартным алгоритмам **рассчитываем первичную освещённость поверхностей от всех источников света**.
- После этого знаем, какая энергия источников света приходится на каждую точку лайтмэпа.
- Затем, собственно, само Radiosity: представив, что **каждый элемент является самостоятельным источником** световой энергии, которую он получил от первичных источников света, **рассчитываем вторичную освещенность**, т.е. учитываем **влияние одних элементов на другие**.
- Затем, учитывая поглощение, проход за проходом **рассчитываем переизлучение** световой энергии между поверхностями сцены до тех пор, пока не установится равновесие.

Вычисление потока энергии

- Поток энергии каждого из окружающих патчей вычисляется как **произведение площади** такого **патча** на **плотность испускаемого им потока энергии** (т.е. radiosity этого патча), поскольку патч настолько мал, что плотность потока в его пределах можно считать постоянной величиной.
- Необходимо учесть **взаимную ориентацию** и **расстояние** между патчами. Для этого вводится так называемый **форм-фактор**.
- **Форм-фактор** просто определяет, **какая часть** всего **потока** излучаемой энергии (radiosity) данного патча **достигает другого** патча.

Вычисление потока энергии

- Основные уравнения расчёта radiosity имеют следующий вид:

$$\mathbf{B}_i = \mathbf{E}_i + \rho_i \sum_{j=1}^{n} \mathbf{B}_j \mathbf{F}_{ij}$$

где $\mathbf{B}_i$ *- radiosity i-го патча,*

$\mathbf{E}_i$ *- диффузное отражение света от прямого источника;*

ρ_i *- коэффициент, характеризующий свойства диффузного отражения поверхности, которой принадлежит патч;*

$\mathbf{B}_j$ *- radiosity j- го патча;*

$\mathbf{F}_{ij}$ *- форм-фактор, определяющий, какая часть радиосити j-го патча достигнет i-го патча;*

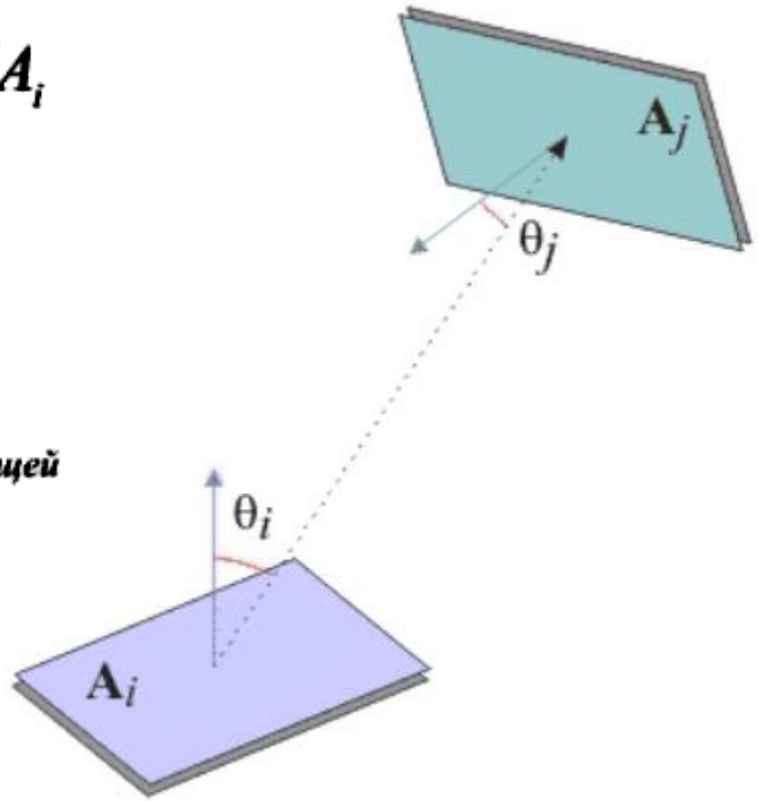
n - количество всех патчей трехмерной сцены.

Вычисление форм-фактора

- Форм-фактор i-го патча определяет, какая часть radiosity j-го патча достигнет i-го патча.

$$F_{ij} = \frac{1}{A_i}\int_{A_i}\int_{A_j}\frac{\cos\theta_i \cos\theta_j}{\pi r^2} HID_{ij}\, dA_j\, dA_i$$

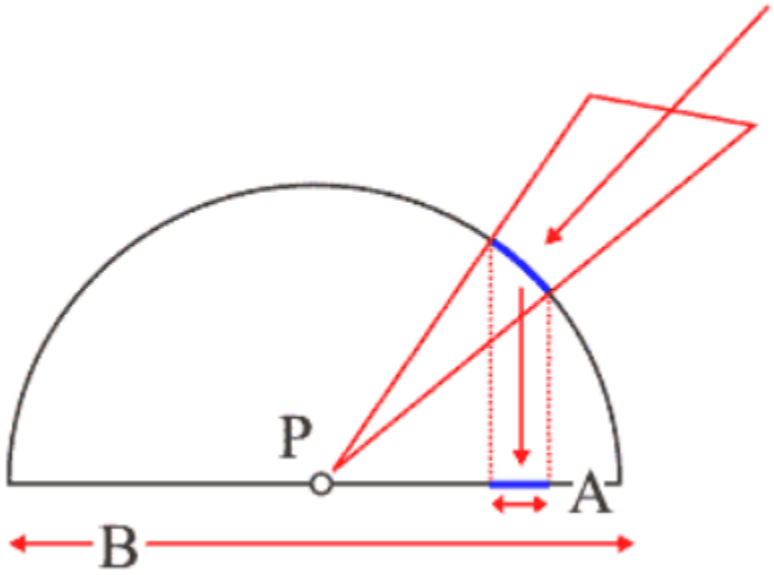
F_{ij} - форм-фактор i-го патча;
A_i , A_j - площади i- го и j-го патчей;
HID_{ij} - функция видимости патчей;
r - расстояние между патчами
θ_i, θ_j - углы между нормалями патчей и линией, их соединяющей



Вычисление форм-фактора

- Формула не учитывает влияния среды, в которой распространяется свет (дым, пыль, туман и т.д.).
- Аналитическое решение этого интеграла в общем случае невозможно, поэтому для его нахождения используются различные численные методы и упрощения.

Расчёт форм-фактора с использованием полусферы

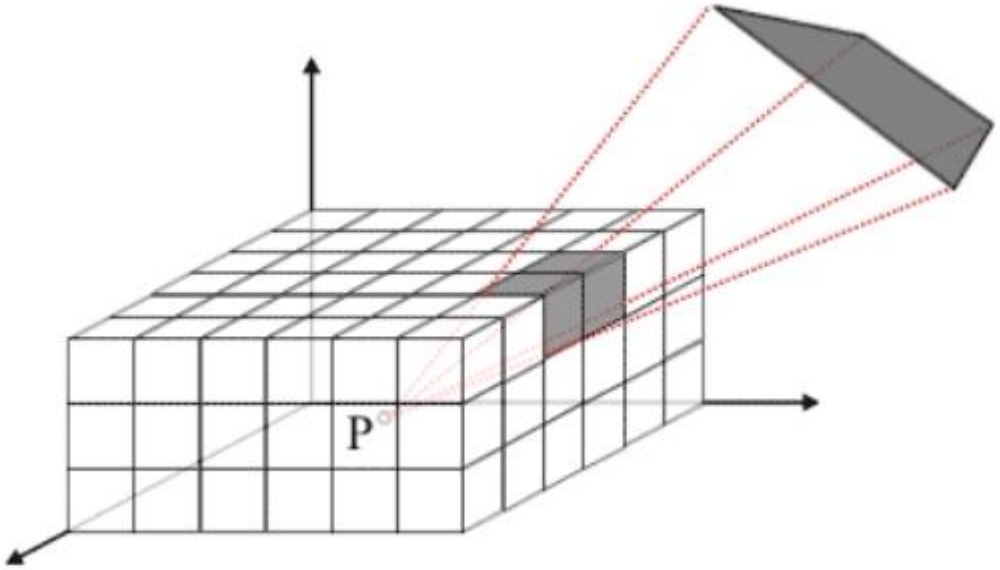


•Считается, что первый патч – точка

•В эту точку помещается полусфера единичного радиуса, второй патч проектируется сначала на поверхность сферы, затем – на её основание.

•Величина форм-фактора тогда будет равна отношению дважды спроектированной площади второго патча к площади основания полусферы.

Расчёт форм-фактора с помощью полукуба



- Патч проектируется на полукуб, вычисляются соответствующие ячейки.
- Форм-фактор патча равен сумме форм-факторов ячеек, на которые «упала тень» патча.

Matrix radiosity: ресурсы

- С математической точки зрения мы здесь имеем **n уравнений** для нахождения **n неизвестных**
- Сцена из 1 миллиона полигонов — патчей
- Система из миллиона уравнений
- Для **каждого** патча должно быть вычислено **n форм-факторов**, определяющих вклады от всех остальных патчей сцены, а для всех **n патчей — n x n форм-факторов**.
- Для сцены из **миллиона патчей** эта цифра составит **10^12**
- Если использовать по **1 байту на фактор**, для хранения всех значений потребуется около **1000 гигабайт**.

Алгоритмы расчета radiosity

- Одним из самых первых алгоритмов стал метод **итераций Гаусса-Зейделя** (**Gauss-Seidel iteration**).
- Второй, и **наиболее распространённый**, метод решения уравнений radiosity носит название метода **прогрессивного уточнения оценок** (**progressive refinement method**).
- Третий, **wavelet radiosity**
- В конце 90-х годов, был разработан ещё один способ решения основных уравнений radiosity, получивший название **Stochastic Relaxation Radiosity**, или **SRR**.

Метод итераций Гаусса-Зейделя

- Пошаговое вычисление radiosity патчей с последующей корректировкой промежуточных значений (называемых оценочными значениями).
- На самом **первом** шаге считается, что **вторичное отражение отсутствует**, тогда **radiosity** всех патчей **равны их собственной эмиссии**.
- На **следующем** шаге вычисленное промежуточное значение **radiosity подставляется в уравнение**, и **находится новое промежуточное** значение, которое будет отличаться по значению в большую сторону.
- Поскольку для реалистичных поверхностей **коэффициент диффузного отражения меньше единицы**, и **любой из форм-факторов** также всегда **меньше единицы**, **добавка** к промежуточному значению радиосити с каждым шагом вычислений **будет все меньше** и меньше.
- На определённом шаге итерации **новое** промежуточное значение **не будет отличаться от предыдущего** вычисленного значения **в пределах заданной точности**. Тогда считается, что искомое значение **radiosity найдено**.
- Хотя этот метод решает часть проблем вычисления значений radiosity, он всё ещё **недостаточно быстр** и требует такого **количества памяти** для хранения переменных, что его **использование для** решения **практических задач почти невозможно**.

Метод прогрессивного уточнения оценок

- Radiosity данного патча рассчитывается как **сумма вкладов radiosity от окружающих** патчей, при этом всякий раз используются все форм-факторы в количестве ***n***x***n***, где ***n*** – количество всех патчей.
- Можно **наоборот** описывать **вклад данного патча в radiosity всех остальных патчей**, т. е. – не собираем свет, а излучаем свет из данного патча в направлении окружающих поверхностей.

$$\Delta \mathbf{B}_i = \rho_i \, \mathbf{B}_1 \, \mathbf{F}_{1i} \, \mathbf{A}_1 / \mathbf{A}_i$$

где $\Delta \mathbf{B}_i$ *- вклад в radiosity i-го патча от первого патча,*

ρ_i *- коэффициент, характеризующий свойства диффузного отражения поверхности, которой принадлежит i-й патч;*

$\mathbf{B}_1$ *- radiosity 1- го патча;*

$\mathbf{F}_{1j}$ *- форм-факторы первого патча в количестве n;*

n *- количество всех патчей трехмерной сцены;*

$\mathbf{A}_1$ *- площадь первого патча;*

$\mathbf{A}_i$ *- площади остальных патчей.*

В практических вычислениях используется следующий алгоритм

1. На первом шаге итерации
   - Считается, что отраженного света пока нет, поэтому radiosity всех патчей определяются **освещённостью прямыми источниками света в соответствии с законом Ламберта + собственное излучение**, если патч принадлежит самосветящейся поверхности.
   - Вычислений форм-факторов тоже пока не требуется.
   - По найденным значениям radiosity производится **упорядочивание патчей в порядке убывания**. Это делается для того, чтобы в последующих итерациях вычисления начинались с патчей, обеспечивающих самые **существенные вклады**. Поскольку речь идет об отражённом свете, это же означает, что такие патчи будут и наиболее сильно освещёнными.

В практических вычислениях используется следующий алгоритм

2. На втором шаге
   - Выбирается первый патч последовательности (наиболее освещённый), и в соответствии с формулой **вычисляются вклады первого патча в radiosity всех остальных патчей**.
   - **Вычисленные вклады суммируются с radiosity** соответствующих патчей и сохраняются **с накоплением в** переменной **неизлученной (unshot) radiosity** соответствующего патча для следующей итерации
   - **Unshot radiosity первого патча обнуляется** после вычисления всех вкладов первого патча – вся "неизлученная" освещенность теперь "излучена".
   - Таким образом, **каждый патч** характеризуется **текущим** значением **radiosity и** переменной **неизлученной radiosity**, представляющей собой сумму вкладов от других патчей, вычисленных в текущей итерации.

В практических вычислениях используется следующий алгоритм

- При вычислении вкладов от второго патча (и последующих) используется накопленная к этому моменту неизлученная radiosity как сумма вкладов от уже обсчитанных патчей (в данном случае – предыдущего, первого патча).
- **После вычислений** всех вкладов переменная **unshot radiosity второго патча тоже обнуляется**, **но её значение будет меняться в процессе итерации** в результате последующих вычислений других патчей.
- Так, значение неизлученной radiosity первого патча, которую мы обнулили раньше, теперь будет равна величине вклада второго патча в радиосити первого патча. **Итерация заканчивается, когда будут вычислены вклады всех патчей**.
- Использование **unshot radiosity** имеет простую **интерпретацию**. Это **свет, излученный патчем** в направлении окружающих поверхностей **и отраженный назад**.
- С каждой итерацией мы снова и снова излучаем этот свет, **интенсивность** его при этом все время **уменьшается**, что и **обеспечивает сходимость**, т.е. конечное число итераций при вычислении полного значения radiosity.

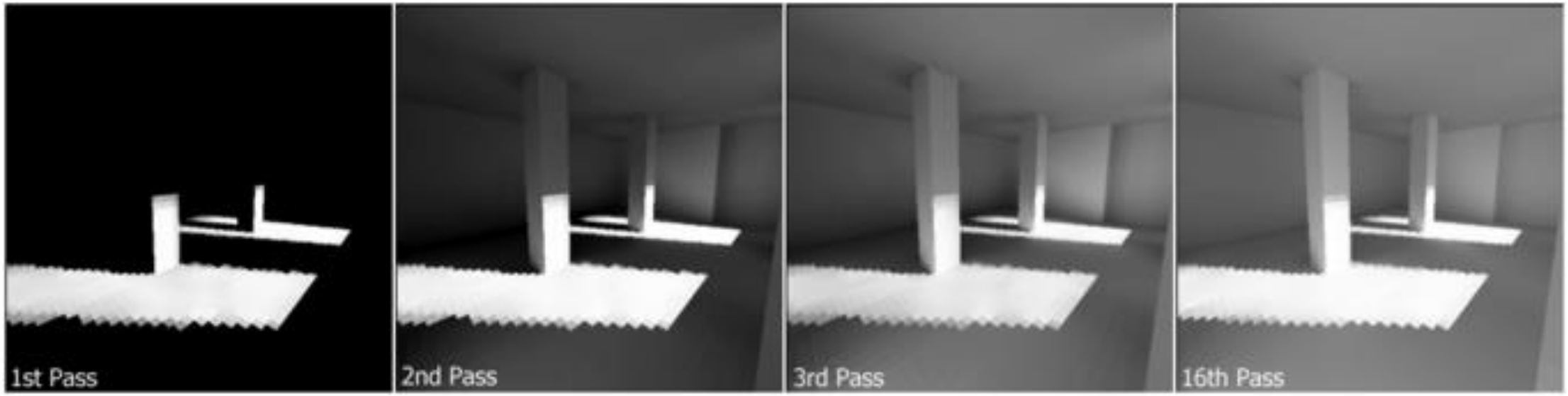
В практических вычислениях используется следующий алгоритм

3. На третьем шаге
   - Итерации выполняются до тех пор, пока значение максимального вклада какого-либо патча не станет меньше некоторой заданной величины.

Пример

- Пусть в нашей сцене имеется всего 3 патча и один источник света E.
- Присваиваем значениям radiosity B(i) и неизлученной энергии U(i) каждого патча величину их диффузной освещённости от источника света E(i).
- Упорядочиваем патчи по величине B, пусть B(1) > B(2) > B(3).
- На второй итерации берём первый патч и используя его значение U(1) находим вклады во второй и третий патчи.
- Вклады суммируем со значением радиосити B(2) и B(3), а также приплюсовываем их к неизлученной энергии U(2) и U(3).
- Затем обнуляем значение U(1) – вклады от неё уже рассчитаны, то есть энергия "излучена".
- Переходим ко второму патчу, используя его значение неизлученной энергии U(2) = E(2) + вклад от первого патча, рассчитываем вклады в первый и третий патчи, обновляем значения радиосити и неизлученной энергии для 1 и 3 патчей, U(2) обнуляем.
- Переходим к третьему патчу, его значение U(3) = E(3) + вклад от первого патча + вклад от второго патча используется для рассчета вкладов в первый и второй патч. После вычисления вкладов U(3) обнуляется. Первая итерация завершена, переходим к следующей и т.д.

Пошаговая визуализация

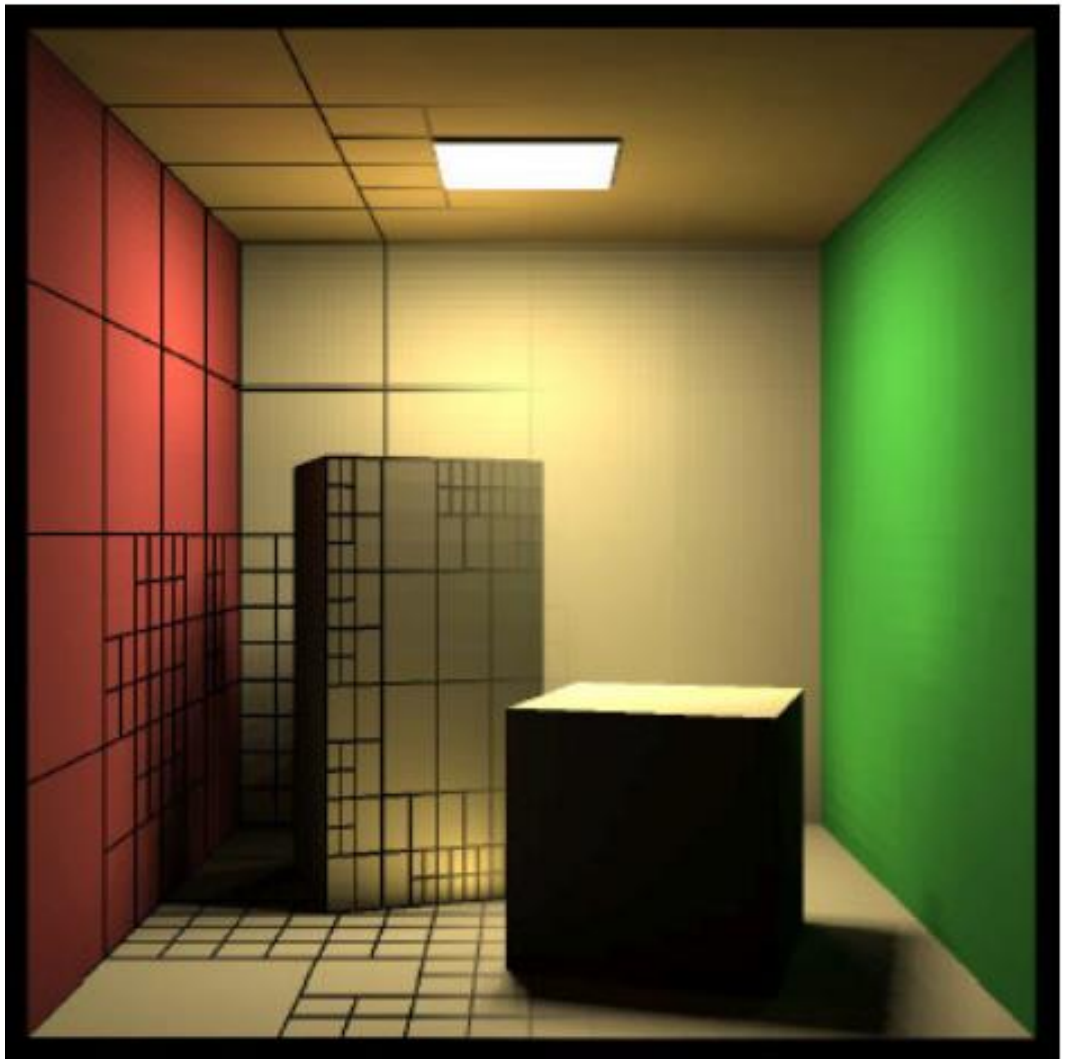


Выигрыш в progressive refinement method (метод прогрессивного уточнения оценок)

- Выигрыш в **скорости расчётов** и **экономия** необходимой для вычислений **памяти** в progressive refinement method оказались настолько велики, что этот метод **впервые** позволил реализовать расчёт radiosity **в практических приложениях** и даже успешно **применяется в приложениях реального времени**.

Progressive refinement method with substructuring

- усовершенствованный **progressive refinement method with substructuring** (**адаптивно разбивает патчи на более мелкие** по площади **в областях с тоновым градиентом** – например, на границах теней).



Wavelet radiosity

- Wavelet метод использует **динамическое представление поверхностей патчами в зависимости от расстояния поверхностей друг от друга**.
- Например, если поверхность А расположена далеко от поверхности Б, то для расчёта вклада radiosity от поверхности А используется грубое представление большими патчами поверхности А.
- Если рассчитывается radiosity поверхности С, расположенной близко к А, то в этом случае А описывается точно, большим количеством патчей.

Практическое использование

- В практических приложениях в основном используется усовершенствованный **progressive refinement method with substructuring** (**адаптивно разбивает патчи на более мелкие** по площади **в областях с тоновым градиентом** – например, на границах теней).

- **progressive refinement** дает лучшие результаты в **простых** сценах, **wavelet radiosity** – в **больших сложных сценах** с множеством объектов. Однако, wavelet radiosity использует гораздо большее количество памяти в вычислениях

Stochastic Relaxation Radiosity, или SRR

- Этот алгоритм расчетов применяется в 3ds max, начиная с шестой версии
- Он основывается на применении метода **Монте-Карло** (М-К) как для решения матричных уравнений radiosity, так и уравнений progressive refinement method.
- В данном случае метод М-К используется **для оценки** величины **суммы вкладов radiosity всех** патчей в данный (или вкладов данного патча во все остальные патчи) **по величине вкладов части из них**.
- Другими словами, этот метод позволяет **вычислять не все вклады, а только часть из них, выбранных по некоторому случайному закону**. Кроме того он же позволяет при вычислениях вкладов **избежать вычислений форм-факторов**.

Общий алгоритм Stochastic Relaxation Radiosity:

- Берем первый патч и на его поверхности некоторым случайным образом выбираем N точек.
- Из каждой точки под некоторым случайным углом испускается луч.
- Луч трассируется до пересечения с ближайшим патчем.
- Вероятность того, что луч пересечет именно этот патч, а не другой, определяется величиной форм-фактора двух соответствующих патчей.
- Именно это обстоятельство позволяет исключить вычисление форм-фактора из вычислений вкладов.

Общий алгоритм Stochastic Relaxation Radiosity:

- Вклад i-го патча, с которым пересёкся луч вычисляется по формуле:

$$B_j = B_j + \frac{1}{N} \rho_j B_i$$

- После расчета вкладов от N случайных патчей, переходим к следующему патчу.
- После перебора всех патчей данная итерация закончена, переходим к следующей, пока величина вкладов не станет меньше заданной величины.

Stochastic Relaxation Radiosity

- Для каждого выбранного патча расчёт вкладов в него от **всех** остальных **патчей заменяется** на расчёт **N вкладов** от **случайно** выбранных патчей, где *N* – число сэмплирующих поверхность патча лучей.
- **В теории** Stochastic Relaxation Radiosity получены математические **формулы**, **определяющие** количество сэмплирующих лучей **N** в зависимости от **требуемой точности** вычисления значений радиосити.
- Значение **N индивидуально для каждого патча сцены** и вычисляется для каждого патча по отдельности, но **один раз для всех итераций**.
- Кроме того, **при вычислении N** требуется рассчитывать **значения форм-факторов** данного патча со всеми остальными, так что совсем без форм-факторов обойтись всё же не удастся.

Stochastic Relaxation Radiosity

- Stochastic Relaxation Radiosity обеспечивает гораздо **более быстрый расчёт радиосити** и требует **меньше памяти** для вычислений, чем все другие рассмотренные методы.

- Есть одно обстоятельство, которое необходимо учитывать при использовании SRR – **случайность** природы расчётов. Это значит, что вполне вероятны ситуации, когда **некоторые мелкие детали** в сцене **могут "выпасть"** из расчетов, если лучи в них не "попали". Для **исправления** такого положения алгоритм расчёта радиосити предусматривает **механизм дополнительного сбора (regathering) излучения** за счёт испускания дополнительных лучей.

Итог — История развития алгоритма

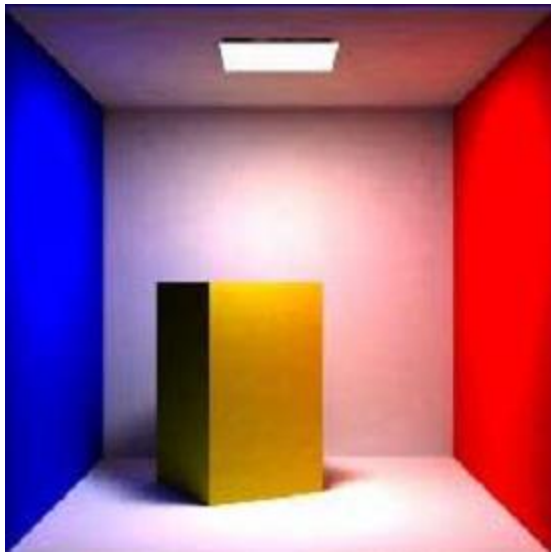
- В ранних версиях radiosity для отображения результата на экране необходимо было полностью просчитать распространение света и освещённость элементов.
- В 1988 разработан метод прогрессивного уточнения: немедленное отображение результата, который со временем уточнялся.
- В 1999 изобретена техника стохастической релаксации излучаемости (Stochastic Relaxation Radiosity).
- Этот алгоритм составляет основу коммерческих radiosity-систем производимых компанией Discreet.

Ray-Tracing vs Radiosity

Ray-Tracing

- Точно рассчитывает **прямое освещение, тени, отражения и эффекты прозрачности**
- **Экономит память**
- Ресурсоёмкий. Время на производство картинки очень зависит от количества источников света в сцене
- Процесс должен повторяться для каждой точки обзора заново
- Не учитывает диффузного переотражения

Ray-Tracing vs Radiosity



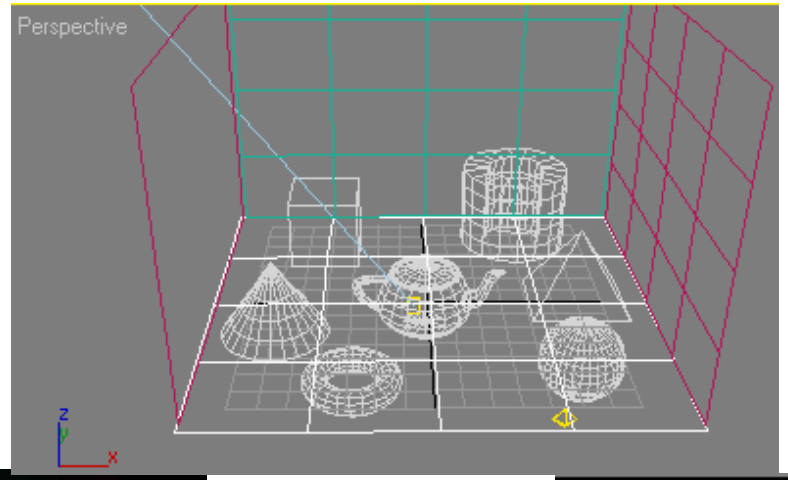
Radiosity

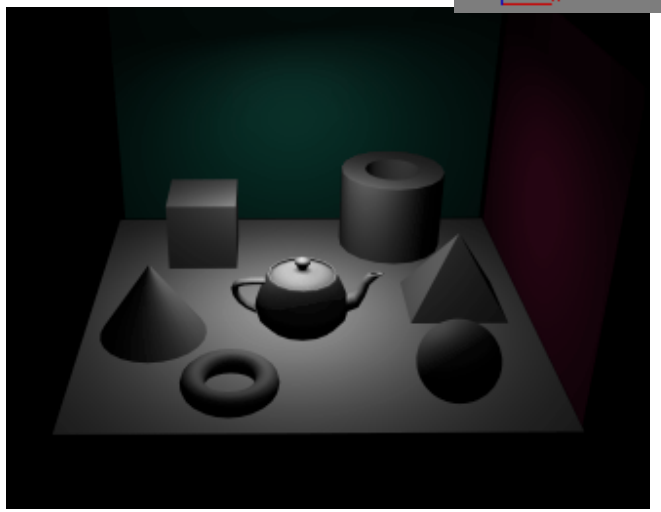
- Рассчитывает диффузное переотражение от поверхностей (**один раз для каждого нового положения источников света**)
- Производит независимые решения, для быстрой визуализации из **любой точки обзора**
- Предлагает непосредственные визуальные эффекты
- 3D-сетка требует больше памяти, чем оригинальные поверхности
- Алгоритм дискретизации поверхности более восприимчив к артефактам, чем ray tracing
- Не работает с эффектами отражения или прозрачности

3ds max

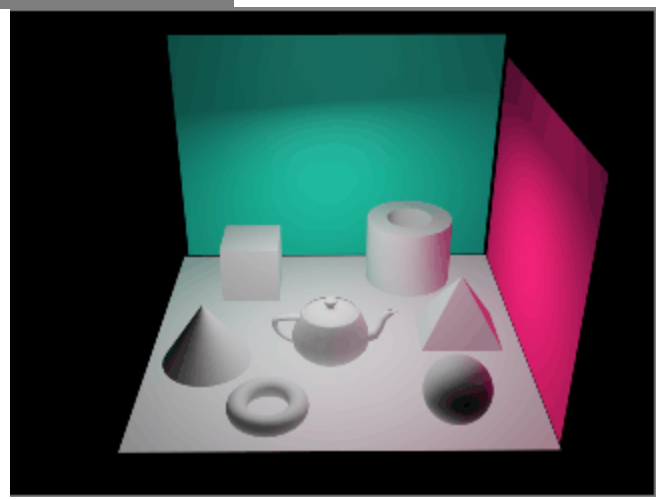
- В 3ds max для имитации **непрямого освещения** используется два алгоритма:
  - **Radiosity** (Излучательность)
  - **Light Trace** (Трассировка света)

3ds max – визуализация



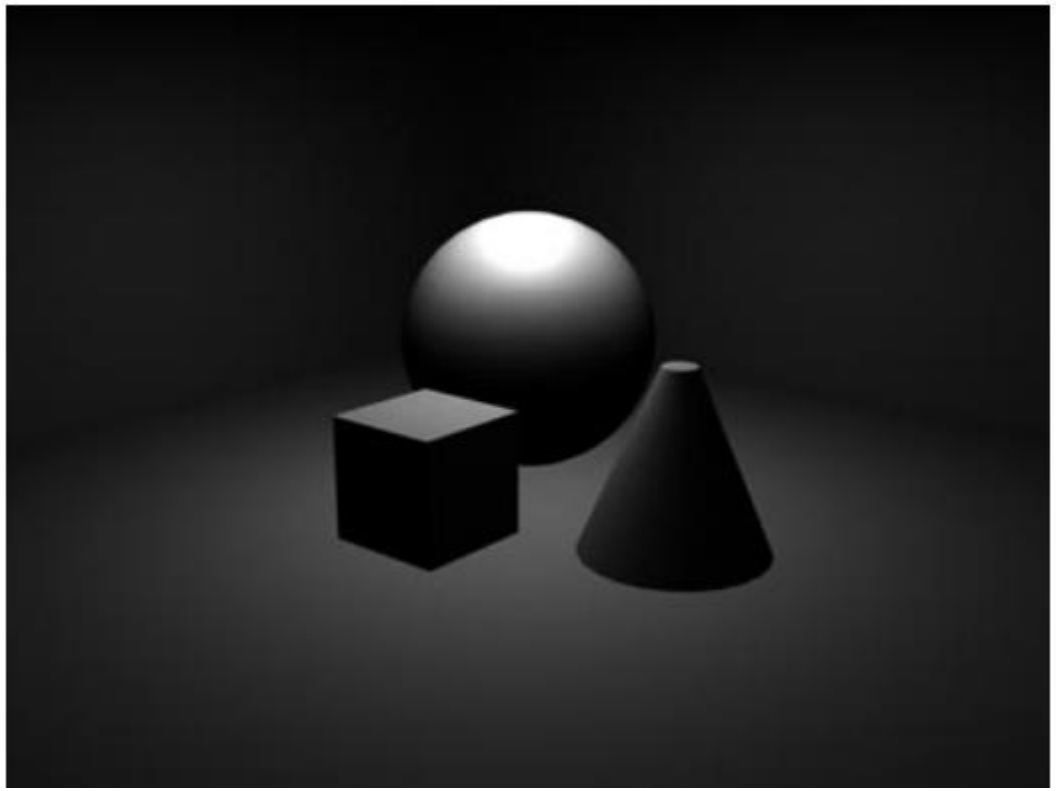


Визуализация сцены без использования Radiosity

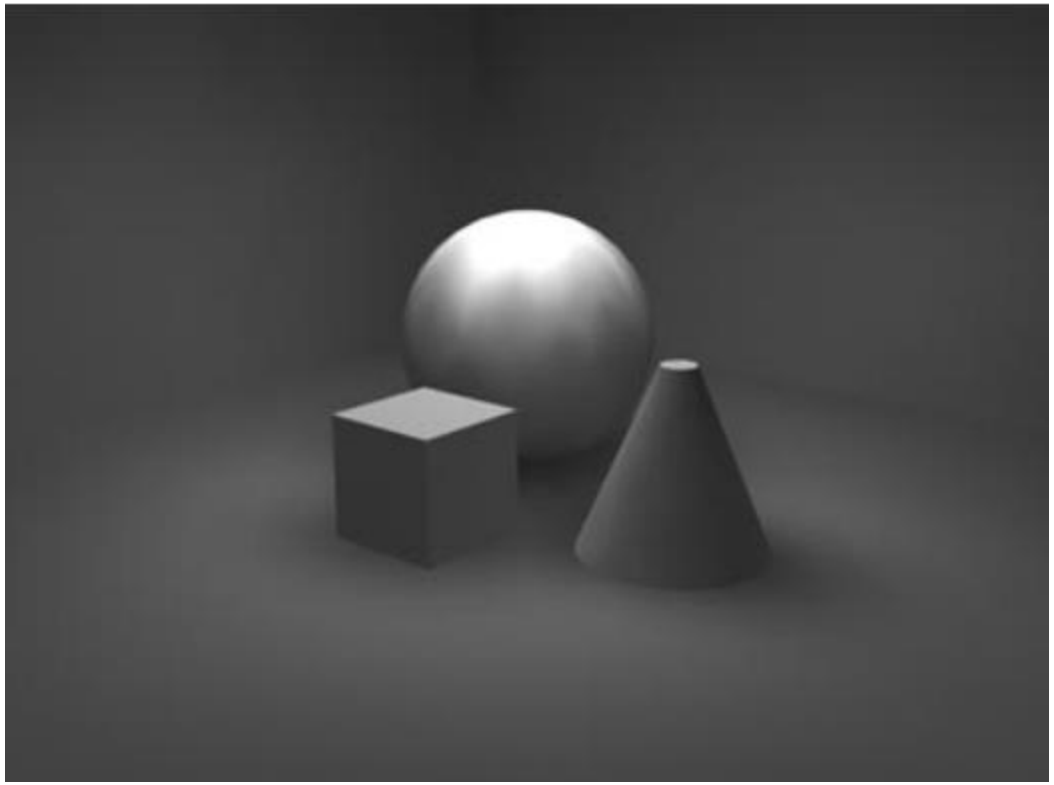


Результат визуализации сцены с учетом Radiosity

Только прямое освещение. Скан-лайн рендер 3ds max 6.



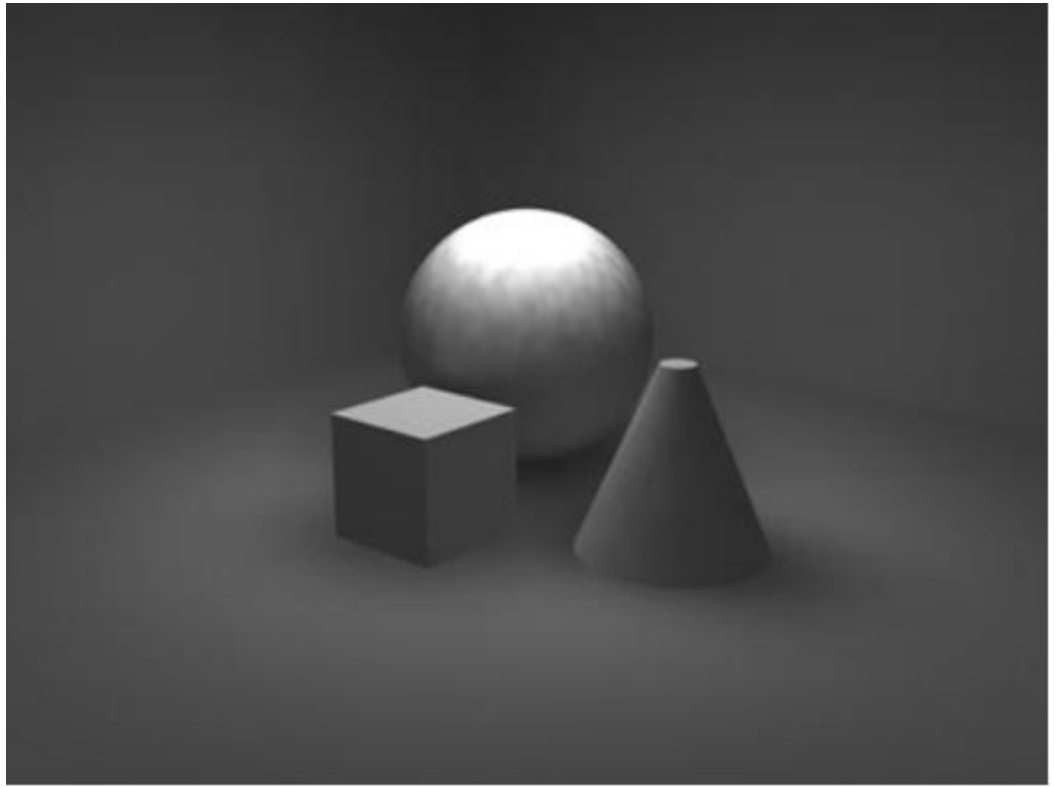
Radiosity рендер 3ds max 6



Только radiosity, без прямого освещения от источников света.

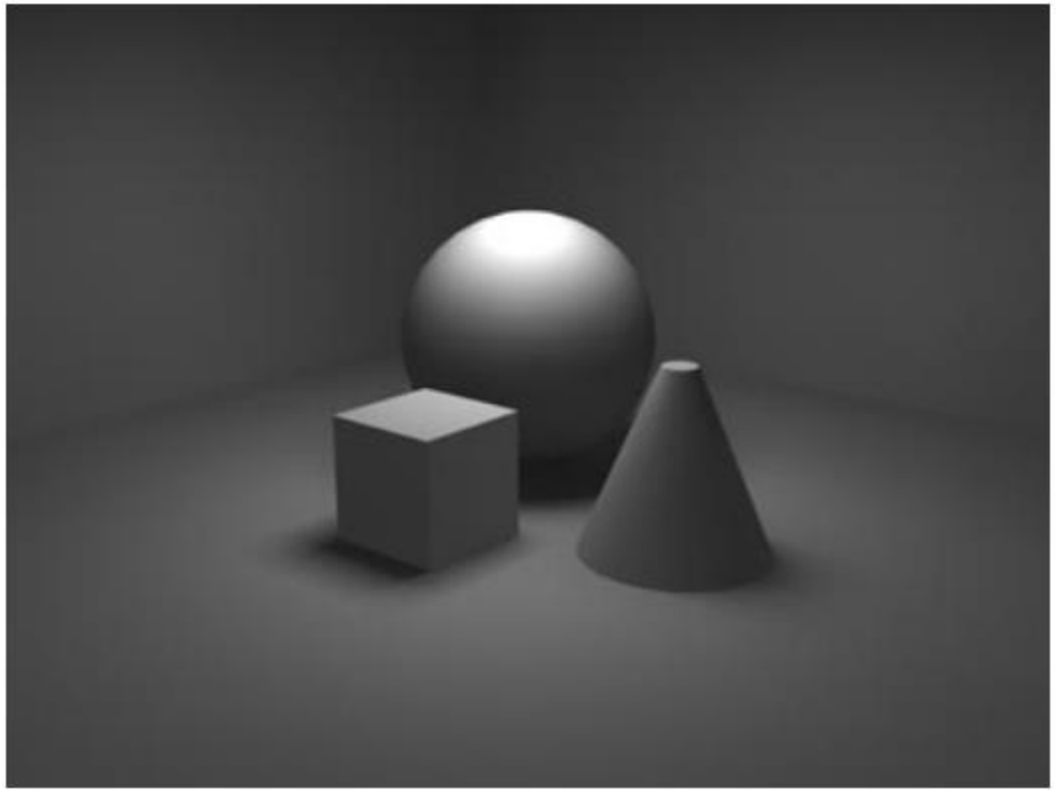
Хорошо заметен шум и полигонная структура сферы – результат использования модели Гуро для сглаживания значений radiosity для разных патчей (полигонов).

Radiosity рендер 3ds max 6



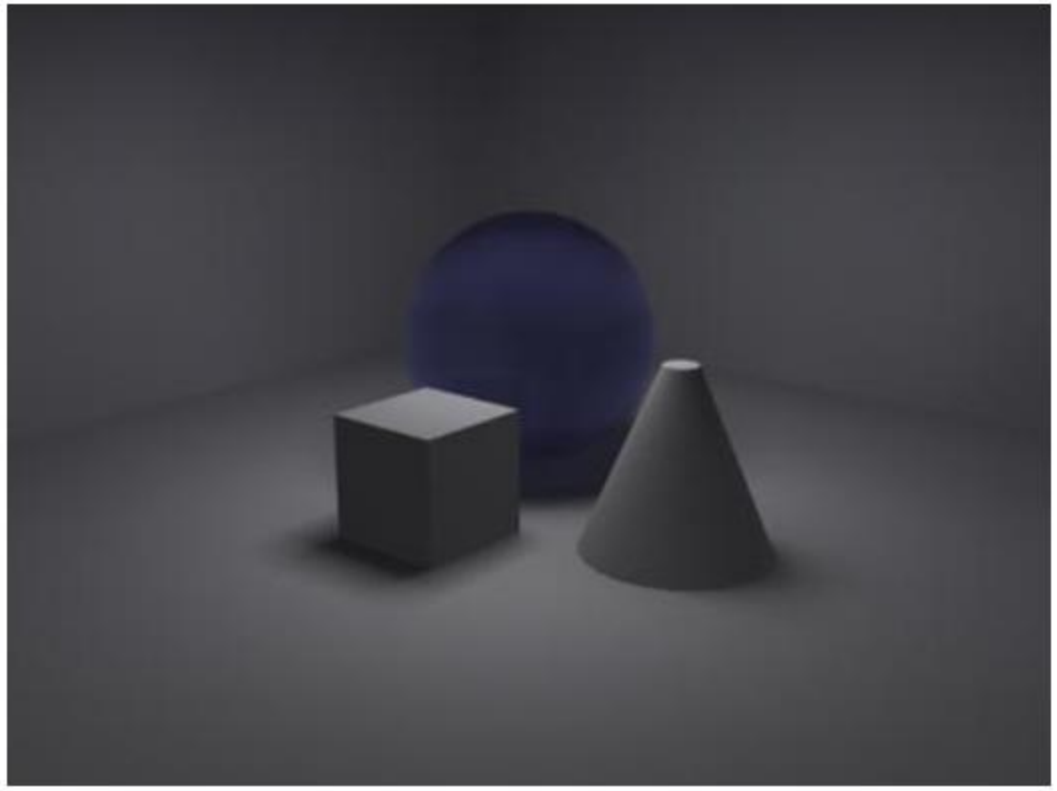
Увеличение точности решения radiosity и количества патчей не устранили шум на сфере.
На кубе и конусе шум отсутствует.

3ds max 6



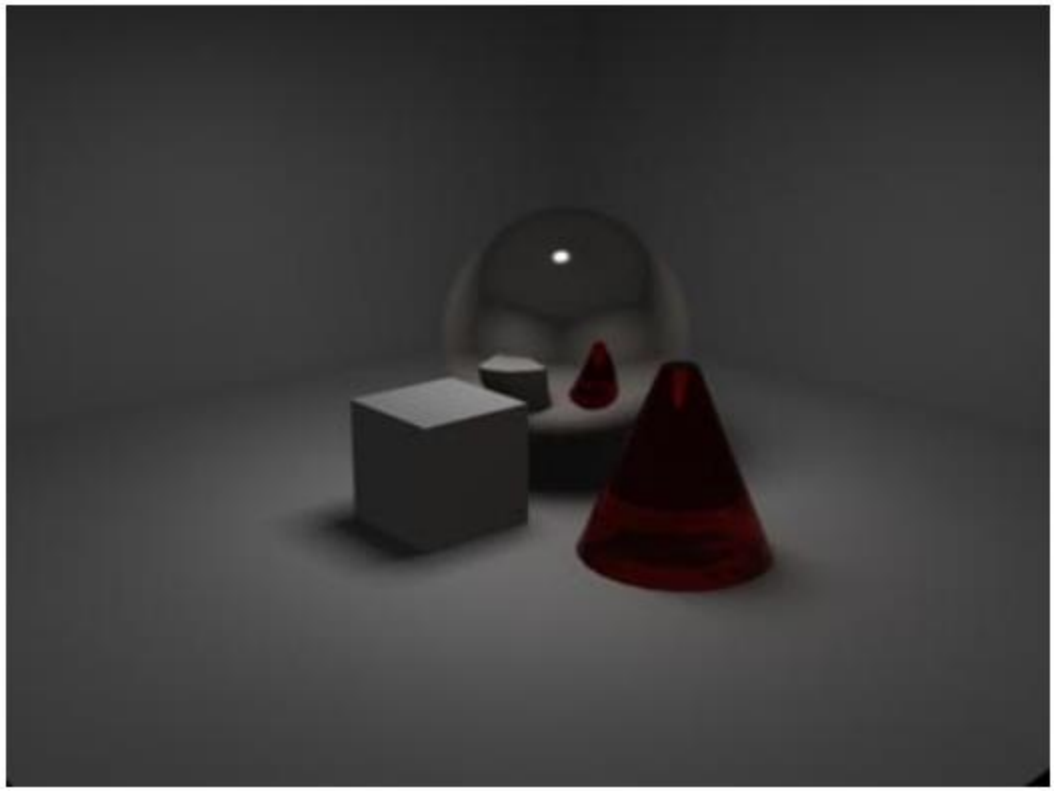
Совместная работа скан-лайн рендера, рейтресера (тени) и radiosity 3ds max 6. Шум не наблюдается, поскольку значения radiosity используются как добавка к расчету прямого освещения, использующего модель Фонга.

3ds max 6



Сфере назначен **прозрачный** материал.
Пример **совместной работы radiosity и рейтресинга 3ds max 6**.
Хорошо заметен color bleeding – голубоватый оттенок на окружающих сферу объектах

3ds max 6



Сфере назначен **отражающий** материал и **прозрачный** материал – **конусу**.
Сканлайн рендер, рейтресинг, радиосити 3ds max 6.

Cyberpunk 2077, A Plague Tale: Requiem и Microsoft Flight Simulator с улучшенной трассировкой лучей

https://shazoo.ru/2022/09/21/132646/cyberpunk-2077-a-plague-tale-requiem-i-microsoft-flight-simulator-s-ulucsennoi-trassirovkoi-lucei