Министерство науки и высшего образования Российской Федерации
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

А. Г. ЗАДОРОЖНЫЙ

КОМПЬЮТЕРНАЯ ГРАФИКА
ВВЕДЕНИЕ В ТРАССИРОВКУ ЛУЧЕЙ

Утверждено
Редакционно-издательским советом университета
в качестве учебного пособия

НОВОСИБИРСК
2021

УДК 004.925.3(075.8)
З-156

Рецензенты:
канд. техн. наук, доцент *В. С. Карманов*
д-р техн. наук, профессор *М. Э. Рояк*

Работа подготовлена на кафедре прикладной математики НГТУ

Задорожный А. Г.
З-156 Компьютерная графика: введение в трассировку лучей: учебное пособие / А. Г. Задорожный. – Новосибирск: Изд-во НГТУ, 2021. – 64 с.

ISBN 978-5-7782-4561-7

В данном учебном пособии рассмотрены элементы теории трассировки лучей и, соответственно, глобального освещения в компьютерной графике. Пособие может быть рекомендовано как для самостоятельного изучения курса «Компьютерная графика», так и для подготовки к практическим и расчетно-графическим заданиям.

УДК 004.925.3(075.8)

ISBN 978-5-7782-4561-7

ВВЕДЕНИЕ

С точки зрения физики, видимость любого объекта является результатом отражения от его поверхности множества лучей света, которые прошли к этому моменту долгий путь поглощений, отражений и преломлений (вследствие чего реалистичное моделирование цвета точки поверхности и является сверхсложной задачей).

Таким образом, при компьютерном моделировании сцен с источниками света возникает необходимость в алгоритме моделирования отражения света от поверхности объектов – *алгоритм переноса света* **(**light transport algorithm**)**. Таких алгоритмов разработано множество в зависимости от учета свойств поверхности и соблюдения баланса между скоростью рендеринга и качеством получаемого изображения.

Другой важный момент при моделировании – это расчет траектории светового луча с учетом возможности его столкновения с различными поверхностями. Обычная растеризация ограничивается в этом случае лишь рассмотрением ситуации первого столкновения луча с поверхностью (в лучшем случае – с учетом первого отражения). Но для качественного решения данной задачи требуется уже тяжеловесные методы глобального освещения, которые позволяют более полно учитывать траекторию лучей при многочисленных взаимодействиях с различными поверхностями.

Как исторически, так и практически, основным методом глобального освещения является метод трассировки лучей.

БАЗОВЫЕ ГЕОМЕТРИЧЕСКИЕ ОБЪЕКТЫ

В трассировке лучей используются различные геометрические объекты, задаваемые как уравнениями, так и набором полигонов (полигональной сеткой). Тип и количество таких объектов зависит от сложности сцены и требуемого качества выходного изображения.

В качестве основной системы координат применяется трехмерная декартова, а в качестве локальной – система координат, лучше всего подходящая для модели конкретного объекта и операций с ним.

К основным (базовым) геометрическим объектам, используемым в трассировке лучей, относятся:

1) Двумерные объекты:
   - треугольник и квадрат,
   - круг и окружность;
2) Трехмерные объекты:
   - прямая, касательная, луч и нормаль,
   - плоскость,
   - сфера и эллипсоид
   - конический цилиндр (обобщение конуса и цилиндра),
   - куб.

В зависимости от ситуации (набора входных данных, удобства применения) уравнения прямой, плоскости и прочих геометрических объектов может быть представлены в различных формах записи:

- явной и неявной,
- параметрической,
- канонической (базовой),
- векторной.

ПРЯМАЯ

Прямая линия является алгебраической кривой первого порядка и в декартовой системе координат задается линейным уравнением двух (трех) переменных.

Прямую в пространстве можно рассматривать, как линию пересечения двух непараллельных плоскостей $a_1 \cdot x + b_1 \cdot y + c_1 \cdot z + d_1$ и $a_2 \cdot x + b_2 \cdot y + c_2 \cdot z + d_2$, система уравнений которых и дает *общие уравнения прямой в пространстве*:

$$\begin{cases} a_1 \cdot x + b_1 \cdot y + c_1 \cdot z + g_1 = 0 \\ a_2 \cdot x + b_2 \cdot y + c_2 \cdot z + g_2 = 0 \end{cases}.$$

В этом случае направляющий вектор $\vec{d} = \left(d_x, d_y, d_z\right)$ можно рассчитать через векторное произведение нормалей обеих плоскостей:

$$\vec{d} = \vec{d}(x, y, z) = \vec{n}_1 \times \vec{n}_2 = \begin{vmatrix} \vec{e}_x & \vec{e}_y & \vec{e}_z \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{vmatrix},$$

соответственно, *нормальным вектором* к прямой является любой вектор, перпендикулярный к направляющему, в частности, и сами нормали данных плоскостей.

Поскольку получение уравнения прямой в явной или неявной форме может быть затруднительно, используется *параметрическое представление в векторной форме*:

$$\vec{r} = \vec{r}_0 + t \cdot \vec{d},$$

где:

$\vec{r}$ – радиус-вектор любой точки (x, y, z), принадлежащей прямой,

$\vec{r}_0 = (x_0, y_0, z_0)$ – радиус-вектор точки (x_0, y_0, z_0) прямой,

$\vec{d} = \left(d_x, d_y, d_z\right)$ – направление прямой,

$t \in (-\infty, \infty)$ – вещественный параметр.

ЛУЧ

Лучом называется полупрямая, т.е. множество всех точек прямой, лежащих по одну сторону (в одном направлении) от заданной ее точки, называемой *началом луча* (начальной точкой).

Из векторного параметрического представления прямой легко получается уравнение луча: в качестве точки $\vec{r}_0$ нужно выбрать начальную точку луча, соответственно параметр t будет меняться в диапазоне $[0, \infty)$:

$$\vec{r} = \vec{r}(t) = \vec{r}_0 + t \cdot \vec{d},$$

где:

$\vec{r}_0 = (x_0,\ y_0,\ z_0)$ – радиус-вектор начальной точки луча,

$\vec{r} = (x,\ y,\ z)$ – радиус-вектор любой точки луча,

$\vec{d} = (d_x,\ d_y,\ d_z)$ – *направляющий вектор* луча,

$t \in [0, \infty)$ – вещественный параметр.

Если параметр t интерпретировать как время, тогда в начальный момент времени $t = 0$ луч находится в своей начальной точке (x_0, y_0, z_0) и с течением времени (с увеличением параметра t) удаляется от нее по направлению $\vec{d}$. Таким образом, при положительных значениях t точки полупрямой расположены спереди (по ходу движения луча). В связи с этим, отрицательные значения t (как и соответствующие точки луча) интереса не представляют.

Параметрическое представление луча также можно записать в векторной форме через единичные орты $\vec{e}_x$, $\vec{e}_y$ и $\vec{e}_z$ декартового пространства:

$$\vec{r} = \vec{r}(t) = x(t) \cdot \vec{e}_x + y(t) \cdot \vec{e}_y + z(t) \cdot \vec{e}_z.$$

НОРМАЛЬ

Нормаль – это обобщение понятия перпендикуляра к плоскости на произвольные поверхности.

В каждой точке поверхности можно провести множество касательных, которые и формируют *касательную плоскость*.

Нормаль к поверхности в заданной точке – это прямая, перпендикулярная касательной плоскости в данной точке поверхности. Нормаль для гладкой поверхности определяется однозначно (рис.1).

Вектор нормали – это вектор единичной длины, параллельный нормали, используется при расчете освещения и определения лицевых граней. По определению, вектор нормали $\vec{n}$ в произвольной точке поверхности, описываемой неявной функцией $F(x, y, z)$, является градиентом данной функции:

$$\vec{n} = grad(F) = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}\right).$$

Замечание. Почти всегда по формулам для расчета вектора нормали получается вектор не единичной длины, необходимо его нормировать.

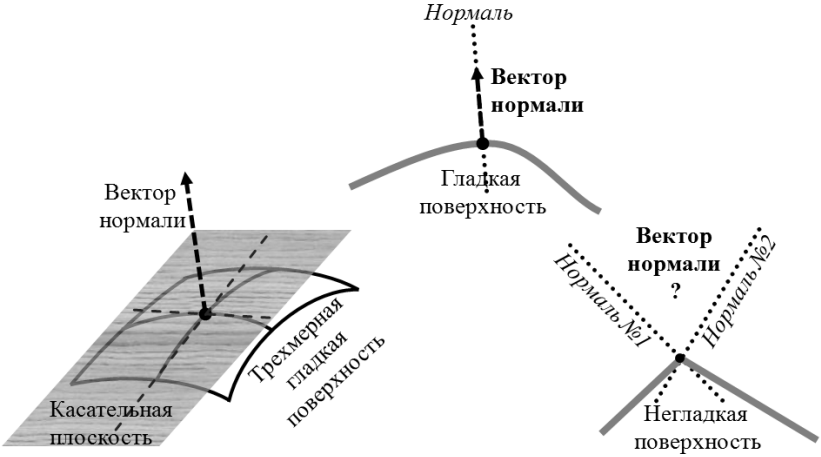


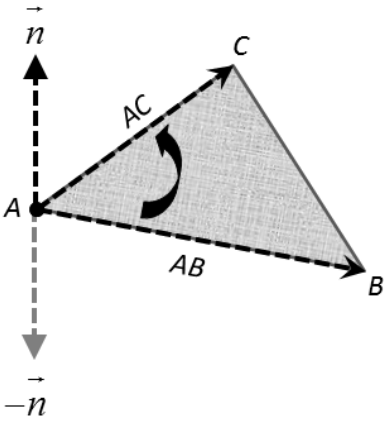
Рис.1. Нормаль к гладкой и негладкой поверхности.

ДЛЯ ПЛОСКОГО ПОЛИГОНА

Для плоских граней (полигонов) нормаль одинакова в любой точке и обычно рассчитывается как векторное произведение двух смежных (для выбранной вершины) неколлинеарных ребер $\vec{u}$ и $\vec{v}$ примитива:

$$\vec{n} = [\vec{u}, \vec{v}] = \vec{u} \times \vec{v} = \begin{vmatrix} i & j & k \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{vmatrix} = \begin{pmatrix} u_y \cdot v_z - u_z \cdot v_y \\ u_z \cdot v_x - u_x \cdot v_z \\ u_x \cdot v_y - u_y \cdot v_x \end{pmatrix}.$$

На рис.2 представлен пример расчета нормали путем векторного произведения в вершине A для невырожденного треугольника ABC, вершины которого заданы против часовой стрелки. Как можно видеть, направление нормали зависит от направления перебора смежных вершин (ребер). Если ребро $\overrightarrow{AB}$ векторно умножить на ребро $\overrightarrow{AC}$, то по правилу буравчика нормаль будет направлена вверх ("наружу"), соответственно, такая грань будет *лицевой* (обход против часовой стрелки). Но если ребро $\overrightarrow{AC}$ векторно умножить на $\overrightarrow{AB}$, то нормаль окажется направленной вниз, такая грань уже становится *нелицевой*.



По правилу правой руки:

$$\vec{n} = \overrightarrow{(B-A)} \times \overrightarrow{(C-A)}$$

По правилу левой руки:

$$-\vec{n} = \overrightarrow{(C-A)} \times \overrightarrow{(B-A)}$$

Рис.2. Нормаль для треугольника.

ДЛЯ НЕСКОЛЬКИХ ПЛОСКИХ ПОЛИГОНОВ

Если все грани (полигоны) лежат в одной плоскости, то и нормали у них направлены одинаково (рис.3), причем, все нормали будут совпадать с нормалью к плоскости после нормирования.

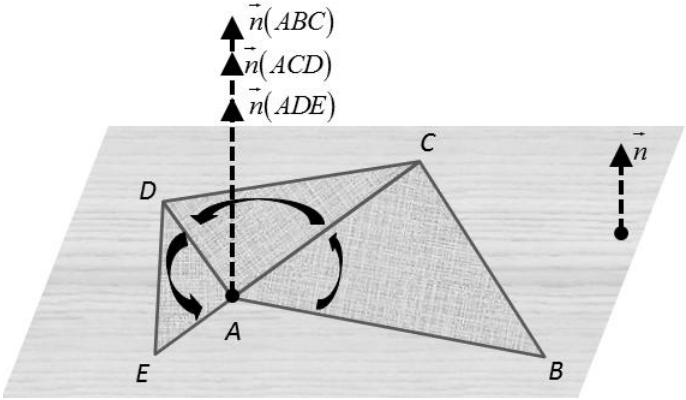


Рис.3. Нормали для треугольников.

Если требуется посчитать единую нормаль в вершине, общей для нескольких нележащих в одной плоскости граней (рис.4), рекомендуется в этой точке рассчитать вектор нормали как векторную сумму нормалей от этих граней. В этом случае стык граней будет сглажен, что больше соответствует реальности.

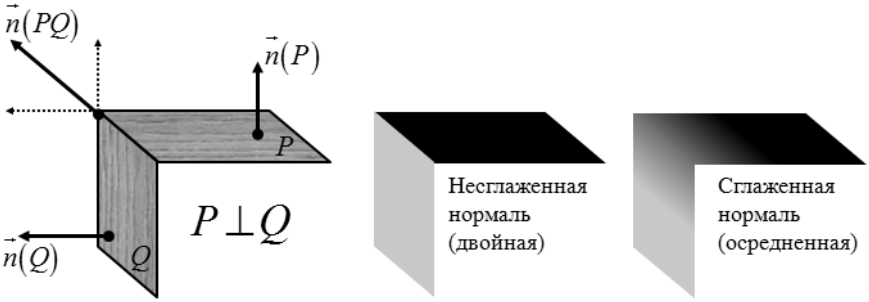


Рис.4. Нормали для треугольников в разных плоскостях.

ДЛЯ НЕПЛОСКОГО ПОЛИГОНА

Когда полигон не является плоскими (например, четырехугольник QUAD), для расчета усредненной нормали рекомендуется использовать устойчивый *алгоритм Мартина Ньюэлла* (Martin Newell):

$$n_x = \sum_{i=1}^{m} (y_i - y_{i+1}) \cdot (z_i + z_{i+1}),$$

$$n_y = \sum_{i=1}^{m} (z_i - z_{i+1}) \cdot (x_i + x_{i+1}),$$

$$n_z = \sum_{i=1}^{m} (x_i - x_{i+1}) \cdot (y_i + y_{i+1}),$$

где:

m – количество вершин полигона с координатами (x_i, y_i, z_i),

$(x_{m+1}, y_{m+1}, z_{m+1})$ – координаты первой вершины полигона.

На рис.5 представлен четырехугольник, у которого одна из вершин приподнята на h. При $h > 0$ данный полигон становится вогнутым, все компоненты исходной нормали $\vec{n} = (0,0,1)$ становятся ненулевыми, а ее направление эквивалентно векторной сумме нормалей двух треугольников, на которые можно разбить четырехугольник.

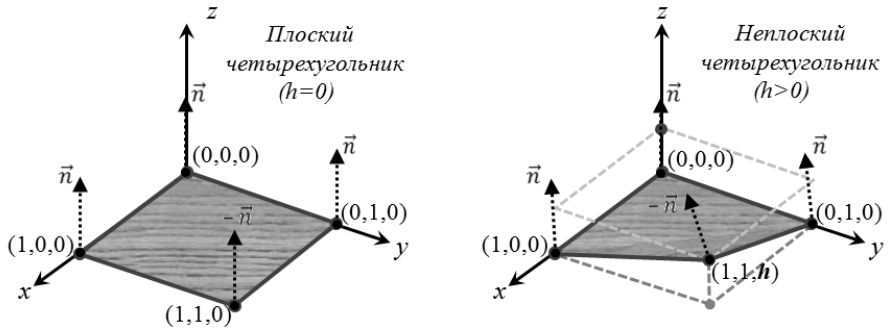


Рис.5. Плоский и неплоский полигоны.

ПЛОСКОСТЬ

Плоскость является алгебраической поверхностью первого порядка, в декартовой системе координат задается линейным уравнением трех переменных. Существуют разные формы записи уравнения плоскости, но в основном используется неявная форма.

Общее (полное) уравнение плоскости (A, B, C, D) имеет вид:

$$A \cdot x + B \cdot y + C \cdot z + D = 0,$$

где A, B, C, D – вещественные константы, причем A, B, C не должны одновременно равняться нулю.

Вектор нормали $\vec{n}$ одинаков для всех точек плоскости и в общем случае не является единичным:

$$\vec{n} = \overrightarrow{(A, B, C)}.$$

НОРМАЛЬНОЕ УРАВНЕНИЕ ПЛОСКОСТИ

Нормальное уравнение плоскости соответствует плоскости с единичной нормалью и получается из общего уравнения путем умножения коэффициентов плоскости на нормирующий множитель

$$k = \frac{1}{\pm\sqrt{A^2 + B^2 + C^2}}$$

и обычно выписывается в виде

$$cos(\alpha) \cdot x + cos(\beta) \cdot y + cos(\gamma) \cdot z = p,$$

где:

α – угол между нормалью и осью OX,
β – угол между нормалью и осью OY,
γ – угол между нормалью и осью OZ,
$p \geq 0$ – вещественное число.

УРАВНЕНИЕ ПЛОСКОСТИ ЧЕРЕЗ 1 ТОЧКУ

Уравнение плоскости, проходящей через точку $r_0 = (x_0, y_0, z_0)$:

$$A \cdot (x - x_0) + B \cdot (y - y_0) + C \cdot (z - z_0) = 0.$$

В векторной форме это уравнение имеет вид:

$$(\vec{r},\ \vec{n}) + D = 0,$$

где $\vec{r}$ – радиус-вектор любой точки (x, y, z), принадлежащей плоскости. Соответственно, точка с радиус-вектором $\vec{r}$ принадлежит плоскости тогда и только тогда, когда

$$\left(\overrightarrow{r - r_0},\ \vec{n}\right) = 0.$$

УРАВНЕНИЕ ПЛОСКОСТИ ЧЕРЕЗ 3 ТОЧКИ

Уравнение плоскости, проходящей через три нележащие на одной прямой точки (x_1, y_1, z_1), (x_2, y_2, z_2) и (x_3, y_3, z_3):

$$\begin{vmatrix} x - x_1 & y - y_1 & z - z_1 \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix} = 0.$$

Векторное уравнение плоскости, проходящей через точку $\vec{r}_0$ и параллельную двум непараллельным векторам $\vec{v}_1$ и $\vec{v}_2$:

$$\vec{r} = \vec{r}_0 + t_1 \cdot \vec{v}_1 + t_2 \cdot \vec{v}_2.$$

УРАВНЕНИЕ БАЗОВОЙ ПЛОСКОСТИ

Базовой плоскостью является плоскость XY, соответствующее обще уравнение плоскости предельно упрощается:

$$F(x, y, z) = z = 0.$$

В этом случае вектор нормали направлен по оси OZ:

$$\vec{n} = (0, 0, 1).$$

СФЕРА

Неявное уравнение для сферы радиусом R и центром в точке $C = (x_0, y_0, z_0)$ имеет следующий вид:

$$F(x, y, z) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - R^2 = 0.$$

Нормаль (в общем случае неединичная) в любой точке $P(x, y, z)$ на поверхности сферы может быть рассчитана следующим образом:

$$\vec{n} = \overrightarrow{P - C}.$$

ПАРАМЕТРИЧЕСКАЯ ФОРМА

В параметрической форме координаты любой точки $P(x, y, z)$ сферы можно записать в виде следующей системы:

$$\begin{cases} x = x_0 + R \cdot sin(\theta) \cdot cos(\varphi) \\ y = y_0 + R \cdot sin(\theta) \cdot sin(\varphi), \\ z = z_0 + R \cdot cos(\theta) \end{cases}$$

где:

$\theta \in [0,\ \pi]$ – угол "вниз" относительно плоскости $z = R$,

$\varphi \in [0,\ 2 \cdot \pi]$ – угол в плоскости XY относительно оси Y.

БАЗОВАЯ ФОРМА

Уравнение базовой (канонической) сферы (единичного радиуса и расположенной в центре системы координат) имеет вид:

$$F(x, y, z) = x^2 + y^2 + z^2 - 1 = 0,$$

причем нормаль в любой точке $P(x, y, z)$ на поверхности сферы совпадет с направлением радиус-вектора этой точки.

ЭЛЛИПСОИД

Неявное уравнение для эллипсоида "радиусом" R и центром в точке $C = (x_0, y_0, z_0)$ имеет следующий вид:

$$F\big(P(x,y,z)\big) = \left(\frac{x - x_0}{a}\right)^2 + \left(\frac{y - y_0}{b}\right)^2 + \left(\frac{z - z_0}{c}\right)^2 - R^2 = 0.$$

Нормаль (в общем случае неединичная) в любой точке $P(x,y,z)$ на поверхности эллипсоида может быть рассчитана следующим образом:

$$\vec{n} = \left(\frac{P_x - x_0}{a^2},\ \frac{P_y - y_0}{b^2},\ \frac{P_z - z_0}{c^2}\right).$$

ПАРАМЕТРИЧЕСКАЯ ФОРМА

В параметрической форме координаты любой точки $P(x,y,z)$ эллипсоида можно записать виде следующей системы:

$$\begin{cases} x = x_0 + a \cdot R \cdot sin(\theta) \cdot cos(\varphi) \\ y = y_0 + b \cdot R \cdot sin(\theta) \cdot sin(\varphi), \\ z = z_0 + c \cdot R \cdot cos(\theta) \end{cases}$$

где:

$\theta \in [0,\ \pi]$ – угол относительно плоскости $z = 0$,

$\varphi \in [0,\ 2 \cdot \pi]$ – угол в плоскости $z = 0$.

БАЗОВАЯ ФОРМА

Уравнение базового (*канонического*) эллипсоида (единичного радиуса и расположенного в центре системы координат) имеет вид:

$$F\big(P(x,y,z)\big) = \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0.$$

КОНИЧЕСКИЙ ЦИЛИНДР (УСЕЧЕННЫЙ КОНУС)

Базовый конический цилиндр (усеченный конус) состоит из трех частей: верхнего и нижнего круглых оснований и боковой поверхности. Очевидно, что при $R = 0$ и $R = 1$ конический цилиндр вырождается, соответственно, в **конус** и **цилиндр** (рис.6).

Переход от этой базовой фигуры к произвольной осуществляется стандартными модельными преобразованиями (подробно они рассмотрены в [2]):

1. Сдвиг объекта в точку $C = (x_0, y_0, z_0)$, которая станет новым центром нижнего основания.
2. Масштабирование по осям (сплющивание или растяжение) с коэффициентами (k_x, k_y, k_z).
3. Поворот по трем осям на углы $(\theta_x, \theta_y, \theta_z)$ или поворот вокруг вектора $\vec{v}$ на угол φ.

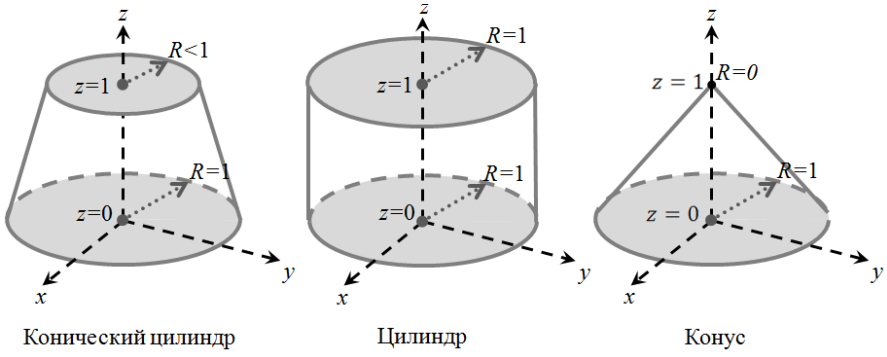


Рис.6. Частные случаи конического цилиндра.

НИЖНЕЕ ОСНОВАНИЕ

Нижнее основание, расположенное на высоте $z = 0$ и являющееся кругом радиуса 1, описывается уравнением

$$F(x, y, z) = x^2 + y^2 \leq 1.$$

Нормаль для любой точки направлена вниз (перпендикулярна плоскости XY):

$$\vec{n} = (0,\ 0,\ -1).$$

ВЕРХНЕЕ ОСНОВАНИЕ

Верхнее основание, расположенное на высоте $z = 1$ и являющееся кругом радиуса R, описывается уравнением

$$F(x, y, z) = x^2 + y^2 \leq R^2.$$

Нормаль для любой точки направлена вверх (перпендикулярна плоскости XY):

$$\vec{n} = (0,\ 0,\ 1).$$

БОКОВАЯ ПОВЕРХНОСТЬ

Боковая поверхность является поверхностью второго порядка и описывается следующим уравнением:

$$F(P) = x^2 + y^2 - (1 + q \cdot z)^2 = 0,$$

где $q = (R - 1)$.

Нормаль в точке $P = (P_x,\ P_y,\ P_z)$ направлена вбок под углом $\varphi \in [0{,}45°]$ к плоскости XY:

$$\vec{n} = (P_x,\ P_y,\ -q - q^2 \cdot P_z).$$

Поскольку у цилиндра боковая поверхность вертикальна ($R = 1$), в этом случае нормаль становится перпендикулярной оси OZ:

$$\vec{n} = (P_x,\ P_y,\ 0).$$

ПОИСК ПЕРЕСЕЧЕНИЯ ЛУЧА С ОБЪЕКТАМИ

Большая часть вычислительного времени при трассировке лучей тратится на проверку пересечений (столкновений) лучей зрения с объектами сцены. При работе алгоритмов проверки пересечений основными факторами являются скорость и точность.

Проблема с точностью заключается не только в вычислительной погрешности, но и в "правильности" построения уравнения луча, когда небольшое изменение направления может привести к другому объекту.

Проблема скорости заключается не сколько в том, как быстро алгоритм вычисляет точку пересечения луча с конкретным объектом, а, скорее, в том, чтобы алгоритм мог быстро определить, когда столкновение не происходит (для этого и используется иерархическое упорядочивание объектов, обхватывающие объемы).

ОБЩАЯ ИДЕЯ ПОИСКА ПЕРЕСЕЧЕНИЯ

В общем случае поиск пересечения луча с некоторой поверхностью (объектом) удобно проводить в ситуации, когда объект задан неявным уравнением

$$F\big(P(x,y,z)\big) = 0,$$

а луч – параметрической формой

$$r(t) = s + t \cdot \vec{d}.$$

В этом случае луч соударяется с объектом в каждый момент времени t^*, когда точка $r(t^*)$ совпадает с поверхностью:

$$F\big(r(t^*)\big) = 0.$$

Для основных объектов данное уравнение решается достаточно просто, поскольку часто после подстановки получается линейное или квадратичное по t уравнение.

НЕСКОЛЬКО РЕШЕНИЙ

В общем случае у уравнения, полученного путем постановки уравнения луча в уравнение объекта, может быть несколько корней (это означает, что луч пересек объект в нескольких точках). В этом случае из всех полученных значений t^k выбирается минимальное положительное значение.

Такой выбор связан с тем, что отрицательные значения t соответствуют ситуации, когда данная точка соударения расположена "позади" наблюдателя. Например, наблюдатель может оказаться как внутри сферы, так и снаружи, в результате чего будут выбраны разные точки пересечения (пример приведен на рис.7).

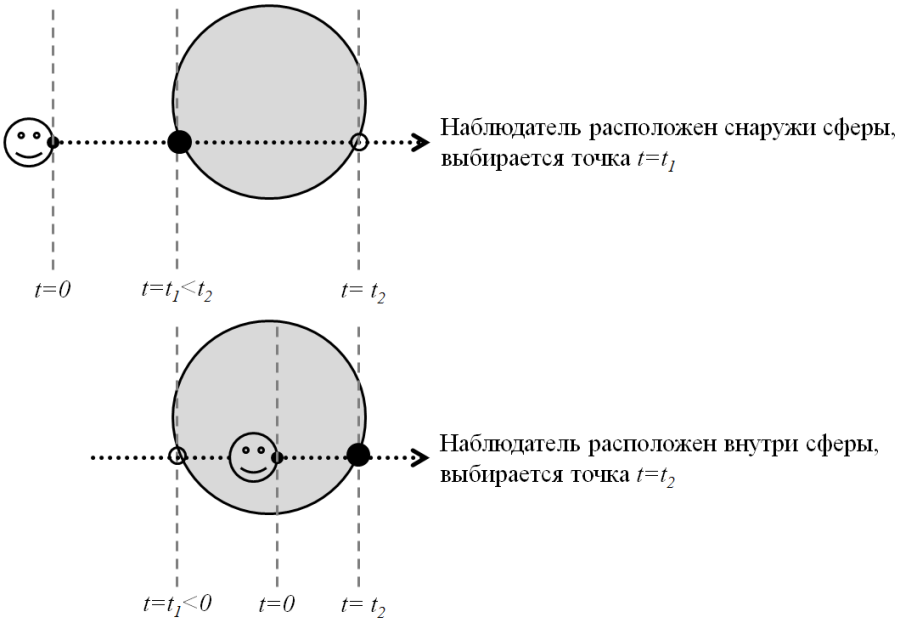


Рис.7. Пример нескольких точек пересечений.

БАЗОВАЯ ФОРМА ЗАДАНИЯ ОБЪЕКТОВ

Объекты удобно задавать и обрабатывать в локальной (модельной) системе координат (когда они представляются в своей базовой форме), откуда с помощью модельного преобразования $T(x, y, z)$ осуществляется переход в мировую систему координат путем умножения координат на соответствующую модельную матрицу M. В этом случае неявное уравнение принимает следующий вид:

$$F\left(T^{-1}(P)\right) = 0.$$

Такие модельные преобразования (сдвиг, растяжение и поворот) обычно выписываются в однородных координатах $(x, y, z, 1)$, как разобрано в [2]. Соответственно, и сам луч тоже необходимо представить в однородных координатах. А поскольку преобразования аффинные, то перевод луча в локальную (модельную) систему координат объекта выполняется следующим образом:

$$M^{-1} \cdot r(t) = M^{-1} \cdot s + t \cdot M^{-1} \cdot d.$$

В результате, процедура нахождения точки пересечения луча и базового объекта разбивается на 3 этапа:

1. Преобразование луча в локальную систему координат рассматриваемого объекта:

$$\tilde{r}(t) = (M^{-1} \cdot s) + t \cdot (M^{-1} \cdot d) = \tilde{s} + t \cdot \tilde{d}.$$

2. Нахождение $t = t^*$ из уравнения $F\left(\tilde{r}(t)\right) = 0$, в случае же нескольких значений t^* берется минимальное неотрицательное (что соответствует ближайшей точке).

3. Вычисление точки пересечения путем подставления полученного t^* в исходное уравнение луча:

$$r(t) = s(x, y, z) + t^* \cdot d(x, y, z).$$

ИСПОЛЬЗОВАНИЕ ЭКСТЕНТОВ

Экстент (extent) объекта – простая форма (оболочка), охватывающая сложный объект. Идея использования экстентов заключается в том, чтобы заключить объекты сложной формы (с которыми вычислительно тяжело искать точки соударения) в объект простой формы (с которым уже легко найти точки соударения). Таким образом, сначала ищутся пересечения лучей с экстентами (что позволяет быстро решить вопрос о непересечении), и лишь после получения положительного ответа – с самими объектами.

На рис.8 приведен пример использования сферических экстентов. Видно, что только 2 из 12 лучей попадает в первый объект, а во второй – и вовсе никто, но при этом каждый луч обязательно просчитывается на пересечение с обоими объектами сложной формы. При использовании экстентов вместо 12 лучей придется проверить только 4 на попадание в первый объект, а со вторым и вовсе не требуется проверок, поскольку ни один луч не попал во второй экстент.

Другой вариант – разбитие пространства сцены трехмерной сеткой, ячейки которой и будут экстентами.

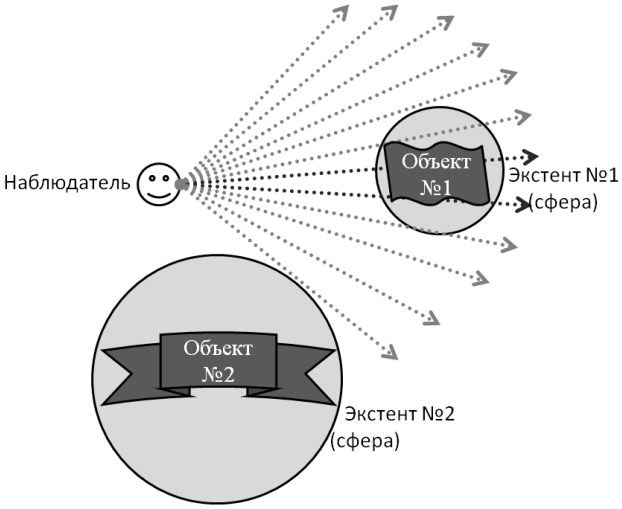


Рис.8. Использование экстентов.

ПЕРЕСЕЧЕНИЕ С ПЛОСКОСТЬЮ

Поскольку плоскость (рис.9) является линейной поверхностью, то при подстановке в него уравнения луча $r(t)$ получается линейное уравнение вида $\alpha \cdot t + \beta = 0$. У данного уравнения нет решений, если луч (Луч №0) параллелен плоскости (вектора $\vec{n}$ и $\vec{d}$ ортогональны).

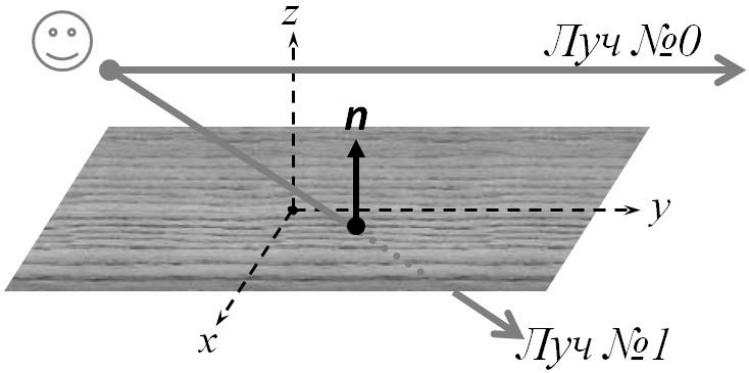


Рис.9. Пересечение лучом плоскости.

БАЗОВАЯ ПЛОСКОСТЬ

В случае базовой плоскости, заданной неявным уравнением $F\big(P(x,y,z)\big) = P_z = 0$ параметр t находится элементарно:

$$t = \frac{s_z}{d_z}.$$

ПРОИЗВОЛЬНАЯ ПЛОСКОСТЬ

В случае произвольной плоскости, заданной неявным уравнением $F\big(P(x,y,z)\big) = A \cdot P_x + B \cdot P_y + C \cdot P_z + D = 0$, параметр t имеет вид

$$t = -\frac{(\vec{n}, s) + D}{\big(\vec{n}, \vec{d}\big)}.$$

ПЕРЕСЕЧЕНИЕ СО СФЕРОЙ

Поскольку сфера (рис.10) является квадратичной поверхностью, то при подстановке в него уравнения луча $r(t)$ получается уже квадратное уравнение вида $\alpha \cdot t^2 + \beta \cdot t + \gamma = 0$. В этом случае возможны три варианта (в зависимости от значения дискриминанта уравнения δ):

1. Нет пересечений (луч №0 не попадает в сферу): $\delta < 0$.
2. Одно пересечение (луч №1 касается сферы): $\delta = 0$.
3. Два пересечения (луч №2 "пробивает" сферу): $\delta > 0$.

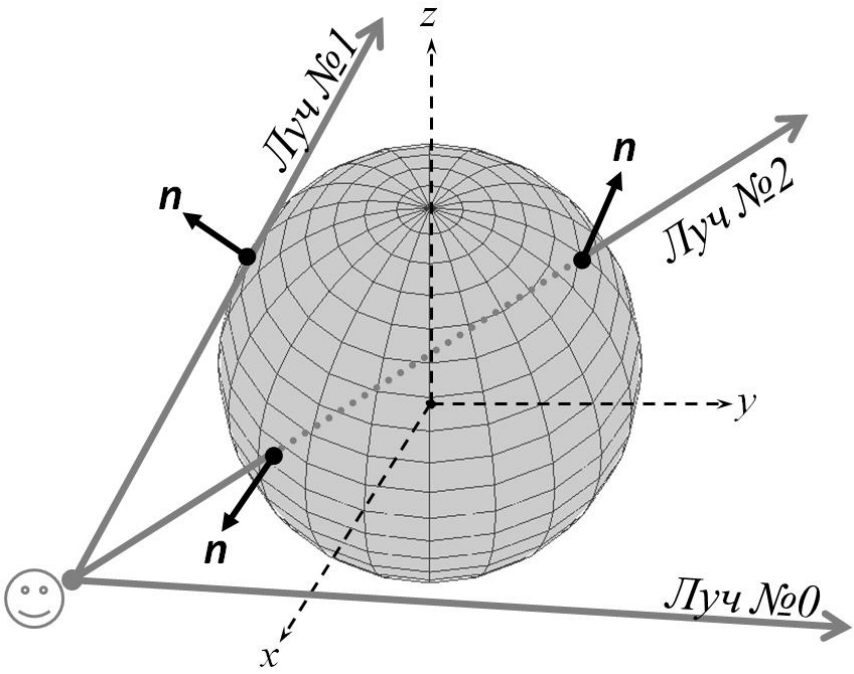


Рис.10. Пересечение лучом сферы.

БАЗОВАЯ СФЕРА

В случае базовой сферы, задаваемой уравнением

$$F\big(P(x,y,z)\big) = P_x^2 + P_y^2 + P_z^2 - 1 = (P,P) - 1 = 0,$$

точка пересечения находится из следующей дроби:

$$t = \frac{-\beta \pm \sqrt{\delta}}{\alpha},$$

где

$$\alpha = \big(\vec{d}, \vec{d}\big),$$
$$\beta = 2 \cdot \big(s, \vec{d}\big),$$
$$\gamma = (s,s) - 1,$$
$$\delta = \beta^2 - \alpha \cdot \gamma.$$

ПРОИЗВОЛЬНАЯ СФЕРА

В случае произвольной сферы, задаваемой уравнением

$$F\big(P(x,y,z)\big) = (P_x - C_x)^2 + \big(P_y - C_y\big)^2 + (P_z - C_z)^2 - R^2 = 0,$$

точка пересечения находится из следующей дроби:

$$t = -\frac{1}{2} \cdot \beta \pm \sqrt{\delta},$$

где

$$\alpha = (C,C),$$
$$\beta = 2 \cdot \big(\overrightarrow{s - C},\ \vec{d}\big),$$
$$\gamma = \big(\overrightarrow{s - C},\ \overrightarrow{s - C}\big) - R^2.$$
$$\delta = \tfrac{1}{4} \cdot \beta - \gamma.$$

ЭЛЛИПСОИД

Неявное уравнение для базового эллипсоида (рис.11) имеет следующий вид:

$$F\big(P(x,y,z)\big) = \frac{P_x^2}{a^2} + \frac{P_y^2}{b^2} + \frac{P_z^2}{c^2} - 1 = 0.$$

В этом случае имеет смысл рассмотреть эллипсоид как сферу, к которой применено преобразования масштаба. В таком случае к нормалям нужно применить обратное преобразование, например, при растяжении сферы по некоторой оси, нормаль должна не растягиваться, а сжиматься по этой оси.

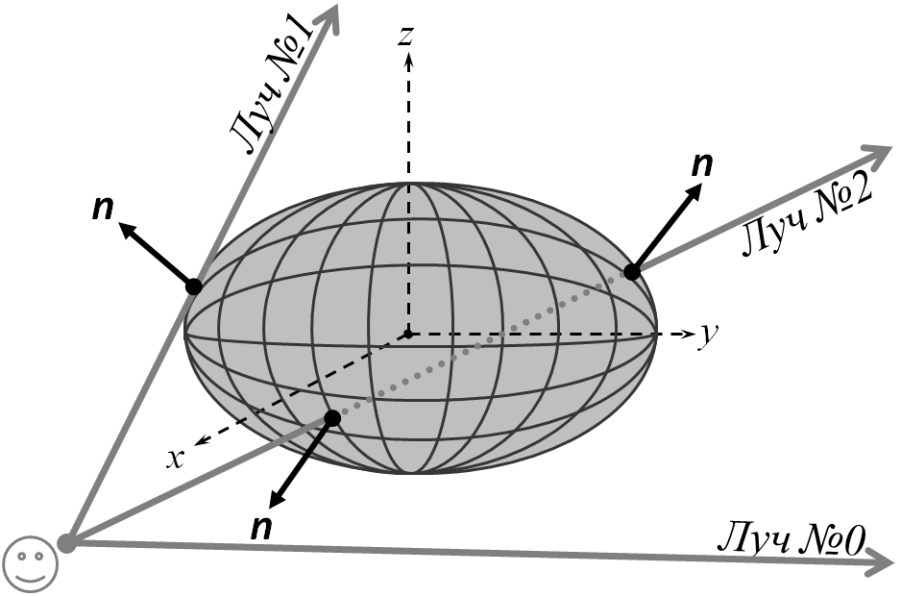


Рис.11. Пересечение лучом эллипсоида.

ПЕРЕСЕЧЕНИЕ С КОНИЧЕСКИМ ЦИЛИНДРОМ

Поскольку конический цилиндр состоит из трех частей, то требуется проверить следующие возможные пересечения (рис.12):

1) Нет пересечений (Луч №0);
2) Одно пересечение боковой поверхности (Луч №1);
3) Два пересечения боковой поверхности (Луч №2);
4) Пересечение боковой поверхности и основания (Луч №3);
5) Пересечение обоих оснований (Луч №4).

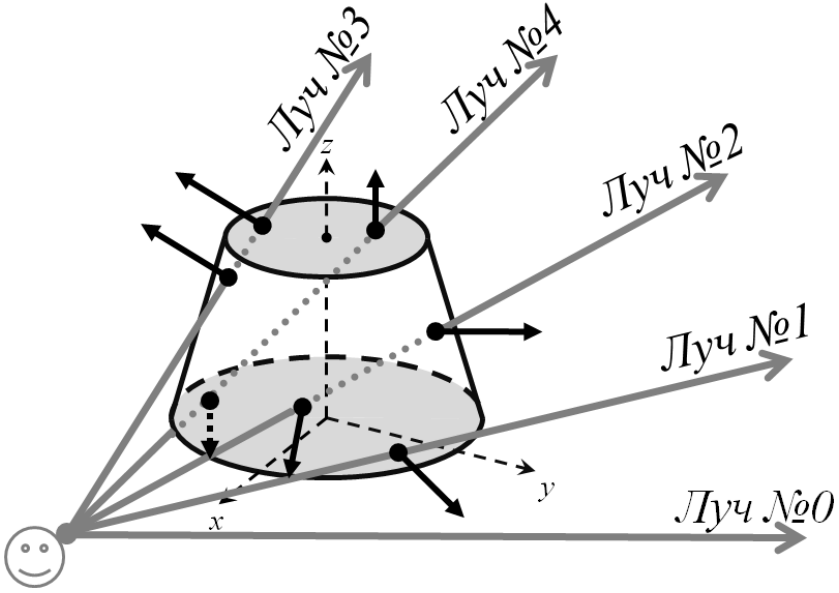


Рис.12. Конический цилиндр.

НИЖНЕЕ И ВЕРХНЕЕ ОСНОВАНИЯ

Общее неявное уравнение для обоих оснований имеет вид

$$F(P) = P_x^2 + P_y^2 < r^2.$$

Соответственно, необходимо найти пересечение луча с плоскостью основания ($z = 0$ или $z = 1$) и проверить на попадание в круг.

БОКОВАЯ ПОВЕРХНОСТЬ

Поскольку боковая поверхность, заданная уравнением

$$F(P) = \{P_x^2 + P_y^2 - (1 + q \cdot P_z^2)^2 = 0$$

является квадратичной, то при подстановке в него уравнения луча $r(t)$ получается квадратное уравнение вида $\alpha \cdot t^2 + \beta \cdot t + \gamma = 0$. В этом случае возможны три варианта (в зависимости от значения дискриминанта уравнения $\delta = \beta^2 - \alpha \cdot \gamma$):

1. Нет пересечений (лучи №0 и №4 не попадают в бок): $\delta < 0$.
2. Одно пересечение (луч №1 касается бока): $\delta = 0$.
3. Два пересечения (луч №2 "пробивает" бок): $\delta > 0$.

Точка пересечения находится из следующей дроби:

$$t = \frac{-\beta \pm \sqrt{\delta}}{\alpha},$$

где

$\alpha = d_x^2 + d_y^2 - \theta^2,$

$\beta = s_x \cdot d_x + s_y \cdot d_y - \sigma \cdot \theta,$

$\gamma = s_x^2 + s_y^2 - \sigma^2,$

$\omega = R - 1, \theta = \alpha \cdot d_z, \sigma = 1 + \alpha \cdot s_z.$

ПЕРЕСЕЧЕНИЕ С МНОГОУГОЛЬНИКОМ

Общий алгоритм для поиска пересечения луча с плоским многоугольником выглядит следующим образом:

1. Находятся коэффициенты A, B, C, D плоскости многоугольника:
    1) Вычисляется нормаль через векторное произведение двух сторон многоугольника (коэффициенты A, B, C);
    2) Вычисляется скалярное произведение нормали и любой из вершин треугольника (коэффициент $-D$).
2. Находится точка пересечения луча с плоскостью.
3. Проверяется полученная точка на попадание в многоугольник.

ТРЕХМЕРНЫЙ МНОГОУГОЛЬНИК

В том случае, когда многоугольник является трехмерным объектом типа куба или тетраэдра, это означает, что у объекта есть набор граней, каждая из которых является многоугольником (например, у параллелепипеда 6 четырехугольных граней, а у тетраэдра – 4 треугольных). Соответственно, процедура поиска заключается в переборе каждой грани трехмерного объекта (рис.13) и поиска пересечения луча с соответствующим многоугольником.

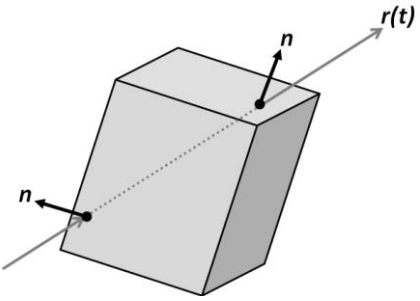


Рис.13. Параллелепипед.

АЛГОРИТМ МОЛЛЕРА-ТРУМБОРА

Алгоритм Моллера-Трумбора – это эталонный алгоритм быстрого определения пересечения луча и треугольника в трехмерном пространстве, который был представлен в 1997 году Томасом Меллером (Tomas Möller) и Беном Трамбором (Ben Trumbore).

Любая точка P треугольника ABC может быть в барицентрических (u, v)-координатах записана следующим образом:

$$P(u,v) = (1-u-v)\cdot A + u\cdot B + v\cdot C$$

Соответственно, при подстановке координат луча получается матричное уравнение

$$\begin{bmatrix} -d_x & B_x - A_x & C_x - A_x \\ -d_y & B_y - A_y & C_y - A_y \\ -d_z & B_z - A_z & C_z - A_z \end{bmatrix} \cdot \begin{bmatrix} q \\ u \\ v \end{bmatrix} = \begin{bmatrix} s_x - A_x \\ s_y - A_y \\ s_z - A_z \end{bmatrix},$$

решив которое можно найти (u, v)-координаты точки пересечения и расстояние q от начала луча до точки пересечения.

Преобразование, описанное выше, может геометрически интерпретироваться как перемещение треугольника к началу координат и трансформация его в единичный треугольник (рис.14), направление же луча при этом будет совпадать с осью OZ.

Пример реализации алгоритма приведен в приложении.

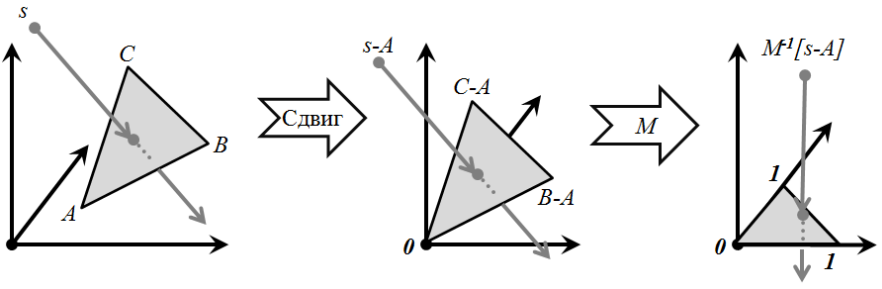


Рис.14. Трансформация треугольника.

РАСЧЕТ ОСВЕЩЕННОСТИ

В общем случае, световая энергия в любой точке сцены, состоит из следующих компонент:

1) Излучаемая (только для источников света);
2) Эмиссионная (излучаемая не источниками света);
3) Отраженная (пришедшая из других точек).

В свою очередь, на отраженную компоненту могут влиять следующие факторы:

1) Энергия источника, дошедшая напрямую;
2) Энергия источника, дошедшая в результате множества отражений и переотражений от объектов сцены;
3) Энергия источника, дошедшая в результате множества преломлений от объектов сцены.

Для аппроксимации учета отраженной компоненты было разработано множество моделей: от простых моделей локального освещения (когда учитывается лишь прямое освещение), до тяжеловесных методов глобального освещения (когда учитывается вклад всех поверхностей в итоговое освещение) типа трассировки лучей и излучательности (radiosity).

В общем случае освещенность в точке описывается основным уравнением освещенности, полученным Джеймсом Кадзией (James Kajiya) в 1986 г.

Более подробно модели локального освещения описаны в [3].

ОСНОВНОЕ УРАВНЕНИЕ ОСВЕЩЕННОСТИ

Суммарная световая энергия, которая излучается из точки x' (точки падения луча света) в направлении точки x'' (рис.15) может быть записана в виде *основного уравнения освещенности*:

$$L(x', x'') = E(x') + \int_x F(x, x', x'') \cdot L(x, x') \cdot C(x, x') \cdot G(x, x') \cdot dx,$$

где:

$L(x, x')$ – световая энергия, излучаемая из точки x в точку x',

$E(x')$ – эмиссионное излучение в точке x',

$F(x, x', x'')$ – функция ДФОС для направлений $(x \to x')$ и $(x \to x'')$,

$C(x, x')$ – индикатор преграды для луча из точки x в точку x',

$G(x, x')$ – форм-фактор поверхностей с точками x и x'.

Как можно видеть, в уравнении рендеринга функция L стоит и в левой, и в правой части (причем под интегралом). Фактически, основное уравнение освещенности является уравнением Фредгольма второго рода и в общем случае решается численно или с использованием метода Монте-Карла (стохастического метода, часто используемого в расчете глобального освещения).

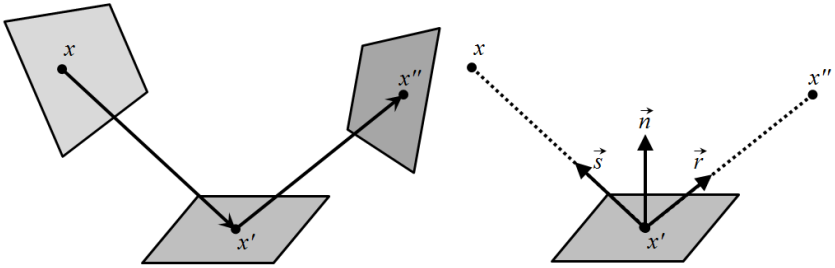


Рис.15. Направления светового потока.

ФУНКЦИЯ ДФОС

Функция $F(x, x', x'') \equiv F(\vec{s}, \vec{r})$ называется функцией ДФОС, *двулучевой (двунаправленной) функцией отражательной способности* (Bidirectional Reflection Distribution Function, BRDF). Данная функция определяет, как свет отражается от непрозрачной поверхности (без учета внутреннего рассеивания), т.е. задает долю световой энергии, пришедшей по направлению $\vec{s}$ (из точки x в точку x') и уходящей затем в направлении $\vec{r}$ (из точки x' в точку x''):

$$F(\vec{s}, \vec{r}) = \frac{dL(\vec{r})}{dL(\vec{s}) \cdot cos\theta \cdot d\vec{s}},$$

где θ – угол между направлениями $\vec{s}$ и $\vec{n}$.

В общем случае ДФОС зависит таких факторов, как свойства материала, длина волны (то есть, цвет) падающего света и его поляризации. При этом ДФОС удовлетворяет условию симметричности (принцип Гельмгольца):

$$F(x, x', x'') \equiv F(x'', x', x).$$

С точки зрения стандартной растеризации ДФОС можно сравнить с текстурой. Только текстура определяет изменение цвета в зависимости от позиции на поверхности материала, а ДФОС – в зависимости от направления освещения или наблюдения.

Простейший способ выбора функции ДФОС – задание ее в виде простой математической функцией с некоторыми параметрами, определяющими свойства материала данной поверхности. Классическим примером данного подхода является модель Ламберта, где

$$F(\vec{s}, \vec{r}) = cos\theta = (\vec{s}, \vec{n})$$

В особо сложных случаях функция ДФОС решается с помощью метода Монте-Карла.

МОДЕЛЬ ОСВЕЩЕНИЯ ФОНГА

В общем виде *модель освещения Фонга*, которая чаще всего используется в компьютерной графике, состоит из суммы фоновой I_a, диффузной I_d и зеркальной I_s составляющей и имеет следующий вид:

$$I = I_a + I_d + I_s = m_a \cdot L_a + m_d \cdot k_d \cdot L_d + m_s \cdot k_s \cdot L_s,$$

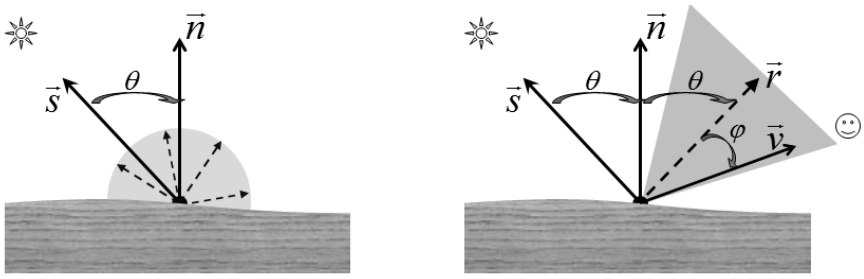
где:

(m_a, m_d, m_s) – коэффициенты поглощения материалом интенсивностей I при попадании света на поверхность;

$k_d = max\{0, (\vec{s} \cdot \vec{n})\}$ – коэффициент потери интенсивности I_d в зависимости от угла падения при диффузном отражении;

$k_s = max\{0, (\vec{r} \cdot \vec{v})^b\}$ – коэффициент потери интенсивности I_s в зависимости от углов падения и направления на наблюдателя при зеркальном отражении.

На рис.16 представлена геометрия диффузного и зеркального отражений.



Диффузная поверхность Зеркальная поверхность

Рис.16. Диффузное рассеивание и зеркальное отражение.

ТРАССИРОВКА ЛУЧЕЙ

Трассировка лучей родилась из метода «бросания лучей» («ray casting»), который был создан для просчитывания гамма-лучей при изучении радиации. Первый вариант для рендеринга был представлен в 1968 г. Артуром Аппелем (Arthur Appel). Суть метода заключалась в генерации луча из точки наблюдения (один луч на один пиксель) и поиске самого близкого объекта, который блокирует его дальнейшее распространение. На основе этих данных с помощью алгоритмов компьютерной графики можно было определить затенение данного объекта. Сам термин «ray casting» появился лишь в 1982 году.

Следующий важный этап начался в 1979 г. Дело в том, что алгоритмы бросания лучей прослеживали путь луча от наблюдателя лишь до первого столкновения с объектом. Тернер Уиттед (Turner Whitted) предложил продолжить этот процесс: луч после попадания на поверхность теперь может создавать три новых типа лучей: отражение, преломление и тень. За счет появления этих вторичных лучей появилась возможность достаточно точно учесть эффекты отражения и преломления, используя законы геометрической оптики. Таким образом, метод трассировки лучей стал первым методом расчета глобального освещения, учитывающим многократные отражения и преломления.

В общем, идея метода трассировки лучей заключается в отслеживание траекторий испускаемых и отражаемых лучей и расчете взаимодействия лучей с объектами в точках их пересечений, от момента испускания лучей источником света до момента попадания в камеру. Под взаимодействием луча с объектом понимаются процессы диффузного и зеркального отражений луча от поверхности и прохождение луча сквозь прозрачные и полупрозрачные объекты. Поведение луча в этом случае описывается законами геометрической оптики.

Различают два подхода к трассировке лучей: метод прямой трассировки (forward ray tracing) и метод обратной трассировки (backward ray tracing).

ЗАКОНЫ ГЕОМЕТРИЧЕСКОЙ ОПТИКИ

Геометрическая оптика – это раздел оптики, изучающий законы распространения света в прозрачных средах, отражения света от зеркальных поверхностей и принципы построения изображений при распространении света без учета его волновых свойств.

Геометрическая оптика неполно описывает оптические явления, поскольку пренебрегает волновыми и квантовыми явлениями, является упрощением более общей волновой оптической теории (в частности, световой луч, являющийся основным понятием геометрической оптики, определяется как световая волна с нулевой длиной). Тем не менее, благодаря своему простому математическому аппарату геометрическая оптика широко применяется во многих задачах, в частности, в той же трассировке лучей.

В основе геометрической оптики лежат несколько простых эмпирических законов:

- Закон прямолинейного распространения света,
- Закон независимого распространения лучей,
- Закон отражения света,
- Закон преломления света,
- Закон обратимости светового луча.

Фактически, из этих законов следуют фундаментальные правила метода трассировки лучей при распространении лучей:

- Траектория луча прямолинейна и обратима, меняется только после контакта с поверхностью;
- Интенсивность луча (фотона) меняется либо с пройденным расстоянием, либо после контакта с поверхностью;
- Освещенность точки поверхности складывается из света дошедших лучей.

ЗАКОНЫ ОТРАЖЕНИЯ

Закон зеркального отражения света определяет изменение направления хода светового луча в результате столкновения с отражающей (зеркальной) поверхностью. У этого закона помимо строгой формулировки есть и широко распространенная, но менее точная.

Строгая формулировка: падающий и отраженный лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения, и эта нормаль делит угол между лучами на две равные части.

Нестрогая формулировка: угол падения равен углу отражения.

Этот закон является следствием применения *принципа Ферма* (свет распространяется по кратчайшему пути) к отражающей поверхности. Закон справедлив не только для идеально отражающих поверхностей, но и для границы двух сред, частично отражающей свет.

Нарушение закона зеркального отражения на поверхностях с мелкими неровностями приводит к *диффузному отражению*, когда свет отражается в произвольном направлении (рис.17)

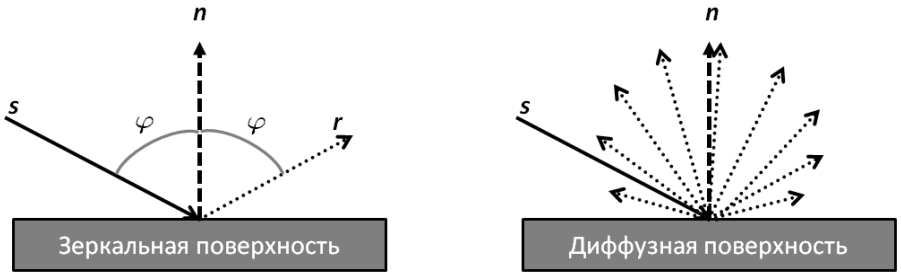


Рис.17. Законы зеркального и диффузного отражения.

ЗАКОН ПРЕЛОМЛЕНИЯ

В общем случае, при попадании луча света на преломляющую поверхность падающий луч разбивается на 2 луча (рис.18): отраженный и преломленный. Отраженный луч отражается от границы раздела сред по закону отражения. Преломленный же луч продолжит распространяться дальше во второй среде, но под уже другим углом. Связь углов падения и углов преломления луча описывается *законом Снеллиуса* (Снелля, Снелла):

$$k_1 \cdot sin(\theta_1) = k_2 \cdot sin(\theta_2),$$

где:

k_1 – показатель преломления среды, из которой свет падает,

k_2 – показатель преломления среды, в которую свет попадает,

θ_1 – угол падения (угол между падающим лучом и нормалью),

θ_2 – угол отражения (угол между отраженным лучом и нормалью).

Если $n_1 \cdot sin(\theta_1) > n_2$, то возникает ситуация полного внутреннего отражения, когда преломленный луч отсутствует, а падающий полностью отражается от границы раздела сред.

Законом Снеллиуса описывает только направление луча, но не изменение его интенсивности.

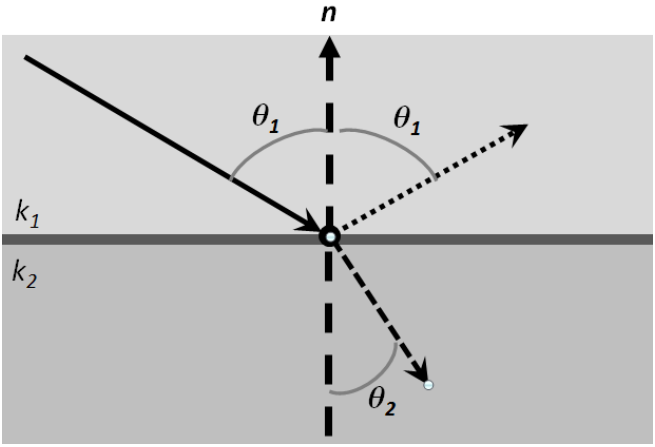


Рис.18. Законы зеркального и диффузного отражения.

ПЕРВИЧНЫЕ И ВТОРИЧНЫЕ ЛУЧИ

В прямой трассировке (forward ray tracing) лучи пускаются из источников света, при этом очевидно, что лишь малая часть таких лучей достигнет наблюдателя (рис.19).

В обратной трассировке (backward ray tracing) лучи пускаются не из источников света, а из глаз наблюдателя, что позволяет просчитывать только необходимые лучи. Типы генерируемых лучей:

1. Первичный луч: из глаза через пиксель до точки первого соударения.
2. Луч тени: из точки соударения до источника (определение видимости источника).
3. Отраженный луч: из точки соударения до следующей точки соударения (при наличии отражающих поверхностей).
4. Преломленный луч: из точки соударения до следующей точки соударения (при наличии преломляющих объектов).

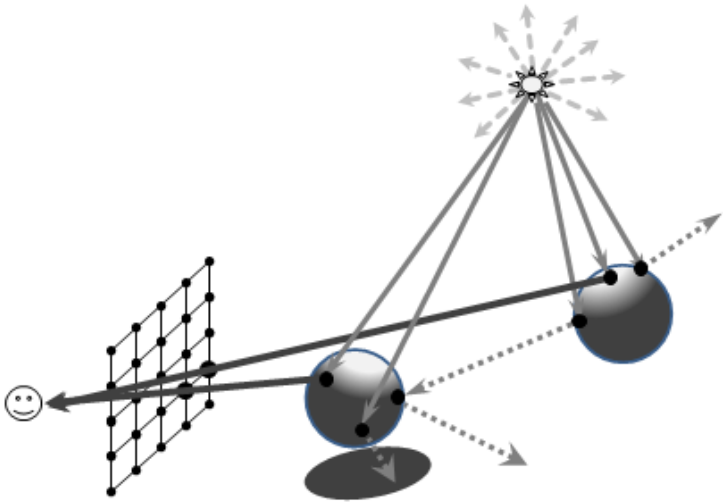


Рис.19. Прямая трассировка.

ТРАССИРОВКА ПЕРВИЧНЫХ ЛУЧЕЙ

Через каждый пиксель экрана выпускается *первичный луч* и ищется его ближайшее соударение с объектами сцены (рис.20). Если соударений не нашлось (луч №1), то пиксель закрашивается цветом фоны.

В противном случае возникает необходимость в расчете прямого освещения (лучи №2-5), для этого из точки соударения в направлении каждого источника испускаются лучи тени, и выполняется расчет первичного освещения по выбранной модели освещения. Как можно видеть, луч №2Т наткнулся на преграду, а для луча №3 и вовсе нет смысла пускать луч тени (поскольку нормаль направлена от источника), поэтому пиксели, соответствующие лучам №2 и №3 закрашиваются фоновым цветом соответствующих объектов. В свою очередь, лучи №4Т и №5Т достигли источника, поэтому пиксели, соответствующие лучам №4 и №4 закрашиваются полноценно по модели освещения.

На этом процесс распространения первичных лучей завершается. Данный метод получил названия «ray casting».

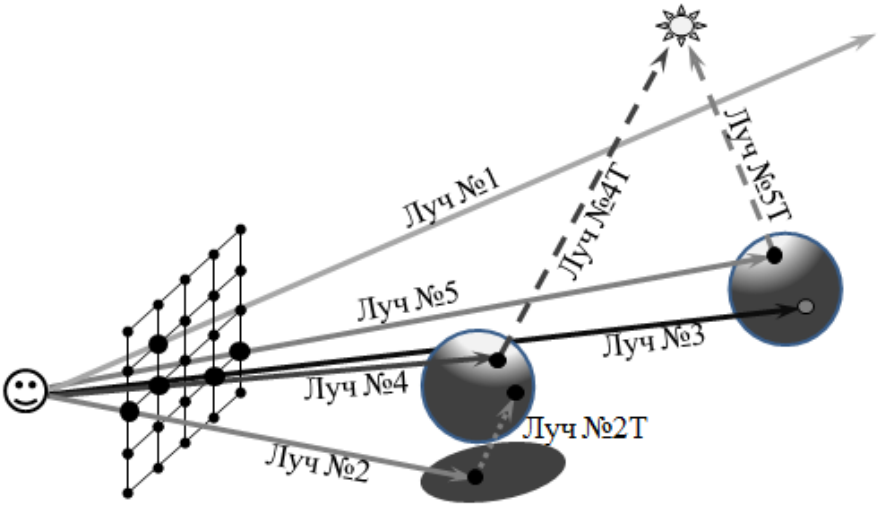


Рис.20. Обратная трассировка.

ТРАССИРОВКА ВТОРИЧНЫХ ЛУЧЕЙ

Если объект обладает свойством отражения или преломления, то возникает необходимость рассчитать и вторичное освещение. Для этого из точки соударения испускаются вторичные лучи отражения и преломления, которые продолжают рекурсивно трассироваться дальше: ищутся новые точки их соударения с объектами, из новых точек испускают уже третичные лучи тени, преломления и отражения и т.д.

Завершается трассировка луча либо по числу итераций, либо по величине приращения освещенности, оказавшейся на данной итерации ниже пороговой, либо по его покидании сцены. Для хранения такой структуры данных удобно использовать двоичные деревья. Освещенность же в корневой точке формируется путем обратного пробега по дереву и

На рис.21 приведен пример распространения вторичных лучей для одного первичного луча *P* и соответствующее двоичное дерево.

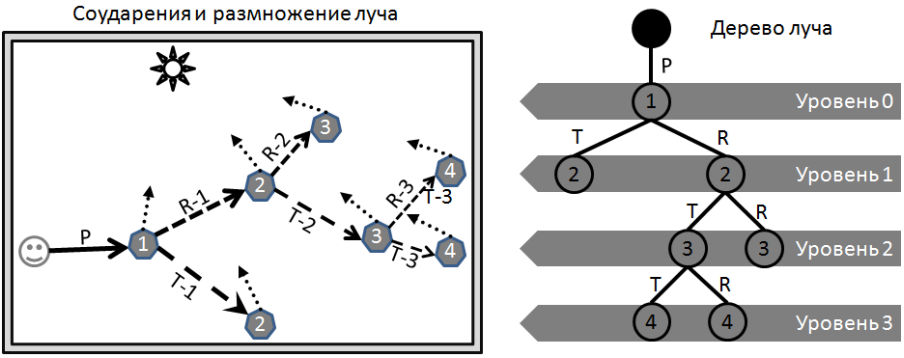


Рис.21. Дерево вторичных лучей.

ЗАДАНИЕ УРАВНЕНИЯ ЛУЧА

1. Задается положение камеры (точки наблюдения, позиции наблюдателя) в точке $Eye = E(x, y, z)$.

2. На расстоянии S от наблюдателя задается экран размером $(2 \cdot W) \times (2 \cdot H)$ так, чтобы камера смотрела в центр экрана под прямым углом. При этом система координат камеры $(\vec{v}, \vec{u}, \vec{w})$ относительно экрана направлена так, чтобы ось $\vec{v}$ была направлена вверх, ось $\vec{u}$ – вправо, а ось $\vec{w}$ – назад.

3. Важно, что размер экрана и расстояние до камеры определяют угол видимости камеры θ, называемый *областью видимости (*field of view, FOV). У людей FOV по горизонтали составляет почти 180°, однако большую часть его составляет смутное периферическое зрение, поэтому часто FOV задают равным 60° по горизонтали и вертикали, что соответствует следующему равенству:

$$S = 2 \cdot W = 2 \cdot H = 1.$$

4. Экран представляется в виде прямоугольника пикселей с разрешением $n_c \times n_r$, соответственно, аспект (соотношение сторон) равен

$$aspect = \frac{n_c}{n_r}.$$

5. Координаты любого (i, j)-пикселя можно рассчитать следующим образом:

$$v_j = -W + W \cdot \frac{2 \cdot j}{V_c}, \qquad j \in [0, n_c],$$

$$u_i = -H + H \cdot \frac{2 \cdot i}{V_r}, \qquad i \in [0, n_r],$$

где:

$$H = S \cdot tg\left(\frac{\theta}{2}\right),$$

$$W = H \cdot aspect$$

6. При условии, что луч начинается $(t = 0)$ в точке E, а при $t = 1$ проходит через (i, j)-пиксель, векторное уравнение луча в параметрической форме можно записать следующим образом:

$$\vec{r}_{ij} = \vec{r}_e + t \cdot \vec{d}_{ij},$$

где:

$\vec{r}_e$ – радиус-вектор точки $E(x, y, z)$,
$\vec{d}_{ij} = V_d \cdot \vec{w} + W \cdot \left(\frac{2 \cdot j}{V_c} - 1\right) \cdot \vec{u} + H \cdot \left(\frac{2 \cdot i}{V_r} - 1\right) \cdot \vec{v}$.

Как можно видеть, с течением времени t луч удаляется от наблюдателя, и если луч на своем пути соударяется с несколькими объектами, то ближайшему объекту соответствует наименьшее значение t. При этом отрицательным значениям t соответствуют объекты, расположенные позади наблюдателя, поэтому должны игнорироваться.

На рис.22 представлено положение камеры, экрана и системы координат. Поскольку позиции фиксированы, в качестве системы $(\vec{v}, \vec{u}, \vec{w})$ удобно брать исходную глобальную систему координат XYZ, где ось OZ направлена вглубь экрана.

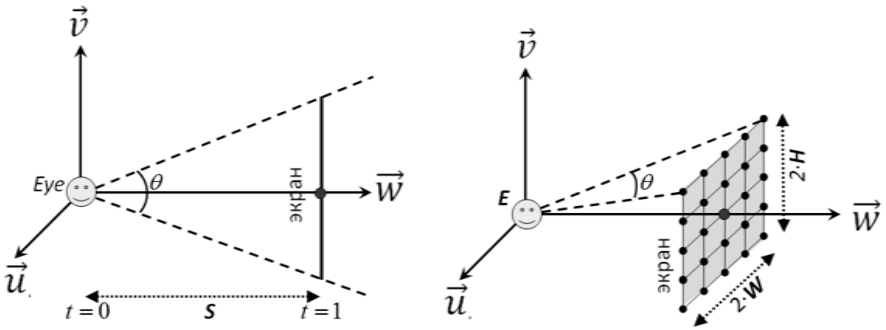


Рис.22. Положение камеры и экрана.

НЕКОТОРЫЕ ПРОБЛЕМЫ ВЫБОРА ЛУЧЕЙ

В классической трассировке лучей луч считается бесконечно тонким, поэтому метод чувствителен к численной погрешности и выбору точки пикселя, через который пускается луч. На рис.23 приведен пример трех вариантов испускания луча: через центр, через угол и середину нижней половины пикселя. Как можно видеть, в этой ситуации результат для пикселя будет разным, поскольку лучи приходят поразному. Для решения данной проблемы можно придать лучу объем (получатся такие модификация метода трассировки лучей, как метод конусов и пирамид) или через пиксель пускать не один луч, усредняя результат (по сути, получится методы сглаживания типа SuperSample Anti-Aliasing или MultiSample Anti-Aliasing). На рис.24 приведен пример, когда пиксель равно разбивается на 4 субпикселя, через центры которых и пускаются 4 луча, каждый из которых участвует в итоговом цвете с весом 1/4, но возможен и стохастический выбор лучей через пиксель.

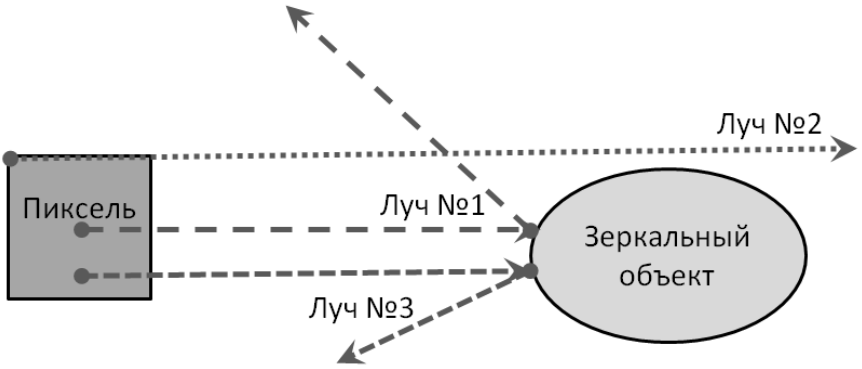


Рис.23. Варианты испускания луча через пиксель.

Помимо этой проблемы существует еще и проблема качественного учета подповерхностных слоев, подповерхностного рассеивания (когда луч неглубоко проникает под поверхность и начинает там отражаться-преломляться), как показано на рис.25. В честности, без учета этого эффекта человеческая кожа будет выглядеть как пластик.

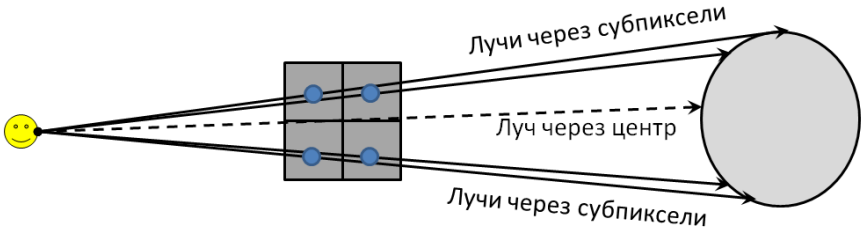


Рис.24. Прохождение лучей через субпиксели.

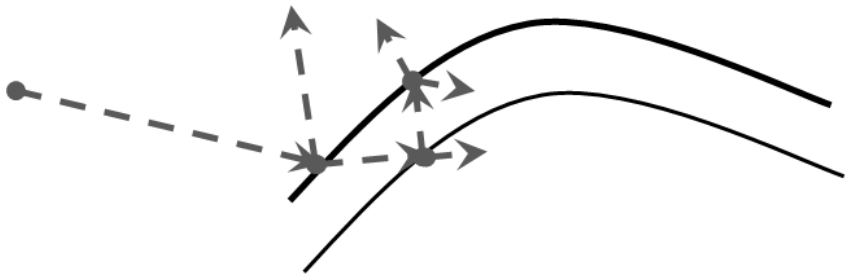


Рис.25. Подповерхностное рассеивание.

ОБРАТНАЯ ТРАССИРОВКА ЛУЧЕЙ

Алгоритм метода обратной трассировки лучей начинается с задания сцены и параметров наблюдателя и пиксельного экрана, после чего для каждого пикселя выполняется следующая процедура:

1. Строится уравнение луча $\vec{r}(t)$.
2. Находится ближайшая точка пересечения со всеми объектами.
3. Рассчитывается "основная" освещенность в найденной точке:
    1) Пускаются теневые лучи ко всем источникам для определения их видимости и (возможно) дальности;
    2) По выбранной модели освещения вычисляется основной цвет.
4. Рассчитывается "вторичная" освещенность (рекурсия):
    1) Если объект зеркальный или преломляющий, то в найденной точке по закону отражения (преломления) строится новый (отраженный или преломленный) луч;
    2) Новый луч рекурсивно трассируется дальше, точки его соударений формируют соответствующее дерево пересечений;
    3) По завершении рекурсии отраженный (преломленный) цвет рассчитывается путем суммирования цвета в узлах дерева.
5. Суммарный цвет в точке вычисляется путем линейной комбинации цветов основного, отраженного и преломленного лучей.

Замечание. Рекурсия трассировки вторичных лучей завершается в следующих случаях:

1) Достигнуто максимальное количество отражений луча;
2) Добавляемая интенсивность цвета незначительна;
3) Луч не пересекается с объектами (уходит в бесконечность).

ОСНОВНЫЕ ДОСТОИНСТВА

1. Физичность процесса.
2. Возможность рендеринга гладких объектов без аппроксимации их полигонами.
3. Возможность рендеринга объектов, заданных уравнениями.
4. Высокая алгоритмическая распараллеливаемость.
5. Автоматическое отсечение невидимых поверхностей.
6. Корректный учет поля зрения (перспективы).
7. Корректный учет отражений (в том числе и взаимных) и преломлений.
8. Легкая реализация сглаживания изображения.
9. Высокая фотореалистичность.
10. Просчет сцены любой сложности.

ОСНОВНЫЕ НЕДОСТАТКИ

1. Источники света являются только точечными, проблематично учесть рассеянное (сумеречное) освещение.
2. Высокая вычислительная сложность, напрямую зависящая как от количества лучей, так и от количества и сложности объектов.
3. В точке соударения луча и поверхности при расчете вторичного освещения выпускаются (и трассируются дальше) дополнительные лучи (отражения, преломления, тени), что резко увеличивает количество трассируемых лучей.
4. В общем случае необходимо просчитывать пересечение каждого луча с каждым объектом.
5. Невозможность корректно учесть некоторые объекты и явления, такие, как: туман (облачность), огонь, сложнопредставимые объекты типа деревьев.

РАЗВИТИЕ ТРАССИРОВКИ ЛУЧЕЙ

Большинство фильмов с синтезированной/рисованной картинкой используют алгоритм под названием REYES, который базируется на растеризации. С развитием же вычислительной техники и соответствующих алгоритмов началось внедрение методов трассировки в производство фильмов. Например, в вышедшем в 2006 г. анимационном фильме «Тачки» («Cars») студия Pixar впервые добавила трассировку лучей к своему движку рендеринга RenderMan, однако использовался трассировщик избирательно, чтобы не перегрузить существующие вычислительные мощности (в основном для отражения глаз на капоте и учета эффектов затенения ambient occlusion). Но уже в 2013 г. в анимационном фильме «Университет монстров» («Monsters University») трассировка лучей полноценно использовалась для освещения и затенения.

Аналогично подходу студии Pixar, корпорации Microsoft и NVIDIA не предлагают «трассировать» всю сцену. Предлагается использовать гибридный подход, когда одна (основная) часть элементов рассчитывается классической растеризацией, а другая (отражения, преломления, тени) – трассировкой лучей. В частности, трассировка может помогать локальным отражениям отслеживать ушедшую из поля зрения геометрию, более точно позиционировать локальное затенение, повышать реалистичность теней.

Про соотношение в современных движках обычной растеризации и трассировки лучей инженер nVidia Дэвид Любке (David Luebke) сделал следующий комментарий: "Растеризация выполняется быстро, но необходимо тщательно продумывать то, как выполнять сложные визуальные эффекты. Метод трассировки лучей поддерживает сложные визуальные эффекты, но необходимо тщательно продумывать то, как сделать его быстрым".

ОСНОВНЫЕ ЭТАПЫ РАЗВИТИЯ ПРОГРАММНОГО И АППАРАТНОГО ОБЕСПЕЧЕНИЯ ТРАССИРОВКИ

Первое изображение, на основе трассировки лучей, было получено в университете Мэриленда еще в 1963 г.

В 1985 году был представлен первый видеоролик для планетарных залов, полностью смоделированный на LINKS-1, разработанном для создания трехмерной графики с использованием трассировки лучей.

В 1984 году была продемонстрирована система моделирования BRL-CAD. Спустя три года для нее был представлен трассировщик лучей, особенностью которого была хорошая оптимизация. Общая производительность при рендере доходила до нескольких кадров в секунду при использовании нескольких машин с разделяемой памятью.

В 2008 г. Intel показала демонстрационные материалы проекта «Quake Wars: Ray Traced». Производительность была на уровне 14-29 кадров в секунду при использовании нескольких четырехядерных процессоров и 20-35 кадров с шестиядерными процессорами.

В 2009 году NVIDIA анонсировала Optix – бесплатный пакет программного обеспечения для работы с трассировкой на видеокартах, который вошел в состав таких популярного графического ПО, как Autodesk Maya и 3DS MAX. На стороне AMD есть аналогичные функции в библиотеках GPUOpen, ставшего потом частью Unreal Engine.

В 2013 был разработан игровой движок Brigade (позже вошедшего в состав Unity), который смог продемонстрировать достойные результаты трассировки путей, при этом нехватка вычислительной мощности приводила не к падению FPS, а к увеличению "шумов" на результирующем изображении.

В 2018 г. Microsoft сделала дополнение к API DirectX 12 в виде DXR (DirectX Raytracing). Позже AMD (создатель GPUOpen) ввела поддержку рейтрейсинга в свой API Vulkan.

В 2018 г. NVIDIA выпустила игровые видеокарты на архитектуре Turing, подразумевающей наличие RT-ядер (для работы именно над трассировкой лучей) и тензорных ядер (Tensor cores).

ФЛАГМАН NVIDIA

В 2018 г. известный разработчик графических процессоров компания NVIDIA представил семейство видеокарт GeForce RTX 20 Series с новой архитектурой Turing (в честь английского математика, логика и криптографа Алана Тьюринга). Данное семейство видеокарт поддерживает трассировку лучей в режиме реального времени благодаря внедрению новых вычислительных блоков – тензорных и рэйтрейсинговых ядер, при этом для увеличения детализации изображения используются решения на базе нейросетей.

RT-ядра выполняют поиск пересечений между лучом и полигонами сцены с использованием метода Bounding Volume Hierarchy (метода иерархической сортировки объектов). В статичной сцене генерация структуры BVH выполняется один раз для всех последующих кадров, но алгоритм также допускает и динамическую коррекцию при преобразовании геометрии. Важную роль в алгоритмах RT играют тензорные ядра, за счет которых алгоритм нейросети сможет компенсировать нехватку плотности лучей путем аппроксимации недостающих результатов.

Несмотря на высокую эффективность новой архитектуры в трассировке лучей, NVIDIA призывает использовать комбинированный подход, в котором RT-ядра привлекаются для таких задач, которые лучше всего решаются трассировкой лучей, а основную работу в построении изображения по-прежнему берет на себя растеризация.

К примеру, растеризация с буфером глубины (z-буфером), преобладающая в играх, гораздо быстрее отсекает невидимые поверхности на раннем этапе рендеринга и полностью заменяет этап первичных лучей в RT. А вот при помощи вторичных лучей можно получить любой набор эффектов освещения: начиная с дозированного формирования теней и отражений и заканчивая полным моделированием физически достоверного освещения.

Трассировка лучей также является частью библиотек GameWorks для симуляции позиционного звука.

БИТВА ВИЗИТОК

Интересной особенностью трассировщика лучей является простой и элегантный алгоритм самой трассировки лучей. В простейших случаях алгоритм можно уложить всего лишь в килобайт обычного кода, однако высокопроизводительные алгоритмы трассировки лучей – совершенно иное дело.

В 1984 Пол Гекберт (Paul Heckbert) в своей книге "Graphics gems IV" была предложена интересная задача-вызов: написать демонстрацию метода трассировки лучей, которая бы умещалась на визитной карточке. Пример варианта самого Гекберта приведен на рис.26.

В 2009 Эндрю Кенслер (Andrew Kensler) придумал 1337-байтный трассировщик лучей размером как раз с визитку (рис.27). Позже, в 2018 на обратной стороне флайера Pixar он разместил вариант трассировщика путей, на рис.28 приведен пример этого кода и соответсвующее построенное изображение.

Примеры взяты с личного сайта разработчика ПО Fabien Sanglard (https://fabiensanglard.net/rayTracing_back_of_business_card/)

```
typedef struct{double x,y,z}vec;vec U,black,amb={.02,.02,.02};struct sphere{
vec cen,color;double rad,kd,ks,kt,kl,ir}*s,*best,sph[]={0.,6.,.5,1.,1.,1.,.9,
.05,.2,.85,0.,1.7,-1.,8.,-.5,1.,.5,.2,1.,.7,.3,0.,.05,1.2,1.,8.,-.5,.1,.8,.8,
1.,.3,.7,0.,0.,1.2,3.,-6.,15.,1.,.8,1.,7.,0.,0.,0.,.6,1.5,-3.,-3.,12.,.8,1.,
1.,5.,0.,0.,0.,.5,1.5,};yx;double u,b,tmin,sqrt(),tan();double vdot(A,B)vec A
,B;{return A.x*B.x+A.y*B.y+A.z*B.z;}vec vcomb(a,A,B)double a;vec A,B;{B.x+=a*
A.x;B.y+=a*A.y;B.z+=a*A.z;return B;}vec vunit(A)vec A;{return vcomb(1./sqrt(
vdot(A,A)),A,black);}struct sphere*intersect(P,D)vec P,D;{best=0;tmin=1e30;s=
sph+5;while(s-->sph)b=vdot(D,U=vcomb(-1.,P,s->cen)),u=b*b-vdot(U,U)+s->rad*s
->rad,u=u>0?sqrt(u):1e31,u=b-u>1e-7?b-u:b+u,tmin=u>=1e-7&&u<tmin?best=s,u:
tmin;return best;}vec trace(level,P,D)vec P,D;{double d,eta,e;vec N,color;
struct sphere*s,*l;if(!level--)return black;if(s=intersect(P,D));else return
amb;color=amb;eta=s->ir;d= -vdot(D,N=vunit(vcomb(-1.,P=vcomb(tmin,D,P),s->cen
)));if(d<0)N=vcomb(-1.,N,black),eta=1/eta,d= -d;l=sph+5;while(l-->sph)if((e=l
->kl*vdot(N,U=vunit(vcomb(-1.,P,l->cen))))>0&&intersect(P,U)==l)color=vcomb(e
,l->color,color);U=s->color;color.x*=U.x;color.y*=U.y;color.z*=U.z;e=1-eta*
eta*(1-d*d);return vcomb(s->kt,e>0?trace(level,P,vcomb(eta,D,vcomb(eta*d-sqrt
(e),N,black))):black,vcomb(s->ks,trace(level,P,vcomb(2*d,N,D)),vcomb(s->kd,
color,vcomb(s->kl,U,black))));}main(){printf("%d %d\n",32,32);while(yx<32*32)
U.x=yx%32-32/2,U.z=32/2-yx++/32,U.y=32/2/tan(25/114.5915590261),U=vcomb(255.,
trace(3,black,vunit(U)),black),printf("%.0f %.0f %.0f\n",U);}/*minray!*/
```

*Рис.26.*Пример кода на визитке (Paul Heckbert)

```
#include <stdlib.h>   // card > aek.ppm
#include <stdio.h>
#include <math.h>
typedef int i;typedef float f;struct v{
f x,y,z;v operator+(v r){return v(x+r.x
,y+r.y,z+r.z);}v operator*(f r){return
v(x*r,y*r,z*r);}f operator%(v r){return
x*r.x+y*r.y+z*r.z;}v(){}v operator^(v r
){return v(y*r.z-z*r.y,z*r.x-x*r.z,x*r.
y-y*r.x);}v(f a,f b,f c){x=a;y=b;z=c;}v
operator!(){return*this*(1/sqrt(*this%*
this));};};i G[]={247570,280596,280600,
249748,18578,18577,231184,16,16};f R(){
return(f)rand()/RAND_MAX;}i T(v o,v d,f
&t,v&n){t=1e9;i m=0;f p=-o.z/d.z;if(.01
<p)t=p,n=v(0,0,1),m=1;for(i k=19;k--;)
for(i j=9;j--;)if(G[j]&1<<k){v p=o+v(-k
,0,-j-4);f b=p%d,c=p%p-1,q=b*b-c;if(q>0
){f s=-b-sqrt(q);if(s<t&&s>.01)t=s,n=!(
p+d*t),m=2;}}return m;}v S(v o,v d){f t
;v n;i m=T(o,d,t,n);if(!m)return v(.7,
.6,1)*pow(1-d.z,4);v h=o+d*t,l=!(v(9+R(
),9+R(),16)+h*-1),r=d+n*(n%d*-2);f b=l%
n;if(b<0||T(h,l,t,n))b=0;f p=pow(l%r*(b
>0),99);if(m&1){h=h*.2;return((i)(ceil(
h.x)+ceil(h.y))&1?v(3,1,1):v(3,3,3))*(b
*.2+.1);}return v(p,p,p)+S(h,r)*.5;}i
main(){printf("P6 512 512 255 ");v g=!v
(-6,-16,0),a=!(v(0,0,1)^g)*.002,b=!(g^a
)*.002,c=(a+b)*-256+g;for(i y=512;y--;)
for(i x=512;x--;){v p(13,13,13);for(i r
=64;r--;){v t=a*(R()-.5)*99+b*(R()-.5)*
99;p=S(v(17,16,8)+t,!(t*-1+(a*(R()+x)+b
*(y+R())+c)*16))*3.5+p;}printf("%c%c%c"
,(i)p.x,(i)p.y,(i)p.z);}}
```

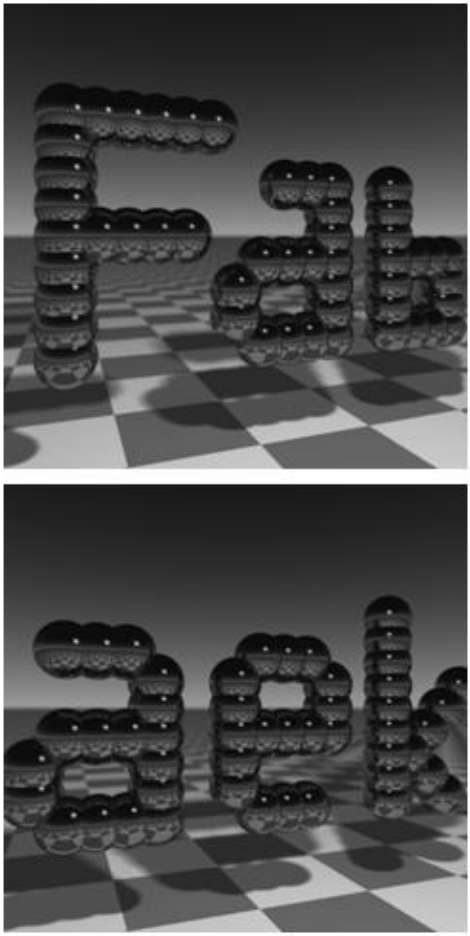


Рис.27. 1337-байтный код на визитке и реализация (*Andrew Kensler*)

Рис.28. 2037-байтный код на визитке и реализация (*Andrew Kensler*).

МЕТОД ТРАССИРОВКА ПУТЕЙ

Трассировка путей (Path tracing) впервые была представлена Джеймсом Кадзией (James Kajiya) в 1986 г.

Трассировка путей является продвинутым частным случаем обратной трассировки лучей: каждый раз, когда луч ударяется о поверхность, выпускается случайный теневой и случайный отраженный луч.

Трассировка путей, вычисляющая интеграл освещенности, очень медленно сходится (скорость сходимости пропорциональна квадратному корню из числа просчитанных путей) и дает характерный шум.

Трассировщик постоянно производит выборку (sampling) пикселей изображения, и чем больше выборок, тем качественнее получается изображение (как правило, для обычных изображений достаточно около 5000 выборок). Огромное преимущество данного метода заключается в том, что можно визуализировать промежуточный результат. Картинка рассчитывается постепенно, и если она устраивает пользователя, алгоритм можно остановить. На рис.29 приведен соответствующий пример (https://en.wikipedia.org/wiki/Path_tracing).

В связи с популярностью, у данного метода развиваются различные модификации, в частности, двунаправленная трассировка путей и алгоритм Метрополиса.

Пример псевдокода для реализации трассировки путей приведен в приложении.

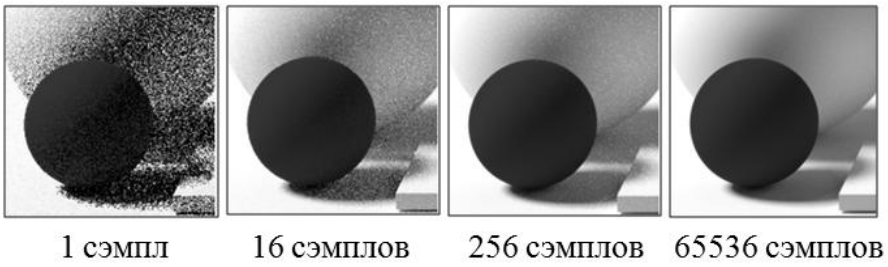


Рис.29. Трассировке путей: пример влияния числа сэмплов на пиксель.

ДВУНАПРАВЛЕННАЯ ТРАССИРОВКА

Одно их популярных направлений развития трассировки путей – это метод двунаправленной трассировки (Bidirectional path tracing), который был предложен Эриком Лафортом (Eric Lafortune) в 1993 г.

Двунаправленная трассировка путей комбинирует вклады от прямой и обратной трассировки при помощи многократной выборки по значимости (Multiple Importance Sampling).

Идея метода заключается в том, чтобы генерировать и объединять два подпути (начиная от источника света и от камеры) в один путем соединения каждого префикса одного подпути с каждым суффиксом другого подпути. Из полученных путей выбираются дающие наибольший вклад в расчет интеграла освещенности

.

АЛГОРИТМ МЕТРОПОЛИСА

Другое популярное направление развития трассировки путей – это метод Метрополиса (Metropolis light transport), который был представлен Эриком Вичем (Eric Veach) и Леонидасом Гимбасом (Leonidas J. Guibas) в 1997 г.

Суть метода заключается в возмущении ранее найденных путей для увеличения производительности в сложных сценах.

Идея этого алгоритма аналогична идее генетических алгоритмов. Для каждого пикселя и источника света алгоритм выбирает некий случайный начальный путь w_0, по которому свет может дойти от источника света до глаза. Затем генерируется последовательность путей $w_0, w_1, \dots w_m$. В этой последовательности каждый следующий путь получен из предыдущего с помощью случайных мутаций. При этом мутации могут приниматься или отвергаться в зависимости от того, какой вклад вносит результирующий путь w_{i+1} в изображение по сравнению с w_i.

МЕТОДЫ ОБЪЕМНОЙ ТРАССИРОВКИ

Трассировка пучков (Beam tracing) – это модификация метода трассировки лучей, в которой лучи заменяются объемными пучками в форме пирамид с многоугольным поперечным сечением. Данный вариант трассировки лучей впервые предложили Пол Хекберт (Paul Heckbert) и Пэт Ханрахан (Pat Hanrahan) в 1984 г.

Трассировка конусов (Cone tracing) – это похожая модификация метода трассировки лучей, в которой лучи заменяются объемными пучками в форме конусов. Трассировку конусов впервые предложил Джон Аманатидес (John Amanatides) в 1984 г.

Схематично, оба метода представлены на рис.30. Как и в трассировке лучей, при столкновении с отражающей или преломляющей поверхностью, создается и начинает трассироваться новый объемный луч.

Оба варианта предназначены для ситуаций, когда объекты представляются в виде полигонов, в противном случае возникают серьезные проблемы учета в объеме сложных поверхностей.

Поскольку в обеих модификациях луч объемный, неограниченной длины (высоты), то при столкновении с полигоном происходит проверка покрытия: если основание пучка только частично попало в полигон, процесс распространения данного луча продолжается дальше, а сам полигон (с цветом и долей попадания) запоминается в стеке. Результирующий цвет учитывается через взвешенную сумму цветов всех полигонов из стека.

На рис.31 приставлены две возможные ситуация: когда пятно пучка полностью укладывается в полигон (в этом случае пиксель закрашивается цветом полигона) и когда пятно пучка поровну укладывается в два полигона на одинаковой глубине (в этом случае пиксель закрашивается полусуммой цветов полигонов). Но возможны также и ситуации, когда пучок пересекает полигоны, расположенные на разном удалении от наблюдателя, и в разной пропорции.

Такой вид реализации используется редко, поскольку задействованные геометрические процессы намного сложнее и, следовательно,

дороже, чем просто пропускание большего количества лучей через пиксель

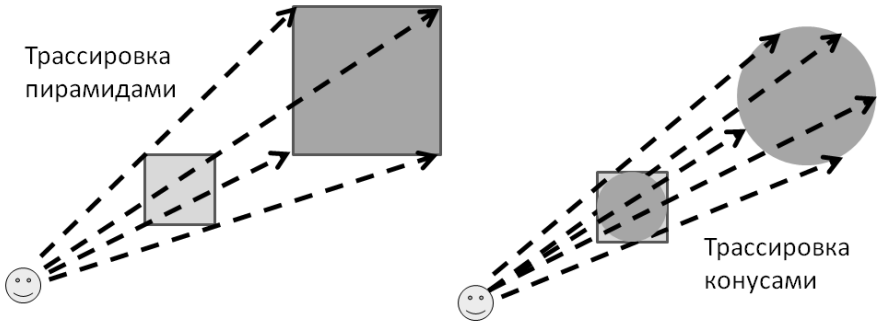


Рис.30. Объемная трассировка через пиксель.

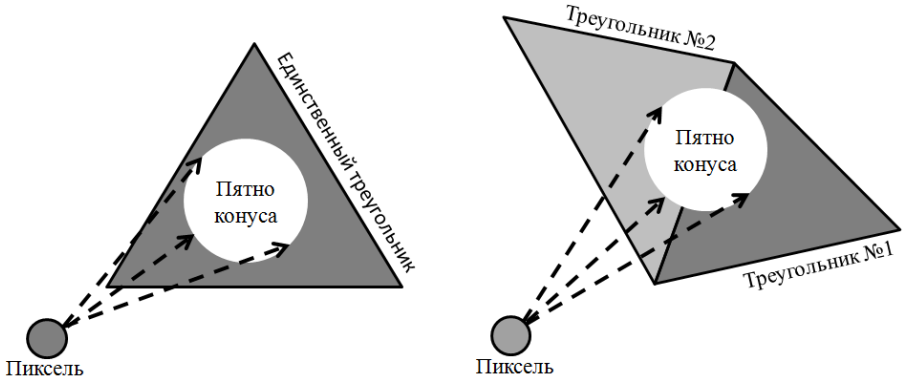


Рис.31. Трассировка конусов: две ситуации.

МЕТОД ФОТОННЫХ КАРТ

Метод фотонных карт (Photon Mapping) является двунаправленным методом, поскольку комбинирует прямую и обратную трассировку лучей.

Фотоны в данном методе – это частицы, переносящие некоторую дискретную порцию световой энергии.

На начальном этапе фотоны испускаются из источника света в соответствии с распределением световой энергии у данного источника, часто этот процесс реализуется стохастическим методом Монте-Карло. В процессе трассировки, фотоны ударяются о различные поверхности сцены (результат всех этих операций сохраняется в специальной фотонной карте), и, в зависимости от свойств материала, фотон может:

- отразиться диффузно (в случайном направлении),
- отразиться зеркально,
- преломиться через поверхность,
- поглотиться поверхностью,
- покинуть сцену.

На втором этапе лучи зрения испускаются из камеры, и при попадании луча на поверхность интенсивность в точке рассчитывается через ближайшие значения в фотонной карте.

Важным достоинством метода является тот факт, что расчеты не зависят от положения камеры, что позволяет во многих случаях рассчитать освещенность всего один раз. Соответственно, первый этап может выполняться очень долго (в зависимости от требований к качеству освещения), зато второй этап выполняется очень быстро (поскольку просчитываются лишь первичные лучи).

Стоит отметить, что в общем случае фотонная карта дает менее точную аппроксимацию освещенности сцены, чем другие методы, особенно при наличии мелких объектов.

МЕТОД РАСПРЕДЕЛЕННОЙ ТРАССИРОВКИ ЛУЧЕЙ

Чтобы получить такие эффекты, как мягкие тени, и визуализировать материалы из реальной жизни, обладающие нетривиальными свойствами, необходимо использовать более сложный метод, чем обратная трассировка лучей.

Одним из таких методов и является метод распределенной трассировки лучей (distributed ray tracing).

В реальности источники света часто не точечные (имеют размер), а материалы не всегда отражают всю энергию в одном направлении по закону “угол падения равен углу отражения”. Такое свойство может быть связано, например, с микрорельефом поверхности или особыми свойствами покрытия.

Из-за того что источник света имеет размер, границы перехода свет-тень получаются нечеткими, потому что на этих границах затеняющим объектом закрыта только часть источника света (рис.32). Чтобы вычислить, какая часть источника света закрыта, до него трассируют некоторое количество теневых лучей с направлением на случайную точку на источнике освещения..

Аналогично поступают, если у материала сложные отражающие свойства – испускается несколько отраженных лучей в разных направлениях, а результат осредняется.

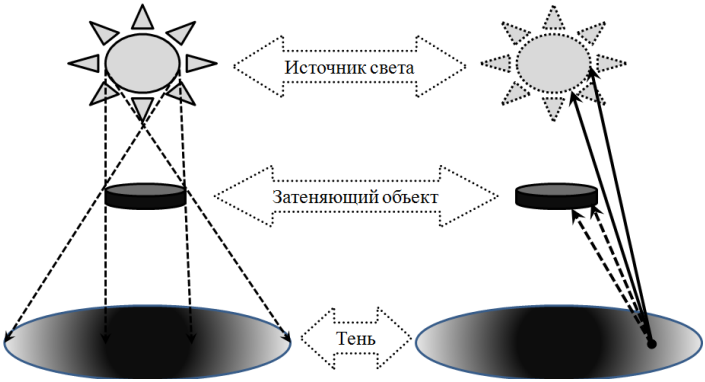


Рис.32. Нечеткие тени.

ПРИЛОЖЕНИЕ А.
ПРИМЕР РЕАЛИЗАЦИИ АЛГОРИТМА МОЛЛЕРА-ТРУМБОРА

```
#include <glm/glm.hpp>
using namespace::glm;

/*
 orig и dir - начало и направление луча
 v0, v1, v2 - вершины треугольника.
*/

/* Функция возвращает расстояние от начала луча до точки пересечения
или 0 */
float triangle_intersection(const vec3& orig,
                            const vec3& dir,
                            const vec3& v0,
                            const vec3& v1,
                            const vec3& v2)
 {
    vec3 e1 = v1 - v0;
    vec3 e2 = v2 - v0;

 // Вычисление вектора нормали к плоскости
    vec3 pvec = cross(dir, e2);
    float det = dot(e1, pvec);

    // Луч параллелен плоскости
    if (det < 1e-8 && det > -1e-8) return 0;
    float inv_det = 1 / det;
    vec3 tvec = orig - v0;
    float u = dot(tvec, pvec) * inv_det;
    if (u < 0 || u > 1) return 0;
    vec3 qvec = cross(tvec, e1);
    float v = dot(dir, qvec) * inv_det;
    if (v < 0 || u + v > 1) return 0;

    return dot(e2, qvec) * inv_det;
}
```

ПРИЛОЖЕНИЕ Б.
ПРИМЕР РЕАЛИЗАЦИИ ТРАССИРОВКИ ПУТЕЙ

```
Color TracePath(Ray ray, count depth)
{
    if (depth >= MaxDepth) return Black;  // Bounced enough times

    ray.FindNearestObject();
    if (ray.hitSomething == false) return Black; // Nothing was hit
    Material material = ray.thingHit->material;
    Color emittance = material.emittance;
    // Pick a random direction from here and keep going.
    Ray newRay; newRay.origin = ray.pointHit;
    // This is NOT a cosine-weighted distribution!
    newRay.direction = RandomUnitVector(ray.normalHit);
    // Probability of the newRay
    const float p = 1 / (2 * M_PI);
    // Compute the BRDF for this ray (Lambertian reflection)
    float cos_theta = DotProduct(newRay.direction, ray.normalHit);
    Color BRDF = material.reflectance / M_PI;
    // Recursively trace reflected light sources.
    Color incoming = TracePath(newRay, depth + 1);
    // Apply the Rendering Equation here.
    return emittance + (BRDF * incoming * cos_theta / p);
  }

  void Render(Image finalImage, count numSamples)
  {
    foreach (pixel in finalImage)
     {
      foreach (i in numSamples)
      {
        Ray r = camera.generateRay(pixel);
        pixel.color += TracePath(r, 0);
      }
      pixel.color /= numSamples;  // Average samples.
    }
  }
```

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Задорожный А. Г.* Введение в двумерную компьютерную графику с использованием библиотеки OpenGL : учебное пособие / А. Г. Задорожный, Д. В. Вагин, Ю. И. Кошкина. – Новосибирск : Изд-во НГТУ, 2018. – 102 с. – ISBN 978-5-7782-3601-1.

2. *Задорожный А. Г.* Введение в трехмерную компьютерную графику с использованием библиотеки OpenGL : учебное пособие / А. Г. Задорожный, М. Г. Персова, Ю. И. Кошкина. – Новосибирск : Изд-во НГТУ, 2018. – 99 с. – ISBN 978-5-7782-3744-5.

3. *Задорожный А. Г.* Модели освещения и алгоритмы затенения в компьютерной графике : учебное пособие / А. Г. Задорожный. – Новосибирск : Изд-во НГТУ, 2021. – 80 с. – ISBN 978-5-7782-4308-8.

4. *Хилл Ф.* OpenGL. Программирование компьютерной графики : пер. с англ. / Френсис Хилл. – 2-е изд. – Санкт-Петербург : Питер : Питер принт, 2002. – 1081 с. : ил. – (Для профессионалов). – ISBN 5-318-00219-6.

СОДЕРЖАНИЕ

Задорожный Александр Геннадьевич

КОМПЬЮТЕРНАЯ ГРАФИКА: ВВЕДЕНИЕ В ТРАССИРОВКУ ЛУЧЕЙ

Учебное пособие

В авторской редакции

Выпускающий редактор И.П. Брованова
Дизайн обложки А.В. Ладыжская

Налоговая льгота – Общероссийский классификатор продукции
Издание соответствует коду 95 3000 ОК 005-93 (ОКП)

Подписано в печать 15.12.2021. Формат 60 × 84 1/16. Бумага офсетная.
Тираж 100 экз. Уч.-изд. л. 3,72. Печ. л. 4,0. Изд. №241/20. Заказ № 26
Цена договорная

Отпечатано в типографии
Новосибирского государственного технического университета
630073, г. Новосибирск, пр. К. Маркса, 20