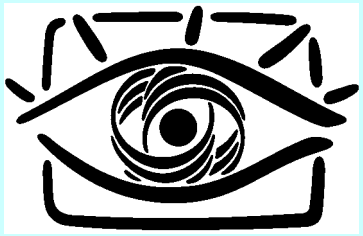


Лекция 13

Синтез изображений (Rendering)

Излучательность (Radiosity)

Трассировка лучей (Ray Tracing)

Экранизация

В этой лекции рассматривается процесс получения изображения из набора трехмерных объектов, источников света и положения камеры (глаза). Этот процесс называется *экранизацией* (rendering).

В области глобального освещения есть два основных алгоритма: излучательность (radiosity) и трассировка лучей (ray tracing). В основе этих алгоритмов лежит решение уравнения переноса лучистой энергии.

Для решения этой задачи необходимо понимать систему зрения, цифровую обработку сигналов, а также физику.

Освещение и отражение

В реальном мире видимый цвет определяется количеством энергии видимого спектра, которая попадает на чувствительные элементы сетчатки глаза

Как правило, мы видим отраженный свет, т.е. энергию, отраженную материалом поверхности.

Отражающие характеристики поверхности определяются отражающими способностями по отношению к волнам различной длины

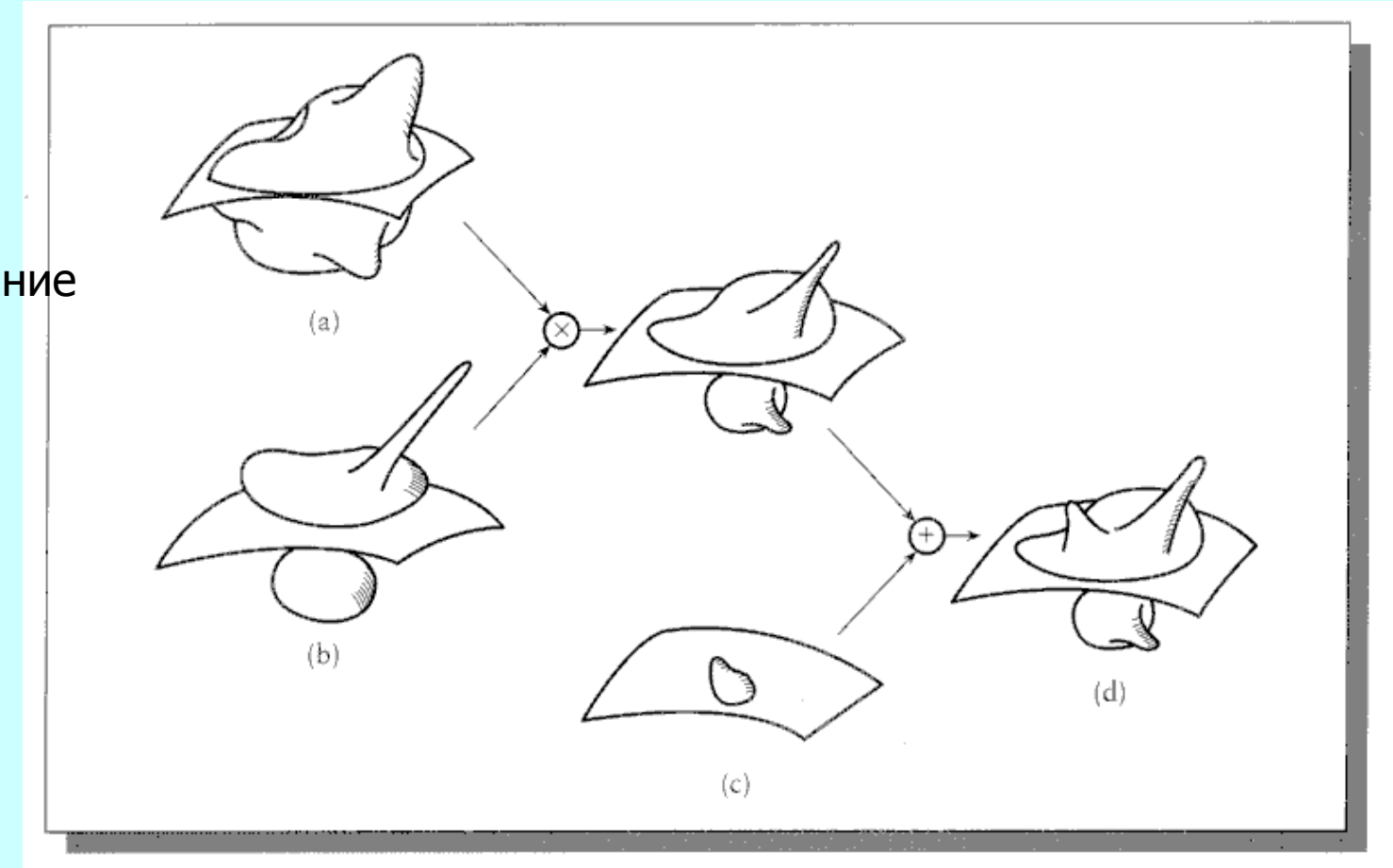
Функции освещения и отражения

(a) Освещение
Illumination

(b) Светорассеивание
BRDF

(c) Эмиссия
Emission

(d) Излучение
Radiance



Уравнение отражения

$$L^r(\vec{\omega}_r) = \int_{\Omega_i} f_r(\vec{\omega}_i \to \vec{\omega}_r) L^i(\vec{\omega}_i)\, d\vec{\omega}_i^{\mathbf{N}}$$

$$= \int_{\phi_i=0}^{2\pi} \int_{\theta_i=0}^{\pi/2} f_r((\theta_i, \psi_i) \to (\theta_r, \psi_r)) L^i(\theta_i, \psi_i) \cos\theta_i \sin\theta_i \, d\theta_i \, d\psi_i$$

$$dA = (r \sin\theta \, d\psi)(r \, d\theta) = r^2 \sin\theta \, d\theta \, d\psi$$

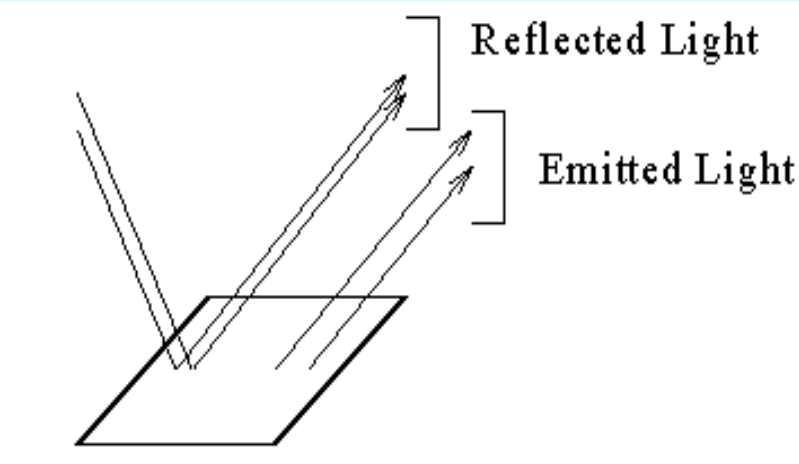
Излучательность

(интегральная светимость)

(Radiosity)

- radiosity *n*. : An illumination algorithm for graphics that accurately computes the effects of diffuse illumination, including multiple diffuse reflections from surfaces, through an energy balancing technique.

Основная идея



- Поделить каждую поверхность в сцене (включая источники) на малые куски
- Построить систему линейных уравнений

B_i = излуч-свет + отраж-свет($\Sigma_j B_j$)

- Решить систему уравнений, найти (RGB) всех кусков
- Экранизировать сцену, используя любой метод, согласованный с закраской Гуро

Обозначения

Radiosity = out-going power per unit area (Watt/meter^2)

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{j-i} \frac{A_j}{A_i}$$

where:

B_i = Radiosity of patch i

B_j = Radiosity of patch j

E_i = Light emitted by patch i

ρ_i = Fraction of incident light that is reflected

F_{j-i} = "Form factor": the faction of energy leaving j that arrives at i

A_i = Area of patch i

A_j = Area of patch j

The **form factor** takes into account the shape and relative orientation of the two patches, and the presence of any obstructing patches.

Форм-факторы

The form factor is symmetric:

$$A_i F_{i-j} = A_j F_{j-i}$$

This simplifies the equation to

$$B_i = E_i + \rho_i \sum_j B_j F_{i-j}$$

or, rearranging terms,

$$B_i - \rho_i \sum_j B_j F_{i-j} = E_i$$

which can be viewed as restating the conservation of energy.

Система линейных уравнений

This gives us the following system of simultaneous equations:

$$\begin{bmatrix} 1-\rho_1 F_{1-1} & -\rho_1 F_{1-2} & \cdots & -\rho_1 F_{1-n} \\ \rho_2 F_{2-1} & 1-\rho_2 F_{2-2} & \cdots & -\rho_2 F_{2-n} \\ . & . & \cdots & . \\ . & . & \cdots & . \\ . & . & \cdots & . \\ \rho_n F_{n-1} & -\rho_n F_{n-2} & \cdots & 1-\rho_n F_{n-n} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ . \\ . \\ . \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ . \\ . \\ . \\ E_n \end{bmatrix}$$

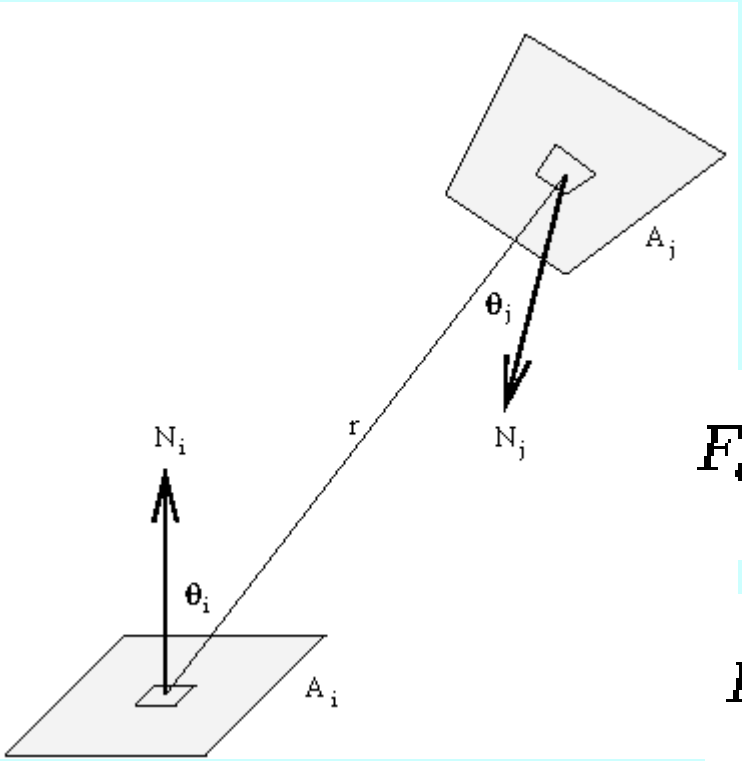
Given the E_i and F_{i-j}, we can solve for the B_i by Gaussian elimination.

Свойства системы линейных уравнений

- ρ_i и E_i зависят от длины волны (RGB).
- $F_{ii} = 0$ для плоских и выпуклых кусков.
- Сумма форм-факторов в каждой строке равна 1. Закон сохранения энергии.
- Поскольку $r < 1$, то матрица с доминирующей диагональю.

Коэффициенты формы

The expensive part of this process is computing the form factors, F_{i-j}. We must compute a form factor for every pair of surface patch



$$dF_{di-dj} = \frac{\cos\theta_i \cos\theta_j}{\pi r^2} H_{ij} dA_j$$

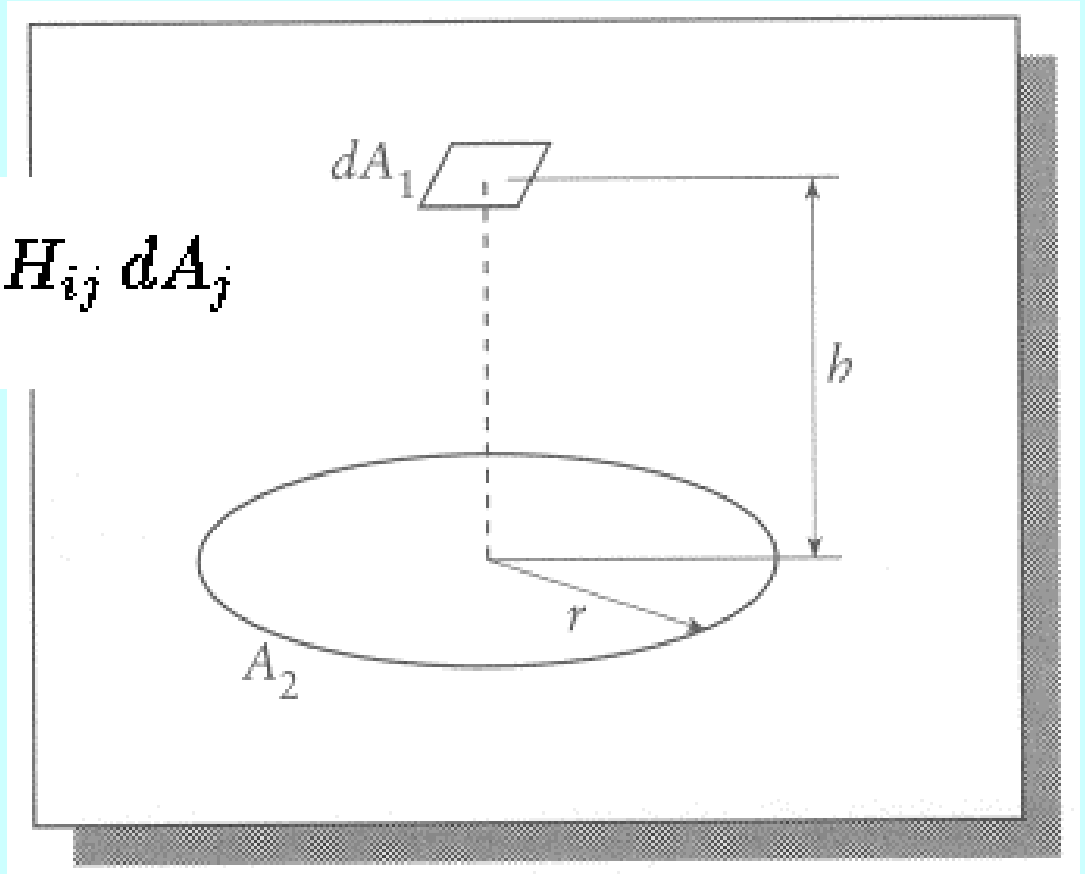
$$H_{ij} = \begin{cases} 1 & \text{if } dA_j \text{ is visible from } dA_i \\ 0 & \text{otherwise} \end{cases}$$

$$F_{di-j} = \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} H_{ij}\, dA_j$$

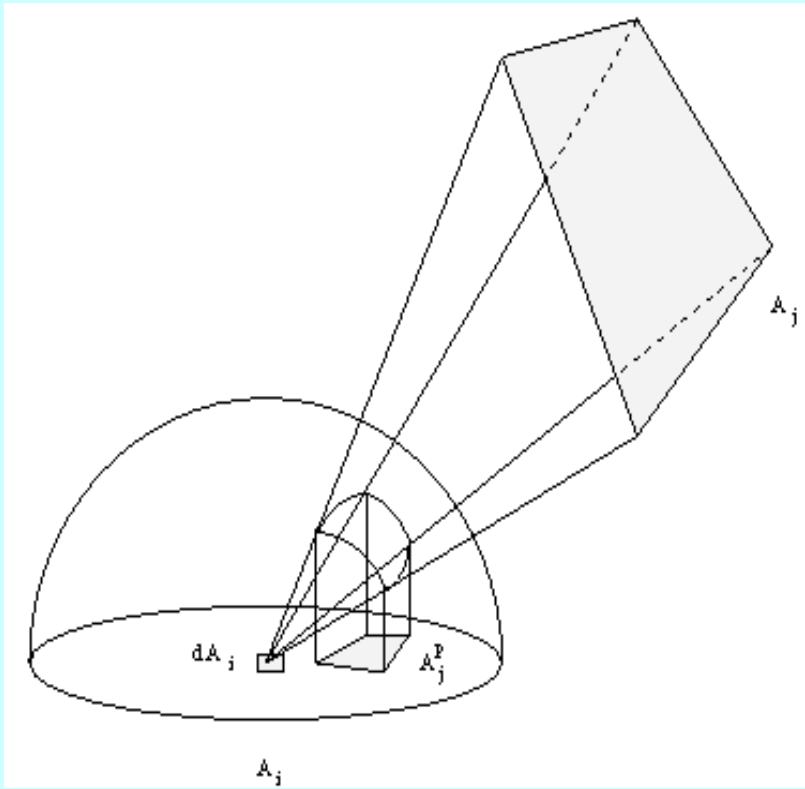
$$F_{i-j} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} H_{ij}\, dA_j dA_i$$

Аналитические методы вычисления форм-факторов

$$F_{di-j} = \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} H_{ij}\, dA_j$$



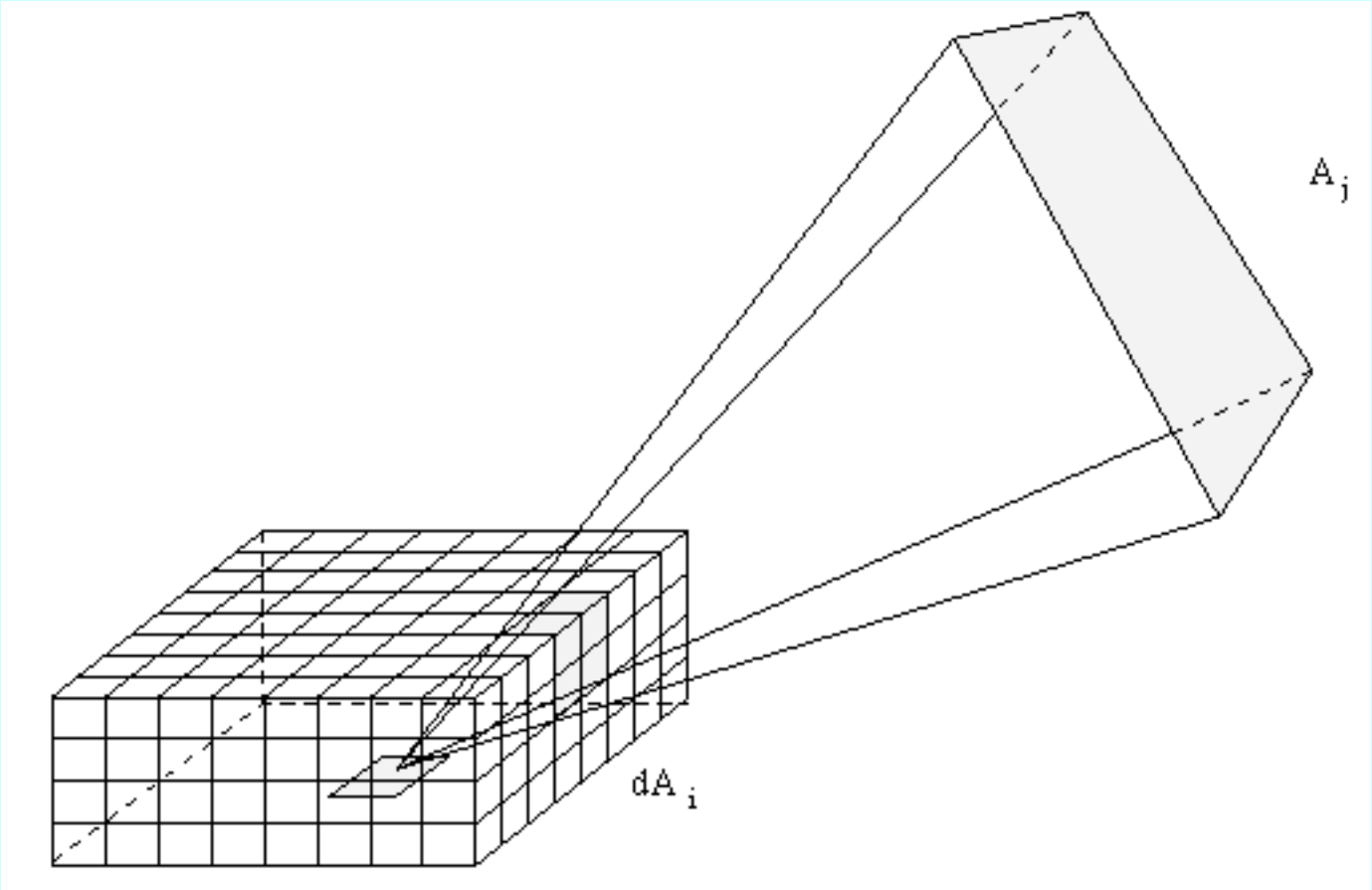
Геометрическая интерпретация



- Построить единичную полусферу над куском A_i
- Спроецировать видимую часть куска A_j на эту сферу
- Спроецировать полученный сферический кусок на основание полусферы и получить A^P_j
- Форм-фактор

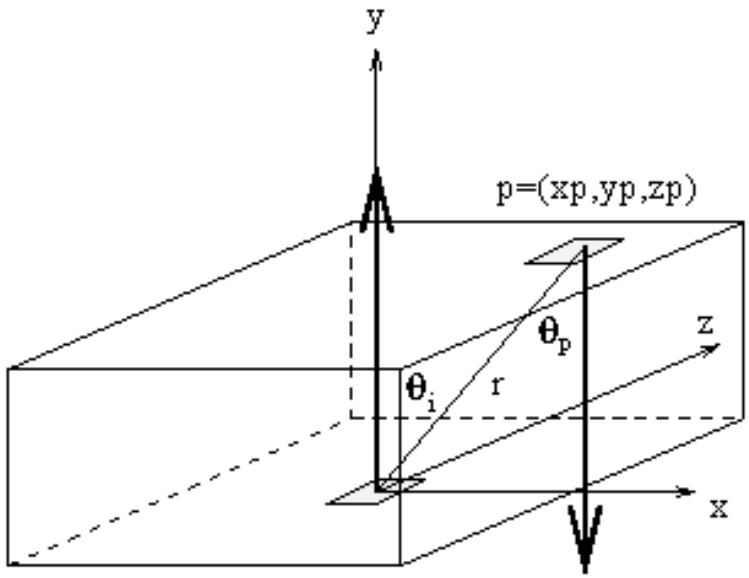
$$F_{di-j} = A^P_j/\pi.$$

Аппроксимация полусферы полукубом



Алгоритм вычисления форм-фактора

- Position unit hemicube over patch.
- Project A_j onto the cells of the hemicube.
- Sum the contribution ΔF_p of each projected cell on hemicube to approximate F_{di-j}.
- Assuming r is much larger than patch sizes, F_{di-j} is a good approximation of F_{i-j}.



The contribution of each hemicube cell is

$$\Delta F_p = \frac{\cos\theta_i \theta_p}{\pi r^2} \Delta A$$

where:

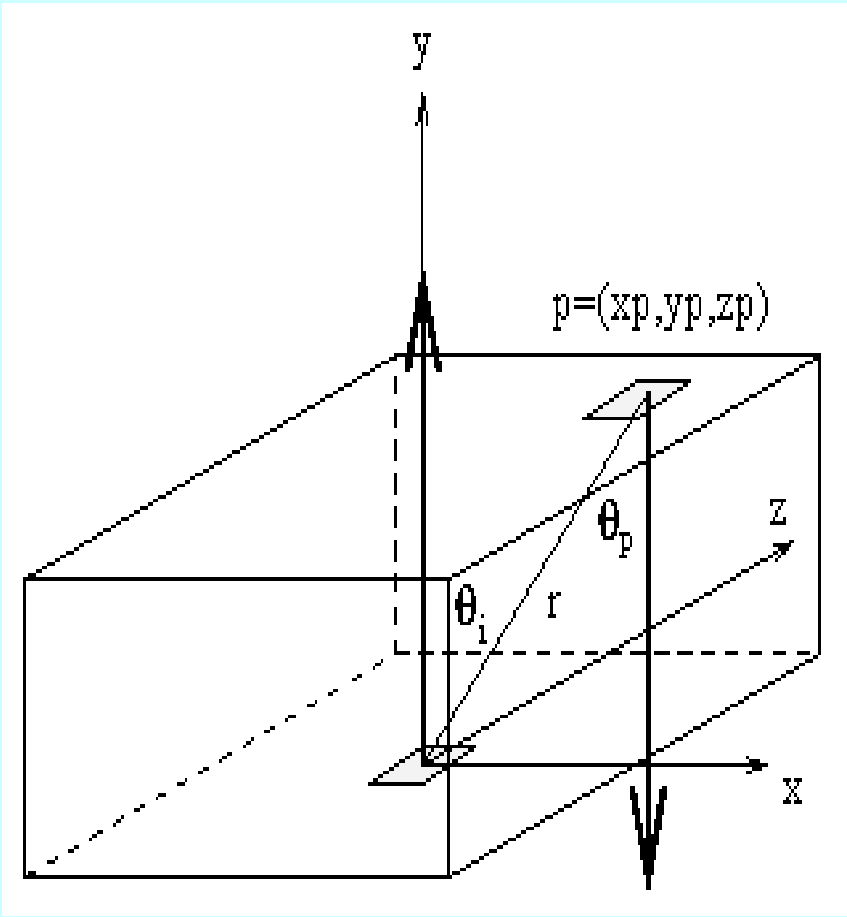
r = Length of line connecting patch center to cell

θ_i = Angle between patch normal and line

θ_p = Angle between inward cell normal and line

ΔA = Area of cell

Вклад элементов верхней грани полукуба



$$r = \sqrt{x_p^2 + 1 + z_p^2}$$

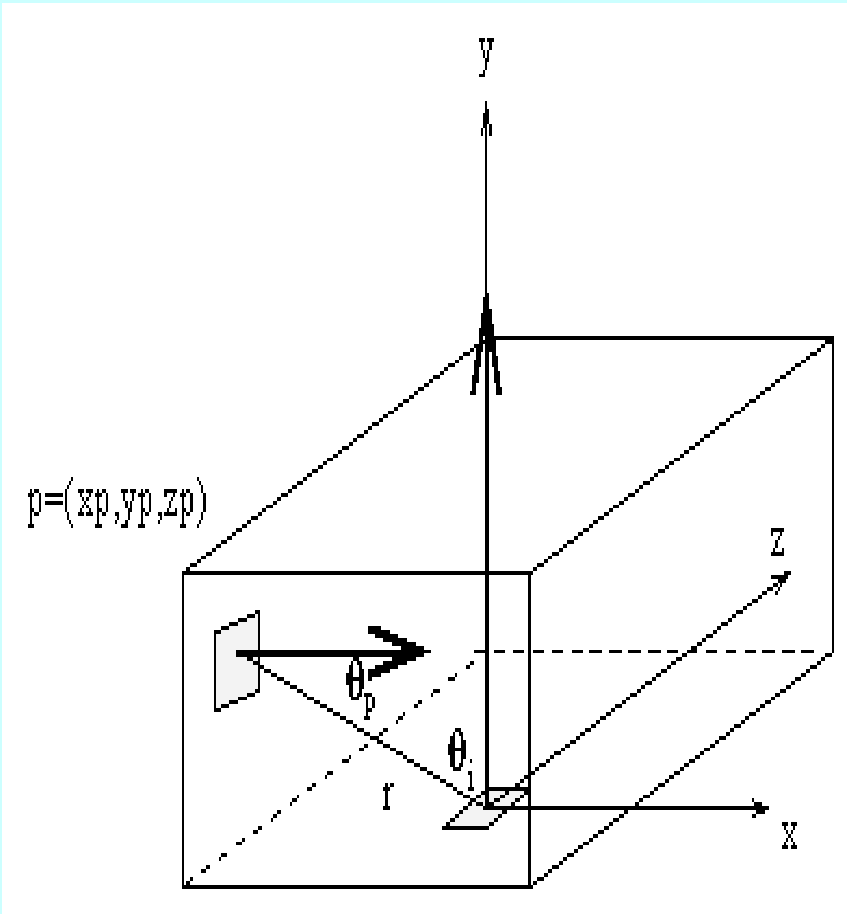
$$\cos\theta_i = (0,1,0)\cdot\frac{1}{r}(x_p,1,z_p) = \frac{1}{r}$$

$$\cos\theta_p = (0,-1,0)\cdot\frac{1}{r}(-x_p,-1,-z_p) = \frac{1}{r}$$

$$\Delta F_p = \frac{\cos\theta_i\cos\theta_p}{\pi r^2}\Delta A = \frac{\frac{1}{r}\frac{1}{r}}{\pi r^2}\Delta A$$

$$= \frac{1}{\pi r^4}\Delta A = \frac{1}{\pi(x_p^2+1+z_p^2)^2}\Delta A$$

Вклад элементов боковой грани полукуба



$$
\begin{aligned}
r &= \sqrt{1 + y_p^2 + z_p^2} \\
\cos\theta_i &= (0,1,0) \cdot \frac{1}{r}(-1, y_p, z_p) = \frac{y_p}{r} \\
\cos\theta_p &= (1,0,0) \cdot \frac{1}{r}(1, -y_p, -z_p) = \frac{1}{r} \\
\Delta F_p &= \frac{\cos\theta_i \cos\theta_p}{\pi r^2}\Delta A = \frac{\frac{y_p}{r}\frac{1}{r}}{\pi r^2}\Delta A \\
&= \frac{y_p}{\pi(1 + y_p^2 + z_p^2)^2}\Delta A
\end{aligned}
$$

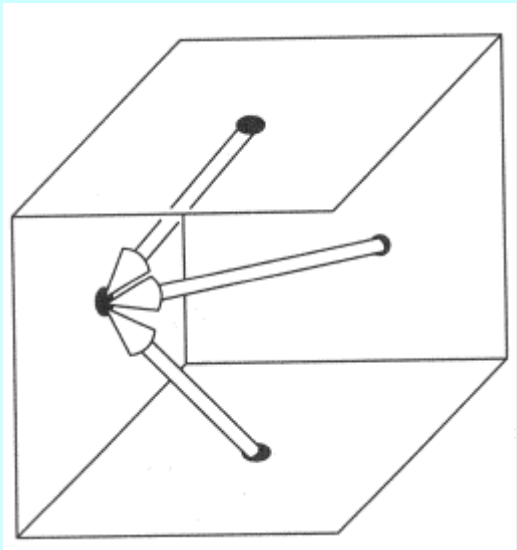
Решение системы линейных уравнений

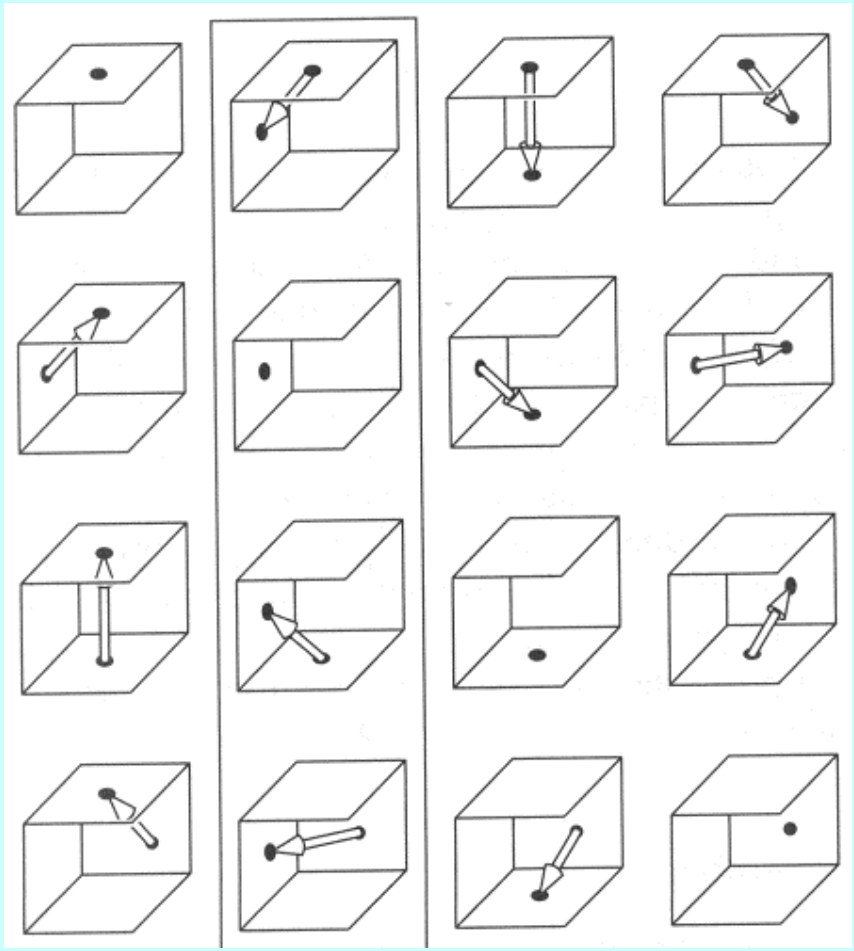
This gives us the following system of simultaneous equations:

$$\begin{bmatrix} 1-\rho_1 F_{1-1} & -\rho_1 F_{1-2} & \cdots & -\rho_1 F_{1-n} \\ \rho_2 F_{2-1} & 1-\rho_2 F_{2-2} & \cdots & -\rho_2 F_{2-n} \\ . & . & \cdots & . \\ . & . & \cdots & . \\ . & . & \cdots & . \\ \rho_n F_{n-1} & -\rho_n F_{n-2} & \cdots & 1-\rho_n F_{n-n} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ . \\ . \\ . \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ . \\ . \\ . \\ E_n \end{bmatrix}$$

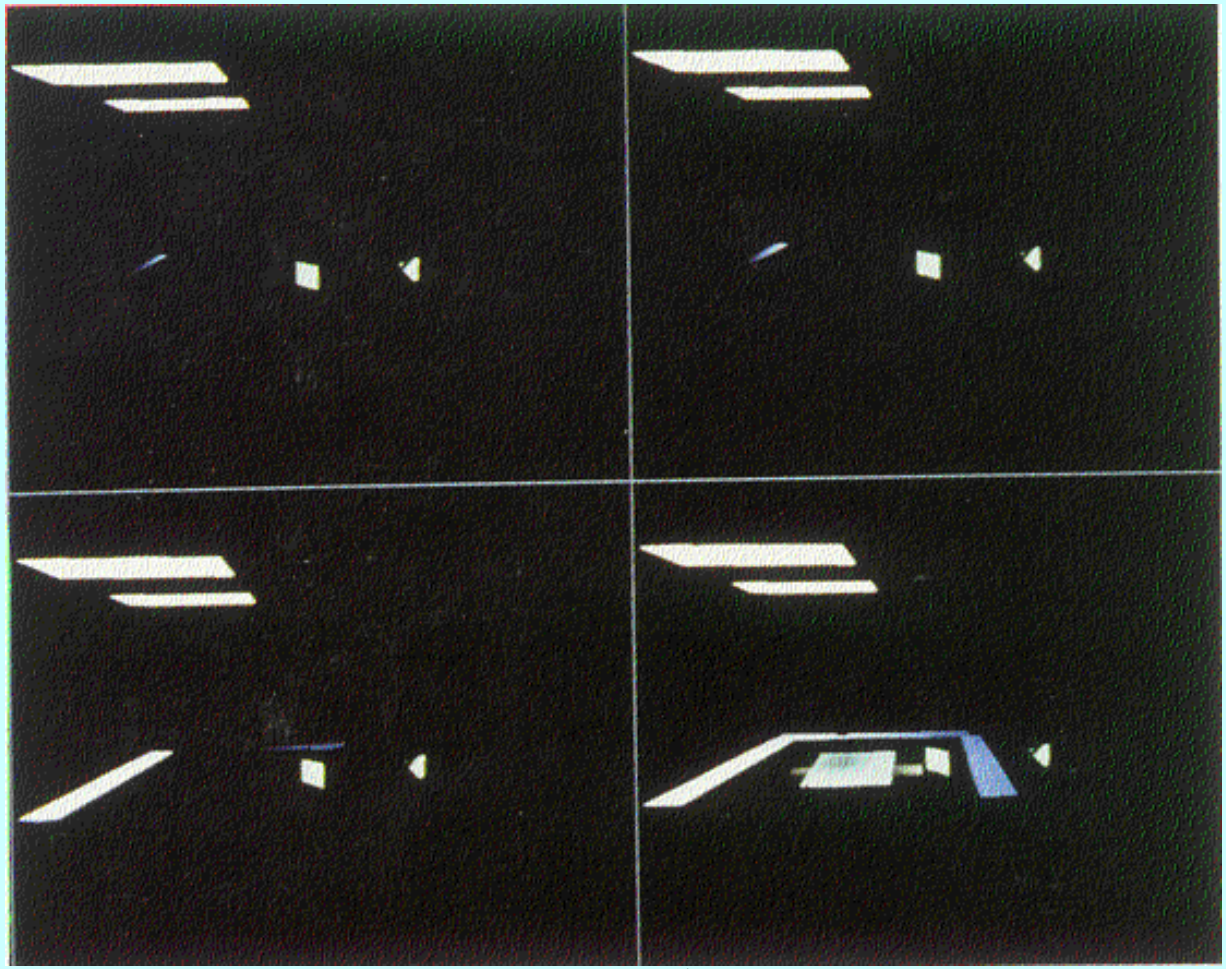
Given the E_i and F_{i-j}, we can solve for the B_i by Gaussian elimination.

Итерации Гаусса-Зейделя

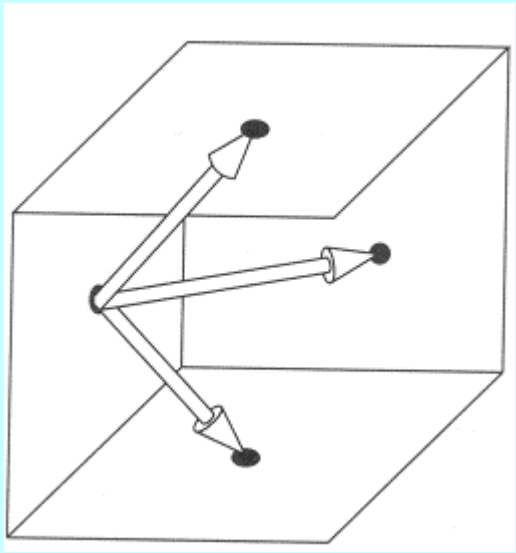


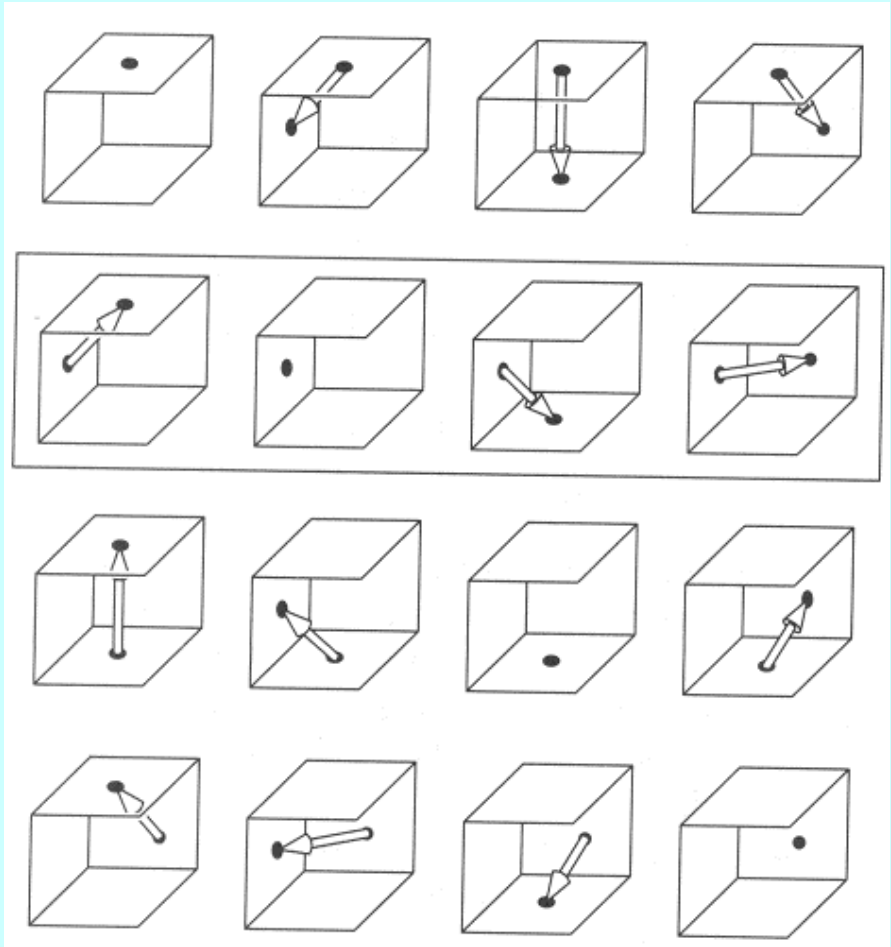


Итерации Гаусса-Зейделя (1, 2, 24, 100 шагов)

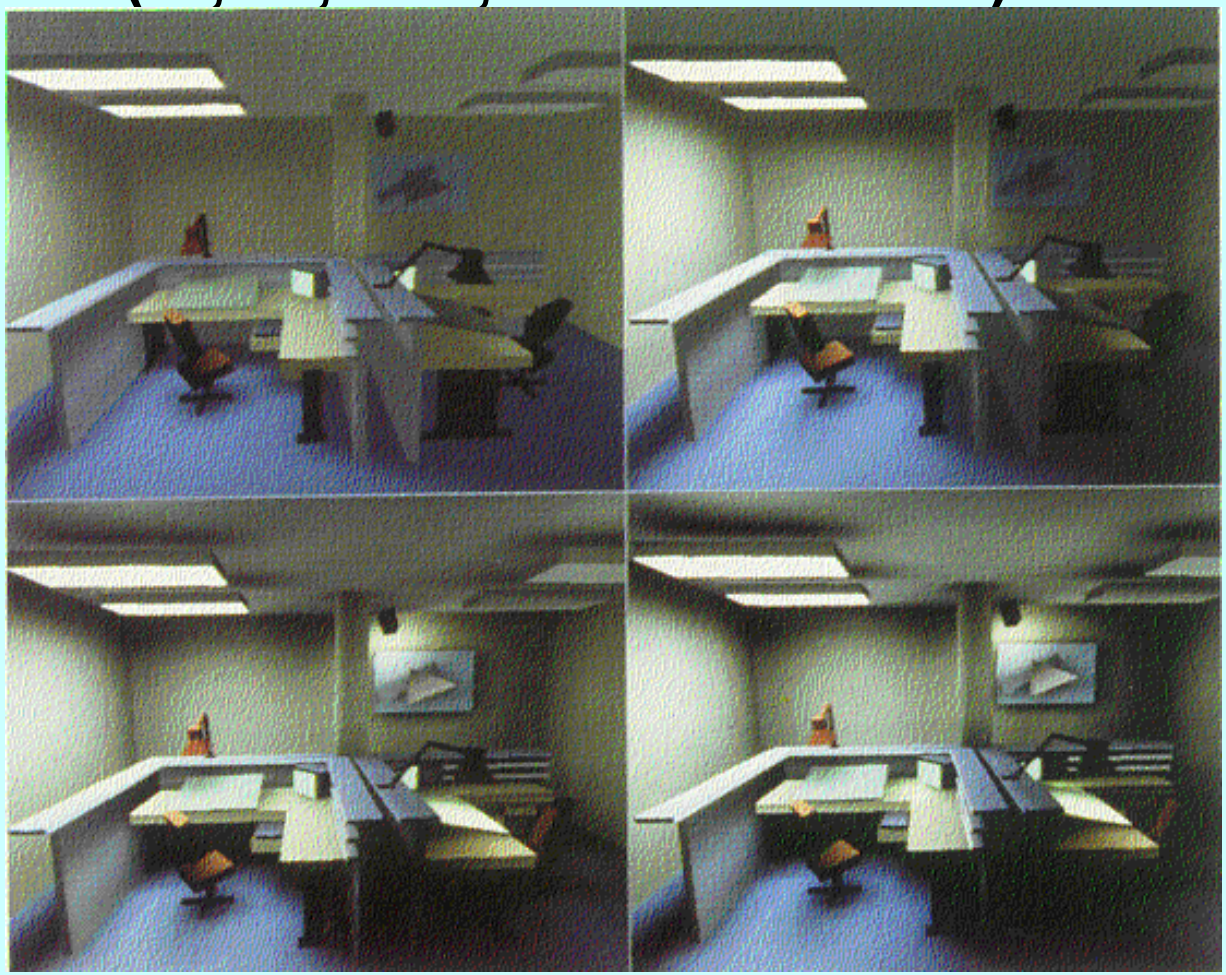


Итерации Саутвелла (Southwell)





Итерации Саутвелла (Southwell)
(1, 2, 24, 100 шагов)

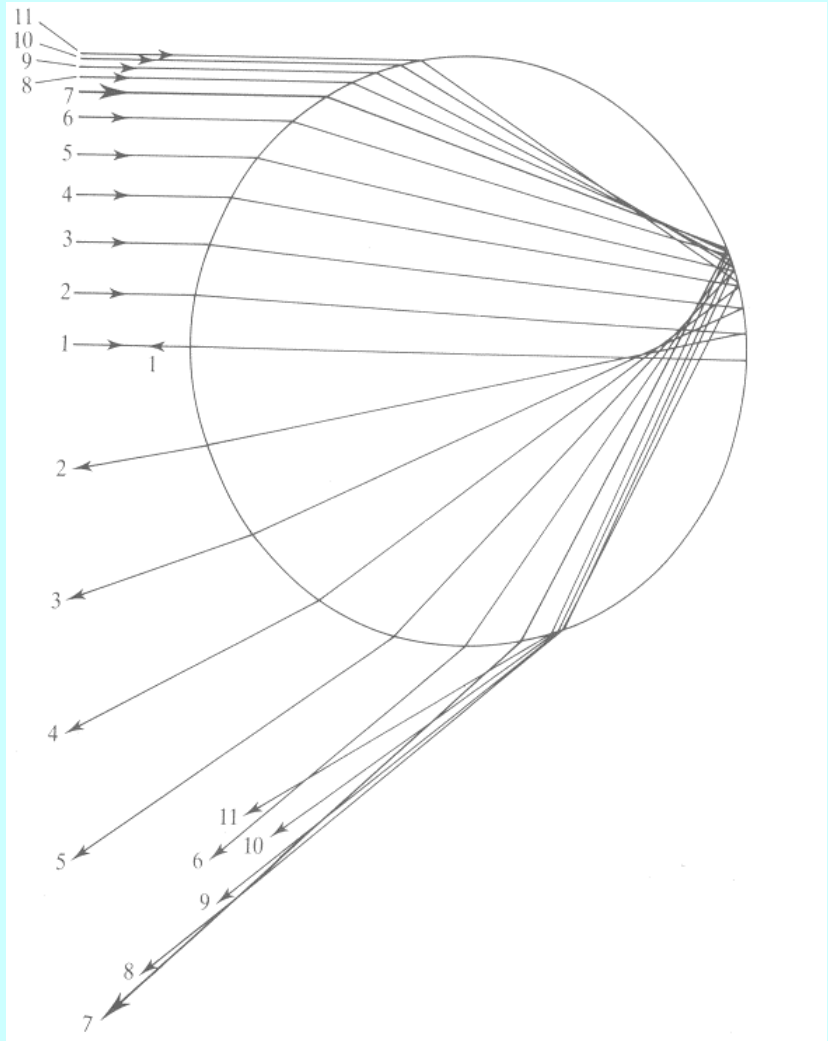


Трассировка лучей

Трассировка лучей

- ray tracing : An approach to rendering based upon simulating the reflection paths of light among the graphics objects to the viewpoint. Rays for each pixel are typically traced backward from the viewpoint and among objects until a light source or data base boundary is reached.

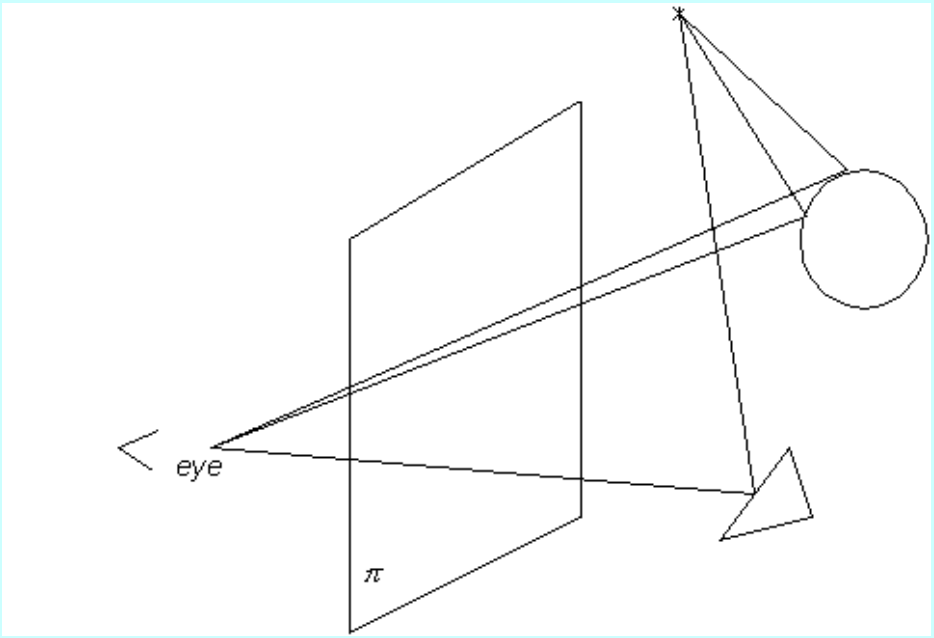
Трассировка лучей через сферическую каплю воды (луч 7 -- луч Декарта [1637])



Прямая трассировка лучей

Каждый источник света испускает лучи света в окружающее его пространство.

Эти лучи распространяются в пространстве и часть из них покидает сцену (не внося никакого вклада в возникающее изображение), а часть падает на поверхности различных объектов сцены



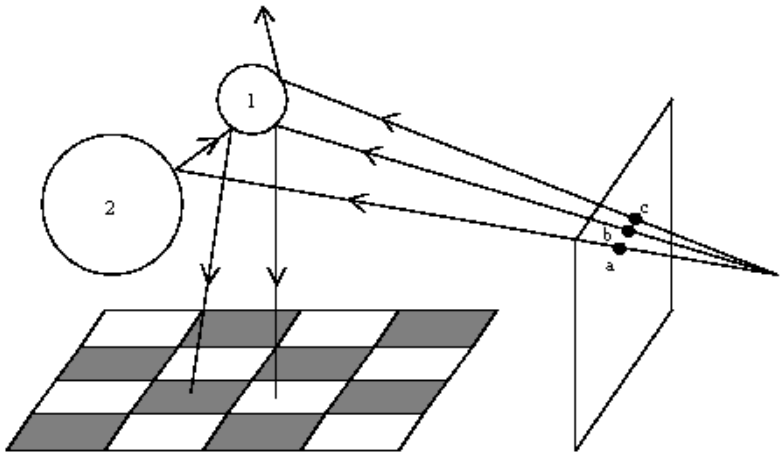
Попав на поверхность объекта каждый луч порождает вторичные отраженные и преломленные лучи

Часть лучей (очень небольшая!) попадает в глаз объектив камеры,
формируя там изображение сцены.

Обратная трассировка лучей

Будем отслеживать луч в обратном направлении – из объектива фотокамеры через заданный пиксел экрана.

Ray Tracing simulates this by **working backward** from the eye. We follow a ray from the eye until it strikes an object. If the object is shiny, we compute a reflection angle and follow the ray until it hits something else. Eventually, the ray will hit a light source or an object with only diffuse reflection.



Обратная трассировка лучей отслеживает только лучи, вносящие значительный вклад в изображение

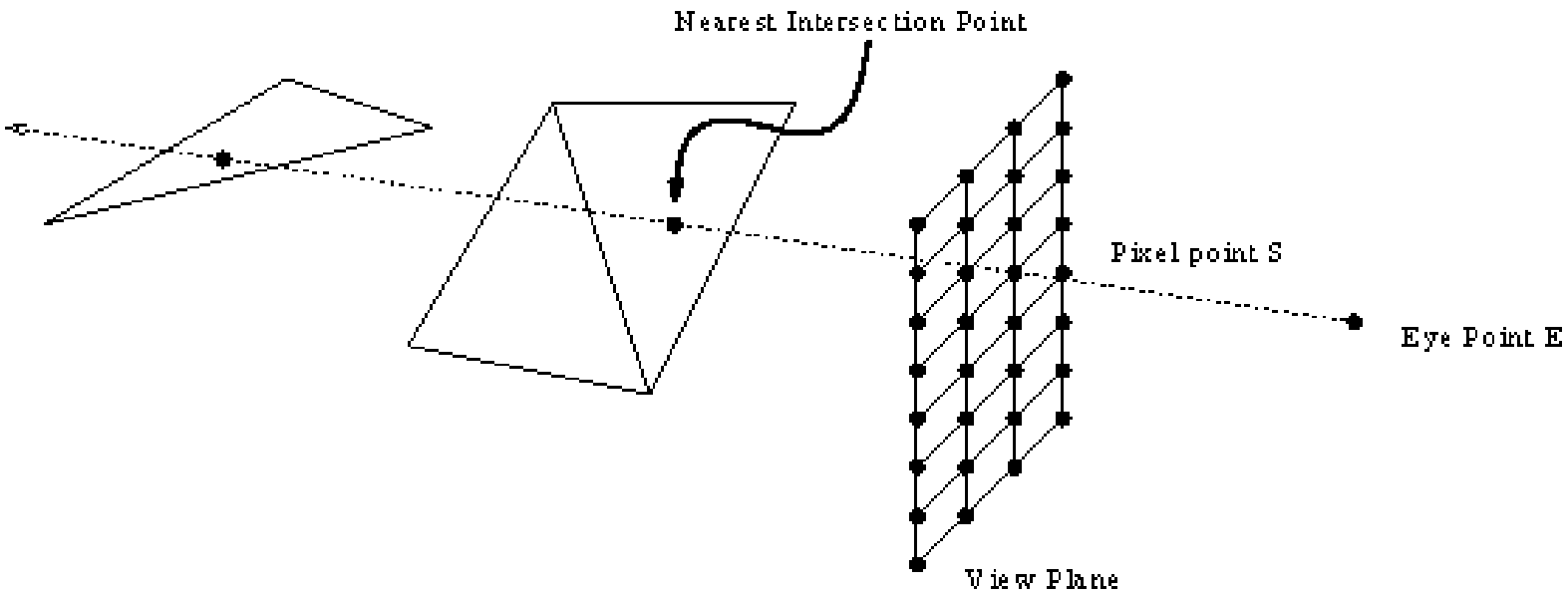
Алгоритм трассировки лучей

1. Через каждый пиксел картинной плоскости выпускается луч в сцену и ищется точка его ближайшего пересечения с объектами сцены. Из этой точки выпускаются лучи ко всем источникам света (для определения их видимости), также выпускаются отраженный и преломленный лучи.

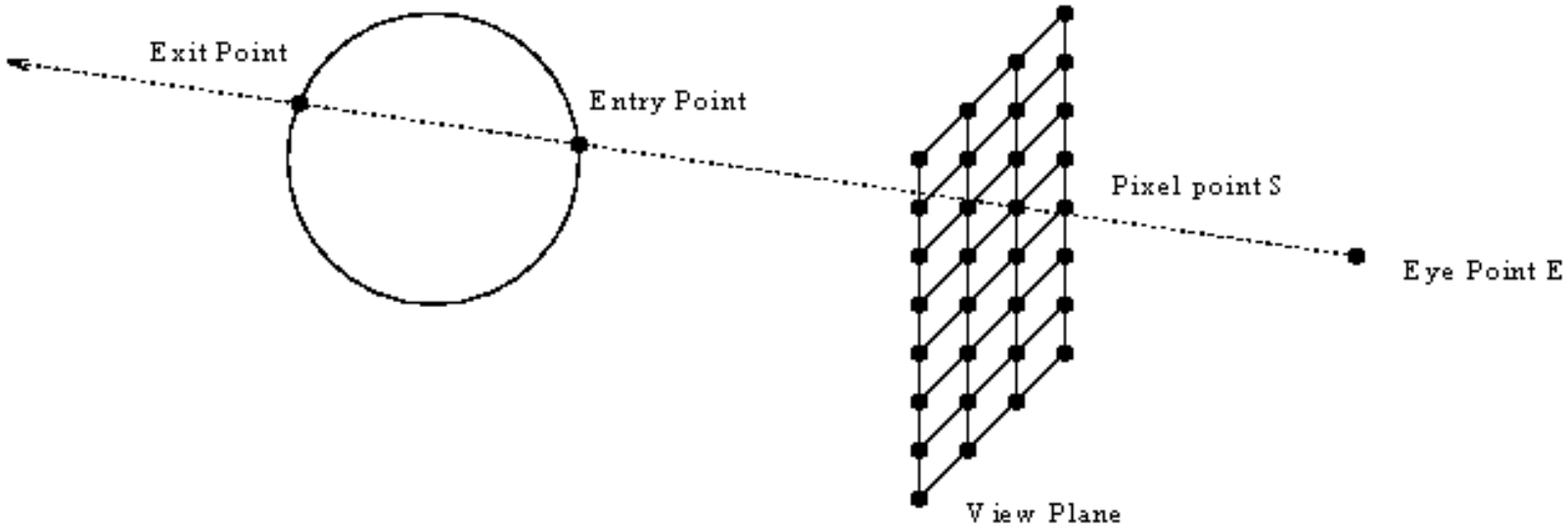
2. Для определения световой энергии, приходящей вдоль отраженного и преломленного лучей, каждый из этих лучей трассируется для определения точки ближайшего пересечения. Затем снова может потребоваться трассировки возникающих отраженных и преломленных лучей.

3. Критерии прекращения рекурсии: заданный уровень рекурсии или заданный вес луча.

Ray Tracing Polygons with Diffuse Reflection

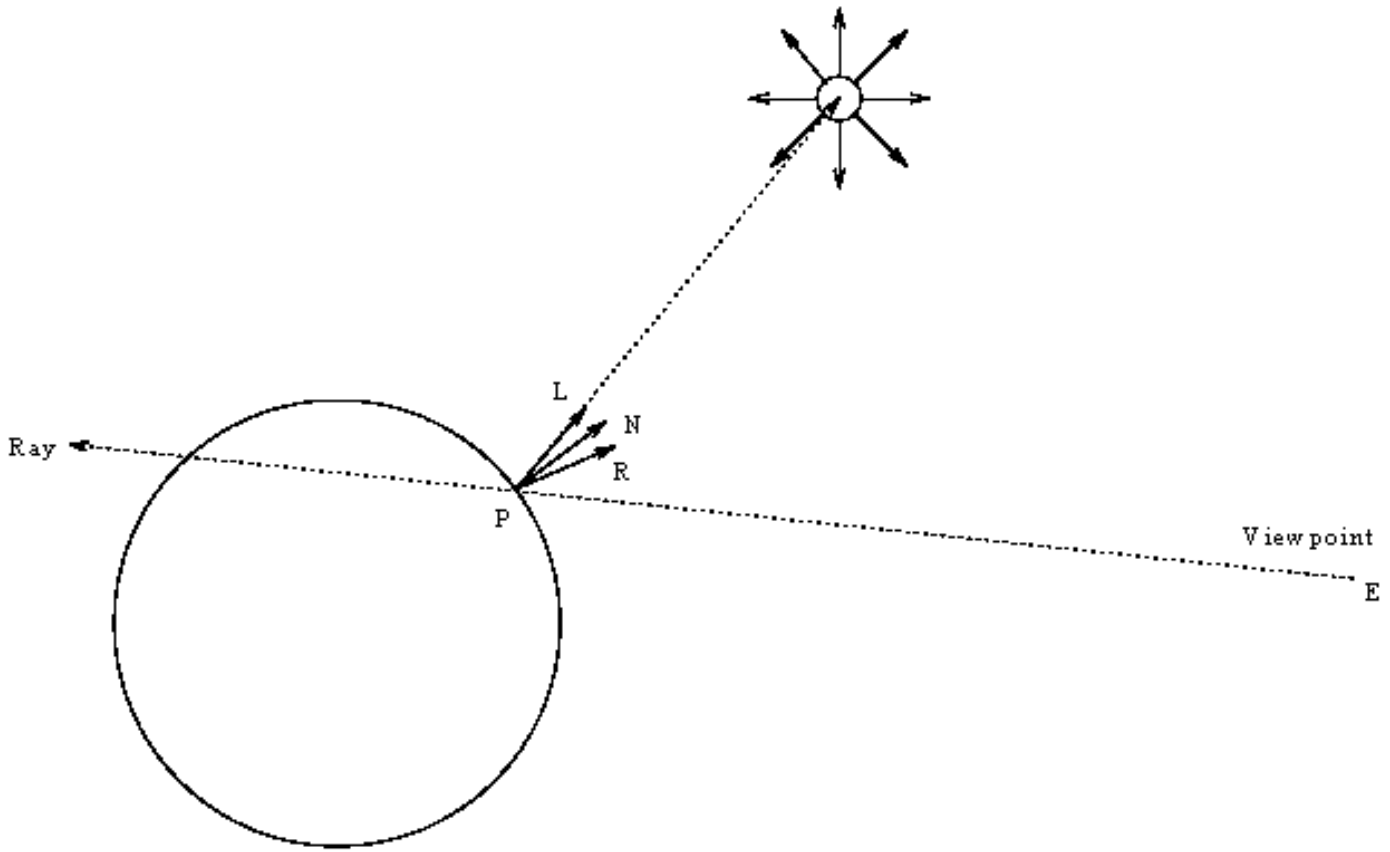


A Simple Example: Sphere with Diffuse Reflection



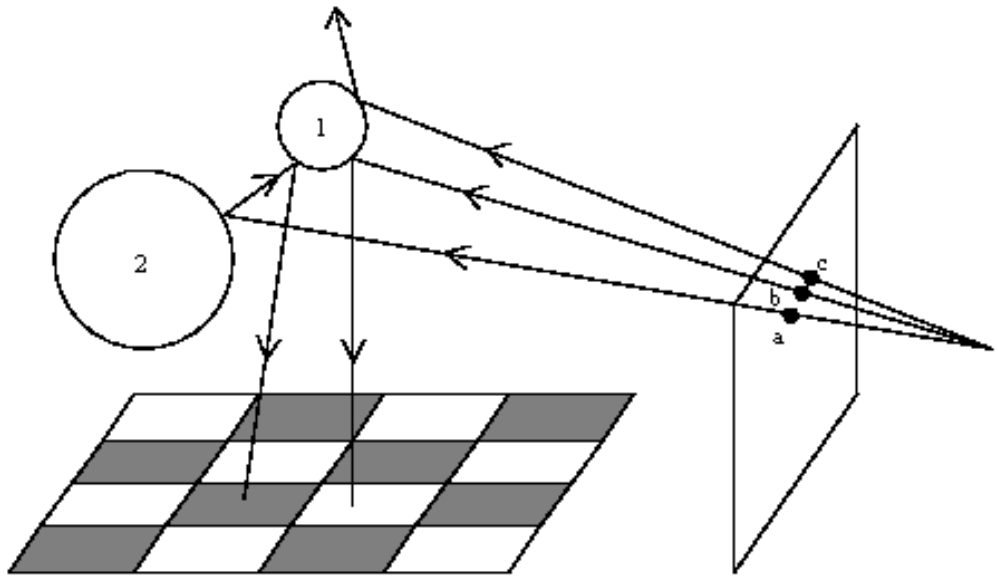
Soft Highlights from Partial Specular Reflection

If the ray reaches the light source, then apply Phong's rule to compute "soft" specular highlight.

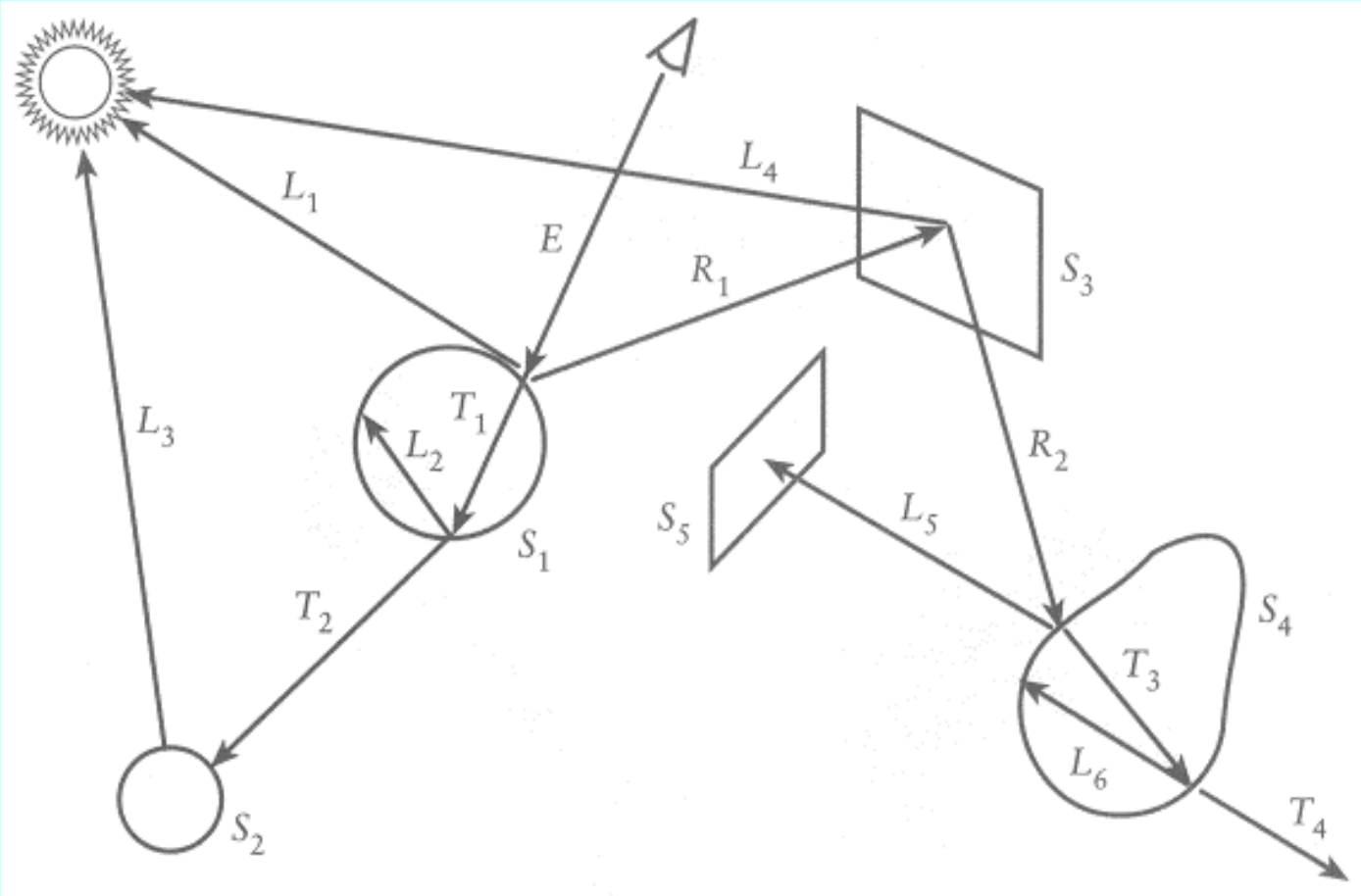


Specular Reflections of Other Objects

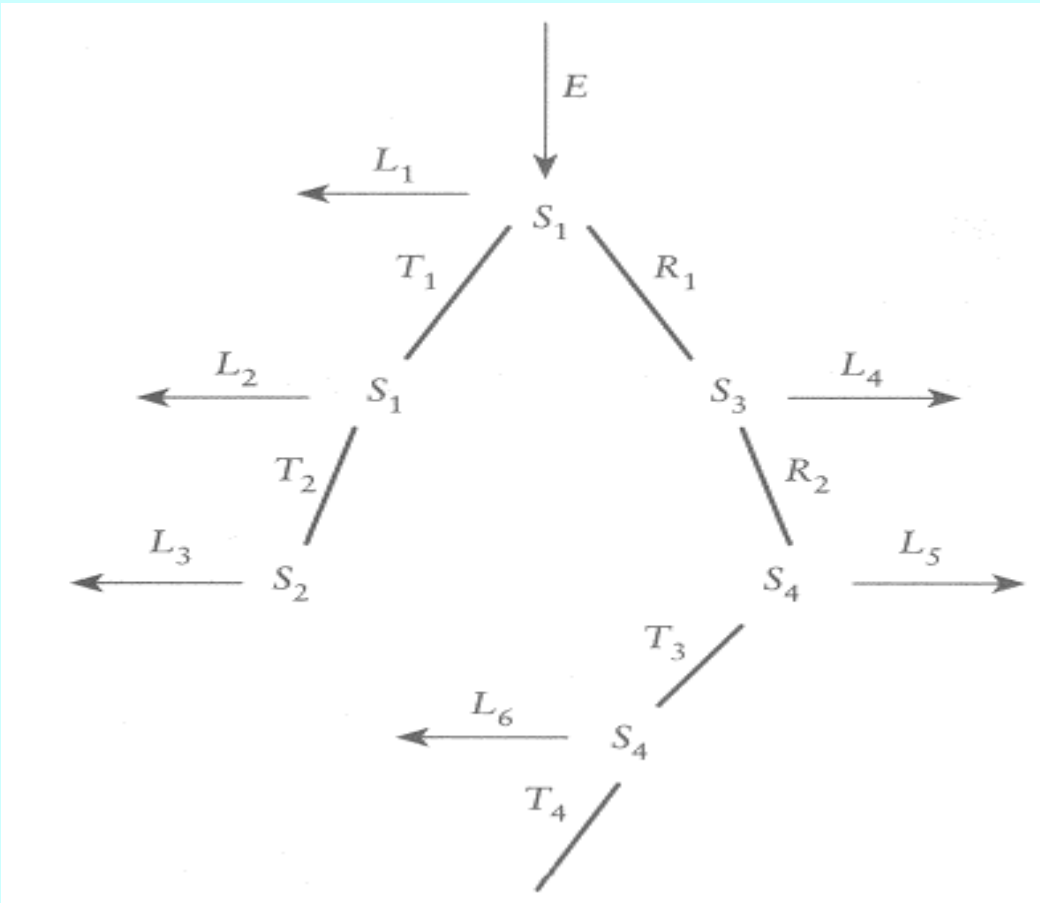
For shiny objects, we must "bounce" the ray off the object to see what it strikes next. Recursion terminates when ray hits non-specular surface or nothing.



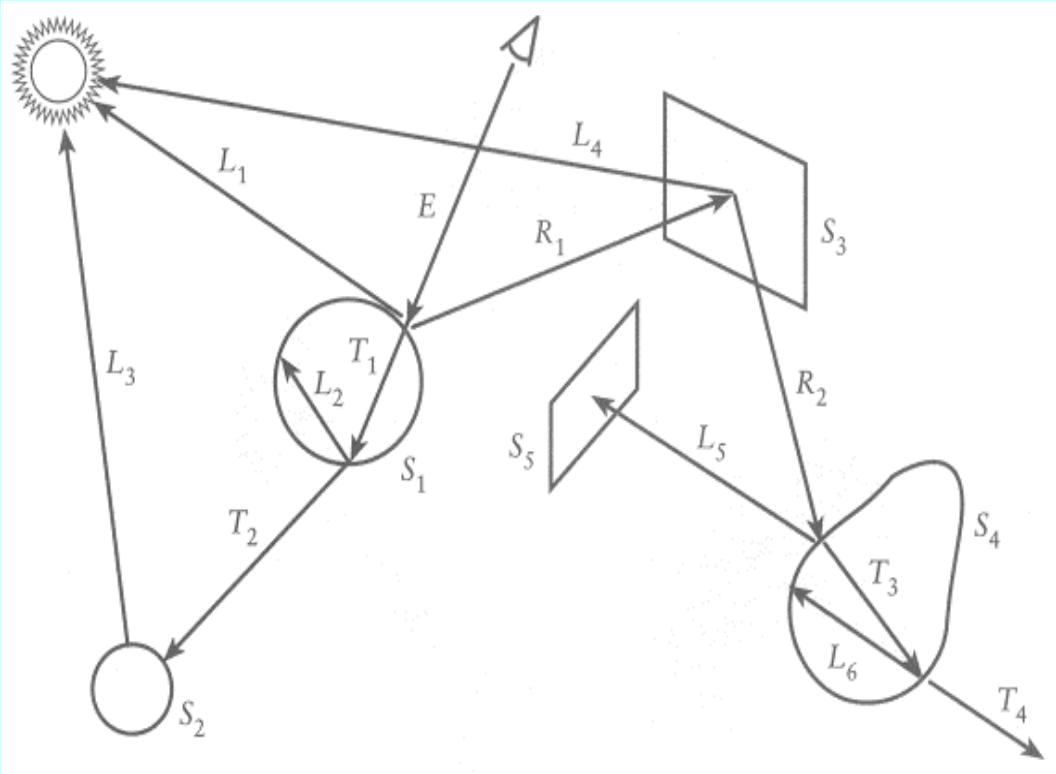
Обратная трассировка сцены

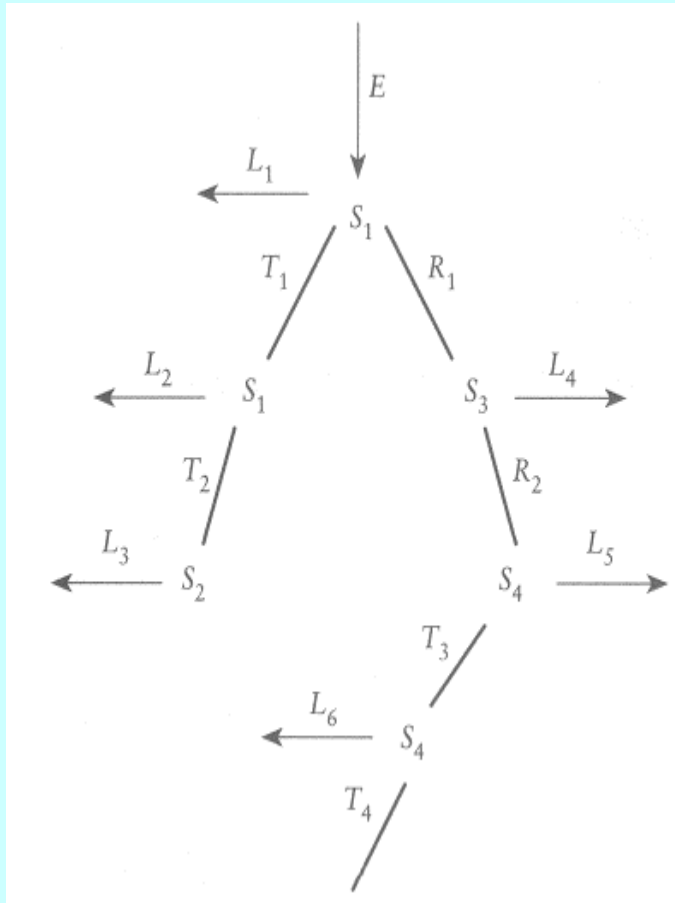


Дерево лучей

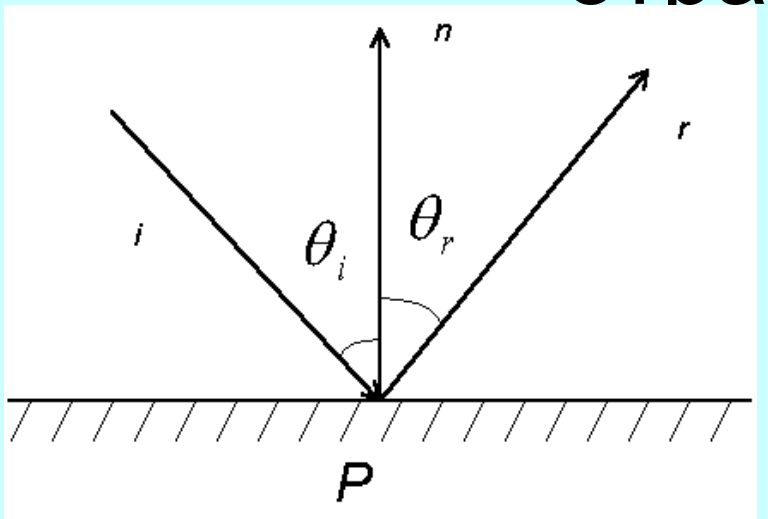


Дерево лучей

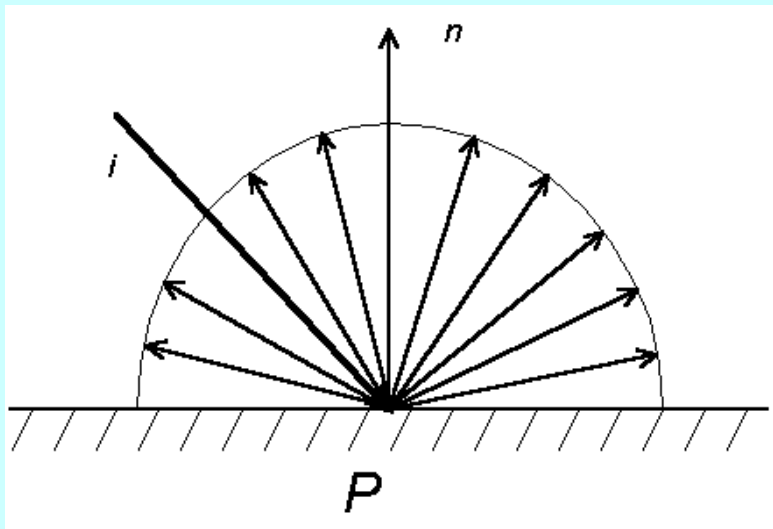




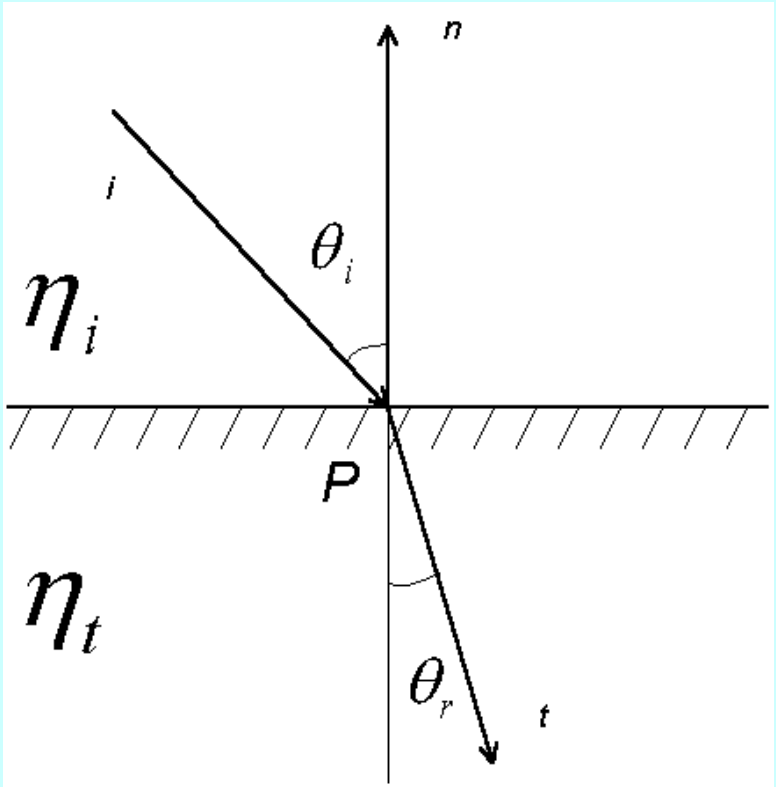
Диффузное и зеркальное отражения



$$r = i - 2(i,n)n$$



Идеальное преломление



$$\eta = \frac{c}{v}$$

v – скорость света в среде

c – скорость света в вакууме

$$t = \eta \cdot i + \left(\eta C_i - \sqrt{1 + \eta^2\left(C_i^2 - 1\right)}\right) \cdot n$$

$$C_i = \cos\theta_i = \left(-i, n\right)$$

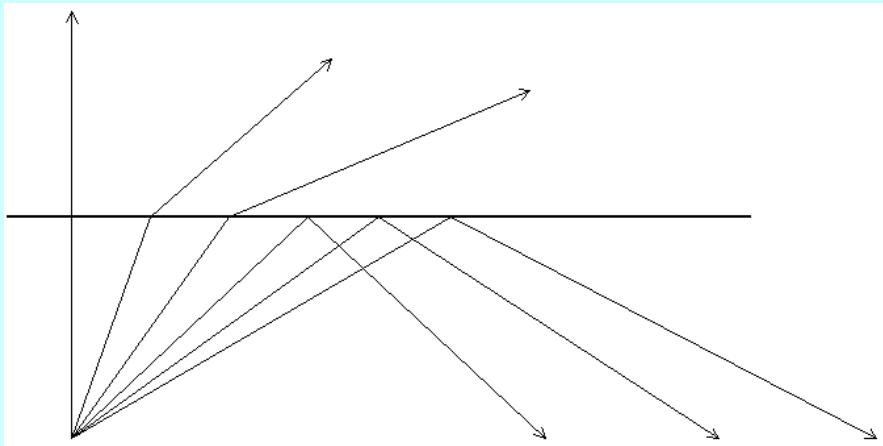
$$\eta = \frac{\eta_i}{\eta_t}$$

Полное внутреннее отражение

Для некоторых углов θ_i падения при выходе из более плотной среды в менее плотную $(\eta > 1)$ выражение под корнем может получиться отрицательным:

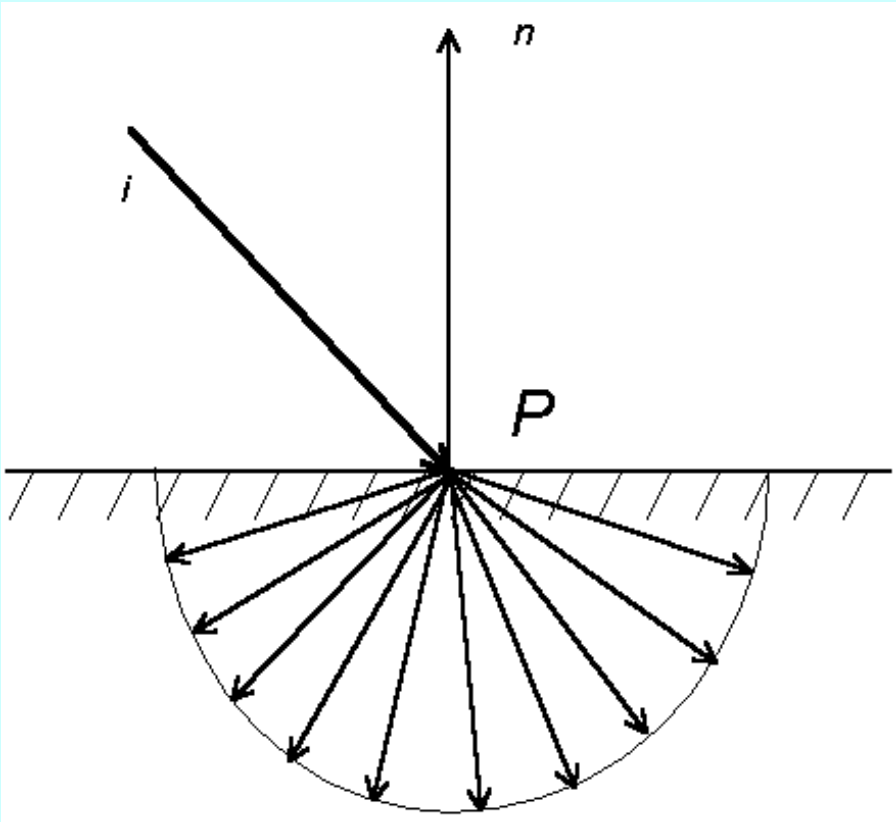
$$1 + \eta^2 \left(C_i^2 - 1 \right) < 0$$

Это соответствует случаю полного внутреннего отражения



Диффузное преломление

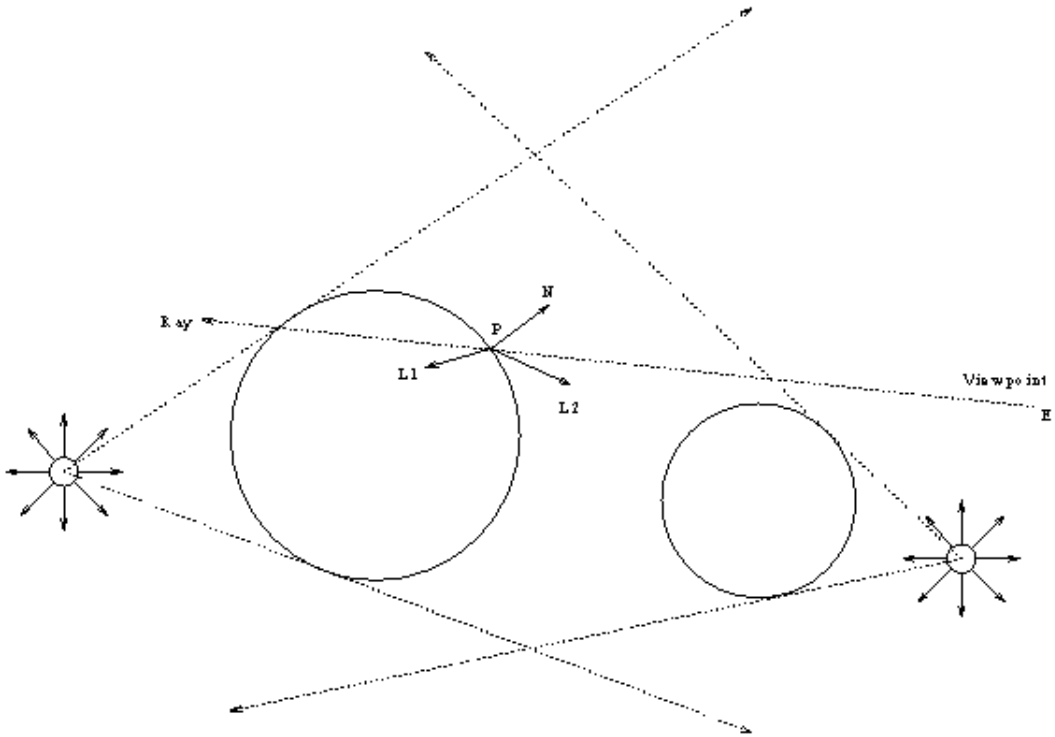
Световая энергия равномерно рассеивается по всем направлениям соответствующего полупространства



Затенение объектов

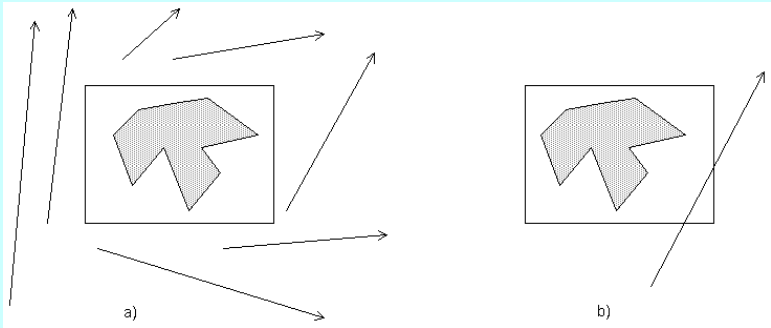
Shadows

To check for shadows, we can define secondary rays from point P to each of the point light sources L_1 and L_2. If a ray hits some object, then the corresponding light source does not strike P.

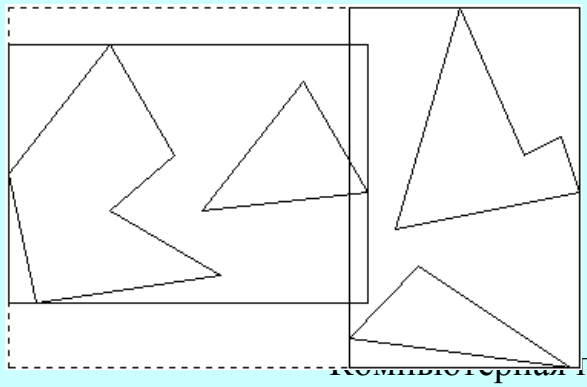


Оптимизация трассировки лучей

Использование ограничивающих тел:



Использование иерархии ограничивающих тел:



Достоинства и недостатки RT

- **Can correctly model shadows and multiple reflected images.**
- **Can handle refraction and transparent objects.**
- **Can handle boolean combinations of objects.**
  Constructive Solid Geometry: CSG.
- **Can handle non-polygonal objects.**
  Still requires computing normal vectors.
- **Very Expensive.**

Достоинства и недостатки обратной трассировки лучей

- (+) Корректно обрабатывает тени, отражения, преломление, полупрозрачные объекты
- (+) Может работать с неполигональными объектами
- (-) Большое время работы
- (-) Нет возможность вычисления вторичного освещения

Итоги

- Методы излучательности и трассировки лучей
  - Глобальное освещение
- Излучательность
  - Решение системы уравнений переноса для вычисления баланса энергии в сцене
  - Работает только для диффузных сцен, дает качественное решение
- Трассировка лучей
  - Обратная трассировка лучей пускает лучи из камеры и вычисляет цвет в точках пересечения с объектами сцены