

Метод трассировки лучей

Метод трассировки лучей является одним из наиболее распространенных методов, используемых для построения фотореалистического изображения сложных сцен с учетом отражения и преломления. Это простой и наглядный метод.

Различают прямую и обратную трассировки лучей. При прямой трассировке мы прослеживаем путь каждого луча, выпущенного из источника света (из каждого источника, если их несколько) по всем направлениям, прослеживая путь луча. Попав на объект сцены, луч может отразиться от него (рассеяться) или преломиться (уйти внутрь объекта). Отраженный луч, распространяясь прямолинейно, снова может попасть на объект сцены и снова отразиться или преломиться. Часть (отраженных) лучей в конце концов попадают в глаз наблюдателя и формируют изображение. Процесс прослеживания множества выпущенных из источника света лучей слишком трудоёмкий, кроме того, только часть оттрассированных лучей вносит вклад в формирование изображения сцены, поэтому чаще применяется метод обратной трассировки.

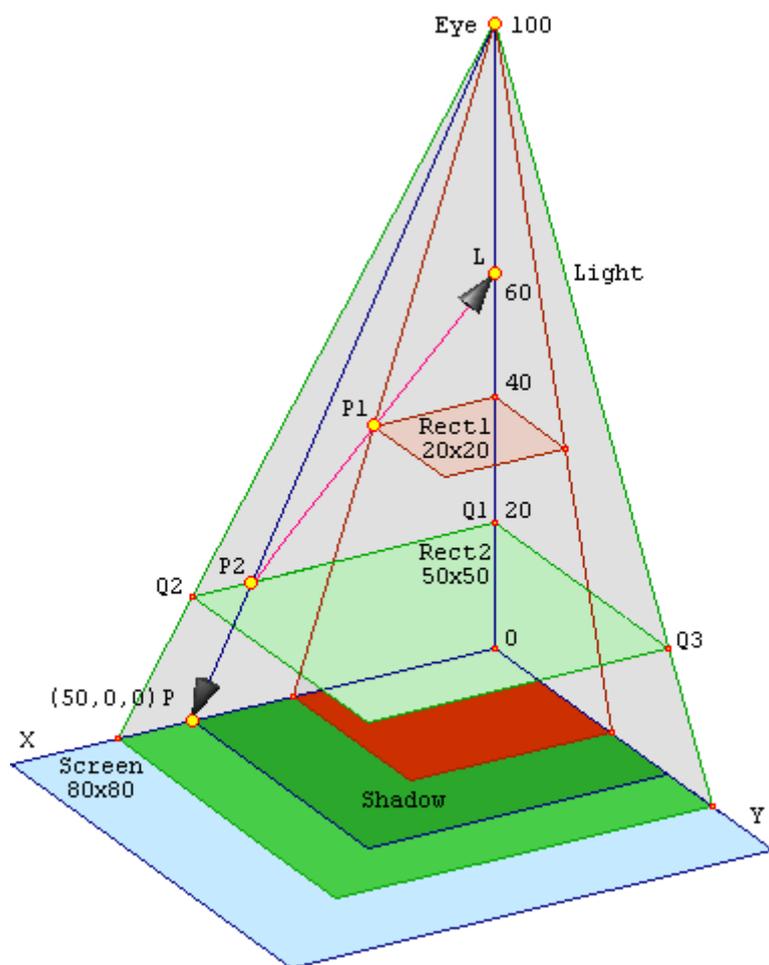


Рис. 1.

При обратной трассировке мы прослеживаем лучи, которые могут прийти в глаз наблюдателя от объектов сцены. Количество первичных лучей при этом ограничивается количеством точек экрана, так как в конечном итоге мы должны вычислить цвет именно точек экрана. Выпущенный из глаза наблюдателя через конкретный

экранный пиксель луч прослеживается до попадания на объект сцены. Если луч попадает на какой-либо объект, прослеживается путь от точки пересечения луча с объектом до источника и вычисляется прямая освещенность. Рис. 1 поясняет модель обратной трассировки лучей.

Расчетная схема

Сцена состоит из двух прямоугольников (первый красный размером 20x20, второй зеленый размером 50x50), расположенных на высоте 40 и 20 соответственно, и из одного источника света, расположенного на высоте 60. Точка наблюдения e_{ye} находится на высоте 100. Изображение проецируется на экран размером 80x80, расположенный в плоскости xz . Ось z направлена вверх.

Для трассировки сцены мы выпускаем лучи из точки наблюдения на экран. Эти лучи проходят через экранные пиксели, таким образом, всего для экрана 80x80 мы выпускаем 6400 первичных лучей.

Первичные трассирующие лучи

Один из лучей (синий) выпущен, например, через пиксель $p(50, 0, 0)$. Он имеет начальную точку $e_{ye}(0, 0, 100)$ и направление $(50, 0, 0) - (0, 0, 100) = (50, 0, -100)$. Его необходимо нормализовать:

```
L = Sqr(50*50 + 100*100) = 111.803
Ray.Direction = (50/L, 0/L, -100/L) = (0.447, 0, -0.894)
Ray.Origin = (0, 0, 100)
```

Пересечение с объектом

Далее мы опрашиваем объекты сцены на предмет того, пересекаются ли они заданным лучом или нет. Так как объекты сцены могут быть разного типа (параллелепипеды, сферы, цилиндры...), то только сам объект может вычислить точку пересечения. Объект типа прямоугольник вычисляет точку пересечения следующим образом:

Определяем косинус угла между нормалью и лучом

```
VD = Normal * Ray.Direction
```

```
ЕСЛИ |VD| > 0.01 ТО
```

```
    ' Если косинус не равен нулю (плоскость видна)
```

```
    ' Определяем расстояние от начала луча
```

```
    ' до точки пересечения (вдоль луча)
```

```
    T = (Loc - Ray.Origin) * Normal / VD
```

```
    ЕСЛИ (T > 0.01) ТО
```

```
        ' Если положение точки в положительном направлении луча
```

```
        ' Вычисляем точку пересечения
```

```
        P = Ray.Origin + Ray.Direction * T
```

```
        ' Вычисляем параметр точки пересечения вдоль оси U
```

```
        U = P * KU - U0
```

```
        ' Вычисляем параметр точки пересечения вдоль оси V
```

```
        V = P * KV - V0
```

```
        ' Если параметры находятся в пределах 0..1 -
```

```
        ' точка внутри прямоугольника
```

```
        ЕСЛИ (U >= 0) И (V >= 0) И (U <= 1) И (V <= 1) ТО
```

```
            Пересекает = ИСТИНА
```

```
        КОНЕЦ
```

```
    КОНЕЦ
```

```
КОНЕЦ
```

Здесь:

\mathbf{Normal} — нормальный вектор к плоскости объекта;

\mathbf{Loc} — вектор начальной точки объекта;

векторы \mathbf{kv} и \mathbf{kv} — базис области прямоугольника в параметрах u и v ;

значения u_0 и v_0 — начальные значения параметров u и v .

Эти значения должны быть вычислены для всех объектов сцены перед началом трассировки.

Параметры объекта ПРЯМОУГОЛЬНИК

Для объекта типа **прямоугольник** параметры вычисляются следующим образом:

Начальная точка объекта

$$\mathbf{Loc} = \mathbf{Q1}$$

Базис плоскости объекта в координатах x y

$$\mathbf{E1} = \mathbf{Q2} - \mathbf{Loc}$$

$$\mathbf{E2} = \mathbf{Q3} - \mathbf{Loc}$$

Нормальный вектор (ненормализованный)

$$\mathbf{Normal} = \mathbf{E1} \times \mathbf{E2}$$

Нормальный вектор (нормализованный)

$$\mathbf{Normal} = \mathbf{Normal} / \|\mathbf{Normal}\|$$

Элементы матрицы

$$s_{11} = \mathbf{E1} \cdot \mathbf{E1}$$

$$s_{12} = \mathbf{E1} \cdot \mathbf{E2}$$

$$s_{22} = \mathbf{E2} \cdot \mathbf{E2}$$

Детерминант

$$D = s_{11} \cdot s_{22} - s_{12} \cdot s_{12}$$

Базис плоскости объекта в координатах uv

$$\mathbf{kv} = (\mathbf{E1} \cdot s_{22} - \mathbf{E2} \cdot s_{12}) / D$$

$$\mathbf{kv} = (\mathbf{E2} \cdot s_{11} - \mathbf{E1} \cdot s_{12}) / D$$

Начальная точка базиса uv

$$u_0 = \mathbf{Loc} \cdot \mathbf{kv}$$

$$v_0 = \mathbf{Loc} \cdot \mathbf{kv}$$

Операция \cdot обозначает скалярное произведение, если оба операнда являются векторами, и произведение вектора на скаляр, если один из операндов вектор, а второй — скаляр. Операция \times обозначает векторное произведение.

Точка q_1 - начальная точка прямоугольника, точка q_2 задает размер вдоль оси x , точка q_3 задает размер вдоль оси y . На рисунке (модель трассировки) эти точки указаны для прямоугольника 2.

Расчет параметров объектов

Исходные точки прямоугольника 1

$$Q1(0, 0, 40), Q2(20, 0, 40), Q3(0, 20, 40)$$

Начальная точка

$$\mathbf{Loc}(1) = (0, 0, 40)$$

Вычисляем базис x y

$$\mathbf{E1} = Q2 - \mathbf{Loc}(1) = (20, 0, 0)$$

$$\mathbf{E2} = Q3 - \mathbf{Loc}(1) = (0, 20, 0)$$

Вычисляем и нормализуем нормаль

$$\text{Normal}(1) = \mathbf{E1} \times \mathbf{E2} = (0, 0, 400) = (0, 0, 1)$$

Вычисляем элементы матрицы

$$s_{11} = \mathbf{E1} \cdot \mathbf{E1} = 400$$

$$s_{12} = \mathbf{E1} \cdot \mathbf{E2} = 0$$

$$s_{22} = \mathbf{E2} \cdot \mathbf{E2} = 400$$

Детерминант

$$D = s_{11} \cdot s_{22} - s_{12} \cdot s_{12} = 160000 - 0 = 160000$$

Базис uv

$$\mathbf{KU}(1) = (\mathbf{E1} \cdot s_{22} - \mathbf{E2} \cdot s_{12}) / D = ((20, 0, 0) \cdot 400 - (0, 20, 0) \cdot 0) / 160000 = (0.05, 0, 0)$$

$$\mathbf{KV}(1) = (\mathbf{E2} \cdot s_{11} - \mathbf{E1} \cdot s_{12}) / D = ((0, 20, 0) \cdot 400 - (20, 0, 0) \cdot 0) / 160000 = (0, 0.05, 0)$$

Начальная точка базиса uv

$$\mathbf{U0}(1) = \text{Loc}(1) \cdot \mathbf{KU}(1) = (0, 0, 40) \cdot (0.05, 0, 0) = 0$$

$$\mathbf{V0}(1) = \text{Loc}(1) \cdot \mathbf{KV}(1) = (0, 0, 40) \cdot (0, 0.05, 0) = 0$$

Исходные точки прямоугольника 2

$$Q1(0, 0, 20), Q2(50, 0, 20), Q3(0, 50, 20)$$

Начальная точка

$$\text{Loc}(2) = (0, 0, 20)$$

Базис $xу$

$$\mathbf{E1} = Q2 - \text{Loc}(2) = (50, 0, 0)$$

$$\mathbf{E2} = Q3 - \text{Loc}(2) = (0, 50, 0)$$

Нормаль

$$\text{Normal}(2) = \mathbf{E1} \times \mathbf{E2} = (0, 0, 2500) = (0, 0, 1)$$

Элементы матрицы

$$s_{11} = \mathbf{E1} \cdot \mathbf{E1} = 2500$$

$$s_{12} = \mathbf{E1} \cdot \mathbf{E2} = 0$$

$$s_{22} = \mathbf{E2} \cdot \mathbf{E2} = 2500$$

Детерминант

$$D = s_{11} \cdot s_{22} - s_{12} \cdot s_{12} = 6250000 - 0 = 6250000$$

Базис uv

$$\mathbf{KU}(2) = (\mathbf{E1} \cdot s_{22} - \mathbf{E2} \cdot s_{12}) / D = ((50, 0, 0) \cdot 2500 - (0, 50, 0) \cdot 0) / 6250000 = (0.02, 0, 0)$$

$$\mathbf{KV}(2) = (\mathbf{E2} \cdot s_{11} - \mathbf{E1} \cdot s_{12}) / D = ((0, 50, 0) \cdot 2500 - (50, 0, 0) \cdot 0) / 6250000 = (0, 0.02, 0)$$

Начальная точка базиса uv

$$\mathbf{U0}(2) = \text{Loc}(2) \cdot \mathbf{KU}(2) = (0, 0, 80) \cdot (0.02, 0, 0) = 0$$

$$\mathbf{V0}(2) = \text{Loc}(2) \cdot \mathbf{KV}(2) = (0, 0, 80) \cdot (0, 0.02, 0) = 0$$

Далее для луча $(50, 0, -100)$ определяем точки пересечения.

Расчет точек пересечения

Прямоугольник 1

$$VD = \text{Normal}(1) \cdot \text{Ray.Direction} = (0, 0, 1) \cdot (0.447, 0, -0.894) = -0.894$$

$VD \neq 0$, следовательно, прямоугольник 1 виден.

$$T = (\text{Loc}(1) - \text{Ray.Origin}) \cdot \text{Normal}(1) / VD = ((0, 0, 40) - (0, 0, 100)) \cdot (0, 0, 1) / -0.894 = 67.082$$

Точка пересечения находится на положительном направлении луча. Вычисляем её:

$$\mathbf{P}(1) = \text{Ray.Origin} + \text{Ray.Direction} \cdot T = (0, 0, 100) + (0.447, 0, -0.894) \cdot 67.082 = (30, 0, 40)$$

Вычисляем параметры точки в базисе uv :

$$u = \mathbf{P}(1) \cdot \mathbf{KU}(1) - \mathbf{U0}(1) = (30, 0, 40) \cdot (0.05, 0, 0) - 0 = 1.5$$

$$v = P(1) \cdot kv(1) - v0(1) = (30, 0, 40) \cdot (0, 0.05, 0) - 0 = 0$$

Параметр $u > 1$, точка находится вне границ прямоугольника и, следовательно, не пересекает объект.

Прямоугольник 2

$$VD = \text{Normal}(2) \cdot \text{Ray.Direction} = (0, 0, 1) \cdot (0.447, 0, -0.894) = -0.894$$

$VD \neq 0$, следовательно, прямоугольник 2 виден.

$$T = (\text{Loc}(2) - \text{Ray.Origin}) \cdot \text{Normal}(2) / VD = \\ = ((0, 0, 20) - (0, 0, 100)) \cdot (0, 0, 1) / -0.894 = 89.443$$

Точка пересечения находится на положительном направлении луча. Вычисляем её:

$$P2 = \text{Ray.Origin} + \text{Ray.Direction} \cdot T = \\ = (0, 0, 100) + (0.447, 0, -0.894) \cdot 89.443 = (40, 0, 20)$$

Вычисляем параметры точки в базисе UV:

$$u = P2 \cdot kv(2) - v0(2) = (40, 0, 20) \cdot (0.02, 0, 0) - 0 = 0.8$$

$$v = P2 \cdot kv(2) - v0(2) = (40, 0, 20) \cdot (0, 0.02, 0) - 0 = 0$$

Так как оба параметра находятся в пределах 0..1, луч пересекает объект.

Расчет освещенности (цвета) точки

Основная задача трассировки — вычислить освещенность в точке пересечения и долю световой энергии, которая уходит от точки в заданном направлении (противоположном трассирующему лучу). Эта энергия складывается из двух частей — непосредственной (первичной) освещенности, которую точка получает от источников света непосредственно, напрямую, и вторичной освещенности, которая точка получает от других объектов.

Непосредственная освещенность точки складывается из диффузного зеркального отражения и зеркального отражения луча от источника света. Диффузное отражение описывается законом Ламберта — падающий свет рассеивается во все стороны с одинаковой интенсивностью, а освещенность точки пропорциональна доле площади, видимой от источника, то есть косинусу угла между лучом света и нормалью:

$$(1) \quad VD = VL \cdot (L \cdot N) .$$

Здесь:

VD — зеркальная диффузная освещенность;

VL — интенсивность источника света;

L — направление на источник света;

N — внешняя нормаль поверхности.

Зеркальное отражение луча света подчиняется закону отражения, — угол падения равен углу отражения, и луч света, нормаль и отраженный луч находятся в одной плоскости. Если считать поверхность идеально гладкой, ровной, то зеркальное отражение, которое уходит в направлении луча трассировки, можно рассчитать через косинус угла между отраженным лучом r и лучом трассировки ray или через косинус угла между падающим L и отраженным r лучами (Рис.2.).

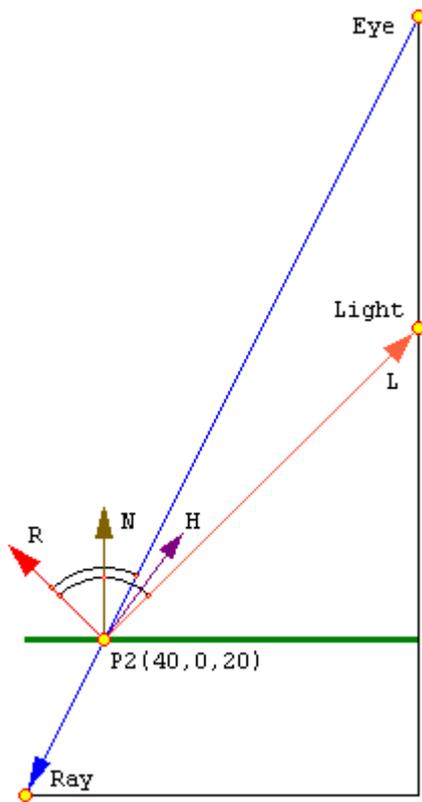


Рис.3.

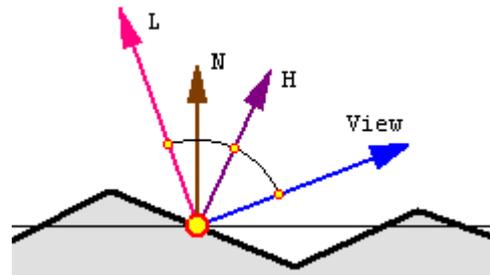


Рис.2.

На практике, однако, поверхность не является идеальной и ее можно представить как совокупность микрограней (микрозеркал). Тогда луч света, попадая в точку поверхности, отражается от некоторого микрозеркала, расположенного достаточно произвольно. Если потребовать, чтобы луч света отражался в направлении точки зрения, получим следующее выражение для вектора нормали микрограней N (Рис.3.)

$$n = (L + View) / \|L + View\|.$$

Величину зеркального отражения в этом случае принимают пропорциональной косинусу угла между нормалью и нормалью микрограней в некоторой степени P , называемой коэффициентом Фонга и учитывающей пространственное распределение света

$$(2) \quad BS = BL \cdot (N \cdot N)^P.$$

Для учета неточности модели вводится понятие фоновой (Ambient) освещенности — равномерное освещение объекта со всех сторон независимо от расположения источника света. Это дает возможность рассчитать освещенность в точке, на которую свет от источника не падает.

Цвет поверхности учитывается только в фоновой освещенности и в диффузном отражении.

Мы получили формулу для расчета непосредственной освещенности

$$(3) \quad B = BA \cdot C \cdot KA + BL \cdot C \cdot KD \cdot SH \cdot (L \cdot N) + BL \cdot KS \cdot SH \cdot (N \cdot N)^P.$$

Здесь:

B — освещенность точки (цвет);

BA — фоновая освещенность (цвет);

c — цвет поверхности;

ка — коэффициент фоновой освещенности (определяется для материала объекта);

вл — интенсивность (цвет) источника света;

кд — коэффициент диффузной освещенности (определяется для материала объекта);

шн — коэффициент затенения источника света (*shadow*);

л — вектор на источник света;

н — вектор нормали к поверхности;

кс — коэффициент зеркальной освещенности (определяется для материала объекта);

н — вектор нормали микрограни;

р — коэффициент Фонга.

Под освещенностью понимается цвет, а именно, веса его трех основных составляющих цветов **r**, **g** и **b**. Поэтому цвет может быть представлен в виде вектора с цветовыми координатами **rgb**, и с ним могут быть произведены векторные операции. Цветовые координаты, однако, могут принимать значения только в определенном диапазоне, например, 0..1 или 0..255. Мы будем использовать второй диапазон. Если при расчете цветовой координаты получится значение, выходящее за границу диапазона, то значение координаты ограничиваем соответствующей границей (цвет не может быть чернее черного и белее белого). При умножении одного цветового вектора на другой цветовой координаты перемножаются и делятся на верхнюю границу диапазона (255).

В приведенной формуле для расчета освещенности предполагается, что источник света один. Если источников света несколько, второе и третье слагаемые рассчитывают как сумму освещенностей, полученных от каждого отдельного источника, принимая за **вл** интенсивность этого источника.

Для расчета сцены используются значения:

ва = (255, 255, 255) = ; (Hue = 160; Saturation = 120; Brightness = 240)

с1 = (234, 21, 21) = ; (Hue = 0; Saturation = 200; Brightness = 120)

с2 = (21, 234, 21) = ; (Hue = 80; Saturation = 200; Brightness = 120)

ка = 0.5;

вл = (255, 255, 255) =  (начальная интенсивность источника света);

кд = 0.5;

кс = 0.5;

р = 3;

Рассчитываем фоновую освещенность:

v = **ва** · **с2** · **ка** = (255, 255, 255) · (21, 234, 21) · 0.5 = (10, 117, 10) = 

Для расчета второго и третьего слагаемых нужно вычислить вектор на источник света **л** и затенение источника **шн**. Затенение источника определяется посредством повторного трассирования сцены с целью определить, виден или нет источник света напрямую, и если не виден, то определить долю световой энергии, которая про

пускается загораживающими источник объектами сцены. Кроме того, доля энергии масштабируется делением затенения на расстояние и умножением на дополнительный параметр `distScale`, в примере равным 30. Дополнительный параметр дает возможность регулировать освещенность.

Вектор на источник света (малиновый луч на рисунке 1) рассчитывается через координаты точки пересечения `p2` и координаты источника света `light`. Он становится новым (вторичным) трассирующим лучом, который обрабатывается при помощи аналогичной процедуры опроса объектов сцены, что и для первичного трассирующего луча.

```
L.Origin = p2 = (40,0,20)
```

```
L.Direction = Light - p2 = (0,0,60) - (40,0,20) = (-40,0,40)
```

Вектор направления нужно нормализовать. Длина вектора до нормализации используется для расчета затенения `sh` источника света. В расчете длина обозначена как `dist`.

```
Dist = Sqr(40·40 + 40·40) = 56.57
```

```
L.Direction = (-40/Dist,0/Dist,40/Dist) = (-0.707,0,0.707)
```

Рассчитываем затенение источника света. При прямой видимости оно определяется по формуле

```
sh = DistScale/Dist = 30/56.57 = 0.530
```

Далее луч на источник `l` трассирует сцену с целью определить пересечение с объектами.

Прямоугольник 1:

```
VD = Normal(1) · L.Direction = (0,0,1) · (-0.707,0,0.707) = 0.707
```

`VD ≠ 0`, следовательно, прямоугольник 1 виден.

```
T = (Loc(1) - L.Origin) · Normal(1) / VD = ((0,0,40) - (40,0,20)) · (0,0,1) / 0.707 = 28.28
```

Точка пересечения находится на положительном направлении луча. Вычисляем её:

```
p1 = L.Origin + L.Direction · T = (40,0,20) + (-0.707,0,0.707) · 28.28 = (20,0,40)
```

Вычисляем параметры точки в базисе UV:

```
u = p1 · kv(1) - u0(1) = (20,0,40) · (0.05,0,0) - 0 = 1
```

```
v = p1 · kv(1) - v0(1) = (20,0,40) · (0,0.05,0) - 0 = 0
```

Таким образом, вторичный луч `l` пересекает прямоугольник 1.

Проверяем, что находится ближе к точке `p2` — точка `p1` или источник света

```
(T < Dist) = Истина
```

Точка `p2` ближе, поэтому прямоугольник 1 загораживает источник света. Вычисляем общее затенение источника, умножая затенение `sh` на коэффициент зеркальной освещенности `ks`

```
sh = sh · ks = 0
```

Следовательно, точка `p2` на поверхности прямоугольника 2 попадает в тень (`shadow`) от прямоугольника 1. Освещенность точки состоит из фоновой освещенности только.

Может оказаться, что другой объект сцены также загораживает источник света. Поэтому при вычислении затенения источника света нужно определить список всех объектов, которые пересекает вторичный луч, и выбрать из них те, расстояние до которых (параметр `t` точки пересечения которых) меньше расстояния до источ

ника света. После этого нужно выяснить, какая доля энергии теряется при прохождении луча через загораживающие объекты. Для этого коэффициент затенения sn умножается на коэффициент kt поверхности каждого из загораживающих объектов. Таким образом учитывается диффузное преломление, которое распространяется одинаково по всем направлениям. Если для какого-либо объекта kt окажется менее 0.01, затенение можно сразу принять равным 0, так как свет полностью гасится непрозрачным объектом.

Чтобы показать, как рассчитываются два других компонента освещенности точки, исключим из сцены прямоугольник 2.

Диффузное отражение

Косинус угла между лучом на источник света и нормалью

$$LN = L.Direction \cdot Normal = (-0.707, 0, 0.707) \cdot (0, 0, 1) = 0.707$$

Произведение цвета источника света и цвета поверхности

$$VL \cdot C2 = (255, 255, 255) \cdot (21, 234, 21) = (21, 234, 21)$$

Общее затенение

$$SN \cdot LN \cdot KD = 0.530 \cdot 0.707 \cdot 0.5 = 0.187$$

Умножаем $VL \cdot C2$ на общее затенение

$$VD = VL \cdot C2 \cdot SN = (21, 234, 21) \cdot 0.187 = (4, 44, 4) = \blacksquare$$

Результат складывается с рассчитанной ранее фоновой освещенностью

$$V = VA + VD = (10, 117, 10) + (4, 44, 4) = (14, 161, 14) = \blacksquare$$

Зеркальное отражение

Определяем нормальный вектор микрограни

$$N = L.Direction - Ray.Direction = (-0.707, 0, 0.707) - (0.447, 0, -0.894) = (-1.154, 0, 1.601)$$

и нормализуем его

$$n = (-0.584, 0, 0.811)$$

Рассчитываем косинус угла между нормалью поверхности и нормалью микрограни

$$NN = n \cdot N = (0, 0, 1) \cdot (-0.584, 0, 0.811) = 0.811$$

Полученное произведение возводим в степень p (коэффициент Фонга)

$$NNP = 0.811^3 = 0.533$$

и умножаем на затенение источника sn и на ks

$$SN \cdot KS \cdot NNP = 0.530 \cdot 0.5 \cdot 0.533 = 0.142$$

Умножаем цвет источника света на полученное значение

$$VS = VL \cdot 0.142 = (255, 255, 255) \cdot 0.142 = (36, 36, 36)$$

Результат складывается с рассчитанной ранее освещенностью (фоновая + диффузная)

$$V = V + VS = (14, 161, 14) + (36, 36, 36) = (50, 197, 50) = \blacksquare$$

На этом вычисление основной части освещенности заканчивается. Осталось выяснить, какой цвет получит точка экрана, если луч не пересекает никакого объекта. В этом случае на данном этапе мы возвращаем цвет фона (`background`), установлен

ный для сцены, равный (128,207,255) =  (Hue = 135; Saturation = 240; Brightness = 180).

На рисунке 4 показан результат трассировки при указанных параметрах.

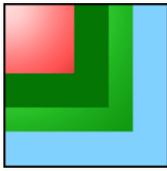


Рис.4.

Программирование

Для выполнения работы используются классы интерфейса `GREEN`.

Основным объектом является сцена `GREENScene`, которая содержит коллекции всех ребер `edges`, граней `faces`, треугольных граней `triangles`, источников света `lights`, точку зрения `viewPoint` и коллекцию объектов `clusters`. Кроме того, сцена содержит три матрицы для хранения исходных (мировых), видовых и экранных координат (имена матриц соответственно `wco`, `eco`, `sco`).

Для работы с разными объектами используется интерфейс `IGREENCluster`, основными элементами которого являются методы `Build`, `ParamsUpdate` и `Intersect`. Разные объекты получаются созданием новых классов (например `Rect`), которые поддерживают этот интерфейс. Для облегчения создания нового класса имеется шаблон (файл `CLUSTTPL.cls`).

Метод `Build` предназначен для создания объекта. Он принимает необходимое число параметров, определяемое разработчиком. Для создания объекта типа `прямоугольник` достаточно иметь два параметра — размер вдоль оси `x` и размер вдоль оси `y`, плюс, например, смещение начальной точки (три необязательных параметра). Создание объекта производится в следующем порядке:

1 рассчитать вершины объекта и занести их координаты в матрицу `wco`. Номера вершин при этом должны быть добавлены в список вершин объекта при помощи метода `AddVector`. Пример выполнения этой части (первая точка прямоугольника):

```
With Scene
  With .WCO
    AddVector .Add.SetValues(X0, Y0, Z0).ID
    . . .
  End With
```

2 зарезервировать место в матрицах `eco` и `sco` на необходимое число вершин:

```
.ECO.Extend UBound(pVecs)
.SCO.Extend UBound(pVecs)
```

3 добавить необходимое число граней в список граней сцены, указав точки в нужном порядке. Точки находятся в массиве `pVecs`, который заполняется методом `AddVector`.

```
With .Faces
  AddFace .Add.Points.SetDirect(pVecs(1), pVecs(2), pVecs(3), pVecs(4))
  . . .
End With
```

4 добавить ребра объекта в список ребер сцены:

```
For Each Face In .Faces
  AddEdges Face.Points.EdgesAddTo(.Edges)
```

Next

5 добавить треугольные грани в список треугольных граней сцены:

```
With .Triangs
    AddTriang .Add.Points.SetDirect(pVecs(1), pVecs(2), pVecs(4))
    . . .
End With
```

Метод `ParamsUpdate` предназначен для вычисления параметров области объекта. Для прямоугольника он описан выше в этом документе на странице 2. В этом методе нужно вычислить (для прямоугольника) параметры `pLoc`, `pNormal`, `pKu`, `pKv`, `pU0`, `pV0` для заданной матрицы `m`. Это глобальные переменные класса. Необходимые для расчета другие переменные должны быть объявлены локально.

Метод `Intersect` вычисляет точку пересечения с заданным лучом и возвращает результаты расчета через объект `GREENInter`. Этот объект содержит переменную `Intersect`, которая указывает на результат вычислений и должна быть установлена в правильное значение (*Ложь* или *Истина*); переменную `id`, которой нужно присвоить идентификатор объекта; переменную `Normal`, которой нужно присвоить значение нормали; переменную `Point`, которой присваивается значение точки пересечения; переменную `Dist`, которой присваивается значение t (см. расчет точки пересечения на стр. 2); также переменную `Surface`, которой нужно присвоить значение соответствующего свойства объекта.

При установке переменной `Normal` нужно получить копию нормали, например:

```
Set Inter.Normal = pNormal.Copy
```

Здесь `Inter` — это локальная переменная типа `GREENInter`. Именно она возвращается этим методом. Она должна быть объявлена с созданием нового объекта

```
Dim Inter As New GREENInter
```

и возвращена при помощи оператора `set`

```
Set Intersect = Inter
```

Параметром метода является луч — объект класса `GREENRay`. Этот объект содержит два вектора. Вектор `origin` — вектор положения начальной точки, вектор `direction` — вектор направления луча. Объект имеет метод `PointAt` для расчета точки на луче через параметр t — расстояние до точки от начала луча. Полученная точка является точкой пересечения. Пример использования метода `PointAt`:

```
Set .Point = Ray.PointAt(Inter.Dist)
```

Для передачи свойств поверхности объекта используется объект класса `GREENSurface`. Он, во-первых, содержит объект класса `GREENColor`, описывающий цвет. Далее он содержит коэффициенты, описывающие свойства поверхности (`ka`, `kd`, `kr`, `ks`, `kt`, `ps`), а также два параметра, описывающие среду — `Atten` (коэффициент затухания) и `Refract` (коэффициент преломления). Эти два параметра могут быть возвращены также как запись о среде типа `GREENMedium`. Среда используется при расчете освещенности от отраженного и преломленного лучей. Свойство `surface` следует копировать методом `copy`:

```
Set .Surface = Surface.Copy
```

При расчете параметров u и v округлите их с точностью 1..3 цифры.

При работе с объектом класса `GREENVector` вам понадобятся операции умножения, сложения, деления, вычисления скалярного и векторного произведений и т.п.

Практически все операции возвращают либо новый объект, либо ссылку на существующий. Это дает возможность выполнять множество операций в одной строке, например:

```
Set pNormal = E1.Product(E2).NormalizeByLength
Set pKV = E1.Mul(S22).Subtract(E2.Mul(S12)).Div(D)
```

Важно следить за тем, чтобы при таких операциях не изменялись исходные объекты. Для этого обращайтесь внимание на то, что возвращает операция — новый объект или текущий. При необходимости используйте метод `copy` для получения копии объекта и работайте с ней.

Объект класса `GREENColor` предназначен для выполнения операций с цветом. В отличие от вектора, операции этого объекта никогда не возвращают новый объект (за исключением метода `copy`), и любая операция изменяет текущий объект. Поэтому сначала нужно использовать метод `copy` для получения копии объекта, а потом уже выполнения операций с ним, например:

```
V.Add Light.Color.Copy.Mul(Inter.Surface.Color.Copy).MulBy(Ks * SH * (HN ^ Ps))
```

Объект `GREENColor` имеет свойства `Hue`, `Saturation` и `Brightness`, предназначенные исключительно для начальной установки цвета, причем свойства следует устанавливать в указанном порядке. При операциях с объектом эти свойства не изменяются, поэтому основными в этой работе являются операции и свойства объекта `r`, `g`, `b`, `RGB`.

Работа над трассировкой начинается с использования готового объекта `GREENRect`. Создание объектов производится внутри процедуры `main` стандартного модуля шаблона проекта. В шаблоне предусмотрено три объекта сцены. Первый объект (`Scene1`) включает в себя 2 прямоугольника с параметрами, использованными в описании. Изучите создание объектов и задание параметров сцены 1. Для трассировки первой сцены ее нужно выбрать:

```
' ***** ВЫБОР СЦЕНЫ ***** '
Set Scene = Scene1
```

Начинайте работу с одним объектом (прямоугольник 2), закомментировав строку создания объекта 1:

```
With Scene1
  ' Set R1 = .Clusters.Add(New GREENRect)
  Set R2 = .Clusters.Add(New GREENRect)
End With
```

Разработайте процедуры для вычисления всех компонентов непосредственной освещенности. Во время отладки установите трассировку только одного луча — описанного в документе и проверяйте получаемые значения по описанию.

Для выполнения трассировки понадобятся следующие процедуры и функции:

1) Процедура, выполняющая трассировку

```
Public Sub Tracing(ByVal hDC As Long, _
  ByVal Y1 As Integer, _
  ByVal Y2 As Integer, _
  ByVal X1 As Integer, _
  ByVal X2 As Integer, _
  ByVal HomeX As Integer, _
  ByVal HomeY As Integer, _
  ByVal Dist As Double)
```

Параметры:

`hdc` — контекст устройства отображения;
`y1, y2` — номер первой и последней строки трассировки;
`x1, x2` — номер первого и последнего пикселя строки;
`homeX` — положение центра устройства отображения;
`homeY` — положение центра устройства отображения;
`dist` — расстояние до устройства отображения;

Параметры считываются с элементов управления формы (в процедуре `Draw`).

Перед вычислением трассирующих лучей:

- 1) Рассчитайте параметры объектов сцены вызовом методов объектов `ParamsUpdate` для матрицы `wco`;
- 2) Вычислите синусы и косинусы углов точки зрения при помощи функции `AngleToSinCos`;
- 3) Вычислите координату `z` плоскости экрана как разность между расстоянием до точки зрения и расстоянием от точки зрения до плоскости экрана
`z = Scene.ViewPoint.Dist - Dist;`
- 4) Запомните координаты точки зрения в переменных `x0, y0, z0`;
- 5) Установите координаты начальной точки луча равными `x0, y0, z0`;

Процедура `Tracing` вычисляет первичные трассирующие лучи при помощи двойного цикла. В циклах параметр `yy` "пробегаёт" значения от `y1` до `y2`, а параметр `xx` — от `x1` до `x2`. Эти параметры используются для позиционирования функции `SetPixel`. Для расчета луча их следует скорректировать:

```
xs = xx - homeX, ys = homeY - yy
```

Эти значения соответствуют плоскости экрана в системе координат, начало которой совпадает с началом мировой системы координат, ось `x` направлена вдоль оси `y`, а ось `y` противоположна оси `x` (если смотреть по рисунку 1). Координаты необходимо пересчитать в мировую систему координат посредством вращения вокруг оси `z` на угол `180-zAngle`, вокруг оси `x` на угол `90-xyAngle` и отражением оси `x`. Для получения вектора направления луча из полученных координат нужно вычесть координаты начальной точки и нормализовать вектор:

```
y = -ys * cz - z * sz
```

```
With Ray.Direction
```

```
.x = y * cx - xs * sx - x0
```

```
.y = xs * cx + y * sx - y0
```

```
.z = ys * sz - z * cz - z0
```

```
.NormalizeByLength
```

```
End With
```

Полученный луч передается в качестве параметра функции трассировки сцены `Trace`, которая возвращает цвет пикселя экрана в объект `Color`. Пиксель мы выводим на контекст устройства `hdc` при помощи системной функции `SetPixel`, используя свойство `RGB` объекта `Color`.

2) Функция для трассирования луча

```
Public Function Trace(ByVal Ray As GREENRay) As GREENColor
```

Она получает сформированный луч и трассирует им сцену путем опроса всех объектов и вызова метода `Intersect` каждого объекта. Результатом метода `Intersect` является объект класса `GREENInter`, который нужно занести в список пересеченных объектов — коллекцию `GREENInters`. Это осуществляется при помощи цикла `For Each`:

```
For Each Q In Clusters
    If Q.Visible Then Inters.Add Q.Intersect(Ray)
Next
```

Здесь `Inters` — объект класса `GREENInters`. Он автоматически упорядочивает добавляемые пересечения по возрастанию расстояния до точки пересечения так, что первым объектом является ближайший. Кроме того, пересечение не будет добавлено в коллекцию, если переменная `Intersect` равна значению `ложь`.

Далее следует проверить, чему равен счетчик коллекции `Inters`. Если менее 1, пересеченных объектов в коллекции нет, следовательно, функция возвращает цвет фона сцены. В противном случае выбирается первый объект и вычисляется освещенность точки пересечения при помощи следующей (описываемой) процедуры. Пример вызова процедуры:

```
Set Trace = Shade(Inters(1), Ray)
```

3) Функция расчета освещенности `Shade`.

```
Public Function Shade(ByVal Inter As GREENInter, ByVal View As GREENRay) As GREENColor
```

Здесь вычисляется освещенность точки пересечения в соответствии с описанием. Здесь же выполняется трассировка вторичных лучей для каждого источника света. Следует помнить, что при добавлении нового источника света его координаты заносятся в матрицу `wco` и номер вектора в матрице запоминается в свойстве `Point` объекта `GREENLight` (это выполняется объектом `scene`).

Для выполнения трассировки вызываем функцию `Tracing` в процедуре `Draw`:

```
With MF
    Tracing .GRADHDC, 0, .GDHeight-1, 0, .GDWidth-1, .GDHomeX, .GDHomeY, .GDDist
End With
```

Точка начала системы координат устанавливается в конце функции `Main`. Для того чтобы выполнить отладку одного луча, установите параметры требуемого пиксела при вызове `Tracing`:

```
Tracing .GRADHDC, 50, 50, 0, 0, .GDHomeX, .GDHomeY, .GDDist
```

После того, как процедуры трассировки будут отлажены, можно приступить к разработке своего объекта типа `прямоугольник`. Для этого включите в состав проекта модуль класса `CLUSTTPL.cls`, задайте ему свойство `Name = Rect`, переименуйте файл в `RECT.cls` и разработайте методы `Build`, `ParamsUpdate` и `Intersect`. Измените ссылки на объекты `R1` и `R2`.

При выполнении работы следует изучить влияние параметров поверхности `KA`, `кв`, `кs`, `ps`, `кт`, а также положение и цвет источника света на результат трассировки.

На этом разработка первой части задания по трассировке завершена.

ЧАСТЬ 2

Теперь нужно вычислить освещенности, которые приходят в точку пересечения по отраженному и преломленному лучу.

Отраженный луч \mathbf{r} соответствует закону идеального отражения — он лежит в одной плоскости с нормалью \mathbf{n} к поверхности и лучом точки зрения \mathbf{e} , а угол отражения луча отражения равен углу падения луча точки зрения. С помощью отраженного луча мы получаем освещенность, пришедшую в точку от других объектов сцены. Эту освещенность мы складываем с непосредственной (первичной) освещенностью точки, предварительно умножив ее на коэффициент вклада отраженного луча k_r . Рисунок 5 поясняет получение отраженного луча.

Здесь: \mathbf{e} — луч от точки зрения, \mathbf{n} — вектор нормали к поверхности, \mathbf{r} — отраженный луч. Для вычисления отраженного луча представим его как линейную комбинацию векторов \mathbf{e} и \mathbf{n}

$$\mathbf{r} = \alpha \cdot \mathbf{e} + \beta \cdot \mathbf{n}.$$

Так как угол падения равен углу отражения, соответственно равны и скалярные произведения лучей падения и отражения с нормалью:

$$-\mathbf{e} \cdot \mathbf{n} = \mathbf{r} \cdot \mathbf{n},$$

откуда мы получаем формулу для вычисления вектора отражения

$$\mathbf{r} = \mathbf{e} - 2 \cdot (\mathbf{e} \cdot \mathbf{n}) \cdot \mathbf{n}.$$

Полученный вектор является единичным. Для вычисления освещенности, полученной по отраженному лучу, мы трассируем им сцену и получаем освещенность точки пересечения этим лучом ближайшей поверхности или освещенность фона. Её мы умножаем на коэффициент вклада отраженного луча k_r , а также учитываем экспоненциальное затухание, если луч распространяется в среде, коэффициент затухания которой отличен от нуля

$$(4) \quad \mathbf{BR} = \mathbf{B} \cdot k_r \cdot e^{-\mathbf{AR} \cdot \mathbf{DR}}.$$

Здесь:

\mathbf{BR} — освещенность, пришедшая по отраженному лучу;

\mathbf{B} — освещенность, полученная в результате трассировки сцены отраженным лучом;

k_r — коэффициент вклада отраженного луча;

\mathbf{AR} — коэффициент затухания среды, в которой распространяется отраженный луч;

\mathbf{DR} — расстояние, пройденное отраженным лучом.

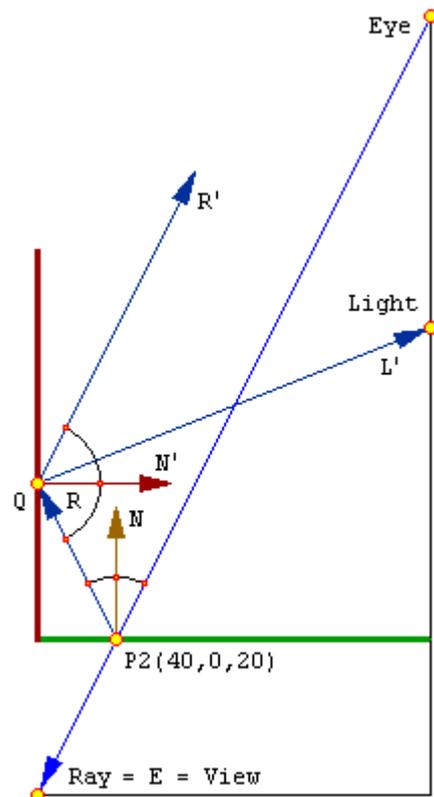


Рис.5.

Для учета распространения света в некоторой среде в программу вводится понятие среды, как записи из двух коэффициентов: коэффициента преломления `Refract` и коэффициента затухания `Atten`.

Коэффициент преломления — это отношение скорости распространения света в вакууме к скорости распространения света в среде. Для хранения и передачи информации о среде используется тип `GREENMedium`. В интерфейсе `GREEN` имеется две переменных, возвращающих описание воздуха и стекла (переменная `air`, коэффициент преломления `1`, и переменная `glass`, коэффициент преломления `1.5`).

Так как отраженный луч снова трассирует сцену, то, в конце концов, мы снова вернемся в процедуру вычисления освещенности точки `shade` и снова начнем формировать отраженный луч. Этот процесс нужно остановить. Для этой цели вводится понятие весового вклада отраженного луча. При первичной трассировке мы укажем весовой коэффициент, равный `1`, как одно из свойств луча (`weight`). Умножая вес на коэффициент вклада отраженного луча `KR`, мы получим новое значение веса, которое используем для трассировки отраженного луча. Когда вес достигнет порогового значения (станет меньше значения `0.01`), мы перестанем выпускать отраженные лучи.

Кроме этого, введем ограничение по уровню рекурсии. Для этой цели используем переменную `level`, предназначенную для учета уровня рекурсии. Сравнивая значение этой переменной с максимальным уровнем `maxLevel`, мы также сможем остановить процесс порождения новых отраженных лучей. Эта переменная увеличивается на значение `1` в начале функции `trace` и уменьшается в конце. Значение переменной `maxLevel` следует установить в начале процедуры `trace`.

Для контроля и отладки используем сцену с двумя прямоугольниками. Первый прямоугольник (красный, `50x50`) расположен в плоскости `y = 50` (получается вращением вокруг оси `y` на угол `-90` и перемещением вдоль оси `x` на `50` и вдоль оси `z` на `20`), второй (зеленый, `50x50`) — в плоскости `z = 20` (тот же самый). Другие параметры трассировки те же, что и в первой части. Коэффициент вклада отраженного луча `KR` для первого прямоугольника равен `0.5`.

В функции `shade`, после вычисления основной доли освещенности вычисляем освещенность по отраженному лучу. Первичный луч проходит через пиксель `P(50, 0)`. Для этого пикселя все расчеты были сделаны ранее, и непосредственная освещенность точки `P2` составила `(50, 197, 50) = `.

Прежде всего проверяем уровень рекурсии и если он равен максимально допустимому, расчет освещенности точки заканчивается. Если уровень рекурсии допустим, формируем отраженный луч. Начало луча — точка пересечения. Копируем ее из объекта `PERЕСЕЧЕНИЕ`

```
R.Origin = Inter.Point.Copy = (40, 0, 20) .
```

Вычисляем весовой коэффициент для отраженного луча

```
Ray.Weight = View.Weight * KR = 1 * 0.5 = 0.5
```

Проверяем величину полученного весового коэффициента. Он превышает значение `0.01`, продолжаем вычисления. Вычисляем косинус угла между нормалью и трассирующим лучом

```
VN = View.Direction * Normal = (0.447, 0, -0.894) * (0, 0, 1) = -0.894
```

Вычисляем направление отраженного луча

$$R.Direction = View - N \cdot (VN + VN) = (0.447, 0, -0.894) - (0, 0, 1) \cdot (-0.894 - 0.894) = (0.447, 0, 0.894)$$

Здесь `view` — это первичный трассирующий луч (параметр процедуры `shadow`). Теперь выпускаем отраженный луч, то есть вызываем процедуру `Trace` с параметром `weight`, равным `RWeight`.

Опускаем вычисление параметров первого прямоугольника, они равны:

$$Normal = (-1, 0, 0); Loc = (50, 0, 20); KU = (0, 0, 0.02); KV = (0, 0.02, 0); U0 = 0.4; V0 = 0$$

Рассчитываем точку пересечения с первым объектом

$$VD = Normal \cdot R.Direction = (-1, 0, 0) \cdot (0.447, 0, 0.894) = -0.447$$

$$T = (Loc - R.Origin) \cdot Normal / VD = ((50, 0, 20) - (40, 0, 20)) \cdot (-1, 0, 0) / -0.447 = 22.36$$

$$Q = R.Origin + R.Direction \cdot T = (40, 0, 20) + (0.447, 0, 0.894) \cdot 22.36 = (50, 0, 40)$$

$$U = P \cdot KU - U0 = (50, 0, 40) \cdot (0, 0, 0.02) - 0.4 = 0.4$$

$$V = P \cdot KV - V0 = (50, 0, 40) \cdot (0, 0.02, 0) - 0 = 0$$

Есть пересечение с первым объектом в точке `Q`, и мы рассчитываем её освещенность.

Фоновая освещенность

$$B' = BA \cdot C2' \cdot CA = (255, 255, 255) \cdot (234, 21, 21) \cdot 0.5 = (117, 10, 10) = \blacksquare$$

Вторичный луч `L'` (на источник света)

$$L'.Origin = Q = (50, 0, 40)$$

$$L'.Direction = Light - Q = (0, 0, 60) - (50, 0, 40) = (-50, 0, 20)$$

Расстояние до источника

$$Dist = \text{Sqr}(50 \cdot 50 + 20 \cdot 20) = 53.85$$

$$L'.Direction = (-50/Dist, 0/Dist, 20/Dist) = (-0.928, 0, 0.371)$$

Затенение источника света

$$SH = DistScale/Dist = 30/53.85 = 0.557$$

Далее луч на источник `L'` трассирует сцену с целью определить пересечение с объектами. Так как пересечений нет, величина затенения не изменяется.

Диффузное отражение

Косинус угла между лучом на источник света и нормалью

$$LN = L'.Direction \cdot Normal = (-0.928, 0, 0.371) \cdot (-1, 0, 0) = 0.928$$

Произведение цвета источника света и цвета поверхности

$$S = BL \cdot C2' = (255, 255, 255) \cdot (234, 21, 21) = (234, 21, 21) = \blacksquare$$

Общее затенение

$$SH \cdot LN \cdot KD = 0.557 \cdot 0.928 \cdot 0.5 = 0.259$$

Умножаем `s` на общее затенение

$$BD = S \cdot SH = (234, 21, 21) \cdot 0.259 = (61, 5, 5) = \blacksquare$$

Результат складывается с рассчитанной ранее фоновой освещенностью

$$B' = B' + BD = (117, 10, 10) + (61, 5, 5) = (178, 15, 15) = \blacksquare$$

Зеркальное отражение

Нормальный вектор микрограницы

$$\mathbf{N} = \mathbf{L}' \cdot \text{Direction} - \text{Ray.Direction} = (-0.928, 0, 0.371) - (0.447, 0, 0.894) = (-1.375, 0, -0.523)$$

и нормализуем его

$$\mathbf{n} = (-0.935, 0, -0.355)$$

Косинус угла между нормалью поверхности и нормалью микрограни

$$\mathbf{nn} = \mathbf{n} \cdot \mathbf{n} = (-1, 0, 0) \cdot (-0.935, 0, -0.355) = 0.935$$

Полученное произведение возводим в степень p (коэффициент Фонга)

$$\mathbf{NRP} = 0.935^3 = 0.817$$

и умножаем на затенение источника \mathbf{sn} и на \mathbf{ks}

$$\mathbf{SN} \cdot \mathbf{KS} \cdot \mathbf{NRP} = 0.557 \cdot 0.5 \cdot 0.817 = 0.227$$

Умножаем цвет источника света на полученное значение

$$\mathbf{BS} = (255, 255, 255) \cdot 0.227 = (58, 58, 58) = \blacksquare$$

Результат складывается с рассчитанной ранее освещенностью (фоновая + диффузная)

$$\mathbf{B}' = \mathbf{B} + \mathbf{BD} = (178, 15, 15) + (58, 58, 58) = (236, 73, 73) = \blacksquare$$

На этом процесс вычисления завершается, так как коэффициент вклада отраженного луча \mathbf{KR} для прямоугольника 1 равен 0. Вычисления возвращаются к расчету интенсивности по отраженному лучу, и полученная интенсивность умножается на коэффициент $\mathbf{KR} = 0.5$

$$\mathbf{B}' = \mathbf{B}' \cdot \mathbf{KR} = (236, 73, 73) \cdot 0.5 = (118, 36, 36) = \blacksquare,$$

после чего складывается с непосредственной интенсивностью

$$\mathbf{B} = \mathbf{B} + \mathbf{B}' = (50, 197, 50) + (118, 36, 36) = (168, 233, 86) = \blacksquare.$$

Результат трассировки приведен на рисунке 6. Хорошо видно отражение красного прямоугольника на зеленом и влияние цвета фона (светло-голубой) на цвет зеленого прямоугольника. Для сравнения на рисунке 7 приведен результат трассировки при $\mathbf{KR} = 0$, а на рисунке 8 — при $\mathbf{KR} = 0.5$ для обоих прямоугольников.

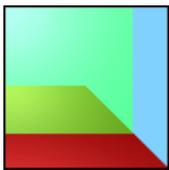


Рис.6.

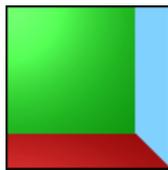


Рис.7.

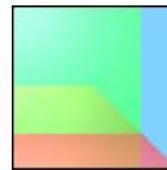


Рис.8.

Не берусь судить, насколько реалистично переданы цвета...

Преломленный луч

Преломление подчиняется закону Снеллиуса, согласно которому векторы View, T и N (Рис.9.) лежат в одной плоскости и для углов между ними справедливо соотношение

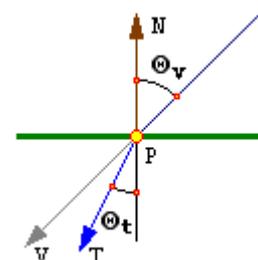
$$(5) \quad n_v \cdot \sin(\Theta_v) = n_t \cdot \sin(\Theta_t).$$

Найдем выражение для вектора \mathbf{t} . Его можно представить как линейную комбинацию векторов \mathbf{v} и \mathbf{n} :

$$\mathbf{t} = \alpha \cdot \mathbf{v} + \beta \cdot \mathbf{n}.$$

Выражение (5) перепишем так:

$$\sin(\Theta_t) = \eta \cdot \sin(\Theta_v), \text{ где } \eta = n_v/n_t.$$



Тогда $\eta^2 \cdot \sin^2(\Theta_V) = \sin^2(\Theta_T)$ или $\eta^2 \cdot [1 - \cos^2(\Theta_V)] = 1 - \cos^2(\Theta_T)$

Учитывая, что $\cos 2(\Theta_V) = (-V \cdot N)$ и $\cos 2(\Theta_T) = (-T \cdot N)$, получим

$$(6) \quad \alpha^2 \cdot (V \cdot N)^2 + 2 \cdot \alpha \cdot \beta \cdot (V \cdot N) + \beta^2 = 1 + \eta^2 \cdot [(V \cdot N)^2 - 1].$$

Из условия нормировки вектора T имеем

$$\|T\|^2 = (T \cdot T) = \alpha^2 + 2 \cdot \alpha \cdot \beta + \beta^2 = 1.$$

Вычитая это соотношение из (6), получим

$\alpha^2 \cdot [(V \cdot N)^2 - 1] = \eta^2 \cdot [(V \cdot N)^2 - 1]$, откуда $\alpha = \pm \eta$ и из физических соображений $\alpha = \eta$.

Второй параметр определяется из уравнения $\beta^2 + 2 \cdot \beta \cdot \eta \cdot (V \cdot N) + \eta^2 - 1 = 0$,

дискриминант которого равен $4\{1 + \eta^2 \cdot [(V \cdot N)^2 - 1]\}$.

Вектор t определяется из решения этого уравнения и равен

$$(7) \quad T = \eta \cdot V + \{\eta \cdot CV - \text{Sqr}[1 + \eta^2 \cdot (CV^2 - 1)]\} \cdot N,$$

где $CV = \cos(\Theta_V) = -(V \cdot N)$.

Случай, когда выражение под корнем (внутри функции Sqr) отрицательно, соответствует так называемому полному внутреннему отражению, когда вся световая энергия отражается от границы раздела сред и преломления фактически не происходит.

В целом вклад преломленного луча рассчитывается по формуле

$$(8) \quad BT = B \cdot KT \cdot e^{-AT} \cdot DT.$$

Здесь:

BT — освещенность, пришедшая по преломленному лучу;

B — освещенность, полученная в результате трассировки сцены преломленным лучом;

KT — коэффициент вклада преломленного луча;

AT — коэффициент затухания среды, в которой распространяется преломленный луч;

DT — расстояние, пройденное преломленным лучом.

Таким образом, полная освещенность точки для одного источника света рассчитывается по формуле

$$(9) \quad B = BA \cdot C \cdot KA + BL \cdot C \cdot KD \cdot SH \cdot (L \cdot N) + BL \cdot KS \cdot SH \cdot (H \cdot N) P + BR \cdot KR \cdot e^{-AR} \cdot DR + BT \cdot KT \cdot e^{-AT} \cdot DT,$$

называемой моделью освещенности Уиттеда (Whitt).

Программирование с преломленным лучом

При расчете освещенности точки прежде всего нужно выяснить взаимное расположение нормали и вектора наблюдения, вычислив их скалярное произведение. В предыдущих случаях косинус угла между этими двумя векторами был отрицателен (хотя рассчитывали мы его всего один раз для отраженного луча). Отрицательное значение свидетельствует о том, что трассирующий луч входит в поверхность со стороны внешней нормали, то есть извне тела. Так как теперь мы должны рассчитывать освещенность от обратных сторон поверхностей, необходимо потребовать, чтобы косинус был отрицательным, а нормаль была обращена против вектора трассирующего луча.

С этой целью расчет косинуса ν_n следует вынести из функции `shade` в расчет точки пересечения, в объект. Объект `ПЕРЕСЕЧЕНИЕ` содержит свойство `cosvn` для этого косинуса. Для фиксации того, является ли трассирующий луч входящим в тело или выходящим из него, объект `ПЕРЕСЕЧЕНИЕ` имеет также логическую переменную `RayIsEntering`. Она нужна для правильного расчета отношения коэффициентов преломления. Если луч входит в тело, то коэффициент преломления среды, в которой распространяется луч, делится на коэффициент преломления среды объекта. Если луч выходит из тела, то наоборот, коэффициент преломления среды объекта делится на коэффициент преломления среды, в которой распространялся луч. Для объекта типа `ПРЯМОУГОЛЬНИК` расчет точки пересечения должен включать установку правильного значения косинуса `cosvn` и переменной `RayIsEntering`. Если переменная ν_n в расчете отрицательна, `cosvn` объекта `ПЕРЕСЕЧЕНИЕ` получает ее значение, а нормаль — копию нормали объекта `ПРЯМОУГОЛЬНИК`. В противном случае `cosvn` получает отрицательное значение ν_n , а нормаль — противоположное значение нормали объекта (`minus`). Переменная `RayIsEntering` для объекта типа `ПРЯМОУГОЛЬНИК` всегда должна быть ложью, так как объект не имеет одного измерения и луч сразу выходит из него.

Для учета среды распространения объект `луч` имеет переменную `medium`. Для первичного луча эту переменную следует установить в значение `air` в процедуре `trace` при формировании луча.

В расчет отраженного луча следует внести изменение: свойству `medium` формируемого луча нужно присвоить свойство `medium` поверхности точки пересечения. Кроме этого, в начале функции `shade` нужно присвоить переменной ν_n значение косинуса `cosvn` из объекта `ПЕРЕСЕЧЕНИЕ`.

После расчета отраженного луча добавляем расчет преломленного. Начальная точка преломленного луча была вычислена ранее (она совпадает с начальной точкой отраженного луча). Рассчитываем коэффициент веса преломленного луча так же, как и для отраженного

```
Ray.Weight = View.Weight * KT,
```

и проверяем его значение. Если полученный вес превышает значение 0.01, продолжаем рассчитывать преломленный луч.

Определяем отношение коэффициентов преломления η в зависимости от того, входит луч в тело или выходит из него:

```
ЕСЛИ Входим_В_Тело ТО
     $\eta = \text{View.Medium.Refract} / \text{Inter.Surface.Refract}$ 
ИНАЧЕ
     $\eta = \text{Inter.Surface.Refract} / \text{View.Medium.Refract}$ 
КОНЕЦ
```

Определяем дискриминант уравнения (7)

```
CV =  $-\nu_n$ 
```

```
D =  $1 + \eta \cdot \eta \cdot (CV \cdot CV - 1)$ 
```

Проверяем случай полного внутреннего отражения. Если дискриминант больше 0.01, вычисляем направление вектора преломления T (в программе вместо T используется переменная `ray`), в противном случае вычисление вклада преломленного луча завершается.

```
T.Direction = View *  $\eta + N \cdot (\eta \cdot CV - \text{Sqr}(D))$ 
```

Вычисляем освещенность, которую несет преломленный луч трассированием сцены вычисленным лучом:

$BT = \text{Trace}(T)$.

Умножаем полученную освещенность на коэффициент вклада преломленного луча и добавляем полученный результат к вычисленной ранее освещенности точки:

$B = B + BT \cdot KT$

Осталось учесть затухание, если оно задано для среды распространения луча. В процедуре трассировки `trace` после вычисления освещенности точки, если точка пересечения существует:

$Atten = \text{Ray.Medium.Atten}$

ЕСЛИ ($Atten > 0.01$) ТО

$\text{Color} = \text{Color.MulBy}(\text{Exp}(-\text{Inters}(1) \cdot \text{Dist} \cdot \text{Atten}))$.

Одновременно в этом месте будет учтено затухание отраженного луча.

На рисунке 10 приведен пример полной трассировки для сцены из стеклянного куба и прямоугольника, расположенного под ним. Прямоугольник имеет коэффициент отражения 0.3, куб имеет оба коэффициента (отражения и преломления) 0.3. Источник света находится возле левой вершины куба. На рисунке 11 источник света между объектами.

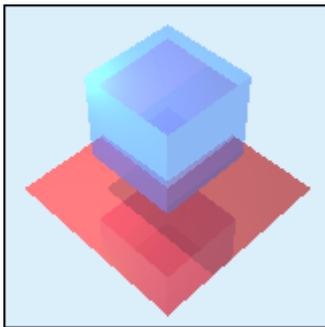


Рис.10.

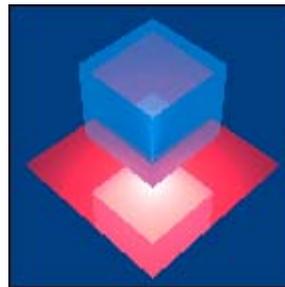


Рис.11.

Расчетные формулы для объекта типа ПАРАЛЛЕПИПЕД

Расчет параметров

Требуются глобальные массивы из 3-х элементов для хранения нормалей, расстояний 1 и 2.

- 1) Начальная точка Loc - 1 (первая, Рис.12.);
- 2) Направляющие векторы $E1$, $E2$, $E3$ - между точками (2-1), (4-1), (5-1);
- 3) Центральная точка $Center = Loc + 0.5 \cdot (E2 + E2 + E3)$;
- 4) Нормали: $N(1) = E1 \times E2$, $N(2) = E1 \times E3$, $N(3) = E2 \times E3$ И нормализовать;
- 5) Расстояния до плоскостей:

$$D1(1) = -N(1) \cdot Loc, \quad D2(1) = -N(1) \cdot (Loc + E3)$$

$$D1(2) = -N(2) \cdot Loc, \quad D2(2) = -N(2) \cdot (Loc + E2)$$

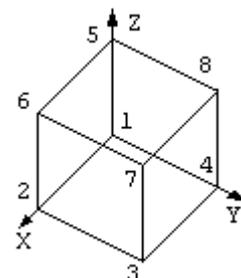


Рис.12.

$$D1(3) = -N(3) \cdot Loc, D2(3) = -N(3) \cdot (Loc + E1)$$

б) Для каждой пары расстояний

ЕСЛИ $D1(i) > D2(i)$ ТО

$$D1(i) = -D1(i), D2(i) = -D2(i), N = -N$$

КОНЕЦ

Расчет точки пересечения

ДЛЯ i ОТ 1 ДО 3

$$VD = Ray.Direction \cdot N(i), VO = Ray.Origin \cdot N(i)$$

ЕСЛИ $VD > 0.01$ ТО

$$T1 = -(VO + D2(i)) / VD$$

$$T2 = -(VO + D1(i)) / VD$$

ИНАЧЕ ЕСЛИ $VD < 0.01$ ТО

$$T1 = -(VO + D1(i)) / VD$$

$$T2 = -(VO + D2(i)) / VD$$

ИНАЧЕ ЕСЛИ $VO < D1(i)$ ИЛИ $VO > D2(i)$ ТО

Не пересекает

ИНАЧЕ

Следующая итерация

КОНЕЦ

ЕСЛИ $T1 > TNear$ ТО

$$TNear = T1$$

$$Index = i$$

КОНЕЦ

ЕСЛИ $TFar > T2$ ТО $TFar = T2$

ЕСЛИ $TFar < 0.001$ ТО Не пересекает

ЕСЛИ $TNear > TFar$ ТО Не пересекает

ЕСЛИ $TNear < 0.001$ ТО Не пересекает

КОНЕЦ ЦИКЛА

$T = TNear$

$$Point = Ray.Origin + Ray.Direction \cdot T$$

$$Normal = N(Index)$$

ЕСЛИ $(Point - Center) \cdot Normal < 0$ ТО $Normal = -Normal$

$$VD = Ray.Direction \cdot Normal$$

ЕСЛИ $VD < 0$ ТО

Входим в тело

Копируем VD и $Normal$

ИНАЧЕ

Выходим из тела

Копируем $-VD$ и $-Normal$

КОНЕЦ

Подготовил Вл. Пономарев. 11.11.2002.