

Задание 2: Излучательность

Фролов Владимир, Алексей Игнатенко

15 апреля 2015 г.

Аннотация

Основная цель данного задания - изучение основ метода излучательности, расчёта глобальной освещенности и базовой визуализации. Задание предполагает реализацию метода излучательности целиком, но для простой трехмерной сцены. При его выполнении придётся столкнуться с трассировкой лучей, либо с проективными преобразованиями. Для выполнения данного задания необходимы базовые знания линейной алгебры и аналитической геометрии, и умение программировать на любом языке. Желательно иметь опыт любой трехмерной визуализации. Объем расчётной части задания (без учёта визуализации, создания окна и.т.д) - около 500 строк кода.

1 Введение

Алгоритм излучательности - один из самых старых методов расчёта глобальной освещенности. Однако, это не означает, что сегодня он потерял свою актуальность. Позже мы рассмотрим применение излучательности в движке «Frostbyte 2» (на котором сделана игра «Battlefield 3») компании «DICE». Но сначала остановимся на описании метода и непосредственно самом задании.

2 Базовые принципы работы излучательности

Излучательность (Radiosity) по определению - это энергия, покидающая поверхность на единицу площади и в единицу времени. Алгоритм расчёта освещения, называемый также словом - «излучательность» исходит в своей основе из закона сохранения энергии и предположения «энергетического равновесия».

Давайте рассмотрим простой пример (комнату из 4 стен в 2D, рисунок 1) и попробуем разобраться, что это означает. Предположим, что одна из стен излучила некоторое количество энергии (световой) в некотором очень коротком промежутке времени (и тут же потухла). Закон сохранения энергии говорит, что вся энергия которая была излучена этой стеной останется внутри комнаты. Для излученного света это означает, что он весь придет на остальные 3 стены.

Далее поверхности могут поглотить часть энергии (перевести её в тепло). Оставшаяся часть энергии отражается обратно в комнату и ... процесс повторяется до бесконечности, а отражённая энергия каждый раз уменьшается, т.к. часть её переходит в тепло. То есть в закрытой комнате (замкнутой системе) вся излучённая однажды энергия с течением времени уйдет в тепло.

Наверняка вы догадались, что описанный процесс происходит в очень короткий промежуток времени, т.к. свет распространяется мягко-говоря быстро. Можно сказать, что именно в этот маленький промежуток времени пока свет от потолка не дошел до стен, и потолок не начал излучать новые порции энергии система не находится в энергетическом равновесии.

Точное моделирование описанного процесса во времени по понятным причинам затруднительно, и не нужно. Гораздо удобнее рассматривать достаточно большой промежуток времени, в котором источник постоянно излучает свет и этот свет постоянно путешествует по комнате. В связи с этим говорят об «энергетическом равновесии». То есть это равновесие между излучённой (световой) и поглощенной (которая перешла в тепло) энергией. То есть утверждается, что уровень освещенности в сцене не колеблется со временем.

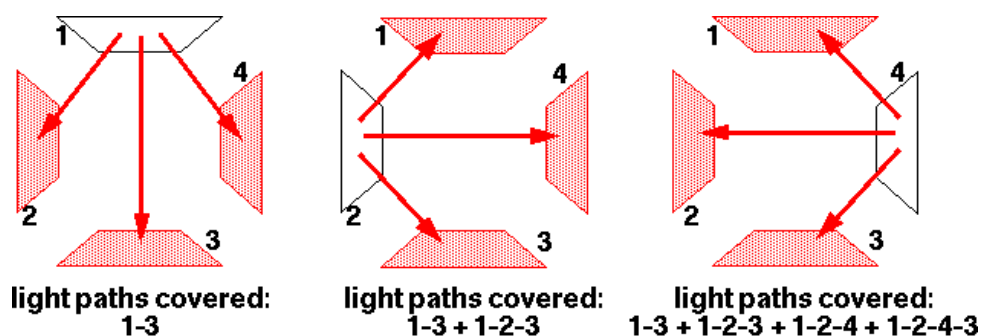


Рис. 1: Простая 2D комната со светящимся потолком.

Таким образом, термин «энергетическое равновесие в системе» для нас на самом деле всего лишь означает, что уровень освещенности в сцене не меняется со временем. То есть мы можем игнорировать время и рассматривать «мгновенное» решение.

3 Алгоритм излучательности в общем

Алгоритм излучательности работает в терминах энергии, переносимой между небольшими площадками (patch).

1. Поверхности сцены разбиваются на конечное число площадок. Площадки имеют малый, но конечный размер и состоит из нескольких последовательных шагов.
2. Источники света также представляются площадками, излучающими энергию.

3. Для каждой пары площадок рассчитывается так называемый форм-фактор F_{ij} . Форм-фактор F_{ij} представляет собой часть энергии, переносимой с патча i на патч j . таким образом получаем матрицу. Из закона сохранения энергии следует, что вся излучённая энергия их патча i энергия должна прийти на какие-то другие патчи. То есть сумма форм-факторов в строке должна быть равна 1.
4. После того как форм-факторы известны, начинается процесс решения линейного уравнения излучательности. Идет расчёт того, какая часть фактической энергии на какие площадки переходит. Здесь возможны как минимум 2 способа - прямой расчёт и решение СЛАУ. Мы опишем прямой расчёт, т.к. он проще в понимании и практичней в реализации (переход к СЛАУ делает площадки равнозначными друг относительно друга, а это не позволит сделать иерархическую излучательность).
5. После того как суммарная излучаемая каждой площадкой известна, можно визуализировать сцену. Поскольку материал Ламберта энергию излучает равномерно во все стороны, то для того чтобы посчитать яркость площадок, необходимо разделить энергию каждой площадки на её площадь.

4 Сцена

В качестве основного геометрического элемента предлагается использовать выровненные по осям координат прямоугольники (Axis Aligned Quad). Этот примитив определяет прямоугольник, находящейся в 1 из 3 плоскостей - XoY , XoZ или YoZ . Его можно задать в виде позиции P и размера в 2 направлениях, перпендикулярных вектору нормали n (рисунок 2). Нормаль n у прямоугольника XoY может быть либо $(0,0,1)$, либо $(0,0,-1)$.

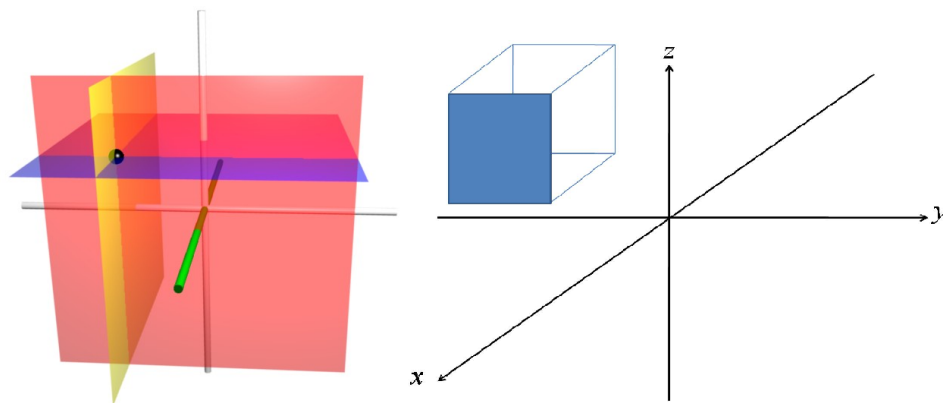


Рис. 2: Выровненные по осям координат прямоугольники и прямоугольный параллелепипед.

С таким примитивом достаточно легко искать пересечение луча. Пусть у вас есть луч с началом в точке O и направлением D . И есть плоскость YoZ . Тогда параметрическая

координата t точки пересечения луча и плоскости будет равна $t_{hit} = (P.x - O.x)/D.x$. Для того чтобы понять, пересекает ли луч ваш прямоугольник, Вам останется проверить, лежит ли найденная точка (выражаемая из уравнения прямой как $P_{hit} = O + D * t_{hit}$) в пределах границ прямоугольника.

Вы можете составить из таких прямоугольников стены комнаты, параллелипипеды и более сложные сцены (рисунок 3). Вы также можете использовать любое другое представление поверхностей сцены.



Рис. 3: Скриншот из игры minecraft.

5 Расчёт форм-факторов

Дифференциальный Форм-Фактор для двух площадок i и j (рисунок 4) записывается следующим образом:

$$FdA_i dA_j = \frac{\cos \theta_i \cos \theta_j}{\pi r^2}$$

Полный форм-фактор для заданного патча будет равен интегралу от дифференциального форм-фактора по площади рассматриваемого патча. Формально, полный Форм-Фактор для двух площадок i и j определяется следующим образом:

$$F_{ij} = \frac{1}{A_i} \int_{x \in p_j} \int_{y \in i_j} \frac{\cos \theta_i \cos \theta_j}{\pi r^2} V(x, y) dy dx$$

Таким образом, мы сделали переход от всех точек сцены к форм-фактору между двумя конкретными площадками. Осталось научиться рассчитывать двойной интеграл по поверхности.

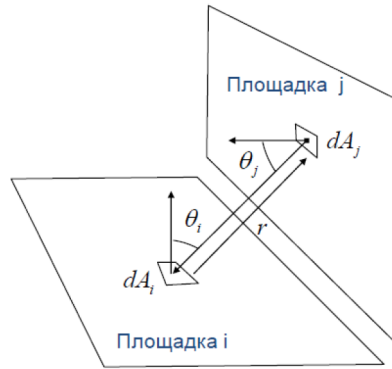


Рис. 4: Взаимное расположение патчей.

Мы приведем здесь простой (и долгий) алгоритм расчёта форм-факторов при помощи трассировки лучей и метода Монте-Карло. Монте-карло интегрирование позволяет посчитать интеграл как сумму большого числа случайных выборок (для форм-факторов нужно примерно от 200 до 1000 выборок). В случае форм-факторов нужно N раз сгенерировать по 2 случайные точки на площадках, посчитать для каждой пары точек дифференциальные форм-факторы и усреднить результат. Полученный результат необходимо умножить на площади обеих площадок (рисунок 5).

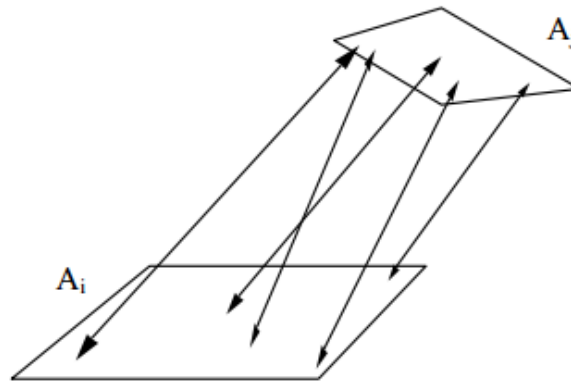


Рис. 5: Интегрирование форм-факторов при помощи Монте-Карло метода и трассировки лучей.

5.1 Свойства форм-фактора

Перечислим основные свойства форм-фактора:

- **Зависимость от геометрии сцены.** *Форм-Фактор зависит исключительно от геометрии сцены, не зависит от положения источников освещения и их интенсивности и от положения камеры.*

- **Обратимость.** Зная соотношение площадей патчей, можно перейти от одного Форм-Фактора к другому: $F_{ij}A_i = F_{ji}A_j$
- **Аддитивность.** Зная Форм-Фактор для пары патчей i и j и Форм-Фактор для пары патчей i и k , объединив патчи j и k , можно вычислить Форм-Фактор для пары патчей i и патча, являющегося объединением патчей j и k . Это свойство будет полезно при реализации иерархической излучательности, когда группу далёких площадок можно рассматривать как 1 большую площадку.
- **Сумма всех возможных Форм-Факторов для каждого патча сцены равна единице.** Вся энергия, покидающая патч, должна прийти на некоторый другой патч.

6 Прямой расчёт излучательности

Алгоритм прямого расчёта достаточно прост. На каждой итерации для каждой площадки собирается вся входящая энергия. В конце итерации она умножается на коэффициент отражения и превращается в исходящую энергию для следующей итерации.

```
struct PatchRadiosity // separate radiosity params from geometry
{
    float3 emmision;
    float3 reflectance;
    float3 incident;
    float3 excident;
    float3 deposit; // this is summ of all passes
};

// std::vector<PatchRadiosity> patchesR(N_PATCHES);

// init phase
for (PatchRadiosity& p : patchesR)
{
    p.excident = p.emmision;
    p.emmision = float3(0, 0, 0);
    p.deposit += p.excident;
}

// do radiosity
```

```

for (int pass = 0; pass < a_numPasses; pass++)
{
    // each patch collects light from the scene
    for (int i = 0; i < patchesR.size(); i++)
    {
        patchesR[i].incident = float3(0, 0, 0);
        for (int j = 0; j < patchesR.size(); j++)
            patchesR[i].incident += patchesR[j].excident*a_formFactors[i][j];
    }

    // deposit emitted energy for current bounce
    for (PatchRadiosity& patch : patchesR)
    {
        patch.excident = patch.incident*patch.reflectance;
        patch.deposit += patch.excident;
    }
}

```

7 Требования

Обязательная часть задания . Необходимо сделать следующее:

1. Реализовать метод излучательности целиком, для простой трехмерной сцены, состоящей как минимум из 1 комнаты с 6 стенами и минимум 1 прямоугольного параллелипипеда, находящегося внутри комнаты.
2. Визуализировать полученное освещение любым удобным способом.
3. Если используете интерполяцию при показе финального освещения, нужно уметь визуализировать отдельно патчи, либо уметь отключать интерполяцию.
4. Обеспечить возможность интерактивно и управляемо изменять начальную излучательность патчей-источников. Допускается несколько возможных реализаций - выбор патча с помощью мышки (рисование по поверхностям сцены), движение по заранее заданным направлениям и клавиатурным стрелкам либо клавишам «WASD». Не допускается случайное перепрыгивание источника с одного патча на другой. Проверяющий должен иметь возможность осветить отдельные выбранные им патчи.
5. При изменении начальной излучательности патчей-источников, освещение должно пересчитываться. Время расчёта не должно превышать 0.5 секунды на средне-

статистическом ноутбуке (в тестовой реализации для 2000 патчей расчёт освещения занимал 2 мс).

6. Время расчёта всех форм-факторов сцене не должно превышать 10 минут на средне-статическом ноутбуке. Хотя бы для одной сцены.
7. Сцена должна быть замкнута, но стены, смотрящие нормалью от наблюдателя не должны рисоваться. Иначе не получится увидеть содержимое комнаты.

На рисунке 6 изображен пример выполненного задания.

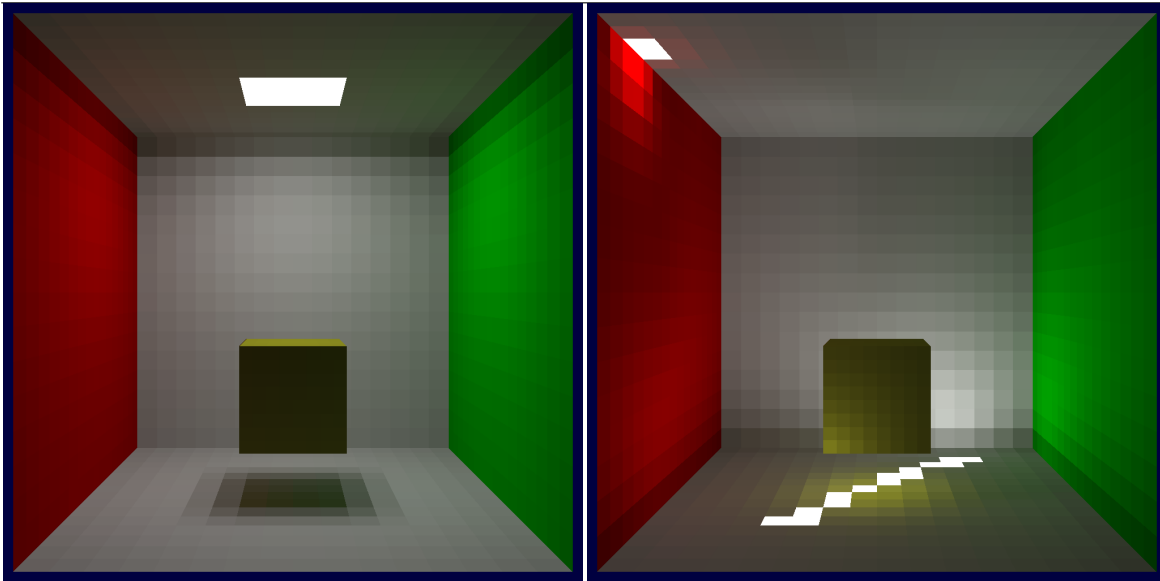


Рис. 6: Пример выполненного задания (на базу).

Что можно делать .

1. Вы можете кэшировать файл с форм-факторами на диск. Однако, программа должна предоставлять возможность пересчитать форм-факторы заново. В `readme.txt` необходимо описать, как это делается.
2. Вы можете использовать любое полигональное представление трёхмерной сцены. Не обязательно ограничиваться только прямоугольными параллелипипедами и стенами, перпендикулярными осям координат. Это упрощение сделано для экономии времени при реализации задания, однако, ни что не мешает Вам реализовать метод излучательности для более сложных сцен.
3. Вы можете обмениваться файлами сцен и форм-факторов с товарищами в любом формате, если вы показываете несколько сцен (как минимум одна должна быть лично ваша!). Вы должны указать в `readme.txt` чьи сцены вы используете. Хотя бы на

одной сцене (а именно вашей) программа должна уметь вычислять форм-факторы заного.

4. Можно писать задание на любом языке программирования.
5. Если вы находите решение через СЛАУ, допускается использование любой сторонней библиотеки решения СЛАУ.
6. Можно использовать любые существующие frame-work-и для визуализации и GUI.
7. Допускается использование сторонней реализации разбиения сцены на патчи в случае использования сложных сцен. К прямоугольникам и кубам это не относится.

Чего нельзя делать

1. Нельзя использовать чужой исходный код в том что касается непосредственно задания за исключением оговоренного выше.
2. Нельзя использовать только чужие сцены с уже рассчитанными форм-факторами. Баллы за базу могут быть снижены в половину. Баллы за дополнительные фишки по сцене не засчитываются.

Подсказки

1. Если вы используете Монте-Карло интегрирование и трассировку лучей для вычисления форм-факторов, проверьте что лучи, определяющие видимость не пересекают исходные площадки для которых считается видимость (то есть пересечения с этими площадками конечно будут, но вы не должны из-за них занулять функцию видимости V).
2. Если вы используете Монте-Карло интегрирование и трассировку лучей для расчёта форм-фактором (лучами), не забудьте умножить полученное значение «частичной суммы/число_выборок» на площади обеих патчей перед тем как поделить результат на площадь i -ого патча.
3. Не забудьте в финале (при визуализации патчей) про гамма-коррекцию. Иначе, вторичное освещение будет выглядеть слабым.

8 Оценки

1. База - 12 баллов.
2. Иерархическая излучательность для большой сцены (число патчей должно быть порядка 100 000, минимум 10 «комнат») - 6 баллов.

3. Сцена, содержащая треугольные меши - 6 баллов.
4. Реализация интересной сцены из квадов и кубиков (2-3 «комнаты» или коридор с 2-3 объектами внутри) - от 2 до 4 баллов.
5. Расчёт освещенности на GPU - 4 балла.
6. Многопоточный расчёт форм-факторов и освещенности +2 балла.
7. Рассчёт форм-факторов на GPU при помощи метода полукуба - 6 баллов.
8. Использование аналогии нусетъта (или тот же метод полукуба) на CPU для расчета форм-факторов - 3 балла.
9. Возможность рисовать мышкой освещение на стенах - 1 балл
10. Возможность покрутить сцену - 1 балл.
11. Возможность путешествовать по комнате или коридору - 1 балл.

Оценки:

1. 8 баллов - 3
2. 12 баллов - 4
3. 16 и выше баллов - 5

9 Применение излучательности в современном мире

Безусловно, изображения на рисунке 6 не выглядят очень реалистично. Отчасти это из-за отсутствия интерполяции при финальной визуализации. В действительности метод излучательности не плох при приближённом расчёте «дальнего вторичного освещения» может конкурировать с такими современными аналогами как LPV и Cone Tracing (рисунок 8).

То есть на практике не нужно пытаться считать при помощи излучательности всю сцену в точности, и тем более не стоит вычислять при помощи неё первичное освещение. В движке «Frost Byte 2» методу излучательности подается на вход упрощенная геометрия (рисунок 7. Излучательность вычисляет только грубую аппроксимацию вторичного освещения и затем полученное решение эксраполируется на детальную геометрию. Локальная детализация вторичного освещения уточняется при помощи «Ambient Occlusion» в пространстве экрана (SSAO), а первичное освещение вычисляется отдельно.

При достаточно небольшом числе площадок финальный расчёт вычислительно прост даже для CPU, чего нельзя сказать о таких методах как LPV и Cone Tracing. Причём, для вторичного диффузного освещения небольшого числа площадок, как правило, достаточно (рисунок 8).

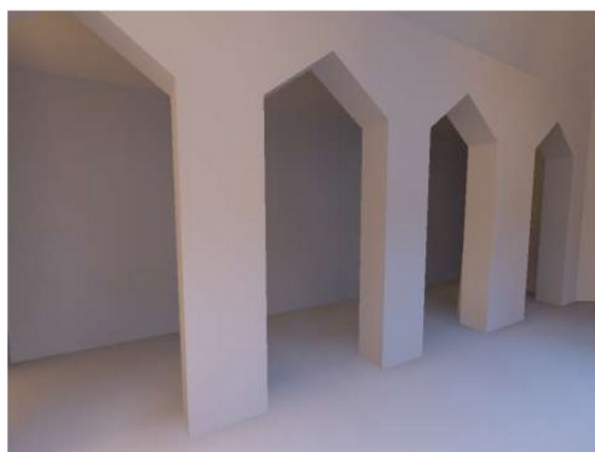


Рис. 7: Детальная геометрия и грубое приближение.

Enlighten Output (Target)



Final Composite

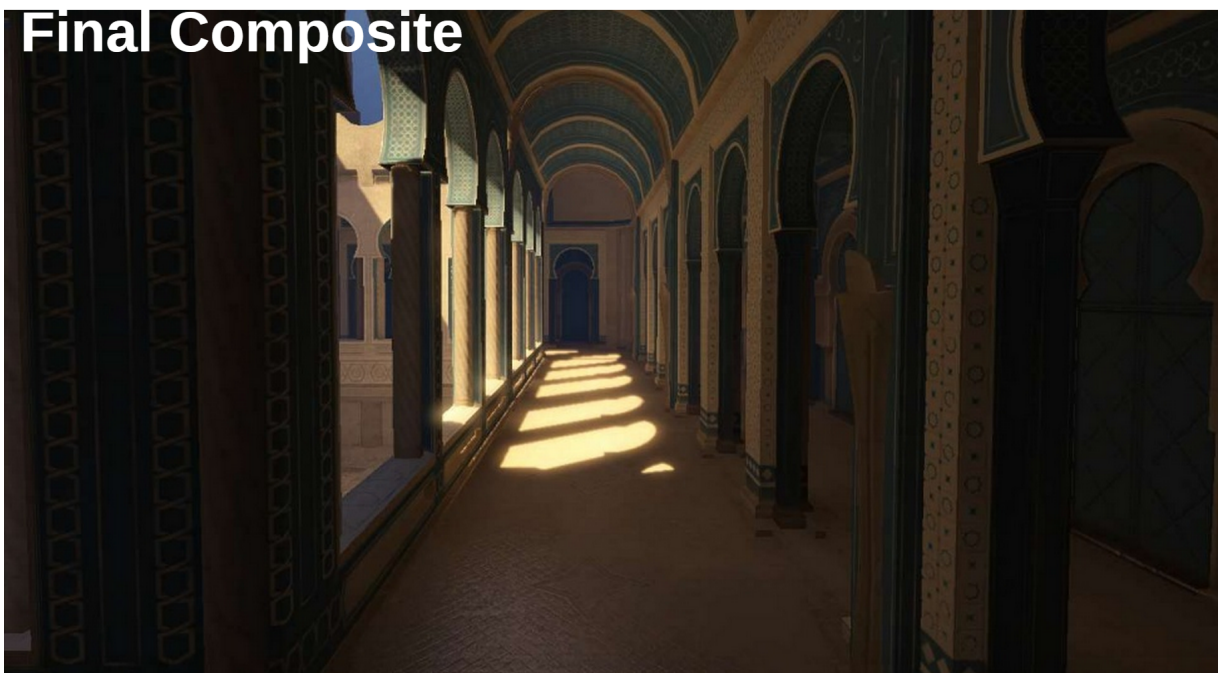


Рис. 8: Использование грубого приближения поверхностей в вижке «Frost Byte 2» и финальное изображение, где вторичное освещение наложено «поверх» детальных поверхностей.