Оглавление

Введение. Задачи синтеза фотореалистичных изображений

Существуют два основных направления создания цифровых изображений: для их автоматического анализа и для показа человеку. При создании изображения для компьютерного анализа не требуется показывать его на экране, достаточно создать матрицу цветовых значений и передать его соответствующей программе. Но когда изображение создаётся для показа человеку, задача становится намного сложнее. Вне зависимости от конкретной цели, которая приводит к созданию изображения, основная задача - это информационная коммуникация с другим человеком. А если требуется передать человеку визуальную информацию, необходимо провести изображение через зрительную систему этого человека. Зададимся вопросом: что нужно для того, чтобы научиться с помощью компьютера синтезировать изображения, неотличимые от реальности (рис 0.1)? Для этого необходимо создать набор цифровых моделей, которые с определённым уровнем достоверности позволяют воссоздать те же процессы, что происходят в реальности. Что же это за модели? На самом деле, часть ответа заключена в самом вопросе, а именно в слове «неотличимые». Похожа картинка на реальность, или нет, решают наши и глаза и наш мозг. Поэтому первым и одним из ключевых элементов синтеза фотореалистичных изображений является понимание принципов функционирования человеческой зрительной системы. Это позволит создать модель зрительного механизма, генерирующую изображения, аналогичные тем, которые формируют наши глаза и мозг, получая информацию из реального мира. Подавая изображение «на вход» системы «глаза-мозг», мы и добьёмся желаемой цели – создания ощущения полной реалистичности синтезированного изображения. Заметим, что эта же задача решается в фотоаппаратах, поэтому часто удобно рассматривать процесс синтеза изображения через создание модели виртуального фотоаппарата, а не глаза (рис. 0.2). Остаётся рассчитать сколько света придёт на каждый светочувствительный элемент фотоаппарата/глаза.

Рис. 0.1. Синетзированное изображение. Высокий уровень реализма обеспечивается моделированием большого количества разнообразных эффектов взаимодействия света с материалами окружающего мира, высокой точностью самого расчёта и, наконец, учётом модели фотоаппарата - т. е. того как формируется итоговое «JPEG» изображение из значений освещённости ячеек матрицы фотоаппарата.

Основная задача реалистичной компьютерной графики — воспроизвести на мониторе изображение модели (объекта), неотличимое от наблюдаемого глазом. В случае виртуального мира можно говорить о задаче создания изображения объекта, неотличимого от реальности (как если бы объект существовал в реальности). Для решения этой задачи необходимо, как минимум, численно измерить свет, попадающий на чувствительный элемент глаза или камеры.

Но это ещё не всё: одних лишь физических расчётов или измерений недостаточно. Необходимо также знать, как устроено человеческое восприятие цвета. В главе 2 мы увидим, что цвет - субъективное для человека восприятие спектра, которое, однако имеет множество различных математических моделей. В главе 3 мы увидим, что на самом деле и этого (моделирование физики света в совокупности с восприятием цвета) недостаточно. Как правило, нужно уметь моделировать всю графическую систему: получение изображения на матрице камеры, хранения и обработка, вывод изображения на монитор (рисунок 0.2) и, наконец, особенности человечесткого восприятия. В первой главе мы сосредоточимся на свете - известном и хорошо изученном физикой явлении.

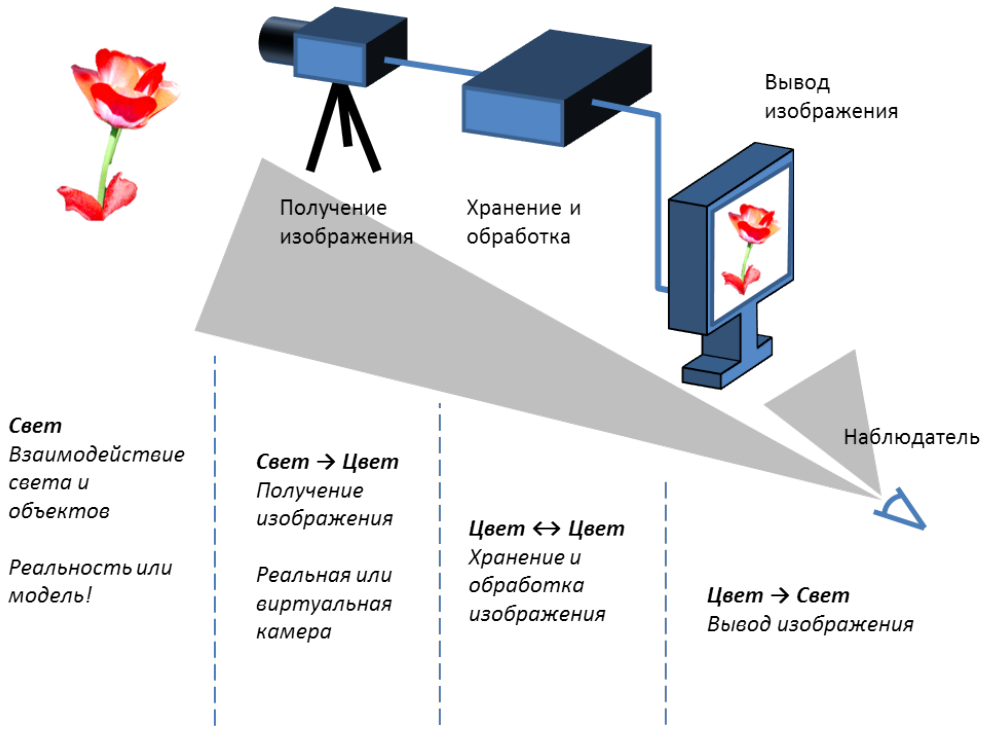


Рис. 0.2. Типичная графическая система и операции со светом и цветом

Глава 1.

Свет как энергия. Радиометрия. Фотометрия

1.1. Свет и его спектральное распределение

Для определения света воспользуемся Большой Советской Энциклопедией:

1) В узком смысле то же, что и видимое излучение, т. е. электромагнитные волны в интервале частот, воспринимаемых человеческим глазом (7,5 · 1014 — 4,3· 1014 Гц, что соответствует длинам волн в вакууме от 400 до 700 нм). С очень высокой интенсивности глаз воспринимает в несколько более широком диапазоне частот. Зависимость чувствительности среднего человеческого глаза от частоты (спектральная чувствительность глаза) характеризуется функцией спектральной световой эффективности (т.н. кривой видности глаза). Эта функция лежит в основе всех светотехнических расчётов. Различие в частоте (или совокупности частот) световых волн в общем — но не в каждом отдельном — случае воспринимается человеком как различие в цвете ...

2) В широком смысле — синоним оптического излучения, включающего, кроме видимого, излучение ультрафиолетовой и инфракрасной областей спектра (диапазон частот приблизительно 3·1011—3·1017 Гц, длин волн в вакууме — от 1 мм до 1 нм). В этом т. н. оптическом диапазоне физические свойства излучения и методы его исследования характеризуются значительной степенью общности (см. Оптика). В частности, именно в оптическом диапазоне начинают отчётливо проявляться одновременно и волновые и корпускулярные свойства электромагнитного излучения [1].

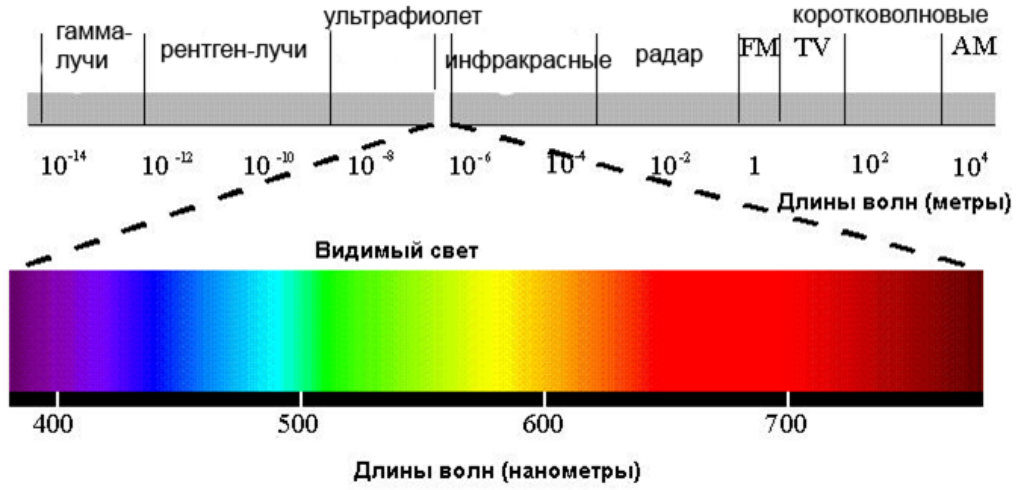


Рис. 1.1. Спектральное распределение

Как и всем электромагнитным излучениям, свету свойственен корпускулярно-волновой дуализм. Это значит, что свет, согласно классической теории имеет волновую природу, но в то же время ведет себя подобно потоку частиц. При этом некоторые эффекты невозможно объяснить на основе корпускулярной теории. Таким образом, возможно рассмотрение света с точки зрения двух теорий: корпускулярной теории (этим занимается геометрическая оптика) и волновой теории (этим занимается волновая оптика). Причины двойственности природы света объясняет квантовая оптика.

1.1.1. Спектральное распределение

Мощность излучения распределена по электромагнитному спектру. Такое распределение называется *спектральным распределением* или просто *спектром.* На рис. 1.1 показан спектр вместе с именованными зонами, соответствующими различным типам излучения. Также выделена видимая часть спектра.

Монохроматическое излучение – это излучение, имеющие только одну частоту и длину волны. Длина волны записывается как λ и измеряется в метрах или производных от метра единицах (см. таблицу 1.1). Частота записывается греческим символом ν и измеряется в циклах (периодах) в секунду. Один цикл в секунду называется один герц (Гц).

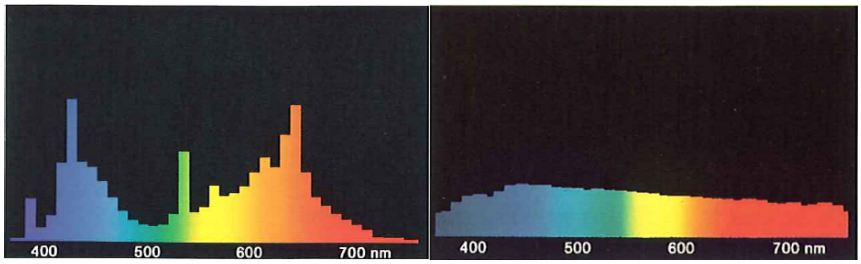


Рис. 1.2. Примеры спектральных распределений для ламп (а) (б)

пикометр	10^{-12}метра = 1/1000 нанометра
нанометр	10^{-9}метра = 1/1000 микрона
микрометр (микрон)	10^{-6}метра = 1/1000 миллиметра
миллиметр	10^{-3}метра=1/1000 метра
метр	1 метр
километр	10^{3}метра=1000 метров

Таблица 1.1. Метр и типичные производные единицы

Длина волны и частота связаны следующим выражением:

$$c = \lambda\nu \tag{1.1}$$

Знание спектрального распределения в потоке энергии очень важно для компьютерной графики. Оптические свойства материалов, такие как отражение, пропускание, поглощение, как правило, зависят от длины волны. Следовательно, эти свойства должны моделироваться также своим спектральным распределением (например, рис. 1.2).

1.2. Геометрическая оптика

Раздел оптики, изучающий законы распространения света в прозрачных средах и принципы построения изображений при прохождении света в оптических системах, называется геометрической оптикой. В этом определении подразумевается, что направление потока энергии светового луча (его ход) не зависит от поперечных размеров пучка света.

В основе геометрической оптики лежат несколько простых законов:

- Закон прямолинейного распространения света

- Закон независимого распространения лучей
- Закон отражения света
- Закон преломления света (Закон Снелла)
- Закон обратимости. Согласно этому закону луч света, распространившейся по определённой траектории в одном направлении, повторит свой ход в точности при распространении и в обратном направлении.

Мы будем в основном рассматривать свет как поток частиц (руководствуясь законами геометрической оптики), поэтому необходимо сразу понять ограничения данной модели, - т.е. какие волновые эффекты не будут моделироваться в рамках представлений геометрической оптики.

Геометрическая оптика предполагает, что направление потока энергии (луча) не зависит от поперечных размеров пучка. Однако, в силу волновой природы света имеют место эффекты дифракции и интерференции. Также геометрическая оптика не учитывает поперечного характера световой волны, поэтому в рамках *традиционной* геометрической теории недоступны эффекты поляризации. Кроме этого в других разделах науки рассматриваются квантовые эффекты распространения света (квантовая оптика) и эффекты, возникающии при взаимодействии световых полей с веществом (нелинейная оптика).

Далее представлено краткое описание основных волновых эффектов.

Дифракция - явление отклонения от законов геометрической оптики, возникающее при сравнимых размерах длины волны и размерах неоднородностей среды. При размерах неоднородностей существенно превышающих длину волны (на 3-4 порядка и более), явлением дифракции, как правило, можно пренебречь.

Интерференция — взаимное усиление или ослабление амплитуды двух или нескольких когерентных волн, одновременно распространяющихся в пространстве, сопровождается чередованием максимумов и минимумов интенсивности.

Поляриза́ция — явление нарушения симметрии распределения возмущений в поперечной волне (например, напряжённостей электрического и магнитного полей в электромагнитных волнах) относительно направления её распространения.

Флюоресценция и фосфоресценция являются частным случаем люминесценции – нетеплового свечения вещества, происходящего после поглощения им энергии. Разница между ними в интенсивности и времени излучения энергии.

Флюоресцирующие материалы излучают в течение крайне короткого времени, тогда как фосфоресцирующие – более низкой интенсивностью и в течение длительного времени (до нескольких часов).

Интересно отметить, что современная наука позволяет расширить геометрическую оптику для того чтобы учитывать все 4 волновых свойства света, описанных выше [2, 3].

1.3. Распространение света

Рассмотрим типичный пример распространения света от источника к приемнику в рамках геометрической оптики (в качестве приёмника выступает элемент матрицы цифровой камеры или чувствительная клетка сетчатки глаза).

Источник света излучает движущиеся частицы, причём мы будем считать, что каждая частица представляет собой излучение на некоторой длине волны. Причины излучения могут быть разными. При тепловом нагреве излучение происходит из-за ускорения колебаний частниц в источнике света, и его температура прямо влияет на спектр излучения. Если температура достаточно высока для излучения в видимом диапазоне, мы видим источник света. Есть и другие причины излучения в видимом диапазоне. Источник света создает *поток излучения*.

Поток, попадая на поверхность, взаимодействует с ней. Некоторые частицы поглощаются объектом, вызывая его нагрев, другие отражаются, третьи проходят через поверхность и продолжают свое движение (в случае прозрачного объекта). Характеристики поверхности объекта задают распределение отраженных частиц в пространстве. Например, для зеркальных объектов подавляющая часть частиц отражается под углом, равным углу падения и в плоскости, задаваемой вектором нормали поверхности и направлением на источник света. Кроме этого, для разных длин волн характеристики поглощения, преломления и отражения могут быть разными. Поэтому объект может казаться цветным при освещении белым светом. Причина в том, что одна часть спектра поглощается сильнее, чем другая.

Взаимодействие потока и поверхности можно охарактеризовать *освещённостью поверхности*, создаваемой потоком. Освещённость равна плотности потока, проходящего через малую площадку поверхности.

Оптическая система камеры отвечает за фокусировку потока излучения на приемнике (матрице камеры или сетчатке глаза). Приемник фиксирует попадание частиц на него. При идеальной фокусировке можно считать, что

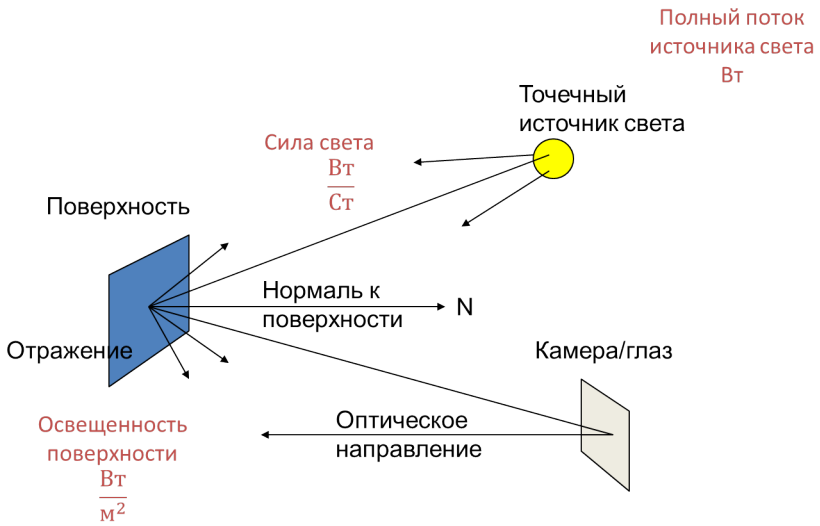


Рис. 1.3. Пример распространения света.

каждой точке приемника соответствует точка поверхности объекта, свет которой достиг приемника. Таким образом, как бы глядя из приемника можно каждой точке объекта сопоставить ее *яркость*, которая зависит как от плотности излучения, так и от ориентации участка поверхности. Точечный источник, площадь которого равна 0, можно описать с помощью *силы света*, которая зависит только от угла распространения энергии.

Для того, чтобы перейти от качественного описания к количественному, необходимо ввести соответствующие единицы и связи величин. Этим занимается радиометрия.

1.4. Радиометрия

Радиометрия — наука об измерении электромагнитного излучения, включая видимый свет. Нам необходимы основные термины, понятия и законы радиометрии для расчета количества света, передающегося по сцене и попадающего на сенсор виртуальной камеры. Радиометрия не учитывает особенностей человеческого восприятия цвета и интенсивности (см. фотометрия, колориметрия).

Радиометрия основана на принципах геометрической оптики. Основные предположения:

Линейность Суммарный эффект двух входных лучей всегда равен сумме эффектов каждого луча по отдельности

Сохранение энергии Рассеиваемый свет не может иметь большую энергию, чем падающий свет.

Отсутствие поляризации . Т.е. изотропная модель распространения (и взаимодействия с поверзхностями) света в пространстве.

Отсутствие флюоресценции и фосфоресценции Поведение света на одной частоте не зависит от поведения на другой

Устойчивость состояния Распределение световой энергии не зависит от времени. То есть мы рассматриваем достаточно большой интервал времени, в котором система находится в энергетическом равновесии. Таким образом, мы пренебрегаем тем фактом, что свет распространяется в пространстве за конечное время и рассматриваем «мгновенное» решение.

Таким образом, в рамках классической радиометрии невозможно измерить результат эффектов дифракции, интерференции, поляризации, флюоресценции, фосфоресценции. Впрочем, последние три эффекта можно сравнительно легко добавить в общую теорию.

1.4.1. Основные термины и их свойства

Воздействие света на глаз или другой регистрирующий аппарат определяется прежде всего энергией переносимой световой волной. Поэтому необходимо рассмотреть понятия и единицы, позволяющие количественно оперировать с такого рода воздействиями.

Фотометрия оперирует четырьмя основными понятиями, которые перечислены ниже вместе с их типичными обозначениями и размерностью.

- Поток Ф, Вт (п. 1.4.2).
- Сила излучения I, $\text{Вт} \cdot \text{Ст}^{-1}$ (п. 1.4.3).
- Освещённость E, $\cdot\,^{-2}$(п. 1.4.4).
- Яркость L, $\text{Вт} \cdot \text{Ст}^{-1} \cdot \text{м}^{-2}$) (п. 1.4.5).

Набор различных понятий необходим для измерений, т.к. разные приборы реагируют на разные световые величины. Например, свет звезды можно оценить через силу излучения, отклик сенсора цифровой камеры пропорционален его освещенности, а человеческий глаз реагирует на уровень яркости источника.

1.4.2. Поток лучистой энергии (Radiant flux)

Лучистая энергия (radiant energy) характеризует энергию некоторого объекта и измерятся в джоулях (Дж). Обычно имеет обозначение Q.

Сама по себе энергия нас не интересует — в рамках радиометрии рассматривается перенос энергии и способы его измерения. Мы рассматриваем энергию в тот момент, когда она по каким-то причинам (например, при нагреве) начинает излучаться некоторым телом и создаёт *поток лучистой энергии*.

Поток лучистой энергии — наиболее «интегральная», обобщающая характеристика излучения энергии телом (или, что эквивалентно — получения энергии телом). Он задаётся как количество энергии Q, излучаемой (или приходящей) за заданное время t, проходящей через заданную поверхность A (описывает скорость излучения):

$$\Phi = \frac{dQ}{dt} \tag{1.2}$$

Мы рассматриваем стационарный процесс, т.е. считаем, что с течением измеряемого времени t поток не меняется. Поток измеряется в ваттах (Вт) — джоулях в секунду. Он также называется мощностью (radiant power) источника.

При рассмотрении потока лучистой энергии необходимо указывать пространственные характеристики объекта или области пространства, где мы измеряем поток.

Поток можно измерить следующим образом: поставить источник энергии и замерить изменение температуры интересующего объекта за заданное время ($Q = mc\Delta t$), где m - масса объекта, а c - удельная теплоёмкость его материала.

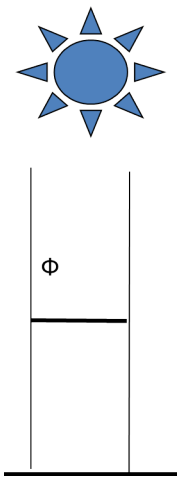


Рис. 1.4. Поток через площадку S

Спектральный поток. Поток лучистой энергии (или просто - «Поток») задает суммарную характеристику излучения по всем длинам волн. Часто необходимо рассматривать спектральный поток Φ_λ, т.е. поток на заданной длине волны. Математически его можно определить следующим образом:

$$\Phi_\lambda = \frac{d\Phi}{d\lambda} \tag{1.3}$$

Спектральное распределение потока энергии важно, например, для вычисления отклика сенсоров на излучение, попадающие на их чувствительную поверхность.

Полный поток. Очень часто применяется понятие *полного потока* (total radiant flux) источника, который можно вычислить как поток через сферу (или другую замкнутую поверхность), описывающую объект (рис. 1.5). Таким образом, полный поток характеризует всю энергию, излучаемую телом за промежуток времени. Величина полного светового потока характеризует излучающий источник и её нельзя увеличить никакими оптическими системами (только перераспределить, сконцентрировать).

Поток позволяет охарактеризовать источник «в общем», с точки зрения мощности излучения. При измерении потока не учитывается ни направление потока, ни площадь излучения.

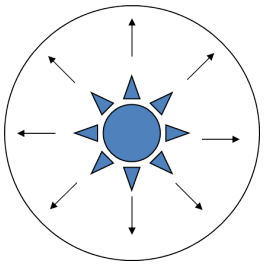


Рис. 1.5. Полный поток.

Для некоторых измерений, однако, требуются более детальные величины, позволяющие оценить степень сконцентрированности излучения энергии в интересующем направлении или на конкретном участке поверхности.

1.4.3. Сила излучения (Radiant Intensity)

Для определения концентрации потока энергии в заданном направлении используется понятие *силы излучения*. Она равна величине плотности потока через малый телесный угол ω.

Определение: *Телесный угол* – часть пространства, ограниченная некоторой конической поверхностью (1.6). Телесный угол измеряется отношением площади S той части сферы с центром в вершине конической поверхности, ко-

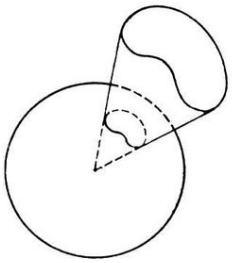


Рис. 1.6. Телесный угол. Поверхность сферы обозначим как Ω

торая вырезается этим телесным углом, к квадрату радиуса R сферы):

$$I = \frac{d\Phi}{d\omega} \tag{1.4}$$

Измеряется в ваттах на стерадиан ($\frac{\text{Вт}}{\text{Ст}}$) и имеет типичное обозначение I.

Сила излучения используется для измерения потока в заданном направлении для точечного источника. При рассмотрении силы излучения игнорируется площадь поверхности светящегося объекта и рассматривается только суммарный поток, посылаемый в телесный угол со всей поверхности объекта. Таким образом, понятие силы излучения применимо либо к объектам, которые можно считать бесконечно малыми (удалённые точечные источники, звёзды и т.п.), либо к объектам, формой которых мы пренебрегаем и рассматриваем только полное излучение со всей поверхности (в этом смысле применяется редко). Для большинства реальных источников сила света сильно меняется для разных направлений. На практике сила света крайне редко используется для чего-либо, кроме описания излучения точечных источников света.

Спектральная сила света. Спектральная сила света I_λ задаётся как поток через малый телесный угол на заданном диапазоне длин волн:

$$I_\lambda = \frac{dI}{d\lambda} = \frac{d^2\Phi}{d\omega d\lambda} \tag{1.5}$$

Связь силы света и полного потока. Если сила света для данного источника одинакова во всех направлениях, то её можно выразить через полный поток Φ как (вывод см. ниже)

$$I = \frac{\Phi}{4\pi} \tag{1.6}$$

Если сила света неравномерна по направлениям, то такое значение определяет среднюю силу света. Средняя сила света может использоваться как характеристика концентрирующей способности оптических систем — чем больше отношение силы света источника в заданном направлении к средней силе света, тем эффективнее оптическая система.

Вывод связи средней силы света и полного потока. Для получения полного потока Φ необходимо просуммировать силу света по всем возможных направлениям (Ω - поверхность сферы), т.е. $\Phi = \int_\Omega I(\omega)d\omega$. Для константного I полный поток оказывается равен $\Phi = I\int_\Omega 1d\omega$, где $\int_\Omega 1d\omega$ — полный телесный угол, равный 4π. Получаем $\Phi = 4\pi I$, из чего сразу следует (1.6).

В общем случае полный поток можно получить путем интегрирования силы света по всем направлениям излучения:

$$\Phi = \int_\Omega I(\omega)d\omega = \int_o^{2\pi}\int_o^{\pi} I(\phi,\theta)sin\theta d\theta d\phi \tag{1.7}$$

1.4.4. Освещённость и светимость

Освещённость (Irradiance) определяется как плотность энергетического потока Φ через заданную площадку A. Т.е. если поток учитывает скорость прохождения энергии через поверхность, освещённость задает поток энергии на единицу площади этой поверхности:

$$E = \frac{d\Phi}{dA} \tag{1.8}$$

Измеряется в ваттах на квадратный метр поверхности ($\frac{\text{Вт}}{\text{м}^2}$), имеет типичное обозначение E.

Из определения освещённости можно видеть, что для набора описывающих сфер и точечного источника в центре, полный поток через них будет одинаковым, а освещённость — разной:

$$E = \frac{\Phi}{S} = \{S = 4\pi R^2\} = \frac{\Phi}{4\pi R^2} \tag{1.9}$$

Аналогично освещенности можно ввести термин *светимость*, который характеризует плотность потока, исходящего с поверхности. Освещённость же характеризует освещение поверхности, т.е. плотность приходящего потока.

Размерность у светимости та же, что и у освещенности (Вт/м^2), типичное обозначение — M.

Освещённость является функцией положения на заданной поверхности и наиболее часто используется, когда необходимо вычислить энергию, падающую на заданную площадку, а распределение энергии по направлениям не важно.

Спектральная освещённость задается как поток через малую площадку на заданном диапазоне длин волн:

$$E_\lambda = \frac{dE}{d\lambda} = \frac{d^2\Phi}{dAd\lambda} \tag{1.10}$$

Освещённость параллельным пучком света Горизонтальный прямоугольник размером WxH освещается параллельным потоком излучения Φ_0 под углом θ к нормали. Какой будет освещённость прямоугольника в зависимости от угла θ (рис. 1.7)?

Пусть $\Phi_0 = E_0 A_0$, где E_0 - освещённость площадки, расположенной перпендикулярно потоку, а A_0 — ее площадь. Тот же самый поток будет падать и на площадку $A = WH$, создавая освещённость $E = \frac{\Phi_0}{A}$. Поток через обе площадки будет одинаковый. Следовательно, получаем: $EA = E_0 A_0$. Из геометрических соображений $A_0 = Acos\theta$, поэтому $E = E_0 \frac{A_0}{A} = E_0 cos\theta$.

По этой причине отклик идеального детектора освещенности будет пропорционален косинусу угла падения параллельного пучка.

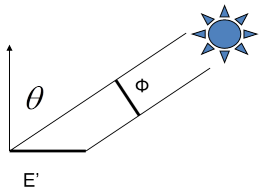


Рис. 1.7. Освещённость и косинус.

Связь силы света и освещенности Рассмотрим освещённость площадки площадью A, расположенную на расстоянии R от точечного источника с силой света I (рис. 1.8).

$$E = \frac{d\Phi}{dA} \overset{\text{следуя (1.4)}}{=} \frac{Id\omega}{dA} \tag{1.11}$$

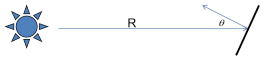


Рис. 1.8. К выводу формулы связи силы света и освещенности.

Учитывая, что $d\omega = \frac{dA\,cos\alpha}{R^2}$, где α — угол между нормалью к поверхности площадки A и направлением на источник, получаем:

$$E = \frac{Id\omega}{dA} = \frac{Icos\alpha}{R^2} \tag{1.12}$$

Полученное выражение представляет собой основной закон освещенности, создаваемой точечным источником (закон обратных квадратов): освещённость поверхности обратно пропорциональна квадрату расстояния от источника до поверхности и прямо пропорциональна косинусу угла межну нормалью к поверхности и направлением на источник. Для протяженных источников света (размерами и неравномерностью потока которых мы не можем пренебречь) зависимость сложнее, хотя на больших расстояниях можно пользоваться и законом для точечных источников.

1.4.5. Яркость (Radiance)

Рассмотрим некоторый источник видимой площади, который нельзя считать точечным, но точная форма которого нам не важна (рис. 1.9).

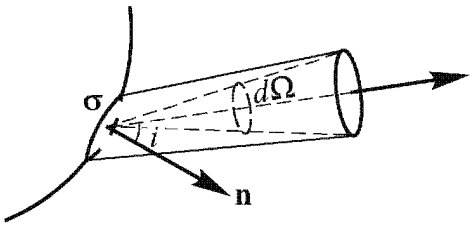


Рис. 1.9. Иллюстрация к определению поверхностной яркости.

Для описания его свойств мы можем использовать следующие единицы (из ранее введенных):

- Поток Ф будет характеризовать излучаемую энергию для этого источника. Это никак не характеризует ни плотность излучения с единицы поверхности, ни распределение направлений излучения, но позволит охарактеризовать полную мощность излучения для данного источника.

- Сила света I для этого источника по заданному направлению позволит вычислить распределение излучения энергии по углам, однако никак не характеризует яркость каждой точки поверхности. Например, если при данной силе света поверхность наблюдается под углом, то такой источник для регистрирующих устройств будет ярче, чем обладающий аналогичной силой света в перпендикулярном направлении, за счет повышения плотности потока в данном направлении, т.к. аналогичный поток будет излучаться с меньшей видимой площади

- Светимость M для данного источника задает полную исходящую энергию с единицы поверхности, однако не учитывает распределение излучения по углам.

Таким образом, для источников, которые нельзя считать точечными, имеет смысл определение понятия поверхностной яркости, или просто яркости. Яркость — это поток, посылаемый в заданный телесный угол с единичной площади источника. Посылаемый в телесный угол поток с участка поверхности пропорционален видимой площади поверхности $A_\perp$ и величине телесного угла. Коэффициент пропорциональности и называется яркостью источника:

$$L = \frac{d^2\Phi}{dA_\perp d\omega} = \frac{d^2\Phi}{cos\theta dA d\omega} \tag{1.13}$$

В приведенном определении (и везде далее) угол θ отсчитывается от нормали к поверхности. Яркость — это функция как поверхности и точки на ней, так и направления. Например, можно говорить о яркости точки на земле в направлении объектива фотоаппарата и т.п.

Из приведённого определения (1.13) видно, что в общем случае яркость зависит от угла наблюдения поверхности. Существуют, однако, источники, для которых поверхностная яркость константна для всех направлений. Такие источники называются ламбертовыми и их свойства подробно рассмотрены в пункте 1.4.7.

Спектральная яркость задается как яркость на единицу длины волны, аналогично другим величинам:

$$L_\lambda = \frac{dL}{d\lambda} = \frac{d^3\Phi}{cos\theta dA d\omega d\lambda} \tag{1.14}$$

Закон сохранения яркости. Понятие яркости источника очень важно в силу закона сохранения яркости, который говорит о том, что при передаче энергии между двумя точками (более точно — дифференциальными площадками)

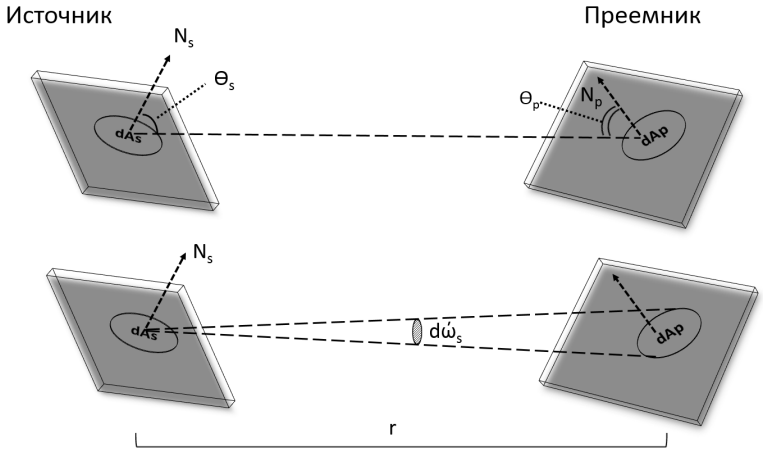


Рис. 1.10. Дифференциальный форм-фактор.

исходящая яркость L_s равна приходящей яркости L_d в случае отсутствия эффекта поглощения энергии средой передачи.

Для доказательства этого определим понятие *дифференциального форм-фактора* для двух малых площадок dA_s и dA_d, расположенных на расстоянии r под углами θ_d и θ_s к отрезку между площадками (рис. 1.10). Телесный угол, который образует дифференциальная площадка dA_d для наблюдателя в центре дифференциальной площадки dA_s составляет

$$d\omega_s = \frac{dA_d cos\theta_d}{r^2} \tag{1.15}$$

Аналогично для наблюдателя на площадке-приемнике:

$$d\omega_d = \frac{dA_s cos\theta_s}{r^2} \tag{1.16}$$

Определим *форм-фактор* как

$$d^2G = dA_s cos\theta_s d\omega_s = \frac{dA_s cos\theta_s dA_d cos\theta_d}{r^2} = dA_d cos\theta_d d\omega_d \tag{1.17}$$

При переходе к протяженным площадкам, форм-факторы можно считать, представляя их как набор дифференциальных форм-факторов (где S_s и S_d - площади соответствующих площадок конечного размера):

$$G = \iint\limits_{S_s S_d} \frac{cos\theta_s cos\theta_d}{r^2} dA_s dA_d \tag{1.18}$$

Далее подставим определение d^2G в определение яркости источника s:

$$L_s = \frac{d^2\Phi_s}{cos\theta_s d\omega_s dA_s} = \frac{d^2\Phi_s}{d^2G} \tag{1.19}$$

Аналогично,

$$L_d = \frac{d^2\Phi_d}{cos\theta_d d\omega_d dA_d} = \frac{d^2\Phi_d}{d^2G} \tag{1.20}$$

Если при передаче поток не теряется (т.е. не происходит поглощения, рассеивания в среде), то $\Phi_d = \Phi_s$. Следовательно, $L_s = L_d$. Зная это, можно рассчитывать освещённость поверхностей, представляя входящий поток через набор параллельных потоков под разными углами и вычисляя яркость источников по соответствующему направлению. На этом принципе, в частности, построены многие алгоритмы синтеза фотореалистичных изображений.

Яркость и человеческий глаз. Как и многие другие детекторы, глаз реагирует на плотность потока через чувствительные элементы сетчатки, т.е. фактически на освещённость. $E = \frac{d\Phi}{dA} = \int L cos\theta d\omega$. Особенности строения глаза позволяют предположить, что свет падает на элементы сетчатки почти перпендикулярно ($cos\theta \approx 1$), а для достаточно удаленных источников, на которых глаз сфокусирован, угловым размером можно пренебречь. В этом случае $E = L$, то есть освещённость равна яркости численно.

Некоторые значения радиометрических величин приведены в таблице

1.2 [4] и проиллюстрированы на рисунке 1.11.

1.4.6. Связь яркости с другими радиометрическими величинами

Большое число понятий, связанных с переносимой светом энергией, обусловлено законом прямолинейного распространения света, в силу которого световая энергия может переноситься по-разному в разных точках поверхности и в различных направлениях. Наиболее дифференциальной характеристикой светового поля служит яркость, определяющая мощность, распространяющуюся в заданном направлении вблизи заданной точки пространства. Сила света описывает мощность, также распространяющуюся в заданном направлении, но от всей поверхности протяжённого источника. Освещённость и светимость

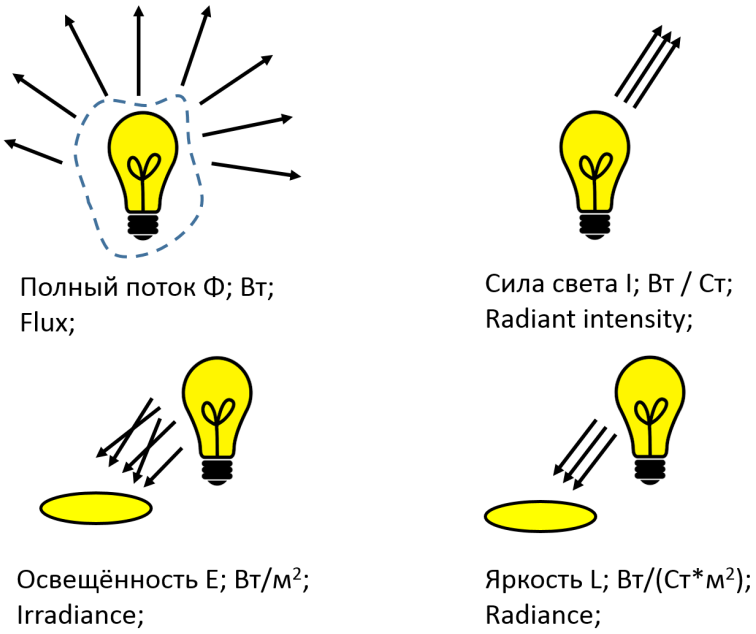


Рис. 1.11. Иллюстрация основных фотометрических величин.

характеризуют мощность, которая распространяется вблизи какой-либо точки поверхности во всех направлениях. Наконец, наиболее интегральной характеристикой является поток - мощность, переносимая через всю поверхность во всех направлениях (рис. 1.11).

Соотношения между яркостью и другими величинами (см. также таблицу 1.3):

$$
\begin{aligned}
I &= \int L_\theta cos\theta dA \\
E &= \int L_\theta cos\theta d\omega \\
\Phi &= \int L_\theta cos\theta d\omega dA = \int I d\omega = \int E dA
\end{aligned}
$$

При наблюдении, например, звёзд, глаз реагирует на свет, испущенный в направлении наблюдателя всей поверхностью звезды, поэтому уместно говорить о силе света звезды (обратите внимание, что для точечных источников, как звёзды, $L = I$). В фотографических приборах неважно, откуда пришел свет на пленку или чувствительный элемент матрицы. Осуществляется интегрирование по направлениям, а значит здесь регистрируется освещённость

Описание	Значение
Полный поток 100 Вт газонаполненной вольфрамовой лампы накаливания	82 Вт
Поток типичного гелий-неонового лазера средней мощности, на частоте 632,8 нм	5 мВт
Поток 40 Вт лампы дневного света	23,2 Вт
Заотмосферная освещённость на средней земной орбите	1367 $\frac{\text{Вт}}{\text{м}^2}$
Земная прямая солнечная освещённость, чистое небо, зима, юго-восток США, полдень	852 $\frac{\text{Вт}}{\text{м}^2}$
Земная полная (полусферическая) освещённость, чистое небо, зима, юго-восток США, полдень	686 $\frac{\text{Вт}}{\text{м}^2}$
Яркость Солнца на поверхности	$2,3 \times 10^7 \frac{\text{Вт}}{\text{м}^2\text{ст}}$
Видимая яркость Солнца с Земли	$1,4 \times 10^7 \frac{\text{Вт}}{\text{м}^2\text{ст}}$

Таблица 1.2. Некоторые значения радиометрических величин

1.4.7. Свойства ламбертовских источников.

Ламбертовский источник - это источник константной поверхностной яркости, для которого поверхностная яркость одинакова для всех направлений ($L = const$, $\forall\omega \in \Omega$). Ламбертовскими они называется потому, что их сила света прямо пропорциональная косинусу угла между нормалью к поверхности и направлением на наблюдателя ($I = I_0 cos\theta$). Такая зависимость называется законом излучения Ламберта. В этом случае $L = \frac{dI_0 cos\theta}{cos\theta dA} = \frac{dI_o}{dA} = const$.

Строго говоря, в природе таких источников не существует. Однако матированное тело или мутная среда, каждая точка которой рассеивает свет одинакого во все стороны, является достаточно хорошей аппроксимацией ламбертовоского источника. Такие среды или материалы можно называть *идеально рассевающими* (или диффузными, diffuse), если они подчиняются закону Ламберта.

Связь светимости и яркости для ламбертовского источника

Для ламбертовского источника существует простая зависимость между светимостью и яркостью:

$$M_{diffuse} = \pi L_{diffuse} \tag{1.21}$$

Доказывается путем вычисления интеграла в определении светимости через яркость (где S - площадь полусферы вокруг некой точки поверхности):

$$M = \int\limits_S Lcos\theta d\omega = L \int\limits_S cos\theta d\omega = L \int\limits_0^{2\pi} \int\limits_0^{\pi/2} cos\theta sin\theta d\theta d\phi = \pi L$$

Аналогично показывается, что освещённость E равномерно освещенной точки составляет πL.

Яркость светящейся полусферы и диска

Рассмотрим светящийся плоский диск S и светящуюся полусферу S'. Предположим, что обе поверхности подчиняются закону Ламберта. В этом случае для наблюдателя диск S будет неотличим от полусферы S', т.к. видимые площади их будут одинаковы, а яркости не зависят от угла с поверхностью. В частности поэтому сферические источники типа Солнца и Луны воспринимаются как плоские диски.

Наименование	Англ. эквивалент	Ед.	Обз.	Определение
Поток энергии	Radiant flux	Вт	Φ	$\Phi = \frac{dQ}{dt} = \int Lcos\theta d\omega dA = \int Id\omega = \int EdA$
Сила излучения	Radiant intensity	Вт/ср	I	$I = \frac{d\Phi}{d\omega} = \int Lcos\theta dA$
Освещённость	Irradiance	Вт/м^2	E	$E = \frac{d\Phi}{dA} = \frac{Id\omega}{dA} = \frac{Icos\alpha}{R^2} = \int Lcos\theta d\omega$
Яркость	Radiance	Вт/(м^2)ср	L	$L = \frac{d^2\Phi}{cos\theta dAd\omega} = \frac{dE}{cos\theta d\omega} = \frac{dI}{cos\theta dA}$

Таблица 1.3. Основные радиометрические величины и их единицы

1.5. Фотометрия

Человеческий глаз чувствителен в ограниченном диапазоне спектра — от 380 до 780 нм. Более того, внутри этого диапазона чувствительность неравномерна. Например, воспринимаемая яркость монохроматического излучения на длине волны 550нм намного больше, чем у излучения такой же мощности на длине волны 650нм. В связи с этим под эгидой Международной Комиссии по Освещению (МКО) были проведены эксперименты, результатом которых стала кривая спектральной световой эффективности $V(\lambda)$ (иногда её называют «Ви-лямбда»). Эта кривая (показана на рис. 1.12) задает усреднённую чувствительность человеческого зрения по отношению к длине волны излучения.

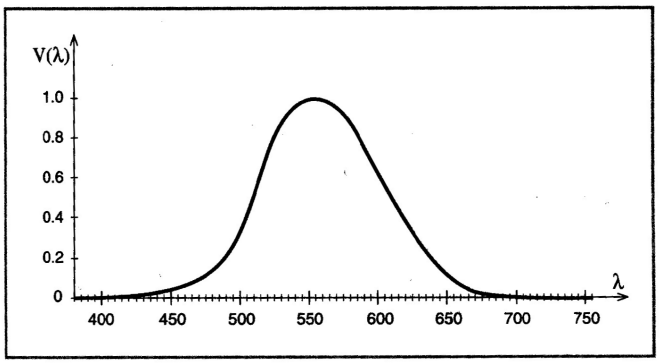


Рис. 1.12. Кривая спектральной световой эффективности

Фотометрические величины могут быть получены из радиометрических с помощью взвешивания спектральных величин функцией $V(\lambda)$:

$$Q_v^f = 683 \int_{380}^{780} Q_v V(\lambda) d\lambda,^1 \tag{1.22}$$

где Q_v– одна из величин Ф, I, E, M или L. Такие величины очень полезны на практике, т.к. их значение соответствует воспринимаемому человеком. Основные фотометрические величины и их соотвествие радиометрическим приведены в таблице 1.4. Световой поток, выражаемый в люменах (Лм) - фотометрический эквивалент энергетического потока. Отсюда можно заметить, что функция $V(\lambda)$ не безразмерна. Ее единицы - люмен/Вт. Другие фотометрические единицы выражаются из люмена так же, как радиометрические из ватт. Сила света измеряется в канделах (Люмен/ст), причем одна кандела равна силе света свечи [1]. Для световой освещенности существует

[1]Константа 683 в (1.22) связана с необходимостью масштабировать фотометрические значения таким образом, чтобы сила света свечи составляла примерно 1 кандел

Радиометрические единицы	Обозн.	Фотометрические единицы	Обозначение
Энергетический поток (radiant flux)	Вт	Световой поток (luminous flux)	Люмен
Энергетическая сила света (radiant intensity)	Вт/ст	Сила света (luminous intensity)	Кандела = Люмен/
Энергетическая освещённость (irradiance)	Вт/м^2	Световая освещённость (illuminance)	Люкс = Люмен/м
Энергетическая яркость (radiance)	Вт/ст/м^2	Световая яркость (luminance)	Нит = Кандел/м

Таблица 1.4. Связь радиометрических и фотометрических единиц

специальная единица – люкс (Люмен/м^2). Яркость выражается канделах на квадратный метр, иногда можно встретить и специальную единицу – нит.

Глава 2.

Цвет. Цветовые пространства. Гамма-коррекция

2.1. Цвет

2.1.1. Цветность и яркость. Задача описания цвета в компьютере.

Восприятие человека позволяет различать различные следующие свойства света: яркость (brightness, luminance), оттенок (hue), насыщенность (saturation) и цвет (color). В этой главе мы рассмотрим аспекты, связанные с восприятием цвета, а именно оттенок и насыщенность. Их нельзя целиком отделить от вопросов восприятия яркости, но подробное рассмотрение этих вопросов мы проведём в третьей секции “Гамма-коррекция”.

Теорию цвета необходимо изучить для разработки алгоритмов, позволяющих добиться точной передачи цвета. Зададимся вопросом, как соответствуют друг другу: цвет, который мы видим в реальности, цвет этого же объекта на экране монитора или проектора, цвет на распечатке фотографии, в графическом редакторе, при использовании OpenGL? Ответ будет разъяснён в этой главе.

Итак, наша задача – описать цвет в машинном представлении. При этом возникает 2 основных проблемы:

1) Ограниченность дискретными конечными числами.

2) Сама возможность представления цвета в виде чисел.

Обе проблемы решаются простым квантованием спектра. Если квантовать энергетический спектр с некоторым шагом (обычно берется шаг 10 нм) и использовать такое представление, то при дискретизации спектра 380-780нм по 10 нм на один пиксель потребуется 40 вещественных чисел. Для цветного изображения размером 1024x1024 пикселей это будет означать, что оно займёт в памяти 160 Мб! Однако это простое решение проблемы представления цвета оказывается избыточным. Далее мы покажем более компактное представление.

2.1.2. Структура и функционирование человеческого глаза

Первичный орган зрительной системы – глаз (рис. 2.1). Это сенсорный орган зрения, преобразующий энергию видимого света в биоэлектрические сигналы нервной системы. Человеческий глаз состоит из светопреломляющего, аккомодационного, адаптационного и рецепторного аппарата.

Светопреломляющий аппарат глаза устроен несколько сложнее, чем может показаться на первый взгляд. Свет поступает на поверхность роговицы глаза, где испытывает первое и второе преломление (на входе в переднюю камеру глаза), затем достигает зрачка, где за счёт малого размера может произойти дифракция. Сильное преломление луча происходит на входе и выходе из хрусталика. В дальнейшем луч аберрирует, проходя через все слои сетчатки. И только после этого он достигает окончаний фоторецепторных клеток.

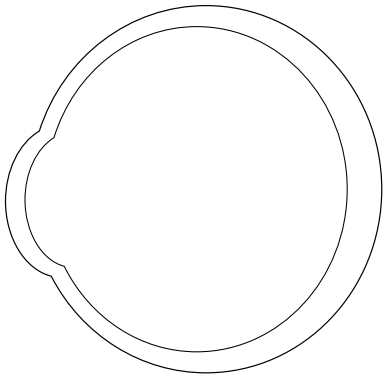


Рис. 2.1. Устройство человеческого глаза.

Механизм аккомодации помогает хрусталику (преломляющей биологической линзе) изменять радиус кривизны (глава 5), за счет чего регулируется расстояние фокусировки. Благодаря этому человек с нормальным зрением способен чётко различать как близкие, так и удалённые объекты.

Зрачок человека также способен менять размер благодаря радужке, это позволяет адаптироваться под изменяющийся уровень освещенности. В темноте

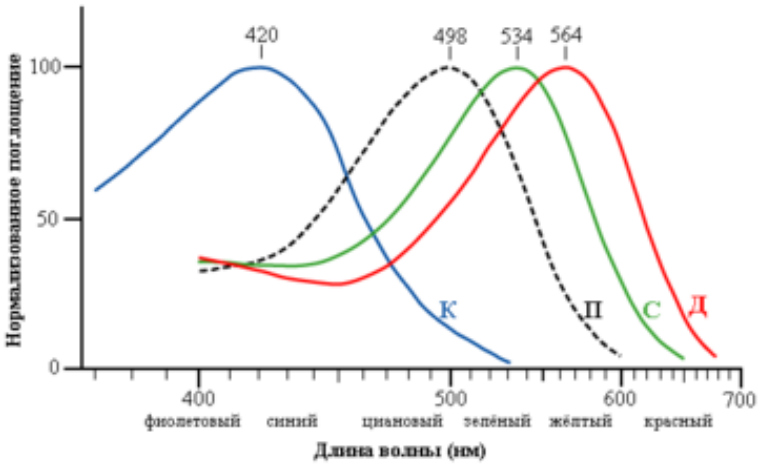


Рис. 2.2. Относительные отклики фоторецепторных клеток сетчатки глаза человека.

зрачки расширяются, при ярком свете сужаются. Это лишь один из уровней адаптации в зрительной системе, регулирующий объём светового потока.

Сетчатка состоит из нескольких слоёв, последний из них (ганглиозные клетки) формирует нейронную сеть для передачи сигнала зрительному нерву. В дальнейшем из всех слоёв сетчатки нас будет интересовать только устройство рецепторного слоя. Фоторецепторные клетки бывают двух типов: светочувствительные палочки и цветочувствительные колбочки. Они работают при разных уровнях освещенности. Палочки срабытывают даже от нескольких фотонов, поэтому работают у человека в условиях ночи. Колбочки дают более детализированное изображение, но требуют больше энергии для срабатывания. Они обеспечивают дневное зрение.

Палочки и колбочки по-разному распределены на сетчатке. В центральной области жёлтого пятна (области наилучшего зрения) преобладают колбочки, на периферии – палочки. Колбочки с разной интенсивностью реагируют на свет определенных длин волн. Всего есть 3 типа колбочек, каждая из которых реагирует на свой диапазон спектра: длинноволновые красные колбочки (L), средневолновые зелёные колбочки (M) и коротковолновые синие колбочки (S). На рисунке 2.2 также приведен график относительной чувствительности палочек к монохроматическому излучению разных длин волн.

2.1.3. Трихроматическая теория

Экспериментальные данные о цветовом восприятии человека позволили выдвинуть так называемую трихроматическую теорию. Идея высказывалась

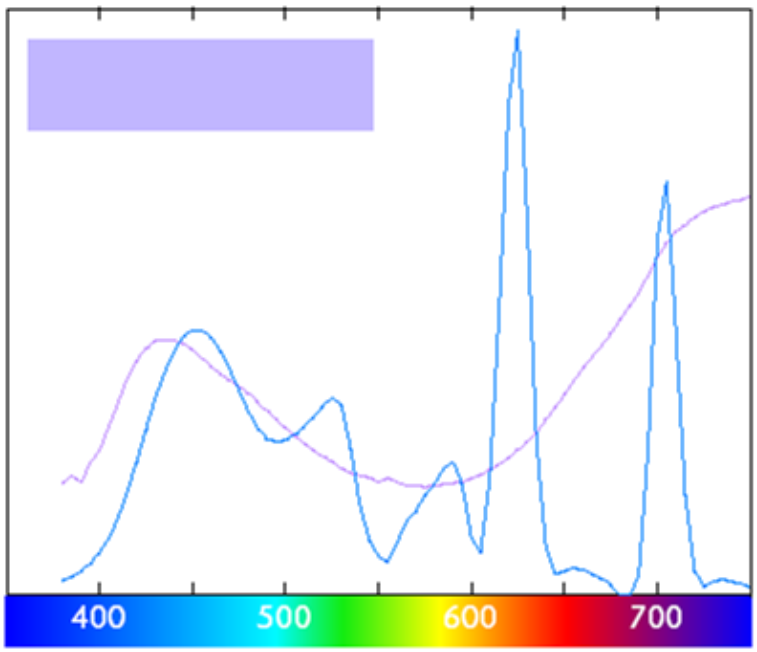


Рис. 2.3. Метамеры. По оси x отложена длина волны. Спектр, обозначенный голубой линией и спектр, обозначенный фиолетовой, с точки зрения человека дают одинаковый фиолетовый цвет.

давно (в том числе М.В. Ломоносовым), но развилась и стала известна в 19 веке благодаря Юнгу, а затем Гельмгольцу. В дальнейшем в 20 веке она подтвердилась более тщательными нейрофизиологическими исследованиями.

Два основных постулата трихроматической теории цветового восприятия таковы:

1) Трихроматия (trichomacy). Весь спектр может быть сведён в точности к трем числам без потери информации для визуальной системы человека.

2) Метамеризм. Все спектры, создающие одинаковые отклики, неразличимы человеком. Такие спектры называются метамерами. На рисунке 2.3 приведен пример двух метамеров, дающих фиолетовый цвет.

Из этих двух постулатов можно сделать главный вывод: не нужно моделировать весь спектр. Для тех частей спектра, которые различает человек, достаточно трёх чисел. Осталось понять, что это за числа. Это было выяснено с помощью оригинальных экспериментов по соответствию цветов, проведённых в 30-е годы 20-го века международной комиссией по освещению (CIE - Commission Internationale de l'Eclairag).

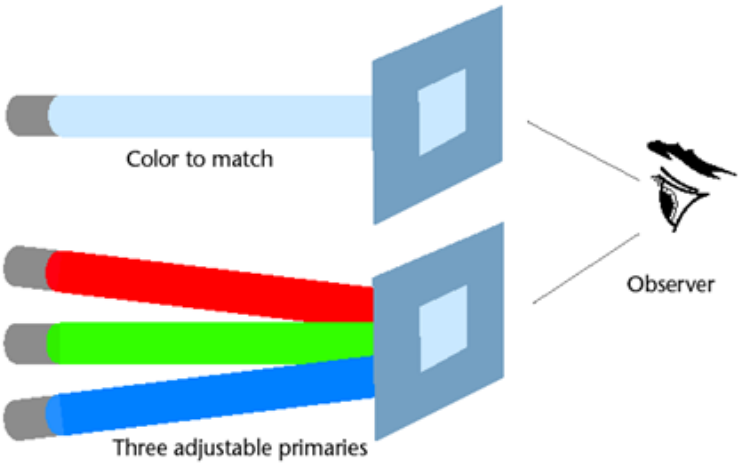


Рис. 2.4. Эксперименты по соответствию цветов.

2.1.4. Эксперименты по воспринимаемому соответствию цветов

Три источника света были направлены на левую часть белого экрана (угловой размер экрана - 2 градуса) таким образом, что их цвет смешивался. Экспериментатор сидел перед экраном, ему давался пульт управления, состоящий из трёх ручек, позволяющих управлять яркостью каждого из трех источников света. Затем на правой стороне экрана показывалась точка некоторого заданного цвета . Задача состояла в том, чтобы настроить ручки таким образом, до совпадения цветов двух точек (рис. 2.4). Лампы были настроены таким образом, чтобы интенсивность каждой контролировалась числом от –1 до 1. При 1 лампа включалась на полную мощность. Если ручка стояла на нуле, лампа была выключена. В положении меньше нуля свет лампы «вычитался» из результирующего. Это достигалось путём увеличения соответствующей компоненты яркости правой точки.

Заданные цвета пробегали спектр 380-780нм с шагом 5нм. Хотя можно пытаться найти соответствие любых цветов, чтобы уменьшить количество, работа велась с монохроматическими цветами. При этом источники света могут быть любые, но такие, чтобы третий нельзя было получить сложением первых двух с некоторыми коэффициентами.

Остановимся подробнее на том, зачем эксперимент был усложнен «вычитанием» цветов из результирующих. Если результирующий цвет может быть получен смешением трёх других, то такое соответствие называется аддитивным (additive). Если же невозможно найти соответствие путём сложения цветов заданных базовых источников света, и необходимо для совпадения один из цветов прибавить к результирующему цвету, то такое соответствие называют субтрактивным (subtractive).

Аддитивное соответствие выражается формулой:

$$C = rR + gG + bB \tag{2.1}$$

где C - заданный цвет, R, G, B - цвета базовых источников света, а r, g, b - коэффициенты модулирования яркости источников. Здесь равенство нужно понимать как воспринимаемое человеком соответствие цветов.

Субтрактивное соответствие можно задать как

$$C + rR = gG + bB \tag{2.2}$$

т.е. условная «красная» лампа прибавляется к заданному цвету вместо подбираемого.

Итак, в результате проведенных экспериментов положения рукояток можно использовать как хранимые коэффициенты для получения цвета любого монохроматического излучения. Однако есть проблемы – результаты верны только для конкретного наблюдателя, для заданных основных цветов (ламп) и только для монохроматических целевых цветов. Для практического использования необходимо расширить их на более широкий класс наблюдателей, на более широкий класс базовых цветов и на как можно более широкий класс целевых цветов.

Для решения первой проблемы обратимся к экспериментам по соответствию цветов, проведённых авторитетной организацией CIE в 1931-ом году. Эти эксперименты были проведены на достаточно большом количестве человек. Хотя результаты разных людей были разными, усреднённый результат при заданных длинах волн для каждой из трёх монохроматических ламп (эти длины волн примерно соответствовали пикам чувствительности разных типов колбочек) был принят за гипотетического стандартного наблюдателя. Поэтому можно считать, что результаты экспериментов могут быть применены к любому человеку с нормальным зрением. Построенные кривые называют кривыми стандартного наблюдателя (рис. 2.5).

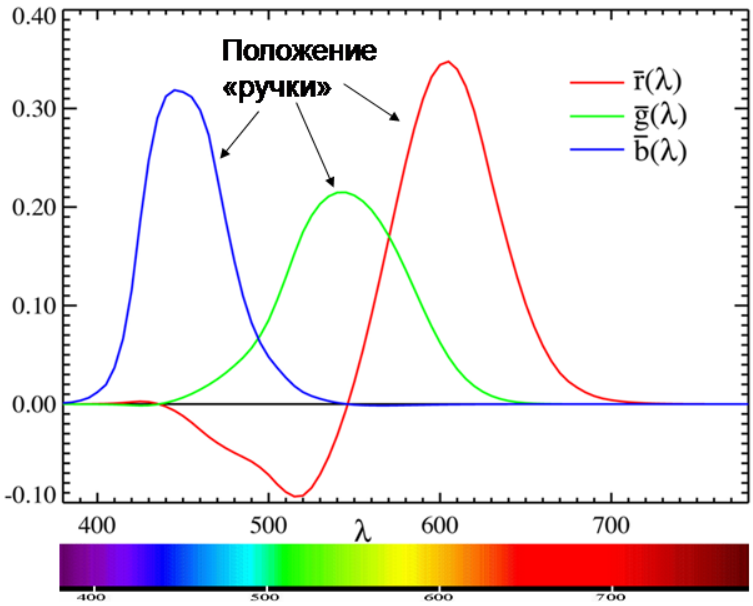


Рис. 2.5. Кривые стандартного наблюдателя.

2.2. Закон аддитивности Грассмана.

Мы знаем:

- что любое излучение можно представить как сумму монохроматических излучений разной интенсивности (амплитуды волны), т.е. $C = \int_{380}^{780} C(\lambda)d\lambda$
- что любой цвет может быть описан тройкой чисел (трихроматия);
- как представить монохроматические цвета с помощью тройки чисел (из экспериментов CIE) для данных базовых цветов.

Возможно ли на основе этой информации найти тройки чисел любого (не обязательно монохроматического) цвета? Ответ – да. Нам на помощь приходит так называемый закон аддитивности Грассмана – эмпирический закон о линейности хроматического человеческого зрения:

- Если наблюдатель задаст цвета лучей 1 и 2 как (R_1, G_1, B_1) и (R_2, G_2, B_2) (где R,G,B - коэффициенты для заданных базовых источников света), и мы сложим эти источники света, то результирующий цвет может быть получен из базовых источников света как $(R_1 + R_2, G_1 + G_2, B_1 + B_2)$.

- Вторая часть закона говорит о том, что соответствие цветов выполняется на всех уровнях интенсивности, т.е. если $C_1 = C_2$, то $kC_1 = kC_2$ (за исключением очень низких и очень высоких уровней интенсивности источников).

Неоднозначность при переводе из RGB/XYZ в спектр

Следствием из закона аддитивности Грассмана является то, что мы можем получить бесконечный набор соответствующих цветов, зная коэффициенты только конечного набора. Любое спектральное распределение может быть получена как взвешенная сумма монохроматических источников. То есть если задать соответствия этих цветов для некоторых базовых источников света, то цвет любого спектрального света будет взвешенной суммой коэффициентов монохроматических цветов.

Пусть спектр света задан функцией $C(\lambda)$, известны кривые стандартного наблюдателя (для заданных базовых источников R, G, B) $x(\lambda)$, $y(\lambda)$, $z(\lambda)$, тогда для монохроматического излучения на волне λ_i получить воспринимаемое соответствие можно будет с помощью коэффициентов

$$X = C(\lambda_i)x(\lambda_i)$$

$$Y = C(\lambda_i)y(\lambda_i)$$

$$Z = C(\lambda_i)z(\lambda_i)$$

Суммируя по всему спектру, получаем:

$$X = \int_{380}^{780} C(\lambda)x(\lambda)d\lambda$$

$$Y = \int_{380}^{780} C(\lambda)y(\lambda)d\lambda$$

$$Z = \int_{380}^{780} C(\lambda)z(\lambda)d\lambda$$

На рисунке 2.6 приведен пример входного спектра (красная линия). Это спектр стандартного источника D65, ему соответствуют координаты $X = 1.0044e + 04, Y = 1.0567e + 04, Z = 1.1507e + 04$. Зеленой линией показан спектр, полученный путем умножения коэффициентов на кривые стандартного наблюдателя. Как мы видим, полученный спектр не совпадает с исходным. Из трихроматического пространства мы не можем однозначно получить исходный спектр.

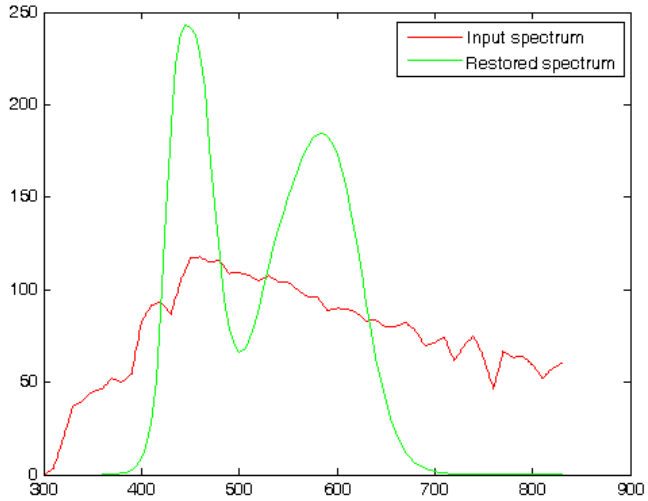


Рис. 2.6. Спектр источника D65 и результат его восстановления по базисным функциям.

2.3. Цветовые пространства

2.3.1. Цветовое пространство CIE RGB

Кривые стандартного наблюдателя и спецификация базовых источников света задают трёхмерное цветовое пространство CIE RGB.

Для любого спектра возможно получение координат в этом пространстве. Процедура их получения была описана в предыдущем пункте. В результате проведённых экспериментов кривые базовых источников были зафиксированы и назвались кривыми стандартного наблюдателя $\bar{r}(\lambda), \bar{g}(\lambda), \bar{b}(\lambda)$ (рис. 2.5).

Поскольку часть кривой $\bar{r}(\lambda)$ лежит в отрицательной области, то не все спектры, порождаемые этим пространством, являются видимыми.

2.3.2. Переход между цветовыми пространствами

Предположим, что мы хотим создать другое цветовое пространство с источниками $X(\lambda), Y(\lambda), Z(\lambda)$. Например, это могут быть реальные источники света для пикселей в LCD-мониторе, проекторе или цвета красок в принтере. Соответственно, необходимо знать коэффициенты яркости именно этих источников, которые позволят нам задавать цвета, привязанные к физически воспринимаемым человеком цветам.

Пусть мы знаем цветовые координаты каждого из трех наших источников в системе CIE RGB $(r_1, g_1, b_1), (r_2, g_2, b_2), (r_3, g_3, b_3)$, т.е. с точки зрения воспринимаемого соответствия цветов:

$$X = r_1 R + g_1 G + b_1 B$$

$$Y = r_2 R + g_2 G + b_2 B$$

$$Z = r_3 R + g_3 G + b_3 B$$

Следовательно:

$$C = xX + yY + zZ = x(r_1 R + g_1 G + b_1 B) + y(r_2 R + g_2 G + b_2 B) + z(r_3 R + g_3 G + b_3 B) = (xr_1 + yr_2 + zr_3)R + (xg_1 + yg_2 + zg_3)G + (xb_1 + yb_2 + zb_3)B$$

Заметим, что коэффициенты при R, G, B формируют оператор перехода из базиса пространства XYZ к базису пространства CIE RGB:

$$r = xr_1 + yr_2 + zr_3$$

$$g = xg_1 + yg_2 + zg_3$$

$$b = xb_1 + yb_2 + zb_3$$

и могут быть записаны в матричном виде следующим образом:

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} r_1 & r_2 & r_3 \\ g_1 & g_2 & g_3 \\ b_1 & b_2 & b_3 \end{pmatrix} \begin{pmatrix} x \\ y \\ x \end{pmatrix}$$

Итак, преобразование между любыми линейными цветовыми пространствами – линейное преобразование. Т.е если мы хотим определить свою цветовую систему, нам необходимо задать параметры базовых источников света, выраженных в координатах другой цветовой системы, для которой известно преобразование в CIE RGB. Понятно, что создавать «цепочки» преобразований до прихода к одной из CIE-систем не самая лучшая практика, поэтому разумно выбрать одно цветовое пространство и считать его стандартом, матрицы перехода в которое должны задавать все другие пространства.

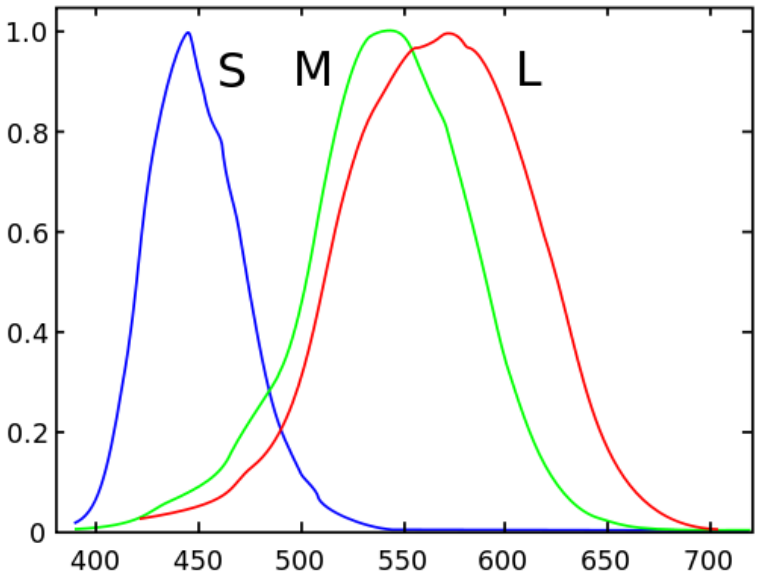


Рис. 2.7. Спектральные кривые в модели LMS

2.3.3. Цветовое пространство CIE XYZ 1931. Диаграмма цветности.

В начале главы мы упоминали, что сетчатка глаза имеет три вида колбочек, отвечающих за цветное зрение. Каждый тип рецептора с разной силой реагирует на видимый свет определённых длин волн. Цветовая модель LMS основана на откликах колбочек человеческого глаза (рис. 2.7). Исторически сложилось, что для измерения цвета используется другое цветовое пространство — XYZ. В 1931 году эта модель была разработана организацией CIE. С тех пор она используется в качестве эталонной модели, с которой соотносятся все прочие цветовые модели.

Как было рассказано в ранее, сначала в результате экспериментов были получены кривые CIE RGB. Члены комиссии решили придумать другое цветовое пространство, связанное с RGB линейным преобразованием. Оно должно было удовлетворять следующим свойствам:

1) Спектральные кривые нового цветового пространства должны быть неотрицательны. В то время это было полезно для упрощения вычислений.

2) Кривая $\bar{y}(\lambda)$ должна была в точности быть равной кривой спектральной чувствительности стандартного наблюдателя $V(\lambda)$. Эта функция описывает изменение воспринимаемой яркости в зависимости от длины волны. Факт, то можно было получить её из линейной комбинации кривых RGB не был гарантирован, но ожидался.

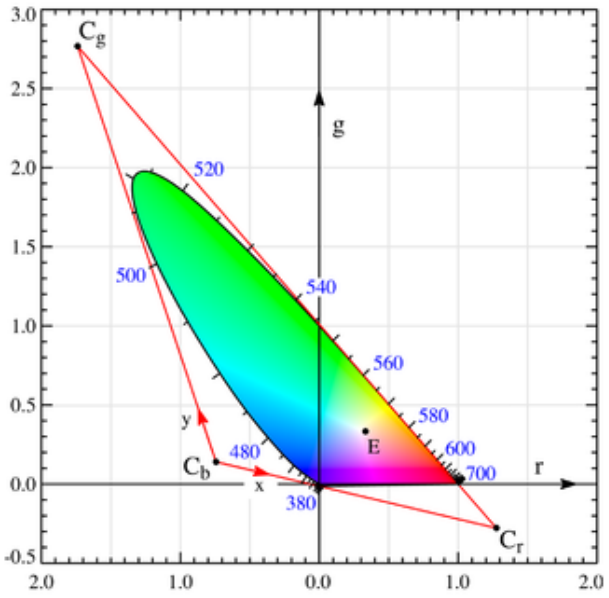


Рис. 2.8. Видимый диапазон цветов в пространстве rgb. $r = R/(R+G+B), g = G/(R+G+B)$.

3) Точка белого (точка на диаграмме цветности, соответствующая белому цвету в данной модели) помещается в точке равной энергии $x = y = z = 1/3$.

4) Видимый диапазон цветов в пространстве rg (см. рисунок 2.8) должен оказаться внутри треугольника [1,0], [0,0], [0,1]. Требовалось, чтобы он как можно плотнее заполнил этот треугольник.

5) Было обнаружено, что можно положить функцию $\bar{z}(\lambda)$ равной нулю ниже 650 нм без превышения ошибки наблюдения.

В терминах геометрии выбор нового пространства эквивалентен выбору нового треугольника в хроматическом пространстве. Это должен быть описывающий область видимых цветов треугольник, исходя из требования 1. Линия C_rC_b фиксируется по требованию 2. По требованию 5 линия C_gC_r должна быть касательной к границе цветового охвата. Точка C_r определена. Требование 4 накладывает ограничения на линию C_bC_g. Таким образом получилось линейное преобразование:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{1}{g_1} \begin{pmatrix} r_1 & r_2 & r_3 \\ g_1 & g_2 & g_3 \\ b_1 & b_2 & b_3 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} = \frac{1}{0.17697} \begin{pmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.00 & 0.01 & 0.99 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

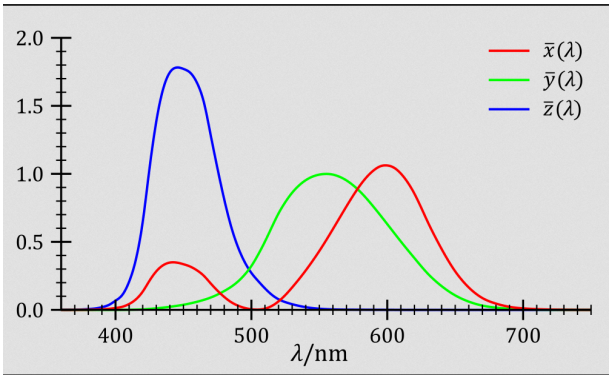


Рис. 2.9. Кривые перехода из пространства RGB в пространство XYZ.

С помощью спектральных кривых цвет XYZ задаётся следующим образом:

$$X = \int_{380}^{780} I(\lambda)\bar{x}(\lambda)d\lambda$$

$$Y = \int_{380}^{780} I(\lambda)\bar{y}(\lambda)d\lambda$$

$$Z = \int_{380}^{780} I(\lambda)\bar{z}(\lambda)d\lambda$$

Компонента Y соответствует визуальной светлоте сигнала, координата Z почти соответствует отклику S-колбочек (чувствительных к синему цвету). Координата X определена таким образом, чтобы всегда быть неотрицательной. Кривые нормируются для соответствия площади. Благодаря такому введению координат, координаты в цветовом пространстве XYZ всегда неотрицательны. Однако, не для каждого цвета XYZ существует физический цвет.

Наглядную интерпретацию ограниченности физических цветов в пространстве XYZ дает диаграмма цветности. Диаграмма цветности (см. рис. 2.10) задаётся в специальных неотрицательных хроматических координатах Yxy и представляет собой срез в пространстве XYZ по поверхности уровня $X + Y + Z = const$ Преобразование из XYZ в xyY осуществляется по следующим формулам:

$$x = \frac{X}{X+Y+Z}$$

$$y = \frac{Y}{X+Y+Z}$$

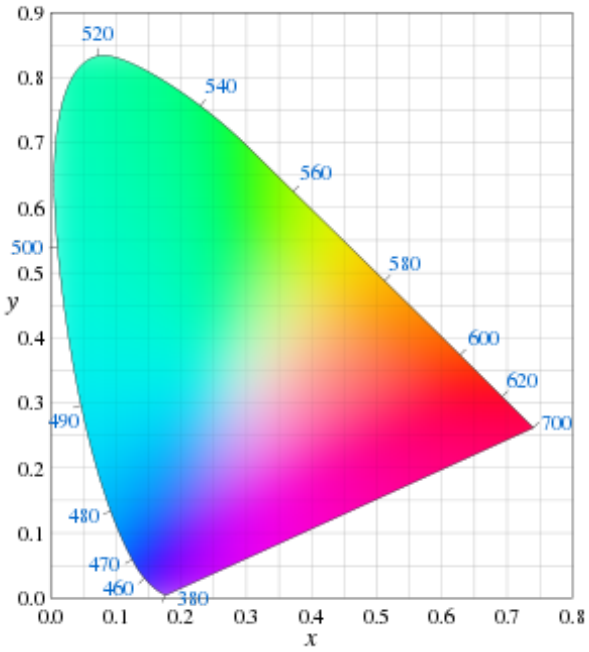


Рис. 2.10. Диаграмма цветности.

На рисунке 2.10 представлена некоторая фигура, заключающая в себе все физически реализуемые цвета. Эти цвета ограничены незамкнутым контуром из монохроматических спектральных цветов. Все цвета за пределами контура существуют только в пространстве xyY, но физически не соответствуют никакому реальному спектру. Эти цвета могут использоваться при расчетах.

Другие трихроматические модели отображаются на диаграмме цветности в виде треугольника. Модель способна представлять только те цвета, которые содержатся внутри данной фигуры. Множество таких точек на диаграмме цветности называется цветовым охватом (color gamut) цветовой модели. Иногда в качестве базовых цветов для трихроматического пространства могут выбираться такие физически не реализуемые цвета. В этом случае построенное пространство может заключать в себе всё множество возможных цветов.

Поскольку модель XYZ исторически устоялась в качестве эталона, все другие модели сводятся к ней путем соответствующих математических преобразований.

2.3.4. Спецификация цветовых пространств. Точка белого.

Вернемся к понятию цветового охвата и базовых (первичных) цветов. Как физические устройства, так и цветовые модели имеют разный цветовой охват. Этот охват в общем случае - некоторая область на диаграмме цветности. В случае не трихроматических моделей (например, для устройств печати) это может другой многогранник, в общем случае — произвольная область. Наибольшее распространение получили именно трихроматические цветовые пространства, наиболее известным из которых является модель RGB.

Дело в том, что RGB — не отдельная конкретная модель, а целый класс цветовых моделей, использующихся для получения, хранения и обработки изображений. Как и другие цветовые модели, конкретные реализации модели RGB связаны с XYZ преобразованиями, поэтому возможна конвертация из одной реализации модели в другую. Характеристики RGB зависят от устройства. Например, у различных мониторов могут отличаться типы базовых излучающих элементов, у фотокамер могут различаться матрицы. В общем случае RGB определяется следующим набором параметров:

- базовыми цветами,
- точкой белого,
- показателем гамма-коррекции.

Базовые цвета соответствуют описанию базовых источников (фосфоров). Как было сказано, они ограничивают цветовой охват устройства. Напомним, что точка белого - это точка на диаграмме цветности, соответствующая белому цвету в данной модели. С понятием точки белого связано понятие цветовой температуры.

В реальном мире мы воспринимаем свет, отраженный от поверхностей окружающих объектов. При этом спектр отраженного света во многом определяется спектром источника в сцене.

Зачастую фотографии, снятые в помещении с лампой накаливания, впоследствии оказываются желтоватыми. В то же время человек этого изменения тона не заметить. Это происходит потому что человеческое зрение адаптируется к цветовой температуре источника. Для того чтобы фотография получилась качественной, фотограф корректирует значение цветовой температуры, соответствующее температуре источников. В современных цифровых фотоаппаратах это зачастую выполняется автоматически или полуавтоматически.

По определению цветовая температура — температура абсолютно чёрного тела, при которой оно испускает излучение того же цветового тона, что и

рассматриваемое излучение. Характеризует относительный вклад излучения данного цвета в излучение источника, видимый цвет источника.

Ниже приведены значения цветовых температур для наиболее распространенных источников освещения:

- 1500—2000 K — свет пламени свечи;
- 2680 K — лампа накаливания 60 Вт;
- 3000 K — лампа накаливания 200 Вт, галогенная лампа, люминесцентная лампа тёплого белого света;
- 3400 K — солнце у горизонта;
- 4000 K — люминесцентная лампа холодного белого света;
- 4300—4500 K — утреннее солнце и солнце в обеденное время;
- 5000 K — солнце в полдень;
- 5500—5600 K — фотовспышка;
- 5600—7000 K — люминесцентная лампа дневного света;
- 6500 K — стандартный источник дневного белого света, близкий к полуденному солнечному свету;
- 6500—7500 K — облачность;
- 7500 K — дневной свет, с большой долей рассеянного от чистого голубого неба;
- 7500—8500 K — сумерки;
- 15 000 K — ясное голубое небо в зимнюю пору;

На диаграмме цветности цветовая температура может быть отображена шкалой, начинающейся среди красных цветов и стремящейся к синим (рис. 2.11).

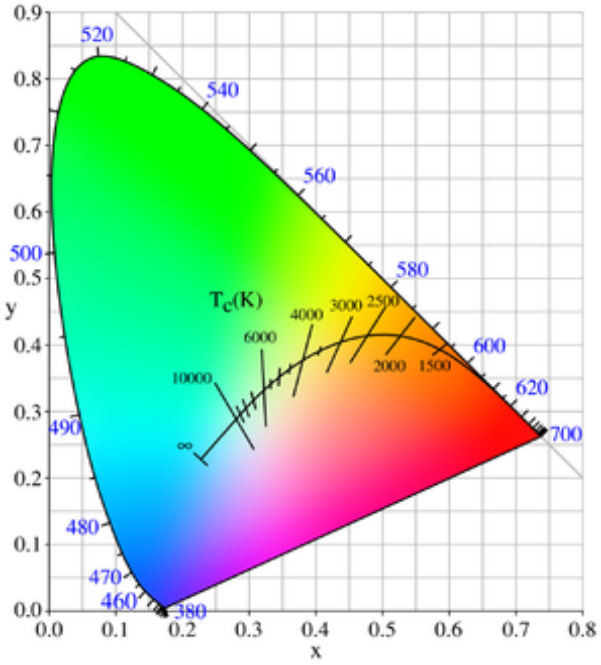


Рис. 2.11. Шкала цветовой температуры на диаграмме цветности.

Существует некоторый набор стандартизированных точек белого - описаний модельных источников освещения, например D55, D65. Поэтому при описании цветовых моделей часто просто называют такую точку белого.

Название	CIE 1931 x	CIE 1931 y	Цветовая температура
E (точка равной энергии)	1/3	1/3	5400
D55	0.33242	0.34743	5500
D65(sRGB)	0.31271	0.32902	6500
D75	0.29902	0.31485	7500
A(лампа накаливания)	0.44757	0.40745	2856

2.3.5. Цветовые профили RGB

Рассмотрим наиболее распространенные цветовые пространства RGB: sRGB, Adobe RGB, ProPhoto.

sRGB

sRGB - формат «по умолчанию», создавался для унификации использования модели RGB в мониторах, принтерах и сайтах в Интернете. sRGB объединил

в себе стандартные основные цвета и гамма-коррекцию, благодаря чему его цвета отображаются на обычных CRT-мониторах и телевизорах. sRGB четко специфицирует показатель гаммы, однако выражает его не в виде фиксированного значения параметра, а в виде двух кусочно-заданных степенных функций. Приближенно можно считать, что степень гамма-коррекции соответствует 2.2. Цветовой охват sRGB небольшой (около 35% видимых цветов). Для перевода линейных значений из пространства XYZ в sRGB используется следующая матрица:

$$M = \begin{pmatrix} 3.2406 & -1.5372 & -0.4986 \\ -0.9689 & 1.8758 & 0.0415 \\ 0.0557 & -0.2040 & 1.0570 \end{pmatrix}$$

Координаты точки белого, таким образом, составляют $\{X, Y, Z\} = \{0.9505, 1.0000, 1.0890\}$. Далее, для каждого канала C производится гамма-коррекция (будет рассмотрена позже):

$$C_{srgb} = 12.92C, C < 0.0031308$$

$$C_{srgb} = 1.055C^{\frac{1}{\gamma}} - 0.055, C > 0.0031308$$

ProPhoto RGB

Цветовое пространство ориентировано на печатный материал, имеет особо широкий цветовой охват. Содержит в себе 90% видимых цветов, в то же время 13% - это мнимые невидимые цвета. За счет этого теряется точность цветопередачи, а потому лучше использовать 16 бит на канал для кодирования (расширенный динамический диапазон, кодировка ERIMM RGB). Пространство редко применяется на практике из-за избыточности. Обычно вместо него используют другое цветовое пространство с расширенным цветовым охватом - Adobe RGB.

Adobe RGB

Цветовое пространство было введено в 1998. Основной целью было обеспечить доступность печатаемых цветов палитры CMYK используя базовые цвета, отображаемые на дисплее монитора. Цветовой охват модели намного шире, чем цветовой охват sRGB, но не такой широкий, как у ProPhoto RGB. Цветовое пространство покрывает примерно 50% видимых цветов. Ниже в таблице приведены параметры модели.

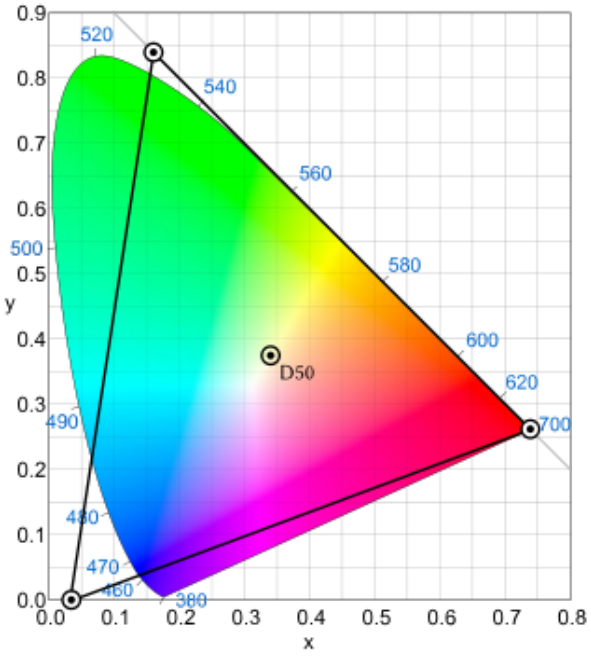


Рис. 2.12. Цветовой охват ProPhoto RGB.

Параметр модели	Значение
Освещённость точки белого	$160.00 \frac{\text{кд}}{\text{м}^2}$
Освещённость точки черного	$0.5557 \frac{\text{кд}}{\text{м}^2}$ (0.34731 % точки белого)
Соотношение контраста	287.9
Уровень рассеянного света	32 lx
Уровень освещенности дисплея	$32.00 \frac{\text{кд}}{\text{м}^2}$ (20% точки белого)
Окружающая освещённость	$2 \frac{\text{кд}}{\text{м}^2}$

Показатель гамма-коррекции 2.19921875. Матрица преобразования координат XYZ в Adobe RGB:

$$M = \begin{pmatrix} 1.96253 & -0.61068 & -0.34137 \\ -0.97876 & 1.91615 & 0.03342 \\ 0.02869 & -0.14067 & 1.34926 \end{pmatrix}$$

sRGB или Adobe RGB

sRGB задано по-умолчанию для системы Windows. На экране стандартного монитора будет это пространство, поэтому готовя фото для выкладывания в интернет, лучше использовать его. То есть повышается совместимость. Также увеличивается точность отображения плавных цветовых переходов. Однако

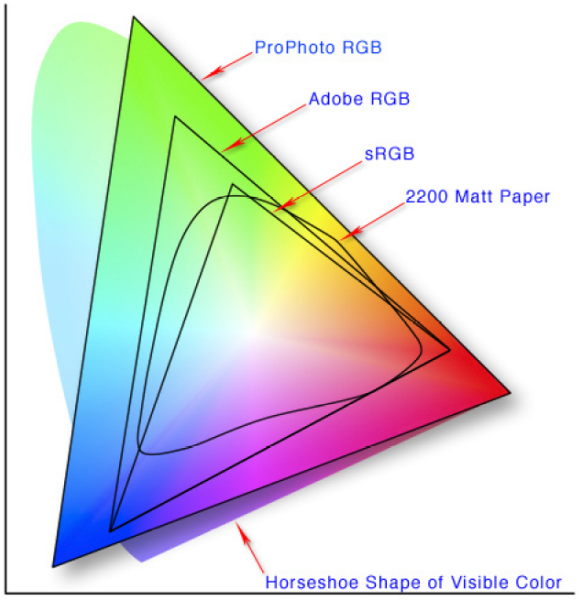


Рис. 2.13. Цветовой охват моделей RGB и печатных цветов.

цветовой охват меньше, поэтому часть цветов теряется. Особенно это плохо при печати на устройствах CMYK, ведь часть потенциальных цветов CMYK не умещается в sRGB, предназначенного для отображения информации на мониторах.

Изначально цветовое пространство Adobe RGB разрабатывалось для стандарта телевидения высокой чёткости — HDTV (High Definition Television) и печати. Оно включает в себя практически весь диапазон цветов CMYK, а также диапазоны цветов для таких устройств, как устройства записи на пленку. Поэтому при печати выдается тот цвет, который и был запечатлен, если аппаратура поддерживает профиль. В то же время в браузере картинки в этом формате смотрятся бледными. Другой недостаток - понижение точности представления данных, так как бит всего 8 на канал, а диапазон шире. Для компенсации можно использовать 16 бит на канал.

На рисунке 2.13 представлено сравнение цветового охвата трёх рассмотренных цветовых пространств, а также цветовой охват чернил на бумаге.

2.3.6. Отображение передаваемого диапазона (gamut mapping)

Из-за разного цветового охвата (gamut) не все цвета одного пространства могут быть однозначно сопоставлены цветам другого пространства. В предыдущем подпункте мы рассмотрели несколько примеров цветовых пространств

(называемых также цветовыми профилями). Мы отметили, что цветовой охват профиля Adobe RGB несколько шире, чем охват sRGB. Не все цвета Adobe RGB могут быть показаны на мониторе. Однако для показа этих цветов нужно каким-либо образом преобразовать изображение. В противном случае данные будут трактоваться как имеющие профиль sRGB и отображаться некорректно, фотографии будут казаться более блеклыми.

Чтобы этого не случилось, нужно преобразовать исходное изображение таким образом, чтобы все его цвета попадали в передаваемый диапазон устройства. Этот процесс называется отображением передаваемого диапазона (gamut mapping).

Другим примером, когда требуется преобразование цветового диапазона, является передача изображения sRGB на печать, т.е. преобразование в CMYK. Например тёмный насыщенный сиренево-голубой цвет обычного компьютерного монитора как правило невозможно распечатать на бумаге с обычным CMYK принтером. Ближайшее приближение по доступному принтеру спектру будет гораздо менее насыщенным.

Предположим, что диапазон значений цвета ограничен интервалом [0,1]. При непосредственном переводе цвета из Adobe RGB в sRGB (через пространство XYZ) могут быть получены 2 некорректных типа значений: отрицательные, либо большее 1. В первом случае невозможна коррекция цветности, во втором - коррекция интенсивности.

Преобразование диапазона цветов может быть локальным или глобальным. Под локальным имеется в виду независимое преобразование значений цвета в каждом пикселе. Например, приведение значений из отрезка [-a,b] к отрезку [0,1]. При этом коэффициенты a и b берутся равными потенциальным экстремальным значениями для данной пары цветовых пространств. Плюсы таких подходов в простоте, скорости и универсальности по отношению к данным. Минусы - в недостаточной реалистичности получаемых изображений.

Глобальное преобразование использует специфическую информацию для каждого изображения. Например, можно определить экстремальные значения каждой компоненты цвета среди имеющихся, а не среди всех потенциальных.

2.3.7. Однородные цветовые пространства

Пространство XYZ - эталонное пространство с неотрицательными значениями цвета, пространство RGB наиболее широко используется в аппаратуре. Однако у них есть некоторые недостатки, один из которых - неоднородность цвета к восприятию. При изменении цветов C1 и C2 на единую величину δ видимое изменение цветов может быть разным. Иллюстрация неоднородности цветового пространства xyY приведена на рисунке 2.14. Показаны 2 пары

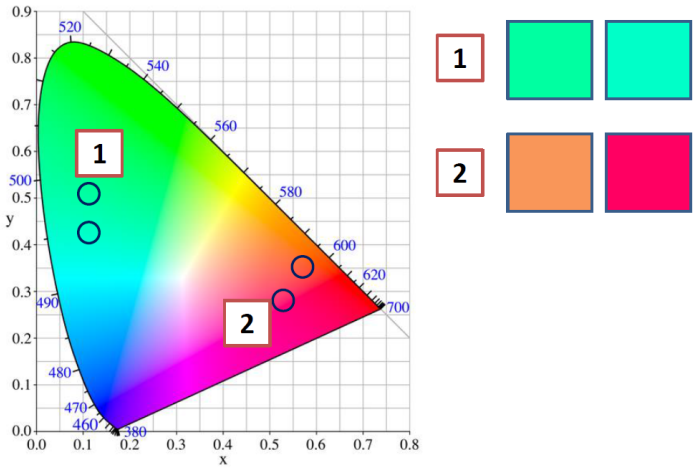


Рис. 2.14. Пары равноудаленных цветов на диаграмме цветности.

цветов, в каждой цвета (точки на диаграмме цветности) отстоят на одинаковом расстоянии. Однако субъективно цвета в первой паре отличаются в значительно меньшей степени, чем цвета во второй.

Возникла цель создать цветовое пространство, изменение цвета в котором было бы линейным с точки зрения человеческого восприятия. То есть одинаковому изменению значений координат цвета должно соответствовать одинаковое ощущение изменения цвета.

Для устранения нелинейности CIE XYZ 1931 был предпринят ряд экспериментов. В 1958 году Ричардом Хантером была предложена модель Hunter «L,a,b» а в 1976 году после устранения разногласий была разработана модель «CIE L*a*b*», которая является сейчас международным стандартом. Стандартным источником является D50.

Оба предложенных стандарта не являются идеальными, так как по-разному ведут себя в различных частях спектра. В Hunter Lab наблюдается сжатие в жёлтой части и расширение в синей. В CIELAB, наоборот, расширение в жёлтой части.

Преобразование из XYZ нелинейное и выглядит следующим образом:

$$L^* = 116 f(Y/Y_n) - 16 \; a^* = 500[f(X/X_n) - f(Y/Y_n)]$$
$$b^* = 200[f(Y/Y_n) - f(Z/Z_n)]$$

$$f(t) = \begin{cases} t(\frac{1}{3}), t > (\frac{6}{29})^3 \\ \frac{1}{3}(\frac{29}{6})^2 t + \frac{4}{29}, t <= (\frac{6}{29})^3 \end{cases}$$

Рис. 2.15. Пары равноудаленных в пространстве Lab цветов.

В цветовом пространстве Lab L означает светлоту (от 0 до 100), a и b - хроматические компоненты (координаты от зелёного до красного и от синего до жёлтого, соответственно). Помимо того, что цветовое пространство обеспечивает однородность изменения цвета, оно также отвечает интуитивным представлениям о природе цвета.

Lab применяется как промежуточное цветовое пространство, не зависящее от устройства, через которое происходит конвертирование данных между другими цветовыми пространствами (например, из RGB сканера в CMYK печатного процесса). Это пространство в силу линейности к восприятию удобно использовать для цветокоррекции. Также в этом пространстве легко можно воздействовать на яркость и контраст изображения.

Основной недостаток модели - нелинейность преобразования в XYZ.

На рисунке 2.15 приведены пары цветов с соответствующими значениями расстояния в пространстве Lab. Видно соответствие прироста разницы между тонами человеческому субъективному восприятию. Первая пара цветов очень похожа, вторая уже сильно различается, а третья вообще содержит два противоположных цвета.

2.3.8. Интуитивные цветовые пространства

В цветовом пространстве XYZ Y отвечает за светлоту, Z связана с откликами колбочек, однако компонента X является искусственной математически вве-

денной компонентой. Поэтому пространство XYZ недостаточно интуитивно. Пространство RGB более интуитивно, однако тоже недостаточно, поскольку промежуточные оттенки (голубой, розовый) приходится каждый раз рассчитывать через другие базовые цвета.

Рассмотренные трихроматические пространства RGB, CMYK удобны с технической точки зрения: они хорошо кодируют информацию, с их помощью удобно получать изображения с камеры, выводить их на экран, печатать на бумаге. Однако когда дело касается обработки изображений, особенно художественной, на первый план выходят другие субъективные характеристики цвета, такие как яркость цвета, насыщенность цвета и цветовой тон.

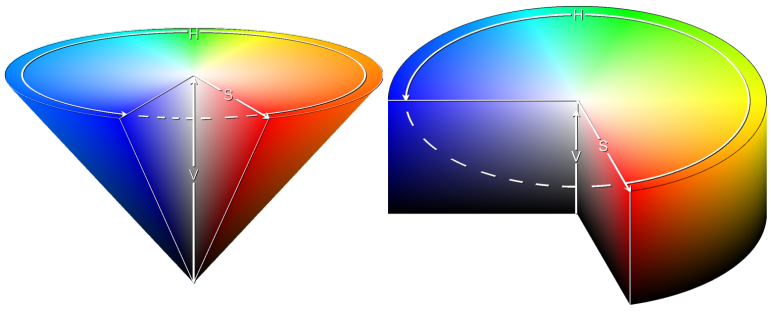
Цветовой тон соответствует доминирующей длине волны. Это вполне интуитивная характеристика, знакомая и понятная даже детям. Насыщенность соответствует яркости, а светлота - белизне цвета.

На рисунке 2.16 приведены примеры пар картинок с разным значением яркости (верхний ряд) и насыщенности (нижний ряд).

Три этих атрибута цвета положены в основу цветового пространства HSV (Hue, Saturation, Value). H обычно принимает значения от 0 до 360 и задан циклически, т.е. субъективные тона в 0 и 360 совпадают. Для наглядного представления палитры цветов используют различные варианты, наиболее распространены цилиндр и конус 2.17.

a) b) c) d)

Рис. 2.16. a) - изображение с низкой яркостью (value), b) - изображение с высокой яркостью. c) - изображение с низкой насыщенностью (hue), d) - изображение с высокой насыщенностью.



a) b)

Рис. 2.17. a) Представление пространства HSV в виде цилиндра. b) Представление пространства HSV в виде конуса.

В этом цветовом пространстве легко получить координаты дополнительных цветов. Дополнительными называются цвета, оптическое смешение которых в равной пропорции приводит к формированию ощущения ахроматического цвета (т.е. серого). В HSV для получения дополнительного цвета достаточно инвертировать первую компоненту Hue относительно середины диапазона.

Есть и другие цветовые модели того же типа, использующие другие субъективные атрибуты, например, HSL (Hue, Saturation, Lightness). Последняя компонента отвечает за светлоту.

2.3.9. Цветовое пространство CIECAM02. Хроматическая адаптация.

Важным свойством зрительной системы является способность адаптироваться к различным освещениям. Благодаря этому обеспечивается цветопостоянство объектов окружающего мира, благодаря этому одно и то же яблоко будет одинаково зелёным и при лампе накаливания, и при дневном свете. В глаз человеку приходит свет источника, отраженный от поверхности. В спектре источника может преобладать тот или иной тон. Мы уже приводили пример того, как неверный выбор баланса белого может повлиять на цвета в отснятом изображении.

Последняя модель, одобренная CIE - это CIECAM02. Модель включает в себя преобразование хроматической адаптации, а также позволяет вычислить 6 атрибутов цвета: яркость, светлоту, цветность (colorfullness), хроматическую насыщенность (chroma), насыщенность(saturation) и оттенок.

Тон имеет аналогичный смысл с тоном в HSV и HSL (однако внутри модели используется промежуточное двухкомпонентное представление цвета), яркость в некоторой степени связана с восприятием отражающей способности объектов, светлота — с белесостью цвета, цветность — мера отличия цветного тона от серого цвета, прочие параметры выражаются через эти базовые.

Всего модель выделяет 4 важные зоны:

1) стимул - рассматриваемая целевая область изображения (её размер согласуется с кривыми стандартного наблюдателя и равен 2 визуальным градусам)

2) промежуточная область (proximal field) - кольцо, захватывающее еще 2 визуальных градуса

3) фон (background) - до 10 градусов. Фон обычно включает в себя всю область монитора

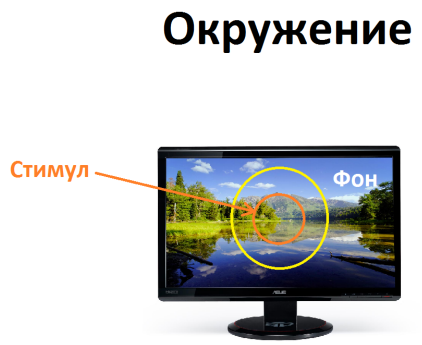


Рис. 2.18. Структура пространства в модели CIECAM02.

4) окружающая область (surround filed) - выходит за пределы монитора, отражает уровень освещенности в комнате, в которой стоит монитор.

Модель берёт в расчёт уровень окружающей освещенности: нормальная (офис, работа за компьютером), полумрак (просмотр телевидения), темно (проектор в тёмной комнате). Для каждого уровня есть предопределённый набор констант, а все промежуточные значения можно получить интерполяцией базовых величин.

Кроме того, требуется определить отношение светимости (luminance) точки белого, измеренной в окружении, к светимости точки белого на дисплее (в фоне). Обычно для этого используют специальное устройство - фотометр.

Основные шаги расчета величин исходя из цвета в XYZ следующие:

1) Конвертация цвета в CAT02 LMS. При этом увеличивается спектральная четкость цвета.

2) Применяется хроматическая адаптация на основе информации об окружении и фоне.

3) Происходит конвертация в пространство LMS, близкое к откликам колбочек.

4) Сжатие откликов колбочек[1].

Эта модель используется для специализированных задач, когда критично обеспечить правильную цветопередачу (например, для демонстрации образцов продукции на мониторах в магазине).

[1]Подробнее про алгоритм можно прочитать в статье Moroney and others “The CIECAM02 Color Appearance Model”

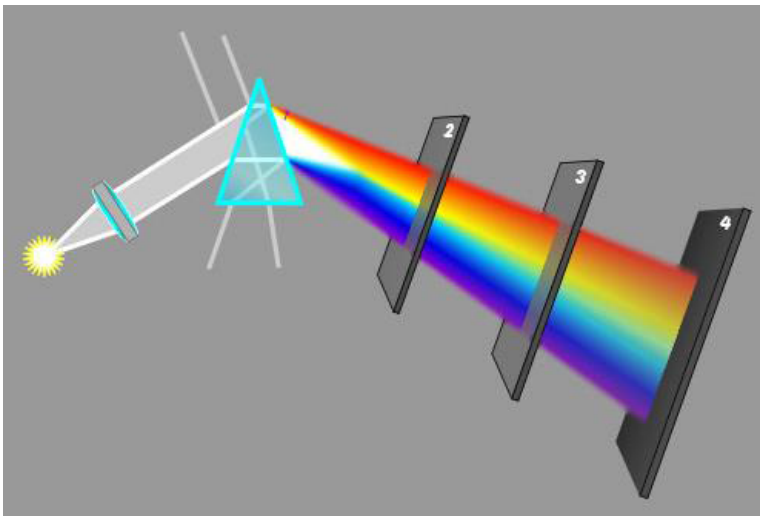


Рис. 2.19. Преломление белого света через призму.

2.3.10. Ограничения трихроматических пространств

Однако даже «продвинутых» цветовых пространств (такие как CIECAM02) недостаточно для решения некоторых задач фотореалистичного синтеза изображений. Их ограниченность следует из самого принципа сведения сложного физического спектра к некоторым базисным значениям. Из трех полученных чисел, очевидно, нельзя однозначно восстановить исходный спектр. Эта потеря информации делает невозможной точное моделирование таких физических явлений как дифракция и интерференция. Оперируя при синтезе изображений с помощью трассировки лучей только тремя компонентами цвета, невозможно смоделировать радугу, полученную в результате преломления через призму (рис. 2.19) с высокой точностью.

В последующих главах будут рассматриваться способы синтеза фотореалистичных изображений. Сейчас отметим только, что зачастую перевод в трихроматическое цветовое пространство лучше осуществлять только на самом последнем этапе, а все промежуточные расчеты выполнять для квантованых спектральных распределений (если имеется спектральное представление материалов и источников в сцене). К сожалению, в большинстве случаев спектрального представления материалов и источников нет, а спектральные рассчёты (например моделирование радуги) произвести нужно. В такой ситуации полезно использовать методы, позволяющие получить *один из возможных* спектров по RGB или XYZ модели [5].

2.4. Гамма-коррекция

Гамма и гамма-коррекция в компьютерной графике – одно из тех понятий, которое на первый взгляд кажется проще, чем есть на самом деле. Начиная разбираться, сталкиваешься с новыми и новыми определениями, новыми и новыми подробностями. В этом пункте сделана попытка тщательно разобраться с гамма-коррекцией, избежав неверных «очевидны» выводов.

2.4.1. Вспомогательные определения

Прежде чем мы перейдем к основному изложению, необходимо вспомнить основные термины, которыми мы будем оперировать.

Энергетическая яркость

Энергетической яркостью (radiance) называется поток энергии, излучаемой некоторой дифференциальной площадкой поверхности в направлении заданного телесного угла (т.е. в заданном направлении). Изменяется в ваттах на квадратный метр на стерадиан $\frac{\text{Вт}}{\text{м}^2\text{ст}}$). Мы будем рассматривать интегральную энергию по некоторому интервалу электромагнитного спектра (скалярная величина).

Энергетическая яркость является объективной линейной мерой количества энергии, попавшей на некоторый приемник с заданного направления или ушедшей в заданном направлении с некоторого источника. Т.е. энергетическая яркость источника со спектром $I(\lambda)$ будет равна $I = \int\limits_{\lambda} I(\lambda)d\lambda$.

Световая яркость

Световая яркость (luminance) относится к фотометрическим терминам и определяется как энергетическая яркость, взвешенная функцией спектральной чувствительности человеческого глаза (luminous efficiency curve) $V(\lambda)$ (показана на рисунке 2.20).

В системе СИ приняты специальные единицы для световой яркости - канделы на квадратные метры $\frac{\text{Кд}}{\text{м}^2}$. Чтобы перейти от $\frac{\text{Вт}}{\text{ст}}$ к канделам, необходимо умножить значение на весовой коэффициент, который получается из определения канделы.

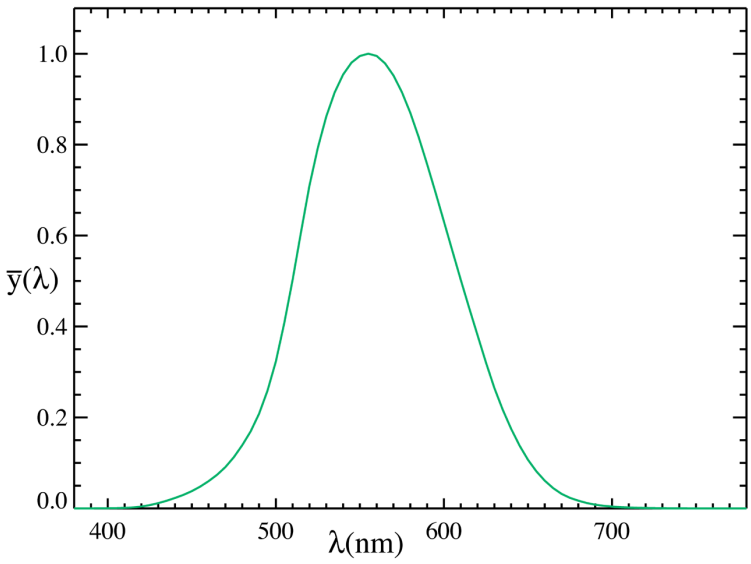


Рис. 2.20. Кривая $V(\lambda)$.

Кандела равна световой яркости монохроматического источника на длине 555 нм, излучающего в данном направлении $\frac{1}{683}\frac{\text{Вт}}{\text{ст}}$.

На длине 555 нм человеческий глаз наиболее чувствителен к световому излучению, поэтому $V(555) = 1$. Т.е. зная кривую $V(\lambda)$ можно выразить световую яркость монохроматического источника на любой длине волны через яркость на длине 555 нм. Таким образом, для любой длины волны, зная спектральную энергетическую яркость на этой длине волны, мы можем получить световую яркость путем взвешивания энергетической яркости функцией $V(\lambda)$, т.е. $I_v(\lambda) = 683V(\lambda)I(\lambda)$ Коэффициент $\frac{1}{683}$ принят для того, чтобы сила света обычной свечи была приблизительно равно одной канделе.

Световая яркость источника с спектром $I(\lambda)$ будет равна $I_v(\lambda) = 683\int\limits_{\lambda} V(\lambda)I(\lambda)d\lambda$.

Такая величина позволяет определять видимую яркость источника.

Расмотрим воображаемый инфракрасный источник с полным потоком излучения $1\frac{\text{Вт}}{\text{ст}}$ в диапазоне от 700 нм до 800 нм и отсутствием излучения в других областях спектра. Его интегральный полный поток $I = \int 1d\lambda = 100\text{Вт}$. В то же время фотометрический полный поток I_v составит около 42 люмен (световая эффективность $\frac{I_v}{I} = 0.42\frac{\text{Лм}}{\text{Вт}}$) . Для сравнения – полный поток лампы накаливания мощностью 100 Вт составляет около 1300 люмен (световая эффективность 13 $\frac{\text{Лм}}{\text{Вт}}$). Т.е. этот источник глазом будет различим как достаточно тусклый.

Восприятие яркости человеком и светлость

Известно, что воспринимаемая человеком яркость источника света существенно нелинейно зависит от фотометрической яркости. Так, если уменьшить яркость источника до 18% по сравнению с базовым, он будет казаться вполовину менее ярким. Эксперименты показывают, что на большей части воспринимаемых яркостей человеческое зрение характеризуется логарифмическим восприятием (т.е. отклик на сигналы, меняющиеся в геометрической прогрессии, будет составлять арифметическую прогрессию). Однако логарифмическая функция обычно заменяется на степенную.

В цветовых системах CIE L*u*v и L*a*b введена соответствующая безразмерная единица «светлости»:

$$L^* = 116 f(Y/Y_n) - 16,$$

где Y_n – светлость некоторого идеально-белого объекта.

$$f(t) = \begin{cases} t^{(\frac{1}{3})}, t > (\frac{6}{29})^3 \\ \frac{1}{3}(\frac{29}{6})^2 t + \frac{4}{29}, t <= (\frac{6}{29})^3 \end{cases}$$

L* лежит в диапазоне от 0 до 100, а единичное изменение немного превосходит порог различимости. В дальнейшем будет подробнее рассмотрена связь законов зрительной системы и гамма-коррекции.

Изображение

Гамма-коррекция рассматривается обычно (и нами тоже) в применении к изображениям. Мы будем понимать термин «изображение» в обычном смысле – как целочисленную матрицу, каждый элемент которой $I \in [0, 255]$ и хранит закодированное значение относительной светимости конкретного пикселя. Т.е. мы полностью исключаем цветовую информацию и сосредотачиваемся на представлении интенсивности.

2.4.2. Линейность кодирования яркости в изображениях

Что кодирует пиксель изображения? Чему соответствует, скажем, значение 0 или значение 125? Если рассмотреть процесс формирования изображения на

чувствительной матрице, скажем, цифрового фотоаппарата, то мы придём к следующему упрощённому представлению.

Чувствительный элемент матрицы представляет собой некоторую чувствительную площадку, закрытую фильтром, пропускающим только определённый участок спектра. Будем считать, что площадка одинакого реагирует на все длины волн Фильтр характеризуется функцией спектральной чувствительности $m(\lambda)$. $m(\lambda) = 1$ означает, что свет на длине волны λ полностью пропущен. Соответственно, $m(\lambda) = 0$ означает, что свет на длине волны λ полностью поглощён фильтром.

Пусть $E(\lambda)$ - спектральная освещённость фильтра ($\frac{\text{Вт}}{\text{м}^2\text{нм}}$). Тогда освещённость E чувствительного элемента составит

$$E = \int_{\lambda} E(\lambda)m(\lambda)d\lambda$$

(взвешенная пропусканием фильтра сумма по всем длинам волн, $\frac{\text{Вт}}{\text{м}^2}$). Предполагая, что отклик чувствительного элемента прямо пропорционален освещенности E и времени воздействия света t , получим энергетическую экспозицию H.

$$H = Et$$

Энергетическая экспозиция измеряется в джоулях на квадратный метр ($\frac{\text{Вт*сек}}{\text{м}^2} = \frac{\text{Дж}}{\text{м}^2}$, т.к. Вт=$\frac{\text{Дж}}{\text{с}}$).

Итак, значение пикселя изображения должно быть линейно пропорционально энергетической экспозиции H, которая, в свою очередь линейно зависит от мощности падающего излучения, взвешенного некоторым фильтром. Фильтры необходимы, чтобы получить цветное изображение (а не суммировать целиком весь спектр).

В случае обычных изображений, целью которых является максимально приближенное к человеческому восприятию картина, суммарный отклик фильтров делается похожим на функцию спектральной чувствительности человеческого глаза, поэтому с некоторым приближением можно перейти к использованию световой экспозиции, которая характеризуется использованием фотометрических единиц освещенности (т.е. освещённость E составит

$$E = \int_{\lambda} E(\lambda)V(\lambda)m(\lambda)d\lambda,$$

где $V(\lambda)$ - функция спектральной чувствительности человеческого глаза). Освещённость в этом случае будет измеряться в люксах (lux), а экспозиция — в люксо-секундах ($\frac{lux}{sec}$).

Для визуализации изображения мы должны обеспечить светимость пикселей на экране пропорционально значению в изображении и, следовательно, пропорционально освещенности сенсора. При этом не будем требовать точного соответствия, т.к. значения светимости пикселей могут зависеть от конкретного монитора. То есть:

$$B_{ij} = a + bI_{ij} = c + dE_{ij}$$

где i,j - координаты конкретного пикселя на изображении, в предположении что они соответствуют пикселям монитора.

Однако на практике выдерживание такой линейности потребует от нас некоторых дополнительных действий, к описанию которых мы и перейдем.

2.4.3. Тракт передачи изображений

Тракт передачи изображений - программно-аппаратный комплекс, предназначенный для получения (захвата или генерации), передачи и визуализации изображений. Пример тракта передачи изображений - система «видеокамера-пленка-видеоплеер-телевизор» или «фотоаппарат-компьютер-монитор». Заканчивается тракт всегда генерацией изображения на устройстве вывода – дисплее. А начинаться может либо с захвата изображения реального мира, либо с генерации изображения тем или иным способом (скажем, алгоритмами компьютерной графики).

Тракт «от света до света» состоит минимум из трех звеньев: приемника (фотоаппарат, сканер, видеокамера), системы хранения/передачи и дисплея. Приемник преобразует свет в код, система хранения код этот хранит и передает до дисплея, который преобразует код в цвет. Тракт для работы с синтезированными изображениями не включает приемника, который заменяется алгоритмом построения изображения.

Задача любого тракта передачи изображений - передать без искажений цвет и яркость исходных (либо синтезированных) изображений (см. Рис. 2.21).

В данной главе мы рассматриваем вопросы передачи яркости, цвет - тема для отдельной работы, этому посвящены многие книги. Гамма-коррекция является составной частью тракта передачи изображений и предназначена для того, чтобы передать изображение без искажения интенсивности света [PoyntonGammaFAQ].

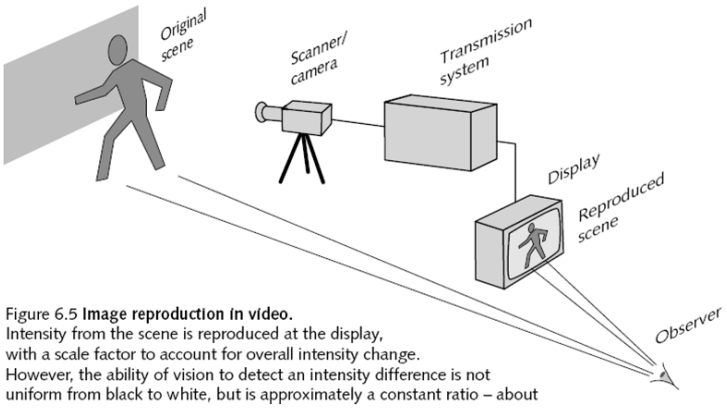


Figure 6.5 **Image reproduction in video.**
Intensity from the scene is reproduced at the display, with a scale factor to account for overall intensity change. However, the ability of vision to detect an intensity difference is not uniform from black to white, but is approximately a constant ratio – about

Рис. 2.21. Тракт передачи изображений.

2.4.4. Передающая функция

С каждым звеном тракта ассоциирована передающая функция.

Пусть функцией приемника будет $C = R(E)$ — преобразование освещенности E в код C.

Функция системы хранения: $C_1 = S(C)$.

Функция дисплея: $I = D(C_1)$.

В идеале

$$D(S(R(E))) = a + bE,$$

т.е. после передачи через все системы и устройства исходное излучение остается неизменным с точностью до линейного преобразования. Это — наша цель, однако её достижению мешают несколько вещей:

- некоторые особенности передающих функцией могут быть обусловлены физическим устройством приёмника и дисплея;
- при хранении информации в цифровом виде неизбежна дискретизация сигнала, что ведёт к потере информации, которую необходимо минимизировать.

Мы будем в основном рассматривать передающую функцию дисплея, т.к. своим появлением гамма-коррекция обязана в основном свойствам этого звена

передающего тракта, а также особенностям человеческого восприятия яркости.

В следующих пунктах мы подробно рассмотрим различные причины появления гамма-коррекции:

- нелинейность передающей функции CRT-дисплеев;
- необходимость нелинейного кодирования яркости для более полного использования ограниченного диапазона представления яркости в ЭВМ;
- особенности восприятия интенсивностей человеческим зрительным аппаратом.

2.4.5. Нелинейность передающей функции дисплея

Исторически необходимость гамма-коррекции была обусловлена особенностями передающей функции CRT-дисплеев.

Рассмотрим передающую функцию дисплея $D(C)$. В данном случае C – это входящий сигнал (напряжение, соответствующие данному пикселю). Интенсивность света, генерируемого физическим устройством обычно не является линейной функцией входящего сигнала. CRT-устройства (телевизоры, мониторы с электронно-лучевой трубкой, также называемой кинескопом) имеют степенную зависимость интенсивности излучения от входящего напряжения:

$$B(C) = A(kC)^{\gamma},$$

где параметр A характеризует максимальную яркость монитора.

Более точно, зависимость имеет вид

$$B(C) = A(k_1 C + k_2)^{\gamma},$$

где k_2 характеризует уровень черного (black-level), т.е. яркость экрана при нулевом сигнале, а k_1 - усиление сигнала (контраст). Большинство различий между мониторами заключается в разной настройке регулировок яркости k_2 и контрастности k_1. Такая зависимость является следствием принципиального устройства электронно-лучевой трубки (см. рис. 2.22).

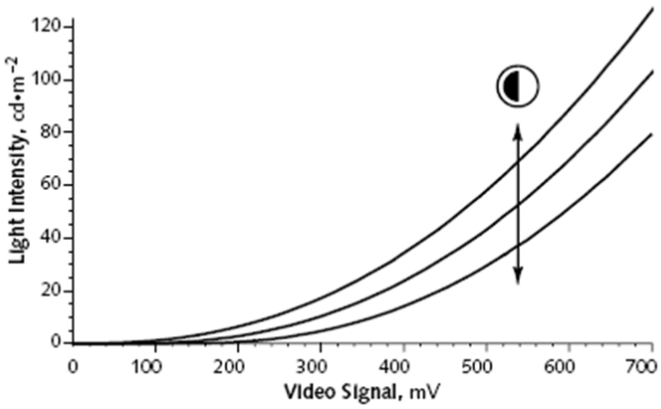


Рис. 2.22. Нелинейность монитора.

Коэффициент γ обычно лежит в пределах 2.3-2.6 (чаще всего около 2.5 [Poynton2003]) и называется гаммой данного устройства (т.е. по названию буквы греческого алфавита, обозначающего коэффициент). Эта нелинейность должна быть скомпенсирована для корректной передачи хранимых яркостей.

Процесс корректировки этой нелинейности и называется гамма-коррекцией. Гамма-коррекция включает в себя преобразование, обратное тому, которое происходит в мониторе (преобразование формы $kC^{\frac{1}{\gamma}}$, см. рис. 2.23).

Для LCD-мониторов передающая функция линейна. Однако они конструируются таким образом, чтобы эмулировать соответствующую кривую для CRT-мониторов. Это делается в основном по двум причинам:

- для того, чтобы можно было безболезненно подключать различные мониторы к одним системам без существенного изменения цвета и яркости изображения;
- нелинейность способствует повышению контрастности изображения и лучшей передаче тёмных оттенков, о чем будет рассказано в следующих пунктах.

Многие программы применяют неполную гамма-коррекцию (т.е. изображение корректируется так, что визуализируемое изображение умышленно остается нелинейным). Часто используется значение гаммы 2.2. Неполная гамма-коррекция увеличивает контраст изображения, что, как считается, предпочитают пользователи. Кроме этого, «недокоррекция» полезна для компенсации эффекта бликования на экране монитора, который ухудшает контраст.

Необходимо заметить, что такая нелинейность мониторов не является по сути дефектом, который необходимо исправлять (на современном уровне развития

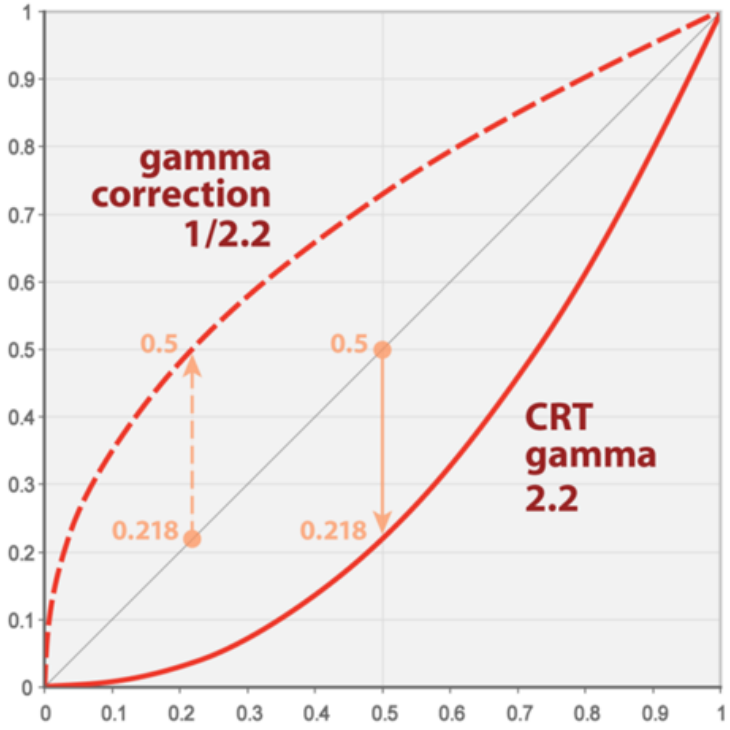


Рис. 2.23. Гамма преобразование и гамма коррекция.

науки и техники закодировать коррекцию нелинейности в аппаратное обеспечение монитора не составило бы больших проблем). Нелинейность CRT практически является точной обратной функцией чувствительности человеческого зрения к яркости и поэтому отклик монитора на входящие сигналы получается воспринимаемо-линейным, что на самом деле очень хорошо.

Кроме этого, сегмент около нуля обычно заменяется на линейное приближение степенной функции. Это делается для минимизации влияния шума в тёмных областях изображения.

2.4.6. Кодирование яркости

Другой причиной для появления гаммы и гамма-коррекции является дискредитированное представление информации в ЭВМ. Традиционное представление цвета в компьютерной графике ограничивает яркость 8-ми битным представлением. Таким образом на весь диапазон яркостей остается только 256 значений. Необходимо использовать их таким образом, чтобы процесс кодирования максимально отвечал свойствам человеческого зрения. Особенность восприятия приводит к тому, что человек лучше различает в тёмных областях.

При линейном представлении 8-битных изображений ошибка (относительный шаг между соседними значениями) меняется от 100% (тёмные пиксели) до менее 1% (при значениях выше 100). Таким образом, соотношение между самым ярким белым и самым тёмным серым, которое может быть закодировано без видимых градаций, составляет 2,55:1 (т.е. min=100, max=255).

Чтобы избежать градаций, необходимо поддерживать шаг 1,01 на всем диапазоне яркостей. При использовании линейного кодирования, абсолютная дельта 0,01 должна поддерживаться на всем диапазоне. Чтобы добиться контраста 100:1, понадобится использовать столько бит, чтобы максимальное значение цвета достигло 10000 (т.е. min=100 max=10000). Наименьшее количество бит на цвет, при котором достигается такое значение - 14 бит на канал.

Хранение 8-битных изображений в нелинейном виде (с гамма-коррекцией) позволит выделить больше бит под тёмные пиксели, что улучшит восприятие при визуализации на мониторе. Монитор выполнит «аналоговое» сжатие диапазона яркостей в тёмной области, что повысит «плотность» градаций, оставляя разницу в пределах 1-2% в широком диапазоне яркостей (см. рис. 2.24).

Гамма-коррекция при кодировании уменьшает ошибку, однако нельзя считать ее универсальным способом кодирования изображений. Кроме того, одного байта на канал яркости часто недостаточно для представления всего

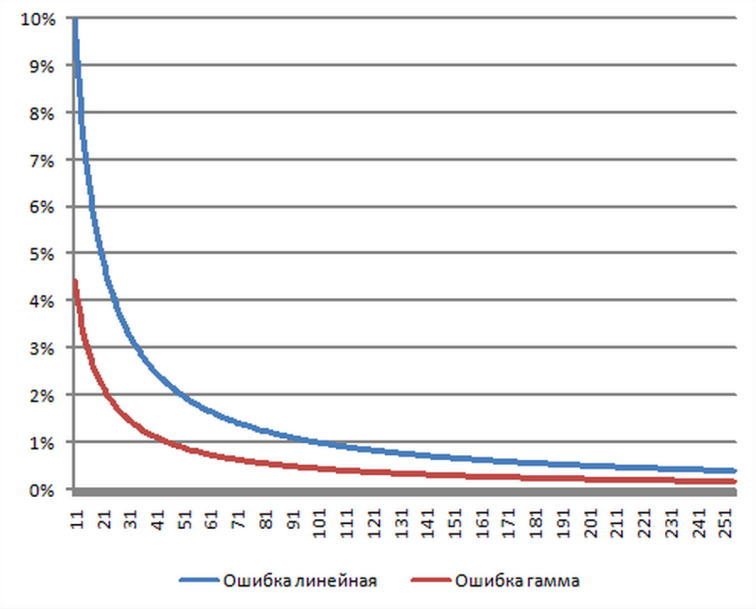


Рис. 2.24. Ошибка линейного кодирования и ошибка гамма-кодирования.

диапазона яркостей, присутствующих в реальном мире. Существуют другие схемы кодирования (например, логарифмические), которые часто применяются для хранений изображений широкого динамического диапазона (см. [Reinhard2005]).

В прошлом пункте мы столкнулись с нелинейностью восприятия цвета человеком относительно цветового пространства XYZ. Мы не заостряли внимание на том, что на самом деле восприятие человека нелинейно не только относительно производных величин, выбранных специально для представления цвета, но и относительно исходных физических величин. Однако это действительно так.

Прежде всего нас интересует восприятие человеком яркости, однако вы, возможно, уже сталкивались с понятием децибелов для звука. Децибел - логарифмическая единица. Величина, выраженная в децибелах, равна численно равна десяти десятичным логарифмам отношения физической величины к одноимённой физической величине, принимаемой за исходную:

$$A_{dB} = 10lg\frac{A}{A_0}$$

где A_{dB} — величина в децибелах, A — измеренная физическая величина, A_0 — величина, принятая за базис.

Одна из причин использования децибелов состоит в том, что восприятие стимулов различного вида носит близкий к логарифмическому характер. Наглядный пример - темперированная шкала частот (музыкальная гамма). Нота “Ля” разных октав фортепиано имеет следующие значения частот (Hz):

27,5 55 110 220 440 880 1760 3520 7040

В 19 веке Вебером, а затем Фехнером был предложен эмпирический закон для описания интенсивности ощущения. Согласно нему, интенсивность ощущения прямо пропорциональна локагифму интенсивности раздражителя:

$$p = kln\frac{S}{S_0}$$

где S — значение интенсивности раздражителя. S_0 — нижнее граничное значение интенсивности раздражителя: если $S < S_0$, раздражитель совсем не ощущается. k - константа, зависящая от субъекта ощущения. Можно заметить, что приведенная формула сходна с выражением физической величины в децибелах.

Позже (в 20 веке) Стивенс показал, что закон Вебера-Фехнера верен лишь для средних значений ощущения некоторых модальностей, а в общем случае зависимость степенная (показатель степени свой для различных условий). Согласно закону Стивенса:

$$Y = kI^a$$

То есть субъективная яркость определяется степенной функцией от уровня физической интенсивности.

Таким образом, степенной закон гамма-коррекции оправдан с точки зрения человеческого восприятия. Рассмотрим подробнее, как гамма-коррекция помогает при кодировании изображения.

Порог различения (Just-noticable difference) - минимальное значение, при котором различаются стимулы. Из приведенных выше законов следует, что порог различения яркости примерно пропорционален её величине. Это означает, что не беря во внимание нелинейность восприятия, мы рискуем неверно закодировать изображение, из-за чего градации яркости получатся слишком разреженными, и пользователь начнет различать ступенчатость.

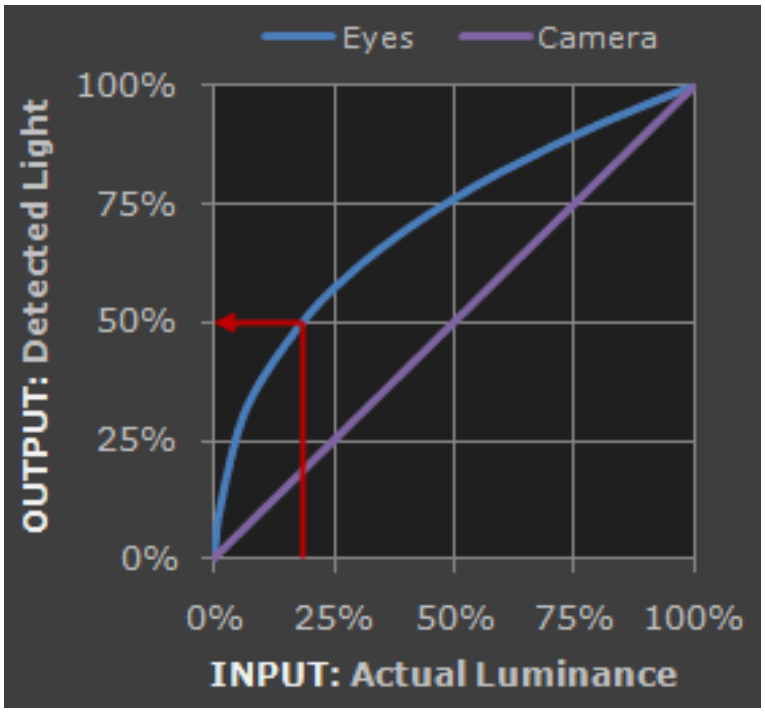


Рис. 2.25. Нелинейность восприятия яркости.

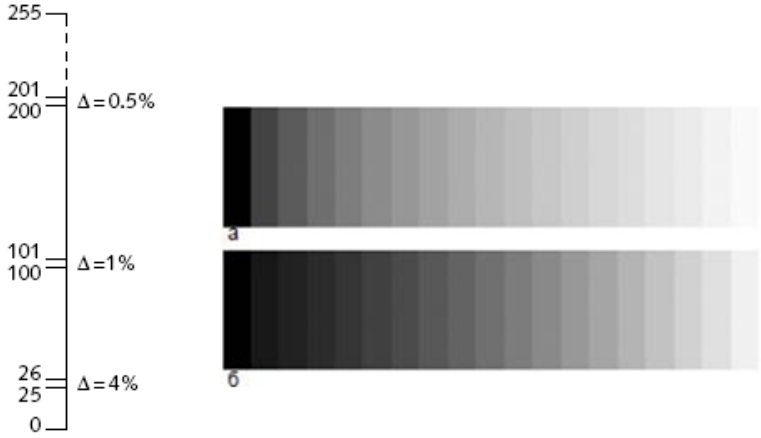


Рис. 2.26. Вверху - физически равномерные яркости, внизу - субъективно равномерные.

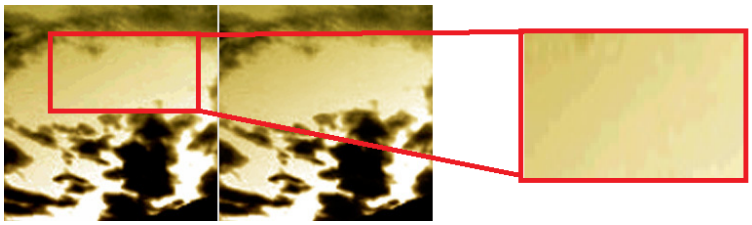


Рис. 2.27. Пример артефактов линейного кодирования.

Считается, что для визуальных стимулов порог различения составляет около 2%. При линейном кодировании с последующей гамма-коррекцией могут возникать артефакты, потому что субъективная мера различия между двумя последовательными целыми значениями яркости (от 0 до 255) будет неравномерной.

При степенном кодировании ошибка уменьшается, поскольку увеличивается равномерность субъективной яркости для последующих градаций. На рисунке 2.26 пример двух шкал яркости без гамма-коррекции и с ней.

2.4.7. Использование гаммы для моделирования процессов адаптации

Одним из возможных проявлений свойства зрения адаптироваться к окружающему освещению можно считать иллюзию одновременного контраста (рис. 2.28). Иллюзия заключается в том, что квадраты одинаковой закодированной яркости могут выглядеть по-разному в зависимости от цвета фона. При ярком свете серое выглядит темнее, чем при тусклом.

Показатель Стивенсона различается для разных уровней освещенности, а значит можно искусственно изменить входные данные по степенному закону для соответствия конкретным условиям просмотра. Похожий, но более сложный механизм используется в модели CIECAM02, упомянутой в предыдущем пункте.

Гамма-коррекция используется в стандартной модели sRGB, а значит широта её применения и простота значительно превосходят хроматическую адаптацию CIECAM02. Подкорректировав параметр коррекции таким образом, чтобы после гамма-коррекции и гамма-преобразования суммарное значение степени превосходило единицу, можно адаптировать видеоряд для показа в тёмных помещениях. Таким образом, повысится контрастность видео.

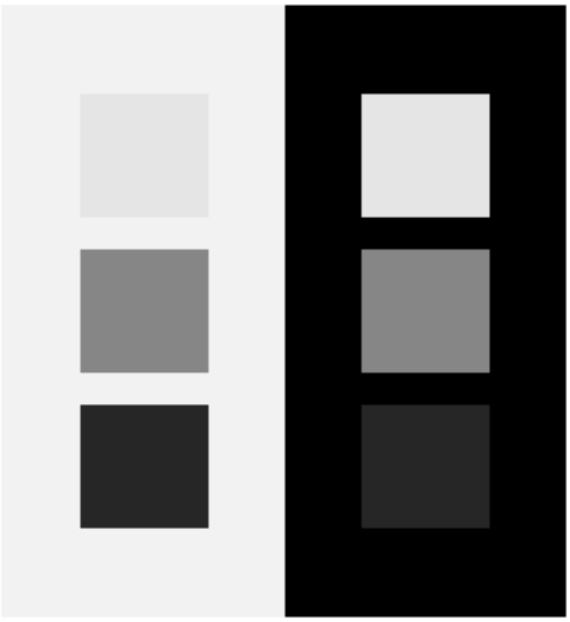


Рис. 2.28. Иллюзия одновременного контраста.

2.4.8. Использование гамма-коррекции в приложениях

Гамма в операционной системе Windows

В большинстве случаев используется показатель гаммы 2.2. Компьютеры Макинтош до 2009 года использовали значение гаммы 1.8. Бинарные данные в файлах изображений и видео закодированы с использованием гамма-кодирования. Гамма для конкретного изображения может быть указана в профиле. Для корректного отображения данных на мониторе, они должны быть приведены в степень, обратную степени гамма-преобразования монитора.

Система управления цветом операционной системы обеспечивает максимально возможную точность передачи цветового содержимого на устройства вывода (выполняют преобразование цветового диапазона - gamut mapping). В ОС Windows хранятся цветовые профили устройств. Цветовой профиль - это файл с цветовыми характеристиками. В профиле хранится информация о том, как именно интерпретирует данные определенное устройство в определенном состоянии. Одно устройство может иметь несколько цветовых профилей. Цветовые профили взаимодействуют с системой управления цветом, обеспечивая приемлемую передачу цветового содержимого независимо

Рис. 2.29. Повышение контрастности за счет изменения результирующей гаммы.

от устройства (с помощью профилей создаются преобразования цветов, которые используют приложения).

ОС Windows поддерживает 2 типа цветовых профилей:

- цветовой профиль системы Windows (WCS - Windows Color System)
- цветовой профиль международного консорциума по цвету (ICC - International Color Consortium)

Цветовые профили можно сопоставлять с устройствами, добавлять новый профиль (например, в результате калибровки экрана), назначать профиль по-умолчанию для устройства.

Общая схема работы с изображениями

При синтезе фотореалистичных изображений зачастую используют готовые данные для создания и детализации объектов виртуальной сцены. Такими данными могут быть, например, трехмерные модели объектов, спецификации отражающих свойств материалов и т.д. Очень часто используют файлы изображений, поэтому важно не только уметь подготовить результат синтеза к выводу на экран, но и уметь интерпретировать данные в сохраненных на диск изображениях.

Наиболее типичными примерами использования изображений при синтезе являются текстуры (рис. 2.32). Текстура как правило является частью описания материала объекта. Хотя в общем случае текстуры могут иметь разную трактовку (текстура отражающих свойств материалов, микрорельеф поверхности, карта нормалей), в большинстве случаев её интерпретируют как цветное изображение, накладывающееся на поверхность полигонов в сцене с целью придания объекту определенного цвета.

Другим примером изображений служат панорамы (cube map, sphere map), выступающие в качестве внешних источников энергии относительно сцены. Сами по себе в отрыве от приложения панорамы представляют собой изображения окружающего пространства.

Алгоритмы синтеза предназначены для работы с линейным представлением излучения.

Однако, как правило, обыкновенные изображения уже содержат в себе гамма-коррекцию. Изображения широкого динамического диапазона (будут рассмотрены на следующей лекции) обычно, напротив, не содержат гамма-коррекции.

Рис. 2.30. a) Карта Земли как текстура для шара.

Рис. 2.31. b) Отражение панорамы в элементе сцены.

Рис. 2.32. Примеры использования текстур при синтезе изображений.

Более точно можно это определить с помощью цветового профиля изображения. Если профиль не указан, то можно считать показатель гаммы равным 2.2. Однако в этом случае нет гарантии, что изображение цвета изображения трактуются правильно.

Общий алгоритм использования изображений в процессе синтеза:

1) Применить гамма-преобразование (2.2 или иное из профиля изображения).

2) Рассчитать освещение, синтезировать выходное изображение.

3) Применить гамма-коррекцию для вывода на монитор ($\frac{1}{2.2}$ или иная).

Рассмотрим модельный пример для простейшего алгоритма синтеза: смешать 2 изображения с прозрачностью α.

Рассмотрим, по каким формулам будет рассчитан выходной свет без использования гаммы и с использованием. Обозначим входные цвета c_1 и c_2.

В первом случае выходной цвет будет равен $c = \alpha c_1 + (1 - \alpha)c_2$

Во втором: $c = ((\alpha c_1)^(\gamma) + (1 - \alpha)(c_2)^(\gamma))^(\frac{1}{\gamma})$

Предположим, что:

$$\gamma = 2.2, \alpha = 0.4$$

Средняя ошибка на всевозможных комбинациях входных цветов составит 11.83, а максимальная 66. На рисунке 2.33 вы вполне можете заметить ошибки невооруженным глазом, сравнив два цвета. Также приведена карта ошибок, где цветом закодирован уровень ошибки для данной пары значений цвета (рис. 2.34).

Можно сделать еще одно замечание. Для сохранения точности операций лучше работать с вещественными числами иначе будут сильны ошибки округления.

Альтернативно можно заранее подготовить линеаризованные данные для передачи на вход приложению, однако в этом случае вновь получим ошибки линейного кодирования.

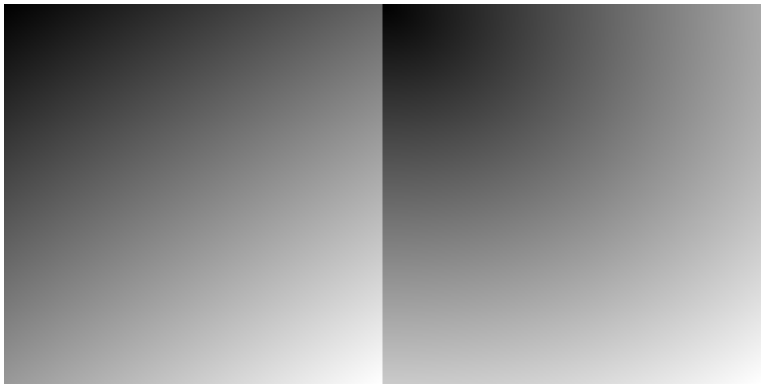


Рис. 2.33. Результирующие цвета для всевозможных значениях входных цветов.

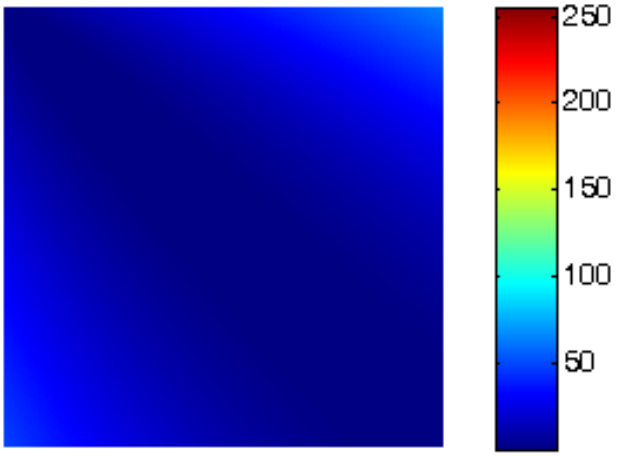


Рис. 2.34. Численная ошибка при комбинировании цветов, если цвета лежат в диапазоне от 0 до 255.

2.4.9. Калибровка монитора

Как определить показатель гаммы у конкретного монитора?

Обычно, у LCD монитора значение гаммы близко к 2.2, но бывают и исключения. Более точно определить значение можно с помощью калибровки монитора. Существует как встроенное средство калибровки Windows, так и множество аналогов других производителей. В Windows калибровка монитора приводит к созданию нового цветового профиля. Монитор можно откалибровать под конкретные условия работы (например, для освещения на конкретном рабочем месте).

Изменение показателя гаммы сопряжено с двумя свойствами отображения: яркостью и контрастом. Как правило, именно эти параметры доступны для регулирования. Для опеределения гаммы достаточно воспользоваться специальной шкалой (рис. 2.35). Пользуются ей таким образом:

1) отодвигаются от монитора до тех пор, пока не перестают различать четкие полосы в левой части

2) находят уровень, для которого совпадают цвета в левой и правой части

3) в найденном уровне написан показатель гаммы вашего монитора.

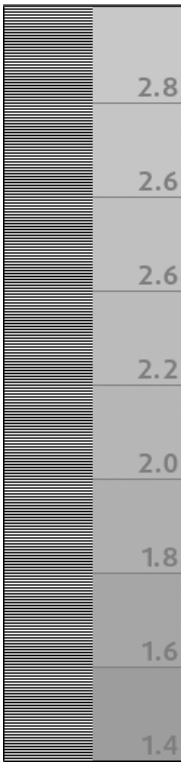


Рис. 2.35. Шкала для определения показателя гаммы монитора.

Глава 3.

Изображения широкого динамического диапазона

3.1. Введение

При создании фотореалистических изображений мы, в конечном итоге, рассчитываем получить неотличимое от реальности изображение. Именно для этой цели и нужны изображения широкого динамического диапазона (high dynamic range images - HDRI), далее HDR-изображение.

Рассмотрим, чем именно ограничен динамический диапазон итогового изображения.

На последнем этапе тракта передачи изображений хранимая информация преобразуется в видимый свет с помощью устройств визуализации. Оказывается, не всех параметров систем визуализации достаточно для точного воспроизведения сохраненной в виде изображения сцены. Хотя с точки зрения разрешающей способности, современые дисплеи удовлетворяют требованиям реализма, цветовые и яркостные характеристики всё ещё не могут соперничать с возможностями восприятия зрительной системы человека. Большинство современных мониторов имеют максимальную яркость 100-200 кд/м2, а контрастность не превышает двух порядков* (*Появляются 10 и 12-битные мониторы, но пока они не широко распространены.). В то же время глаз способен работать с 10-12 порядками яркости (солнечный день и безлунная ночь).

Однако работа с изображениями не ограничивается выводом фотографий на дисплей. Как было сказано ранее, процессу визуализации предшествует хранение и обработка изображений. Устройство фотокамеры позволяет регулировать поток через диафрагму, однако в одном кадре зачастую невозможно

запечатлеть реалистичную сцену. Хотя динамический диапазон пленки выше динамического диапазона монитора, его так же не достаточно для описания всех различимых человеком яркостей.

По этим двум причинам изображения узкого динамического диапазона (8-битные) более широко используются. Однако в процессе синтеза работа с HDR-изображениями де факто стала обязательной. Они могут использоваться как в процессе визуализации, так и в качестве промежуточного результата для хранения физических значений яркости. Кроме того, существуют способы получения HDR-изображений реальных сцен с помощью стандартных технологий съемки. Для воспроизведения таких изображений разрабатывают специальные алгоритмы, называемые операторами тональной компрессии (tone-mapping operators).

Если вы хотите, чтобы результат вашей визуализации был максимально реалистичен, вам так или иначе необходимо уметь получать, хранить и обрабатывать HDR-изображения.

Вначале рассмотрим современные устройства визуализации и выявим их ограничения более детально.

3.1.1. Геометрическое разрешение дисплея

Представим, что мы не ограничены в параметрах яркости и разрешения цифрового изображения. Подсчитаем, какой дискретизации по геометрическим характеристикам и цветовым характеристикам достаточно для реалистичной (не отличимой от реальности) визуализации.

Разрешение зоркого глаза – примерно 1' (угловая минута). Значит, на расстоянии 1 метра человек способен различить изменение освещенности на площади равной приблизительно трети миллиметра (3.1):

$$\Delta = tan(1')d \approx 0.29mm$$

Таким образом, монитор с диагональю 22 дюйма (дюйм $\approx$ 2.54 см) и отношением сторон 16:9 должен иметь минимальное разрешение 1678X944(3.2).

$$diag = \frac{22*25.4}{0.29} \approx 1926pix$$

$$x = \frac{16*diag}{\sqrt{16*16+9*9}} \approx 1678pix$$

$$y = \frac{9*diag}{\sqrt{16*16+9*9}} \approx 984pix$$

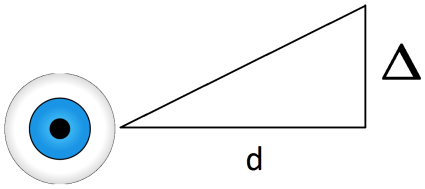


Рис. 3.1. Разрешающая способность глаза человека.

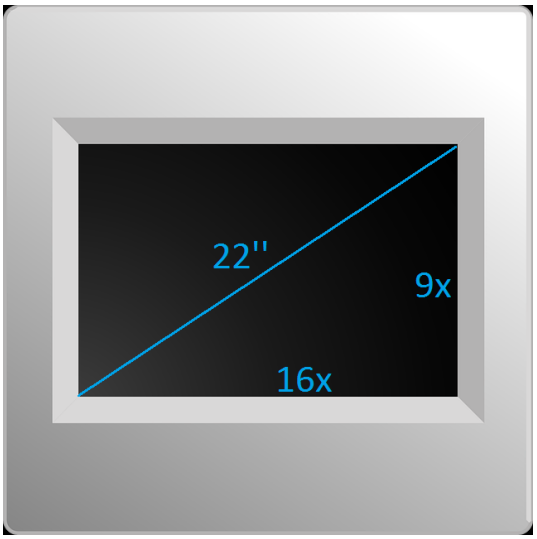


Рис. 3.2. Необходимая разрешающая способность монитора.

Понятно, что все зависит от расстояния просмотра, контрастности, условий освещения. Но, тем не менее, можно сказать, что в среднем разрешения современных мониторов и телевизоров вполне достаточно для того, чтобы человек не различал отдельных пикселей.

3.1.2. Цветовое разрешение дисплея

Однако с цветовым разрешением дисплея всё обстоит несколько сложнее. Рассмотрим отдельно тон, насыщенность и светлоту.

Считается, что человек различает 150 спектральных тонов и 10-60 оттенков насыщенности в зависимости от длины волны [6]. Таким образом, при фиксированной яркости различимы несколько тысяч разных цветов. Это приводит нас к заключению, что для кодирования цвета достаточно 8 бит на канал. С помощью 24-битного представления можно передать все различимые человеком цвета (без учета светлоты).

Возможности человеческого глаза на данный момент существенно превосходят возможности устройств получения изображений и устройств вывода изображений. Считается, что человек в состоянии адаптироваться к освещению в пределах 10 порядков изменения яркости. При этом одновременно глаз различает до 5 порядков. Современные мониторы дают максимум 2 порядка. Таким образом, даже если в сцене оказалось несколько объектов различной

яркости: яркое небо, солнце, освещенная земля, тени, то визуализировать всё это разнообразие на мониторе или телевизоре не получится.

Полученные яркости растянутся на доступный диапазон, реалистичность резко ухудшится, всё качество визуалзации будет потеряно. Существует огромное кол-во задач и сцен, в которых это имеет большое значение. То, как мы видим солнце в реальной жизни, разительно отличается от того, как мы видим солнце на фотографии, где солнце может иметь яркость, сравнимую с яркостью иных объектов.

3.1.3. Статический и динамический диапазон устройств

Динамический диапазон сцены - отношение максимальной яркости к минимальной. Для измерения динамического диапазона устройств так же можно использовать линейное соотношение максимума и минимума. На практике минимум освещенности никогда не бывает равен 0, так как у монитора всегда есть минимальный уровень отражаемого света. Нулевое значение данного линейного отношения в точке означает отсутствие воспринимаемой величины цвета. Линейные соотношения зачастую оказываются очень большими и неудобными в использовании.

Поэтому используются другие модели, например:

1) Стоп (двоичный логарифм отношения) – часто применяется в фотографиях

2) Децибел (десятичный логарифм отношения) - применяется для одномерных сигналов

Таким образом, при помощи логарифма мы можем измерить диапазон яркости и громкости в стопах и децибелах.

Статический диапазон устройства – разница между максимальной и минимальной освещенностью в данный момент времени, а динамический диапазон устройства – во все моменты времени.

Например, статический диапазон человеческого глаза приблизительно равен 4 порядкам, а динамический диапазон - десяти. Такой скачок достижим за счет способности глаза к адаптации. Адаптация происходит как на уровне реакции зрачка, так и на уровне нервных клеток сетчатки.

Статический диапазон монитора – 2 порядка линейной интенсивности. Однако, увеличивая мощность подсветки, можно увеличить максимальную яркость монитора, и, следовательно, динамический диапазон. Этим фактом часто пользуются производители мониторов, указывая завышенные значения

Фон	Яркость $[cd/m^2]$
Ночное небо без луны	0.00001
Ночное небо с луной	0.001
Ясный день	1000
Солнечный день	10000
Монитор (CRT, LCD)	1000

Таблица 3.1. Яркость некоторых сцен

контрастности. Нужно понимать, что при активизации подсветки яркость самого темного пикселя также увеличивается. Это значит, что реальный контраст и диапазон отображаемой фотографии не изменится. Максимум, что можно сделать – изменить уровень подсветки для общего уровня яркости сцены: если визуализирован солнечный день, увеличить подсветку, если мрачное подземелье – уменьшить. С практической точки зрения, гораздо важнее одновременный уровень яркости для демонстрации и темных и светлых объектов в один и тот же момент времени.

Ниже приводится таблица типичных яркостей излучения (в канделах на метр квадратный). Из таблицы видно, что яркость современных мониторов корректно отображает яркость ограниченного набора сцен. Остальные сцены можно отобразить лишь приблиежнно, используя специальные техники для повышения реализма.

3.1.4. LDR и HDR

Итак, изображение широкого динамического диапазона – изображение, диапазон яркости которого превышает 8 бит (то есть динамический диапазон превышает 256:1). Все изображения, которые вы видите в этой книге – изображения узкого динамического диапазона, т.к. они ограничены возможностями выводящего устройства (экрана или бумаги). Ниже представлены 2 пары изображений (рис. 3.3,3.4). Слева показано LDR, справа – специальным образом визуализированное HDR. Заметьте, что в первой паре на левом изображении недосвечена лодка, а небо пересвечено. В реальности мы бы увидели и детали лодки, и детали на небе, но возможностей фотоаппарата не хватило, чтобы запечатлеть всё в одной экспозиции. На правом изображении это было компенсировано, поэтому результат более близокк реальности. Во втором примере на левом изображении были потеряны детали за окном.

Изображение широкого динамического диапазона может быть использовано везде, где может быть использовано обычное изображение. Обратное неверно. Поэтому область применения HDR охватыает множество применений LDR.

Рис. 3.3. Слева пример фотографии (городской пейзаж) без использования алгоритма тональной компрессии. Справа пример фотографии (городской пейзаж) с использованием алгоритма тональной компрессии.

Рис. 3.4. Слева пример фотографии (интерьер) без использования алгоритма тональной компрессии. Справа пример фотографии (интерьер) с использованием алгоритма тональной компрессии.

Рис. 3.5. Слева пример синтеза без использования алгоритма тональной компрессии. Справа пример синтеза с использованием алгоритма тональной компрессии.

Вот лишь краткий список приложений, в которых использование HDR критично:

- Физически-корректная визуализация (техники глобального освещения)
- Эффекты дополненной реальности (спецэффекты в кино, мультфильмы)
- Симуляция человеческого зрения
- Профессиональная фотография
- Компьютерные игры

Список приложений со временем будет только расширяться. Результаты алгоритмов фотореалистичного синтеза, которые будут рассматриваться во второй части курса, без потерь описываются изображениями широкого диапазона. Для представления их на экране нужно использовать методы тональной компресии, которые будут рассмотрены в 4 пункте.

В заключение пункта приведем пример результатов фотореалистичного синтеза с использованием HDR (рис. 3.5).

3.2. Получение изображений широкого динамического диапазона

Источником изображений широкого динамического диапазона может служить либо виртуальная реальность, либо реальный мир.

В первом случае, у нас имеется некоторый алгоритм синтеза изображений. В этом алгоритме мы можем напрямую работать с физическими величинами (можем даже проводить вычисления для спектральных распределений). В результате мы получаем энергетический срез сцены в пространстве экрана. Таким образом, фактически в алгоритме синтеза мы получаем изображение широкого динамического диапазона. Далее нужно сохранить результат в некотором формате, минимизируя по возможности потери (об этом в секции 3 или же вывести изображение на монитор (об этом в секции 4).

Во втором случае физическое излучение реального мира фиксируется с помощью устройств ввода (фото- и видео-камер). У регистрирующей аппаратуры существуют физические ограничения.

1) Матрица фотоаппарата имеет довольно узкий динамический диапазон. Современные цифровые зеркальные камеры поддерживают 14 бит на канал (т.е. 16384:1). Этого достаточно для многих сцен, но диапазон уже, чем диапазон человеческого зрения. Более простые фотоаппараты в принципе не обладают столь хорошими матрицами.

2) Программное обеспечение фотоаппарата зачастую сжимает информацию до 8 бит на канал.

Итак, возникает ситуация, когда полученное с помощью фотокамеры изображение не может передать весь динамический диапазон реальной сцены. Что делать в таком случае? Как получить изображение широкого динамического диапазона при ограниченных возможностях матрицы фотоаппарата?

3.2.1. Получение HDR-изображения из набора LDR-изображений.

Фотограф имеет возможность регулировать время выдержки, в течение которого свет попадает на матрицу. Благодаря этому он может получить ряд изображений с разной экспозицией. Если время выдержки мало, то на матрицу поступит энергия лишь с тех направлений, где она высокая. Полученное изображение будет содержать детали в самых ярких областях сцены, а во всех других изображение будет темным. Если увеличивать время выдержки, то в темных областях начнут проявляться детали, а в светлых произойдет переполнение, пересветка области. Таким образом, снимая одну и ту же сцену с разной экспозицией, можно получить набор LDR-изображений, содержащих достоверную информацию о разных частях сцены. Набор таких изображений дан на рисунке 3.6.

На левом избражении плохо различим дом и площадка перед ним, зато хорошо видно дорожку и листву дерева на переднем плане. На среднем изображении хорошо различимы детали дома, травы и листвы на заднем плане,

Рис. 3.6. Вверху набор фотографий с разной выдержкой. Внизу те же фотографии, цветами выделены области с хорошо различимыми деталями.

зато дорожка уже не видна. На третьем хорошо различима площадка перед домом, а всё остальное – слишком темное.

Формализуем задачу и опишем алгоритм поиска физических значений яркости по набору фотографий с разной выдержкой.

На входе:

- набор цифровых фотографий $j = 1..n$, 24 бит на пиксель; в процессе съемки освещенение не меняется, объекты в кадре не движутся;
- для каждой фотографии абсолютное или относительное время экспозиции Δt_j.

На выходе: HDR-изображение.

Алгоритм основан на принципе обратимости, который является физическим свойством как фотохимических, так и электронных систем получения изображения. Откликом точки на пленке или элемента сенсора будет экспозиция X. Принцип обратимости говорит о том, что экспозиция может быть задана как произведение освещенности $E[\frac{\text{Вт}}{\text{м}^2}]$ и времени t, т.е. имеет единицы $[\frac{\text{Дж}}{\text{м}^2}]$. Таким образом, имея значения X и t, можно найти исходную освещенность точки, которую можно считать пропорциональной искомой световой яркости L. К сожалению, значение пикселя в фотографии не равно X. Это происходит по нескольким причинам:

1) внутрикамерная обработка и необходимость сохранения я 24х-битный формат приводят к нелинейному кодирования цвета;
2) из-за ограниченной чувствительности сенсоров при малых t или E значение экспозции будет столь мало, что сенсор даст нулевой отклик. А при больших значениях, наоборот - максимальный для данного сенсора.

Будем считать значение i-го пикселя в изображении j равным Z_{ij}. Тогда

$$Z_{ij} = f(X) = f(E_i t_j)$$

где f - характеристическая кривая (функция) камеры. Она преобразует экспозицию в значение пикселя. Для построения HDR-изображения эту функцию необходимо знать. Для учебных целей можно принять ее равной кривой гамма-коррекции с $\gamma = 2.2$ Так как эта функция монотонна, то она обратима. Если предположить, что эта функция известна, то $f^{-1}(Z_{ij}) = E_i \Delta t_j$ Тогда

$$E_i = \frac{f^{-1} Z_{ij}}{\Delta t_j}$$

Проблема заключается в том, что таким способом нельзя восстановить освещенность по одному изображению, т.к. информация в недосвеченных и пересвеченных пикселях будет потеряна. Также нельзя доверять яркости пикселей около нуля и близких к насыщению. В пикселях около нуля сказывается цифровой шум, а в ярких пикселях возможные засветки от ярких источников света, попавших в соседние пиксели. Поэтому надо брать информацию из всех изображений с разными весами. Введем функцию $w(Z)$, которая возвращает вес пикселя Z в зависимости от его яркости и с этими весами усредним значения одного пикселя i для разных значений экспозиций j. Таким образом

$$E_i = \frac{\sum_{j=1}^{n} w(Z_{ij}) \frac{f^{-1}(Z_{ij})}{\Delta t_j}}{\sum_{j=1}^{n} w(Z_{ij})}$$

Для выбора функции $w(Z)$ возможны варианты. Если предположить, что $Z_{ij} \in [0; 1]$, то можно использовать функцию типа $1 - (2z - 1)^{12}$ (по примеру [1]). Отдельно придется обрабатывать ситуации с нулем в знаменателе и, возможно, настраивать функцию для борьбы с цифровым шумом. Результирующие E_i представляют собой значение одного канала изображения. Все приведенные выше рассуждения применются к каждому каналу и результат сохраняется (без нелинейных преобразований) в один из HDR-форматов. В описанном выше алгоритме есть два важных момента. Первый – выбор весовой функции $w(z)$ и второй – неизвестный вид кривой отклика $f(X)$. Неаккуратное исполнение этих моментов может привести к тому, что после деления значения интенсивности пикселей будут сильно отличаться.

3.2.2. Выбор весовой функции

Если неправильно учтена кривая отклика камеры, то эта ошибка будет систематической, иначе причиной несовпадения может быть одна из следующих причин: шум, изменение сцены за время фотографирования, краевое или близкое к краевым значение пикселя. Проблемы такого типа решаются выбором весовой функции. При этом весовая функция может, в принципе, вовсе обнулить веса у некоторых значений.

Весовую функцию можно выбрать несколькими способами. Например, не учитывать все недосвеченные (менее 0) и пересвеченные (более 255) пиксели, а остальные, поделенные на экспозицию члены, усреднять.

Однако этого часто бывает недостаточно. Темные пиксели содержат серьезный шум. Белые пиксели могут быть бликом. Функция смешивания присваивает каждому значению яркости (от 0 до 255) некоторый вес. Примеры таких

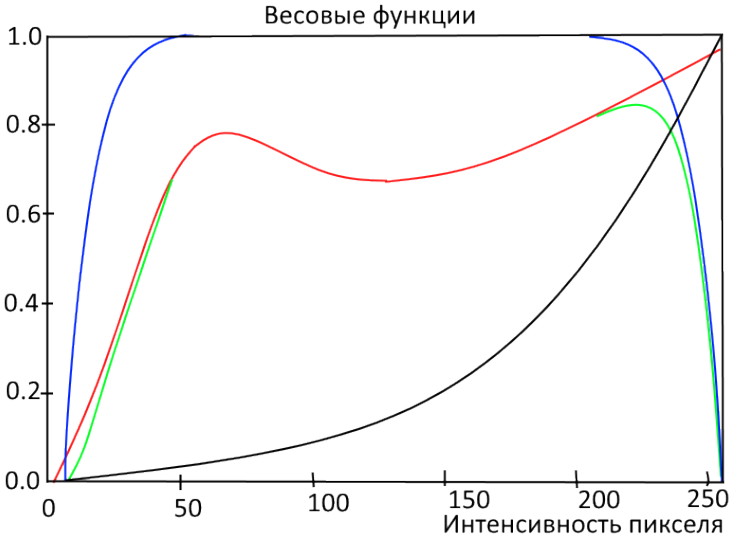


Рис. 3.7. Различные весовые функции. Черная линия - обратная кривая отклика камеры, синяя линий - hat-функция $(1-(2z-1)^{12})$, красная - функция Mitsunaga-Nayar, зеленая - произведение функций Mitsunaga-Nayar и hat-функции.

функций представлены на изображении 3.7. Они уменьшают вес у пикселей с низким уровнем доверия.

На рисунке 3.8 показан результат работы одного из методов построения HDR по LDR.

Слева цветом закодировано, из какого изображения был взят пиксель: красный означает, что из самого темного, желтый – из среднего, синий – из яркого. Правее показан результат визуализации изображения широкого динамического диапазона, при этом на мониторе старались передать все яркости, присутствующие в HDR. В итоге мы получили иображение, в котором содержится вся информация о нашей сцене, в то время как ни в одном из исходных всей информации не было.Можно считать, что в полученном изображении содержатся яркости, пропорциональные абсолютным фотометрическим, и использовать изображение для расчетов (например, в качестве карты окружающей освещенности). Чтобы получить абсолютные значения яркости, нужно использовать базовый калиброванный объект, яркость которого мы точно знаем. А желательно произвести серию тестовых замеров для минимизации экспериментальной ошибки.

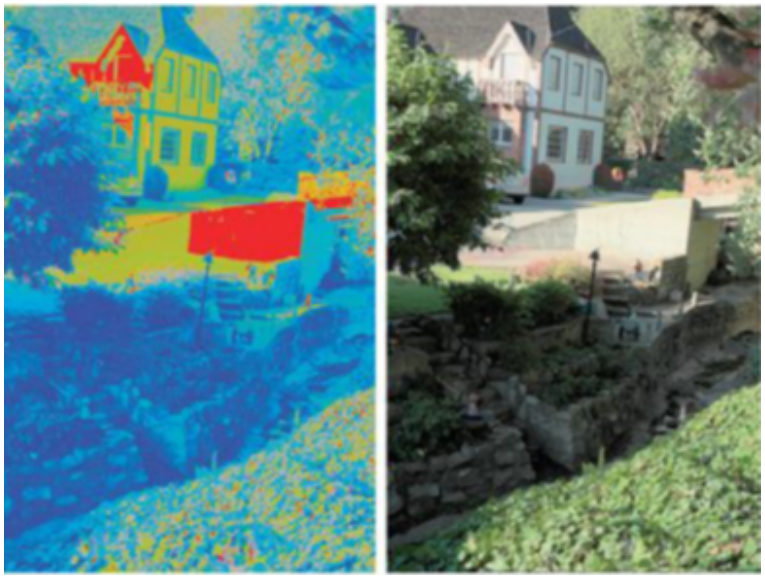


Рис. 3.8. Результат работы метода (справа) и закодированный индекс изображения (слева).

3.2.3. Восстановление кривой отклика камеры.

Кривая отклика камеры может быть точно не известна. Кривые отклика реальных фотоаппаратов практически никогда не соответствуют гамма-функции, так как производители фотоаппаратов не публикуют свои кривые отклика, которые уменьшают шум, увеличивают насыщенность, корректируют цвета. Поэтому базовый алгоритм, представленный в пункте 2.1, будет работать с огрехами. Использование неверной кривой отклика приводит к тому, что отношения между яркостями в результирующем изображении получаются нелинейными. Однако, эту кривую можно посчитать прямо из исходных изображений. Вернёмся к уравнению $f^{-1}(Z_{ij}) = E_i \Delta t_j$. Возьмем логарифм от обеих частей и обозначим $g = ln(f^{-1})$ Тогда

$$g(Z_{ij}) = ln(E_i) + ln(\Delta t_j)(*)$$

Новая функция g также монотонная и гладкая. Множество значений её аргумента – конечно. $Z_{ij} \in [Z_{min}, Z_{max}]$. Нам нужно минимизировать ошибку в системе уравнений типа (*). У нас $Z_{max} - Z_{min} + N$ неизвестных, где N – количество пикселей. Значит, необходимо больше уравнений, чем $Z_{max} - Z_{min} + N$. Кроме минимизации ошибки, нам необходимо также, чтобы результирующая функция была гладкой. Поэтому модифицируем результирующий функционал ошибки:

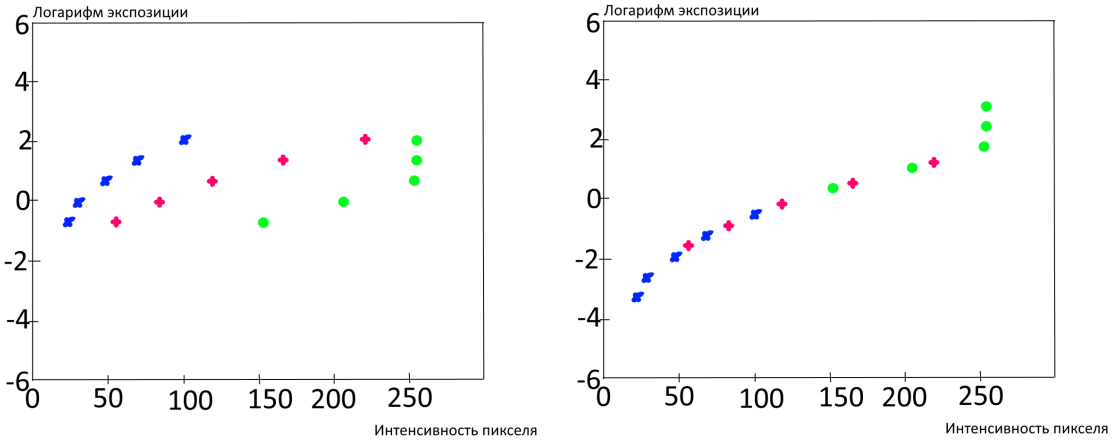


Рис. 3.9. Иллюстрация идеи восстановления кривой отклика.

$$O = \sum_{p=1}^{N} \sum_{j=1}^{p} (g(Z_{ij}) - ln(E_i) + ln(\Delta t_j))^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}} g''(z)^2$$

Таким образом получается расширенная система уранений, которую можно решить любым способом, например сингулярным разложением (SVD - singular value decomposition)[7]. На самом деле при добавлении постоянной к E, система уранения останется верной, поэтому нужно дополнительно зафиксировать одну из точек результирующей функции. Например, можно ввести нормировку:

$$g(\frac{Z_{min}+Z_{max}}{2}) = 0$$

На краях функция будет менее гладкой. Можно тут тоже использовать весовую функцию w внутри функционала минимизации:

$$O = \sum_{p=1}^{N} \sum_{j=1}^{p} w(g(Z_{ij}) - ln(E_i) + ln(\Delta t_j))^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}} wg''(z)^2$$

Для восстановления кривой не обязательно использовать все точки изображения, необходимо лишь, чтобы $N(P-1) > Z_{max} - Z_{min}$. То есть случайным образом выбирается несколько точек с разной яркостью. Для них выбираются значения для всех выдержек. На рисунке 3.9 представлены такие значения 3 пикселей для 5 выдержек. По горизонали отложено пиксельное значение яркости, по вертикали – значения правой части (*). В процессе оптимизации подбирается такое решение (множители для этих рядов), чтобы все точки легли на одну кривую (справа).

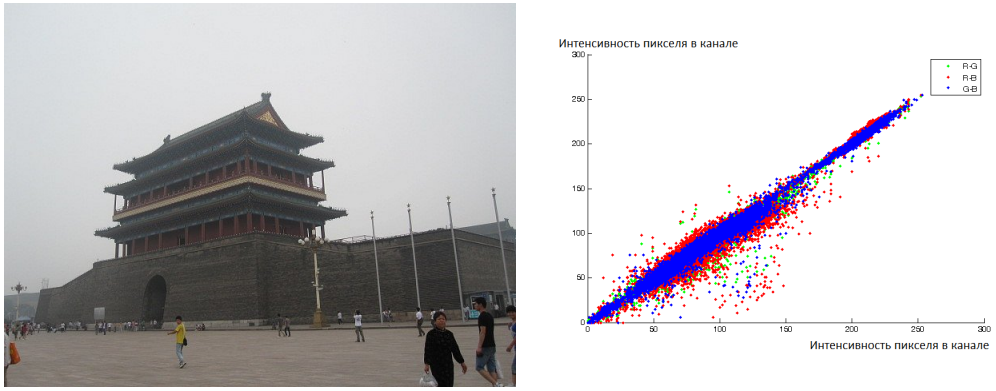


Рис. 3.10. Слева - изображение реального мира, справа - график корреляции каналов (value).

3.3. Хранение HDR-изображений

При фиксированной яркости в реальных изображениях разброс цвета по оттенкам относительно небольшой. Яркие цвета чаще бывают белыми, а насыщенные яркие цвета – довольно редкое явление даже для изображений широкого динамического диапазона. Это наблюдение можно использовать как идею для раздельного кодирования яркости и цвета. При кодировании нам нужно меньше бит выделять на цвет, чем на яркость. Рассмотрим, например, 3.10: изображение и диаграмму корреляции цветовых каналов для него. Видно, что относительно диагонали разброс мал, то есть цвета очень сильно коррелируют. Значит, яркость имеет более сильный разброс, чем цвет.

3.3.1. Способы кодирования яркости

В прошлой главе мы дали пример гамма-кодирования и обратили внимание на выравнивание различимой разницы по яркости с точки зрения человека при таком подходе.

$$\overline{I} =$$

$I^{\gamma}, \gamma -$параметр кодирования, $I -$входное значение, $\overline{I} -$квантованное значение

Цифровое кодирование требует квантования, при этом неизбежны ошибки. Эти ошибки нужно сохранять ниже различимого человеком порога. Поскольку отклик глаза на интенсивноть нелинеен, то и квантизация должна быть нелинейна, иначе будет заметна ступенчатость при переходе от светлого к темному. Если использовать гамма-кодирование, то этот эффект можно частично компенсировать. Однако для HDR-изображения гамма-кодирование

уже не столь хорошее решение, поскольку наблюдатель адаптируется под текущий уровень освещения. Если умножить яркость на достаточно большой коэффициент, то ошибка уменьшится, но сократится разрешение по яркости (уменьшится диапазон представимых значений) – см. график 3.11. Существуют и другие способы кодирования, например, логарифмическое кодирование и кодирование мантисса-экспонента. При логарифмическом кодировании яркость кодируется в соответствии с формулой:

$$\overline{I} =$$

$k^I, k-$параметр кодирования, $I-$входное значение, $\overline{I}-$квантованное значение

Если известны границы диапазона I_{min} и I_{max}, то формула может быть переписана следующим образом:

$$I_{out} = I_{min}(\frac{I_{max}}{I_{min}})^v, v \in [0, \frac{1}{n}, \frac{2}{n}, \dots 1]$$

Соседние значения при логарифмическом кодировании отличаются на постоянную:

$$(\frac{I_{max}}{I_{min}})^{\frac{1}{n}}$$

Кодирование мантисса-экспонента по сути близко к логарифмическому. Число в этом представлении кодируется в виде комбинации мантиссы, обычно задаваемой в некоторых фиксированных пределах (например, от 0 до 1, или от 1 до 2) и экспоненты, значению степени при представлении числа в некоторой системе счисления (например, десятичной). Ошибка для этого типа кодирования является кусочно-монотонной функцией.

$$\overline{I} = M10^e, e - \text{экспонента}, M - \text{мантисса}, \overline{I} - \text{квантованное значение}$$

На графике 3.11 показаны графики относительных ошибок в процентах для каждого из перечисленных видов кодирования.

Резюмируем вышесказанное. При уменьшении интенсивности ошибка при гамма-кодировании растет, но в узком диапазоне при условии адаптированного зрения ошибка при гамма-кодировании будет составлять менее одного процента. Логарифмическое кодирование имеет постоянную ошибку. Кодирование мантисса-экспонента тоже имеет малое значение ошибки, однако она не монотонна.

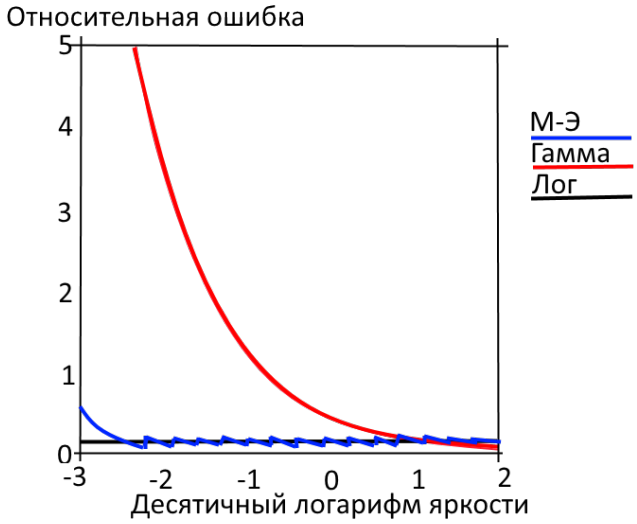


Рис. 3.11. Ошибки разных методов квантизации яркости в HDR-изображениях. По вертикали отложена относительная ошибка, по горизонтали – десятичный логарифм кодируемой яркости.

3.3.2. Форматы хранения HDR-изображений

Radiance RGBE Encoding (HDR)

В 1985 Вард начал изобретение физической системы визуализации Radiance. Поскольку система вычисляла фотометрическую яркость, казалось необуманным выбросить информацию, получая на выходе изображение. Поэтому использовалась 4-байтное представление с общей экспонентой. Кодирование свободно распространялось вместе с системой визуализации. По сей день формат RGBE остается самым популярным форматом хранения HDR-данных. В каком-то смысле является аналогом формата bmp для HDR в силу своей простоты. В формате используется предположение о том, что у цвета нет сильного варьирования по каналам, и это значит, что можно поделить значения каналов на яркость, чтобы уместить цвет в 8 бит, а оставшуюся информацию о самой яркости хранить отдельно. Один байт использовался для мантиссы красного цвета, другой – для зеленого, третий – для синего, четвертый дополнительно кодирвал яркость. При этом яркость кодируется логарифмически. Итого на цвет уходит 24 бита (24 битное представление цвета), на яркость – дополнительные 8 бит. Базисное основание равно двум. Общая формула:

$$E = \lceil log_2(max(R_W, G_W, B_W) + 128) \rceil$$

$$R_M = \lfloor \frac{256 R_W}{2^{E-128}} \rfloor G_M = \lfloor \frac{256 G_W}{2^{E-128}} \rfloor B_M = \lfloor \frac{256 B_W}{2^{E-128}} \rfloor$$

$$R_W = \frac{R_M + 0.5}{256} 2^{E-128} G_W = \frac{G_M + 0.5}{256} 2^{E-128} B_W = \frac{B_M + 0.5}{256} 2^{E-128}$$

Ошибка алгоритма составляет около 1%, покрывается 76 порядков яркости. Формат очень простой, кодирование и декодирование происходят очень быстро. Поддерживается RLE-кодирование (данное представление позволяет цепочке одинаковых цветов кодироваться как совокупность «цвет»+ количество далее идущих подряд пикселей с данным цветом). При HDR-кодировании в итоге в среднем можно получить 25-процентное сжатие. Хотя эта модель стала своего рода стандартом, ей присущи серьезные недостатки:

1) Избыточность динамического диапазона (примерно 62 порядка лишние с точки зрения восприятия человека, для других задач этот ранг может быть не избыточен). Было бы лучше, если бы формат имел меньший диапазон, но лучшую точность при том же числе бит.

2) Мантиссы могут принимать только положительные значения и не могут покрыть весь видимый диапазон. В случае использование вместо RGB XYZ увеличивается ошибка квантования из-за еще большего количества неиспользуемых значений.

3) Распределение ошибки неравномерно по отношению к восприятию. Шаги могут быть различимы для насыщенного голубого и розового (5%).

TIFF (Tagged Image Format)

Одной из первых групп, создавших стандарт кодирования изображений широкого динамического диапазона, был отдел компьютерной графики компании Lucasfilm, который в середине 80-х отделился и стал известен как Pixar. Требовалось сохранять визуализированные изображения. Плёнка способна сохранять динамический диапазон, превосходящий отображаемый на дисплее, при этом кривая скорее логарифмическая. В соответствии с этим в стандарте также используется логарифмическое кодирование.

В формате используется 3 канала, но для каждого отводится 11 бит информации. Таким образом, удалось закодировать динамический диапазон 3600:1 (недостаточно широкий) с шагом в 0.4% (глаз различает 1%, то есть точность вполне достаточная). Цвета не могут принимать отрицательные значения, значит не возможно представить весь цветовой охват.

В дальнейшем расширения формата позволили сохранять в себе 32-битные числа с плавающей точкой, т.е. 96 бит на пиксель. При этом размер файла, разумеется, получался огромным.

LogLuv TIFF

Более эффективное кодирование формата TIFF – LogLuv-кодирование. В 1997 Вард решил исправить ошибки, допущенные при составлении RGBE HDR. Он представил кодек LogLuv в библиотеке TIFF. Это кодирование основано на восприятии и спроектировано таким образом, чтобы шаг квантизации соответствовал порогам обнаружения контраста и цвета. Ошибка при этом квантовании всегда ниже различимого диапазона. Отделив цветность и яркость и применив логарифмическое кодирование к яркости, Вард получил очень эффективную квантизацию. Существует 3 варианта логарифмического кодирования.

1) 10 бит на яркость и 14 бит на пару (u',v'). Это было сделано в основном для доказательство, что использование моделей восприятия помогает эффективнее использовать память. В этом случае можно было на неразличимом уровне работать с цветовым охватом и 4.8 порядками яркости. Пара значений кодировалась индексом на диаграмме цветности (см. 3.12)

2) 16 бит на яркость, динамический диапазон 38 порядков, шаг квантизации 0.3%

3) 16 бит на яркость и по 8 бит на каждый цветовой канал.

Формат не получил широкого распространения. Часть людей предпочитает пользоваться знакомым цветовым пространством (RGB). Объективно это очень хороший, компактный формат, позволяющий наиболее эффективно использовать память запоминающих устройств.

OpenEXR

Формат был представлен в 2002 как открытый формат, с тех пор широко используется. Это оболочка общецелевого назначения для 16-битных данных. Другое название этого формата “S5E10” (“Sign plus 5 exponent plus 10 mantissa”). OpenEXR также поддерживает стандартный 32-битный вещественный цвет с плавающей точкой. Современный формат, был разработан как открытый стандарт. С одной стороны удобный, т.к. открыт для всех спецификаций, есть открытые библиотеки для загрузки и сохранения. С другой стороны - формат был разработан для реализации весьма серьезных алгоритмов для хранения HDR-изображений. В последние годы его поддержка есть во многих программах редактирования изображений.

Может представлять отрицательные значения цвета, покрывает видимый цветовой охват и диапазон 10.7 порядков яркости с относительной точностью в

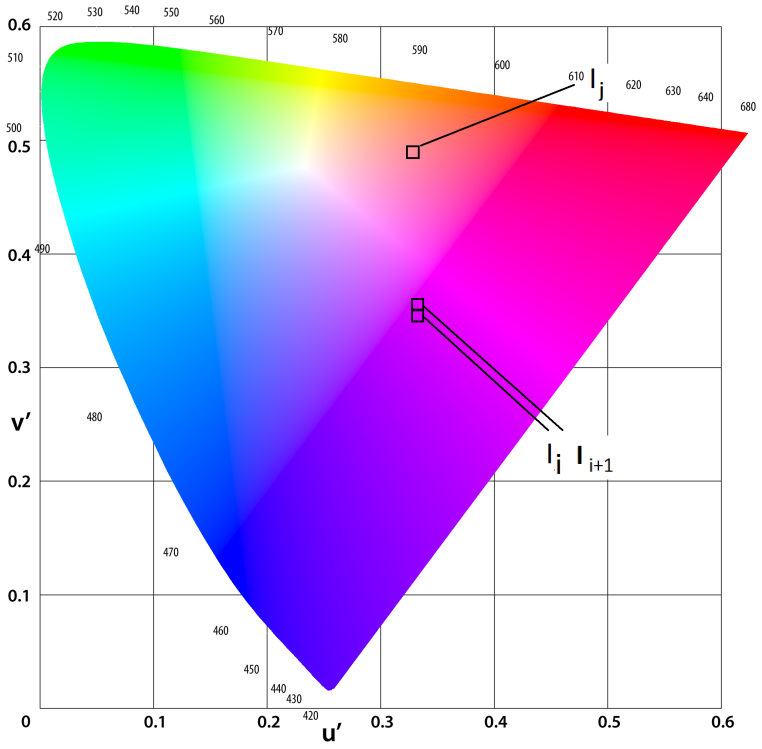


Рис. 3.12. Принцип кодирования пары (u',v') в формате LogLuv TIFF. Индекс кодирует область цветовой диаграммы. В пределах одного индекса разница между цветами незаметна человеком.

0.1%. Поскольку человек может одновременно видеть не более 4 порядков яркости, можно использовать этот формат данных совместно с архивацией. Особенности формата OpenEXR:

1) Ограничено максимальное значение (при этом мы всегда можем использовать некоторый нормировочный коэффициент, теряя в диапазоне представимых значений).

2) Шаг квантования 0.1%. Такой маленький шаг выбирать разумно, когда требуется дополнительно обработать изображение. Ведь после обработки ошибка возрастает.

3) Есть разные форматы сжатия, в том числе и современное сжатие без потерь на основе вейвлетов.

4) Спецификация OpenEXR предлагает возможность использования дополнительных каналов, куда можно записать информацию о прозрачности или глубине.

5) Формат хранения half float поддерживается современными графическими картами.

6) Библиотка распространяется свободно.

Форматы с обратной совместимостью

Идея использования форматов с обратной соместимостью (например, JPEG-HDR) состоит в том, что данные могут интерпретироваться как LDR-изображения и воспроизводиться классическими просмотрщиками изображений, поддерживающих JPEG. А специальные просмотрщики, знакомые с расширением стандарта могут обрабатывать и воспроизводить изображение широкого динамического диапазона.

В метаданных может храниться уменьшенная копия изображения, в пиксели которой записаны коэффициенты, на которые нужно умножить значения исходного пикселя для восстановления изображения широкого динамического диапазона. Если программы не знают о дополнительном изображении в метаданных, они покажут основную каринку без усиления, иначе интерполируют коэффициенты масштабирования для получения HDR-изображения.

Таблица 3.2подводит итог рассмотренных форматов хранения HDR [2]. По таблице можно сделать выводы, что наиболее удачными на сегодняший день являются форматы Open EXR и LogLuv TIFF. Форматы RGBE,XYZE избыточны, форматы sRGB,Pixar TIFF недостаточны для представления различимых глазом цветов. У формата Open EXR более широкое распространение,

Формат хранения	Покрытие	Диапазон (порядки)	Шаг квантования	Бит на пиксел
sRGB	Нет	1.6	Переменный	24
RGBE	Нет	76	1%	32
XYZE	Да	76	1%	32
Pixar TIFF	Нет	3.8	0.4%	33
LogLuv 24	Да	4.8	1.1%	24
LogLuv 32	Да	38	0.3%	32
EXR	Да	10.7	0.1%	48

Таблица 3.2. Форматы хранения HDR-изображений

а возможность эффективного сжатия без потерь компенсирует избыточный объем данных пикселя.

3.4. Визуализация HDR-изображений. Алгоритмы тональной компрессии.

Большинство современных средств визуализации рассчитано на работу с LDR-изображениями. В последнее время появились мониторы с более высоким динамическим диапазоном, позволяющим превосходить ограничения стандартных LCD-дисплеев (кратко будет рассказано в конце секции). Однако основная масса дисплеев оперирует 256 градациями яркости.

Пусть имеется некоторое HDR-изображение, содержащее яркости в пределах шести порядков (от 0 до 100 000). Для представления модельного изображения на экране нужно тем или иным способом сопоставить каждому из 100 001 значений яркости в данном пикселе экрана значение в диапазоне монитора (от 0 до 255). При этом, разумеется, произойдет потеря информации.

Алгоритмы тональной компрессии стараются уменьшить динамический диапазон изображения таким образом, чтобы минимизировать потерю информации с точки зрения наблюдателя. Алгоритмы тональной компрессии отвечают на вопрос: “Как в условиях ограниченного диапазона оборудования отобразить на нём изображения, таким образом, чтобы с точки зрения наблюдателя отображенное изображение воспринималось идентичным исходной сцене?”

Простейший вариант отображения диапазонов – обрезать значения, или взять их с некоторым сдвигом. Это приведет к тому, что всё, что было ярче верхней границы или темнее нижней, просто станет белым, и все детали за пределами порогов станут неразличимы. Ясно, что для большинства изображений

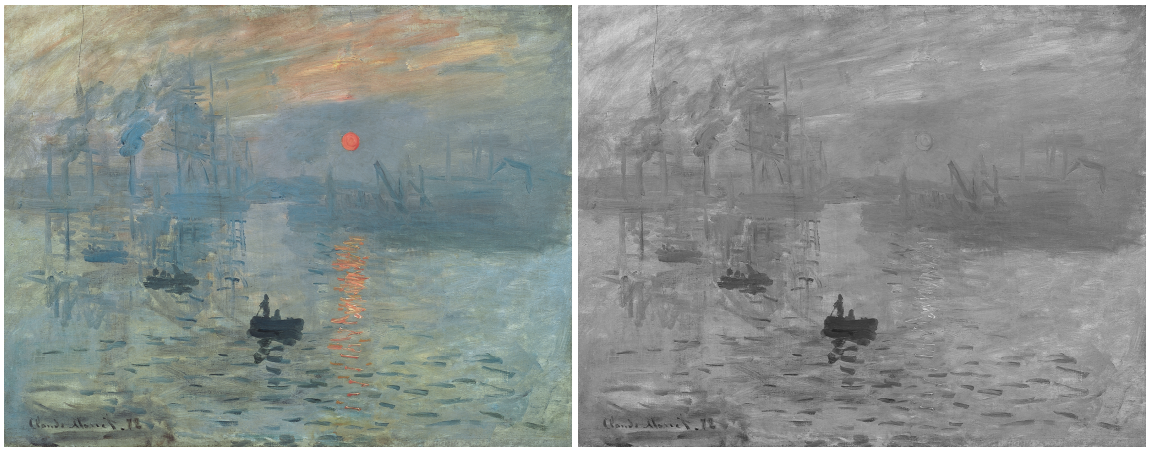


Рис. 3.13. Клод Моне. «Впечатление. Восходящее солнце», 1872. Слева - цветная репродукция, справа - черно-белая.

это приведет к потери информации. Другой простой вариант – нормировать изображение. В этом случае мы теряем детали на изображении (т.е. если на картинке солнце яркостью 100000 и остальная сцена, которая укладывается, скажем, в 500 – то на сжатом изображеии мы не увидим ничего, кроме солнца). Можно применить также нелинейную функцию, которая переведет значения от 50 000 до 100 000 в диапазон от 150 до 255, а все остальное - в диапазон от 0 до 150.

Цель алгоритма тональной компрессии – показать на экране изображение, максимально близкое к тому, что увидил бы человек, находясь в сцене. Решаема ли данная задача в принципе? Оказывается, вполне решаема. На протяжении столетий она успешно решалась художниками, отображающими окружающий мир с помощью ограничений имеющейся палитры. При этом вы не раз сами убеждались в достигаемом ими реализме. В своей работе художники использовали собственную зрительную систему для верификации получающегося правдоподобия. Эль Греко использовал насыщенные цвета противоположных оттенков для увеличения видимого диапазона сцены. Также он прорисовывал контуры темными и светлыми мазками, что увеличивало воспринимаемый контраст изображения. На рисунке 3.13 изображено знаменитое потоно Моне «Impressions at Sunrise». Солнце и облака имеют одинаковую физическую яркость. Однако насыщенный красный солнца помещен на насыщенный синий фон неба, что усиливает впечатление. Справа приведен обесцвеченный вариант изображения, на котором солнце почти незаметно.

После эры художников, за дело взялись фотографы. Выдающимся мастером можно считать Анселя Адамса, он первым систематически измерил диапазон всего имеющегося оборудования. Изобретенная им зонная система позволила с точностью предсказать, какие детали он может заснять на пленку, поэтому

он мог выставить правильные настройки съемки до спуска затвора. Кстати, его идеи легли в основу одного из алгоритмов тональной компрессии, о котором будет рассказано далее.

3.4.1. Классификация операторов тональной компрессии

Операторы тональной компрессии целенаправленно разрабатываются учеными вот уже без малого полвека. Классифицировать все имеющиеся на сегодняшний день наработки довольно сложно, однако некоторые общие закономерности выявить, все-таки можно. Среди операторов тональной компрессии можно выделить минимум 2 основных типа:

- глобальные, пространственно-равномерные операторы,
- локальные, пространственно-неравномерные операторы.

Первые еще иногда называют TRC(tone reproduction curve), вторые – TRO(tone reproduction operator)[5]. Глобальные операторы используют идентичную кривую сжатия для всех пикселей изображения. Локальные операторы используют информацию о пикселе и его окружении для того, чтобы скорректировать значение яркости вокруг этого пикселя.

Иногда в литературе [6] можно встретить выделение третьего типа операторов – временные тональные операторы. В таких операторах как правило учиываются не только пространственные, но и временные законы восприятия зрительной системы. Далее мы не будем останавливаться на них подробнее. Интересующиеся могут самостоятельно изучить статьи [5] и [6].

В дополнение к сжатию диапазона яркости, тональная компрессия может подражать воспринимаемых характеристикам, воспроизводя изображения, возбуждающее в зрительной системе тот же отклик, что и реальная сцена. Например, оператор тональной компрессии может пытаться сохранять такие характеристики как контраст, яркость или четкость – черты, которые могут быть утеряны при сжатии. Для корректного сжатия диапазона нужно разрабатывать психо-физиологические модели восприятия цвета и видимого контраста. Часто методы тональной компрессии сосредотачиваются на единичных аспектах, таких как восприятие яркости. Это происходит, потому что отсутствуют глубокие знания об устройсиве зрительной системы человека. Однако такие алгоритмы не эффективны, поэтому часть исследователей не рассматривает устройство зрительной системы человека вообще и работает исключительно с экспериментальными данными.

Резюмируя, можно выделить вторую плоскость классификации алгоритмов тональной компрессии:

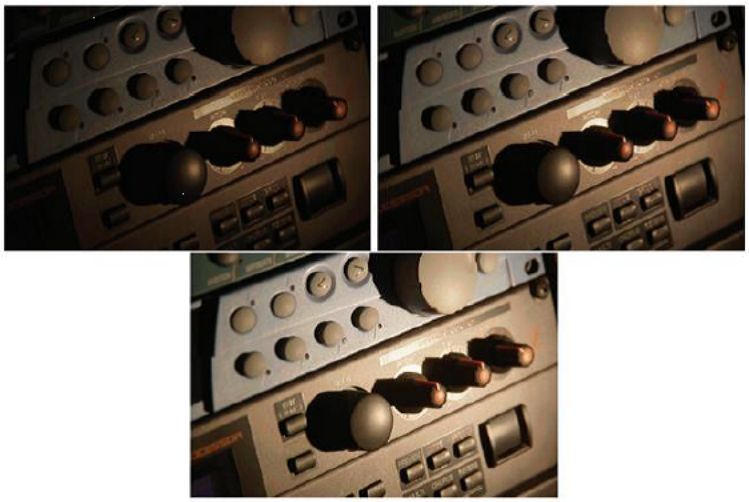


Рис. 3.14. Пример работы одного алгоритма для разной нормировки физической яркости.

- экспериментальные,
- теоретические.

Нужно также отметить важность нормировки изображений (3.14). Для исходного изображения широкого динамического диапазона настоящая яркость, как правило, неизвестна. Поделенная на экспозицию, яркость изображения всегда получается приведенной к некоторому диапазону в зависимости от того, какой алгоритм синтеза мы используем. Это приводит к проблемам, т.к. если у нас в некоторых пикселях изображения яркость, близкая к нулевой на грани распознавания человеческим глазом, и нужно визуализировать ее на мониторе так, как эту яркость увидел бы человек, то должны работать специфические механизмы (колбочки, а не палочки или палочки, а не колбочки), в зависимости от уровня яркости, при котором ведется наблюдение на мониторе. Вообще говоря, алгоритмы могут зависеть от абсолютной яркости.

Если для той или иной задачи требуется восстановить точные значения освещенности для реальной сцены, то для этого как правило необходимо иметь дополнительную информацию об устройстве, на котором производилась съемка (фотоаппарате). Каждому фотоаппарату соотвествует свой коэффициент, связывающий значения на матрице с физической энергией. Этот коэффициент можно рассчитать по специальным калибровочным объектам.

В случае, если изображение создается в результате синтеза, необходимо аккуратно учитывать все физические преобразования яркости в результате отражений и преломлений, чтобы получить на результирующем изображении

корректные физические значения. Соответсвенно, часть алгоритмов тональной компрессии (обычно из числа теоретических) требует физических значений яркости на вход, а другая часть не требует.

3.4.2. Глобальные операторы тональной компрессии

Глобальные операторы имеют простой вид и очень высокую скорость работы, однако при любом выборе кривой мы теряем либо в бликах, либо в объектах, расположенных преимущественно в тени, ведь чем больше диапазон, тем больше значений нам нужно отобразить. Заметный плюс глобальных операторов – отсутствие артефактов на полученном изображении, так как изменения с пикселями происходят гладко, плавно, и локальные соотношения яркости между соседними пикселями не изменяются. Глобальные методы плохо работают с изображениями, где области интереса (детали) равномерно расположены по всему диапазону яркости.

С примером глобальных операторов мы уже сталкивались на прошлой лекции, когда говорили о гамма-коррекции. На самом деле гамма-коррекцию можно произвести и с изображением широкого динамического диапазона.

Исторически так сложилось, что в разные временные периоды исследователи стремились к сохранению в алгоритмах тон-маппинга различных психифизиологических характеристик. Первой такой характеристикой была яркость.

Впервые в 1984 Ngai и Miller [8] использовали экспериментальные данные для сопоставления яркости реальной и отображаемой сцены для цели определения освещенности пикселя в их системе визуализации. Они использовали психофизиологические данные из работы [9]. Далее Tumblin and Rushmeier [10] таже сфокусировались на сохранении общего впечатления наблюдателя о яркости сцены. Они создали модель наблюдателя – математическую модель зрительной системы, которая включала свето-зависимые визуальные эффекты по преобразованию физических значений освещенности в воспринимаемую яркость.

Далее глобальные методы сместились в сторону сохранения контраста. В работе [11] были сделаны попытки свести вычислительную сложность алгоритма к минимуму, умножая реальные значения на коэффициент. Основываясь на контрастной чувствительности глаза, он представил адаптацию зрения как сдвиг в абсолютной яркости освещенности, необходимой, чтобы наблюдатель заметил вариацию яркости. Недостатком его метода было откидывание очень ярких и темных яркостей. Эксперименты, на которые он опирался, проводились в лабораторных условиях и не учитывали сложности типичных условий наблюдения.

Работа [12] представила технику выравнивания гистограммы для отображения воспринимаемо точных тонов в HDRI. Фокус был сделан на различимости объекта и контрасте изображения, вторичная цель была в воссоздании субъективного отклика наблюдателя о консистентности наблюдаемых сцен. Они использовали тот факт, что зрительная система чувствительна к относительным, а не абсолютным изменениям яркости, поэтому яркое должно быть ярким, а тусклое тусклым вне зависимости от абсолютных значений. Кумулятивная гистограмма яркости используется для нахождения кластеров яркостных уровней и присваивает им итоговые уровни на основе контрастной чувствительности.

В [13] представили 2 новых тональных оператора для имитации некоторых процессов адаптации зрительной системы человека. В процессе рендеринга послойный метод конструирует изображение из нескольких слоев освещения и свойств поверхности. Это делается путем разделения изображения на слои и сжатия слоя освещения с сохранением отражательности и прозрачности, что уменьшает контраст с сохранением деталей. Второй, фовеальный метод, итеративно настраивается для сохранения деталей в регионе взгляда (на который наблюдатель указывает мышью) и сжимает остальное. Оба эти оператора вычислительно не затратные. Послойный метод подходит для статических синтетических сцен, а фовеальный – для интерактивных.

Алгоритм Drago

В статье [14] предлагается быстрая высококачественная техника тональной компресии для отображения высококонтрастных изображений на устройствах с ограниченным динамическим диапазоном светимости. Метод основан на логарифмическом сжатии значений освещенности, таким образом имитируется отклик человека на свет. Представлена функция смещения мощности, автоматически варьирующая основание логарифма. Алгоритм удовлетворяет следующим требованиям:

1) согласованность с восприятием человека для изображений естественных сцен;

2) дополняемость и расширяемость;

3) минимизация артефактов (инверсия контраста, ореолы);

4) интуитивные параметры настройки;

5) интерактивность.

Яркость выходного изображения зависит от освещенности сцены. Значит, требуется найти множитель для яркости, аналогичный экспозиции. Для статических изображений вычисляется логарифмическое среднее сцены на основе значений освещенности в пикселях (адаптация к мировой освещенности). Для интерактивного приложения вычисляется логарифмическое среднее региона, окружающего пиксель (центр фиксации взгляда) и свернутого двумерным гауссовым ядром. Площадь семплируемого региона и фильтра Гаусса - 15% изображения, но может настраиваться. Этот метод можно дополнить системой слежения за глазами. Коэффициент нормировки также регулируется для настройки яркости выходного изображения под условия монитора (освещения в комнате).

Базисная функция для рассчета выходной яркости такова:

$$L_d = \frac{log(L_w+1)}{log(L_{max}+1)}$$

Основание логарифма x варьируется между 2 и 10. При $x < 2$ присутствует слишком много деталей, сложно настроить яркости, при $x > 10$ идет большая потеря контраста. Основание логарифма интерполируется между двумя крайними значениями с использованием функции сдвига:

$$bias_b(t) = t^{\frac{log(b)}{log(0.5)}}$$

Итоговая функция тональной компресии используется для вычисления отображаемого значения L_d. L_{dmax} характеризует максимальную освещенность дисплея. Параметр функции смещения обозначается b.

$$L_d = \frac{0.01 L_{dmax}}{log_1 0(L_{wmax}+1)} \frac{log(L_w+1)}{log(2+8\frac{L_w}{L_{wmax}})^{(\frac{log(b)}{log(0.5)})}}$$

b настраивает сжатие в ярких областях и различимость деталей в темных областях (параметр интерполяции). Оптимальный диапазон значений параметра $b \in [0.7, 0.9], b = 0.85$ – по результатам опросов наилучшее значение. Оказалось, что яркость изображения примерно удваивается для $b = 0.85$ и утраивается для $b = 0.7$ по сравнению с яркостью для $b = 1.0$. Это влияет на реализм изображений, несмотря на увеличение контраста. Поэтому мировая яркость должна быть домножена на дополнительный коэффициент:

$$L_{wa} = \frac{L_{wa}}{(1+b-0.85)^5}$$

Далее должна выполняться гамма-коррекция до вывода на экран.

На рисунке 3.15 приведены примеры результатов работы алгоритма. В первом случае результат удовлетворительный, однако на втором примере стало слишком белёсым, пропал контраст.

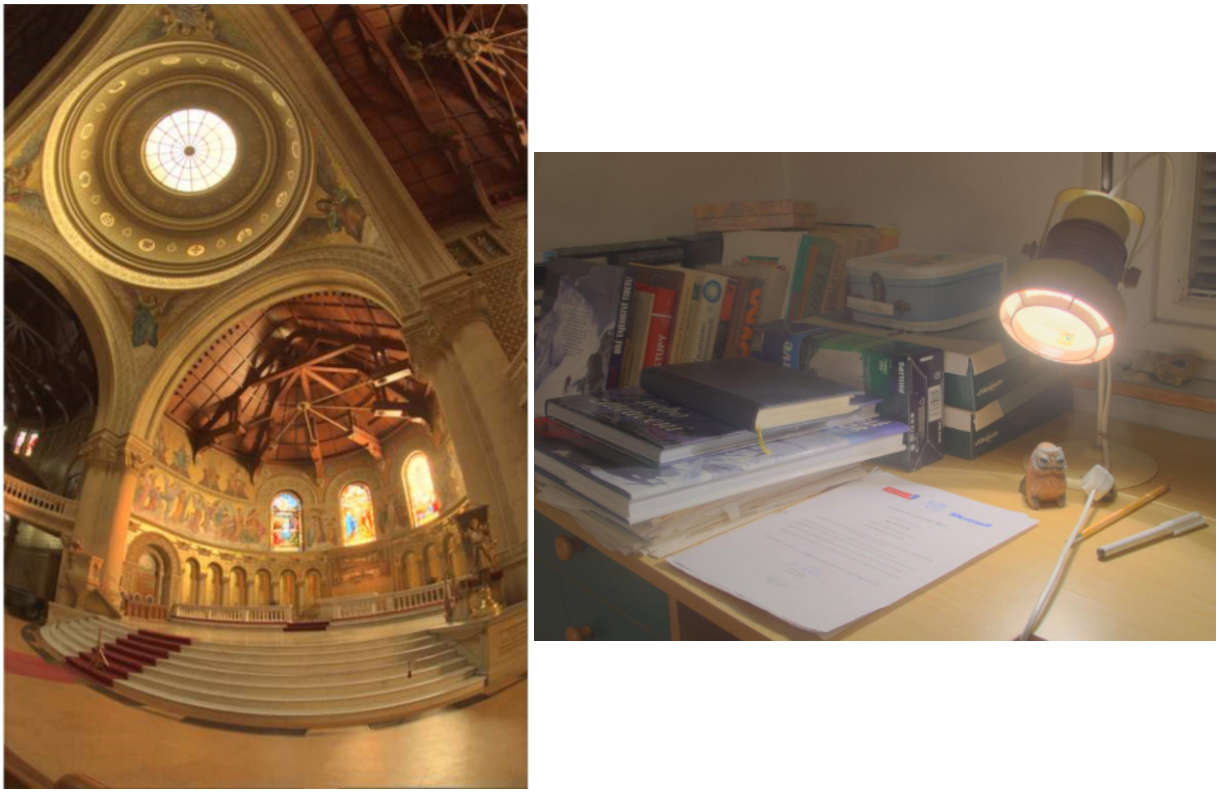


Рис. 3.15. Примеры работы алгоритма.

3.4.3. Пространственно-зависимые операторы тональной компрессии

Среди пространственно-зависимых операторов можно выделить несколько подгрупп:

1) локальные операторы, действующие в ограниченной области вокруг данного пикселя;

2) частотные операторы, использующие разложение изображения на две и более частотых компонент;

3) градиентные операторы, модифицирующие изменения яркости на изображениях.

Впервые попытка создания алгоритма пространственно-зависимой тональной компрессии была проведена в 1968 году Oppenheim, Schafer and Stockham's [12]. Идея их алгоритма была такова. Изображение делится на компоненту освещения (освещенность) и компоненту отражения (отражательность). Компонента освещения содержит большие перепады яркости, состоит из низких частот, а компонента освещения – из высоких. Поэтому низкие частоты – это широкий динамический диапазон, а высокие – узкий динамический диапазон. Уменьшив низкие частоты в пространстве Фурье, можно сжать HDR данные с сохранением высоких чатот. Далее появилось сразу несколько методов, одни из которых следователи этой идее (частотные методы), а другие выбрали свои направления исследований (локальные и градиентные методы).

Локальный оператор тональной компресии. Photographic Tone Reproduction for Digital Images [13]

Эта статья базируется на зонной системе фотографа Анселя Адамса. В результате получается простой и быстрый экспериментальный метод (без моделирования зрительной системы человека). Информации об абсолютных значениях яркости не требуется, так как используется пользовательский интерактивный ввод. Суть метода такова:

1) весь диапазон яркости разбивается на совокупность зон;

2) средняя визуальная яркость в сцене сопоставляется средней зоне;

3) пространственно-равномерный оператор производит сжатие;

4) при этом для сохранения различимости важных областей используется пространственно-неравномерный оператор ретуши («dodging and burning»)/

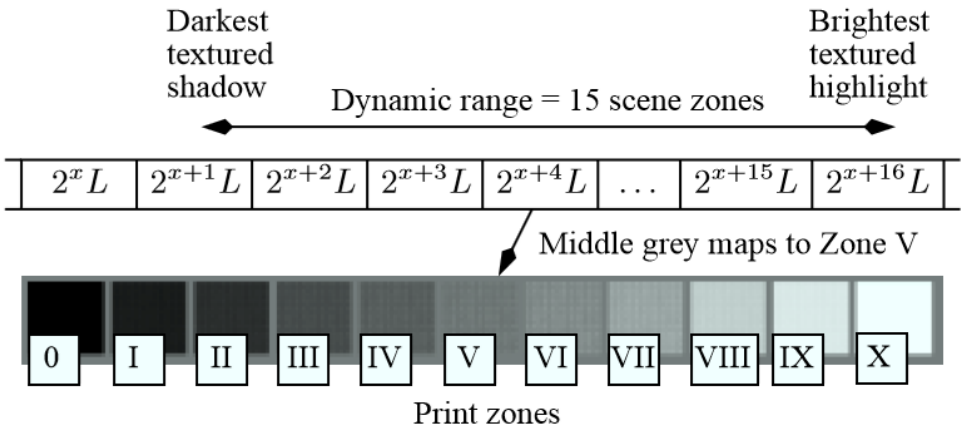


Рис. 3.16. Зонная система.

Зонная система - это система практических замеров, когда фотограф использует информацию о замерах яркости для улучшения качества будущего снимка. Зоны обозначаются римскими цифрами от 0 до X (каждая следующая вдвое ярче предыдущей), означающими примерный диапазон освещенности сцены, а также приблизительную отражательность снимка. Всего печатных зон 11, а зон сцены намного больше. Зоны сцены требуется отобразить в печатные зоны (Рис. 3.16).

Сначала фотограф определяет средне-серый цвет. Это субъективно средняя яркость области сцены, обычно соответствует печатной зоне V. Затем он определяет фотографический динамический диапазон. Динамический диапазон – это соотношение яркостей в самой светлой и самой темной областях в сцене. Фотографический динамический диапазон – соотношение яркостей в регионах, где различимы детали. Из-за логарифмической зависимости зон от яркости, можно выразить фотографический динамический диапазон как разность между максимальной и минимально-различимой зоной сцены.

Кроме того, важным является понятие ключа сцены. Ключ сцены изображает общую яркость сцены: сцена субъективно яркая, нормальная или темная. Ретушь - техника проявки, когда часть света удаляется с области снимка в процессе проявки (dodging), или наоборот добавляется больше света (burning). Сначала фотограф определяет освещенность средне-серой поверхности и отображает её в зону 5, отвечающей 18% отраженности снимка. Для сцен с высоким ключом серый будет темным регионом, для сцен с низким – одним из светлых регионов. Выбор зависит от художника, но можно использовать 18%-ную серую карту для автоматизации процесса. Затем фотограф считывает светлые и темные регионы для определения динамческого

диапазона сцены. Если диапазон не превышает 9 зон, то вся сцена со всеми деталями попадет в снимок. Иначе некоторые зоны отобразятся в абсолютно черный или абсолютно белый. Там, где произошла потеря деталей, можно использовать ретушь для внесения локальных изменений. Описанную процедуру сложно автоматизировать, например, непонятен выбор ключа. Тем не менее, субъективных настроек у алгоритма всего несколько. Ключ сцены - средний логарифм яркости.

$$\overline{L_w}(x,y) = \frac{1}{N} e^{sum_{x,y} log(\delta + L_w(x,y)}$$

В этой формуле$\overline{L_w}(x,y)$ – яркость пикселя, суммирование производится по всем пикселям изображения (N – число пикселей), δ- маленькое число для предотвращения сингулярности. Если ключ сцены нормальный, то отобразим полученное значение в 18% (значение ключа a = 0.18), иначе в некоторое другое:

$$L(x,y) = \frac{a}{\overline{L_w}(x,y)} L_w(x,y)$$

Обычно сцены с нормальным диапазоном содержат несколько элементов с высокой яркостью (например, на небе). В традиционной фотографии сжимают обычно и низкие и высокие яркости, но в современной стараются сжать в основном высокие. Пример такого тонального оператора:

$$L_d(x,y) = \frac{L(x,y)}{1+L(x,y)}$$

Эта формула гарантированно отобразит весь имеющийся диапазон, но это не всегда желательно. Иногда удобно выбрать некоторое ограничение сверху (L_{white}), все значения выше которого можно отобразить в белый цвет. Модифицируем формулу:

$$L_d(x,y) = \frac{L(x,y)}{1+L(x,y)}(1 + \frac{L(x,y)}{L_{white}^2})$$

Технику ретуши можно представить как выбор значения ключа для каждого пикселя. Ретушь обычно применяется для региона, ограниченного большим контрастом. Например, черное дерево на светлом фоне. Размер локального региона вычисляется на основе многомасштабной меры локального контраста [14]. Эта функция конструируется для каждого масштаба s и каждого пикселея (x,y) разницей двух гауссиан вида:

$$R_i(x,y,s) = \frac{1}{\pi(\alpha_i s)^2} e^{-\frac{x^2+y^2}{\alpha_i s^2}}$$

Больший гауссиан в 1.6 раз превосходит меньший. Большой гауссиан текущего масштаба становится малым следующего. После свертки изображения с гауссианами, получаются функции отклика V_i:

$$V_i(x, y, s) = L(x, y)R_i(x, y, s)$$

Наименьший Гауссиан будет едва больше пикселя. Интеграция производится в терминах функции ошибки:

$$V(x, y, s) = \frac{V_1(x,y,s) - V_2(x,y,s)}{\frac{2^\phi a}{s^2} + V_1(x,y,s)}, \text{где} a \text{ - ключевое значение}, \phi \text{ - параметр четкости.}$$

Окрестность пикселя – максимальная площадь без резких изменений контраста. Уравнение вычисляется для разных маштабов. Там где градиент яркости мал, мала разница между V1 and V2. Поэтому, для выбора окрестности, V просеивается по порогу для выбора хорошего маштаба s_m. Начиная с наименьшего масштаба, ищем первый масштаб, где выполняется неравенство:

$$|V(x, y, s_m)| < \epsilon$$

Далее используем локальное отображение:

$$L_d(x, y) = \frac{L(x,y)}{1+V_1(x,y,s_m(x,y))}$$

Частотный оператор тональной компрессии

Основная идея частотных операторов состоит в разложении изображения на две компоненты: освещенность и отражательность. Человек более чувствителен к отражательности, поэтому можно сжимать освещенность, а отражательность оставлять без изменений. Далее мы подробнее изложим суть двух работ.

Авторы первой установили ограниченность области применения глобальных операторов тональной компрессии. Они намеренно не включали процессы адаптации и психофизиологические модели в свой оператор, а экспериментировали с пространственно-зависимым оператором. Метод теоретический, работает с физическими яркостями.

Авторы второй работы используют метод сохранения граней на основе билатеральной фильтрации. Базовый уровень (освещенность) получается в результате фильтрации, у неё уменьшается контраст. Итоговый метод получается быстрый, устойчивый. Метод экспериментальный, тональный оператор не пытается моделировать зрение.

Рис. 3.17. Зонная система.

Spatially nonuniform scaling function for high contrast images [15]

Одноканальные изображения, сгененрированные компьютером, домножаются на пространственно-неоднородные коэффициенты для отображения на стандартном CRT-дисплее. Пиксели с одинаковой яркостью на исходном изображении могут иметь разную интенсивность в результирующем изображении. Вопросы адаптации остаются в стороне.

Тестовое изображение – комната, освещенная одной лампой накаливания. Это изображение получается в результате трассировки лучей, также добавляется рассеянная компонента для аппроксимации непрямого освещения. Яркости консистенты с яркостями реальной комнаты. Задача ставится так. Требуется отобразить одно изображение в другое, минимизируя ошибку с точки зрения восприятия.

Изображение реального мира отображает любое значение из отрезка [0;1] в излучательность: $R = r|r : [0,1]^2 \to [0,\infty)$ Изображение реального мира (представленное функцией $F(x,y)$) отображается в растровое изображение P размера $(n_x, n_y)P = 0,\ldots,n_x - 1x0,\ldots,n_y - 1 : F = f|f : P \to [0,\infty)$ Растровое изображение вычисляется из реального изображения с помощью процесса, известного как фильтрация. В компьютерной графике процесс фильтрации обычно сводится к семплированию. Конечный динамический диапазон оборудования требует дальнейшего шага нормализации. Нормализованное множество растровых изображений: $\hat{F} = \hat{f}|\hat{f} : P \to [0,1]$ Дискретная

природа дисплея требует дискретного представления цвета каждого пиксела: $D = d|d : P \to l_0, l_1, \dots l_n$

Далее на экране получается растровое недискретизированное изображение, потому что применяется размытое ядро в результате излучения каждого фосфора. Если бы у нас была мера воспринимаемой близости, то можно было бы просто выбрать по ней ближайший к f элемент $\hat{f}$. Физические ограничения глаза создают артефакты. Внесение этих артифактов в изображение может увеличить реализм. Однако это нужно делать с осторожностью, так как мозг частично компенсирует эти артефакты. Поэтому авторы статьи предлагают не стимулировать локальный контраст, а стимулировать блик, объясняя это тем, что локальный контраст воспроизводим на дисплее. Рассматривается базовый способ отображения с обрезанием высоких яркостей:

$$\hat{f}(i,j) = \begin{cases} \frac{f(i,j)}{I_{clamp}}, f(i,j) < I_{clamp} \\ 1, \text{иначе} \end{cases}$$

Для изображений широкого динамического диапазона изображение либо выглядит слишком темным, если порог отсечения большой, либо возникают артефакты, если маленький. Можно уменьшить артефакты, используя более непрерывное преобразование, но тенденции сохранятся. Поэтому необходимо использовать пространственно-неравномерное отображение:

$$\hat{f}(i,j) = g(i,j,f,f(i,j))$$

Более удобно представить трансформацию как произведение оригинального изображения и коэффициента масштабирования:

$$\hat{f}(i,j) = S(i,j)f(i,j), S(i,j) = \frac{g(i,j,f,f(i,j))}{f(i,j)}$$

Из определения следует, что допустимые функции отображения должны лежать между этими граничными функциями в каждой точке (i,j):

$$S_0(i,j) = 0, S_{max}(i,j) = \frac{1}{f(i,j)}$$

Глаз более чувствителен к локальным изменениям яркости, вызванными локальными изменением отражательной способности объектов при переходе между соседними пикселами изображения, чем к глобальным изменениям яркости всего изображения. В результате этого, слабые пространственные изменения яркости в некоторой мере игнорируются глазом. Это значит, что

Рис. 3.18. Зонная система.

мы можем отобразить изображение с более широким динамическим диапазоном на дисплей с более узким динамическим диапазоном. Поэтому, если S имеет слабый градиент, то результирующее изображение не должно слишком отличаться от исходного. S выбирается пропорциональным обратному размытому изображению f:

$$S(i,j) = \frac{1}{kf_{blur}(i,j)}$$

Далее авторы специфицирует точный вид фильтра и значение константы на основе экспериментальных данных. После этого у них получаются изображения, содержащие темный ореол вокруг источника освещения (Рис. 3.18).

Для его компенсации они предлагают моделировать свечение вокруг источника (blooming). Оно наблюдается у объектов с высокой интенсивностью в результате преломления в глазу. Монитор не может создать достаточную яркость, чтобы воспроизвести этот эффект.

$$\gamma(i,j) = \begin{cases} k, i = j = 0 \\ \frac{1-k}{I} F(i,j), \sqrt{i^2+j^2} <= \frac{w}{2} \\ 0, \text{иначе} \end{cases}$$

W – ширина фильтра

$$F(i,j) = |\sqrt{i^2+j^2} - \tfrac{w}{2}|^n, n > 1$$

$$I = -F(i,j) + \sum_{i=-\frac{w}{2}}^{\frac{w}{2}} \sum_{j=-\frac{w}{2}}^{\frac{w}{2}} F(i,j)$$

N – контролируемый параметр.

Разумеется, при этом могут быть ошибки в изображениях с высокоинтенсивными не источниками, а отражателями. Однако, если метод применяется вместе с синтезом изображений широкого динамического диапазона, то можно одназначно установить, где находится источник и создать там эффект свечения.

Fast Bilateral Filtering for the Display of High-Dynamic-Range Images[16]

Метод основан на декомпозиции изображения на 2 масштаба. Базовый уровень кодирует изменения большого масштаба. Уровень детализации кодирует малые изменения. Только базовый уровень испытывает уменьшение контраста, поэтому сохраняются детали. Базовый уровень находится с использованием билатерального фильтра, сохраняющего границы. Это нелинейный фильтр, вес каждого пикселя вычисляется с использованием фильтра Гаусса, умноженного на функцию влияния (действует в пространстве яркости), что увеличивает вес пикселей с большой разницей интенсивности. Затем фильтрация ускоряется с помощью кусочной аппроксимации в области интенсивности и подходящим семплированием, благодаря этому достагается оптимизация в 2 порядка. Быстрый метод, не требует настройки параметров.

Преложенный метод связан с анизотропной диффузией. Размытие по Гауссу может быть интерпретирвано в качестве решения уровнения теплопроводности: $\frac{\delta I}{\delta t} = -\Delta I$. То есть, интенсивность каждого пикселя рассматривается как тепло и распространяется в течение времени к своим 4 соседям. Функция краевой остановки g изменяет проводимость в зависимости от градиента изображения. Это предотвращает прохождение теплового потока через края:

$$\frac{\delta I}{\delta t} = -div[g(||\nabla I||)\nabla I]$$

Ранее другими исследователями было предложено 2 выражения для функции краевой остановки:

$$g_1(x) = \frac{1}{1+\frac{x^2}{\sigma^2}}\ g_2(x) = e^{-\frac{x^2}{\sigma^2}}$$

Где σ – параметр масштабирования в пространстве интенсивности. Специфицирует, какая величина градиента должна остановить диффузию. Дискретное уравнение диффузии:

$$I_s^{t+1} = I_s^t + \frac{\lambda}{4} \sum_{p \in neighb_4(s)} g(I_p^t - I_s^t)(I_p^t - I_s^t)$$

T – дискретный шаг по времени. λ –скаляр, определяющий уровень диффузии. Это популярное средство сохранения граней, но очень медленное из-за дискретного процесса, результат зависит от времени останова, так как диффузия сходится к равномерному изображению. Билатеральная фильтрация используется как альтернатива. Это нелинейный фильтр, где выход – взвешенное среднее входов. Вес пикселя завивит также от функции g в пространстве интенсивности, которая уменьшает вес пикселей с большой разницей интенсивности (аналог функции остновки в анизотропной диффузии):

$$J_s = \frac{1}{k(s)} \sum_{p \in \Omega} f(p-s) g(I_p - I_s) I_p, \text{где} k(s) \breve{} \text{нормализующий член} : \\ k(s) = \sum_{p \in \Omega} f(p-s) g(I_p - I_s)$$

Уменьшение контраста с сохранением границ может порождать артефакты гало для четких границ из-за вспышек у высококонтрастных границ. Авторы предлагают такое объяснение. Артефакты возникают в тех местах, где недостаточно информации о соседях для отделения черт большого масштаба и малого масштаба. Избежать этой проблемы помогает нормализующий фактор k. Его можно использовать для определения пикселей, которые нужно исправить. Единственные контролируемые пользователем параметры метода – это общая яркость и базовый контраст. Хотя предоставляются и автоматические значения. Благодаря устранению эффекта гало, это один из самых хороших методов. Примеры работы дан на Рис. 3.19.

Градиентный тональный оператор. Gradient Domain High Dynamic Range Compression[17]

Этот метод концептуально простой, эффективный устойчивый. Увеличивается амплитуда больших градиентов. Новое изображение с более низким динамическим диапазоном получается путем решения уравнения Пуассона для модифицированного поля градиентов. Результаты демонстрируют, что метод способен значительно уменьшить динамический диапазон и сохранить детали, отсутствуют такие дефекты как гало, инверсия градиента, потеря локального контраста. Также метод значительно улучшает обычные изображения, привнося детали в темные регионы. Получаются хорошие результаты для панорамных видео-мозаик, обычных высококонтрастных фотографий и медицинских изображений. Идея метода похожа на идею частотных методов. Зрение чувствительно не столько к абсолютной яркости, сколько к отношению локальных интенсивностей, что уменьшает эффект больших глобальных изменений, ассоциируемых с разницей в освещении. Изменение яркости на

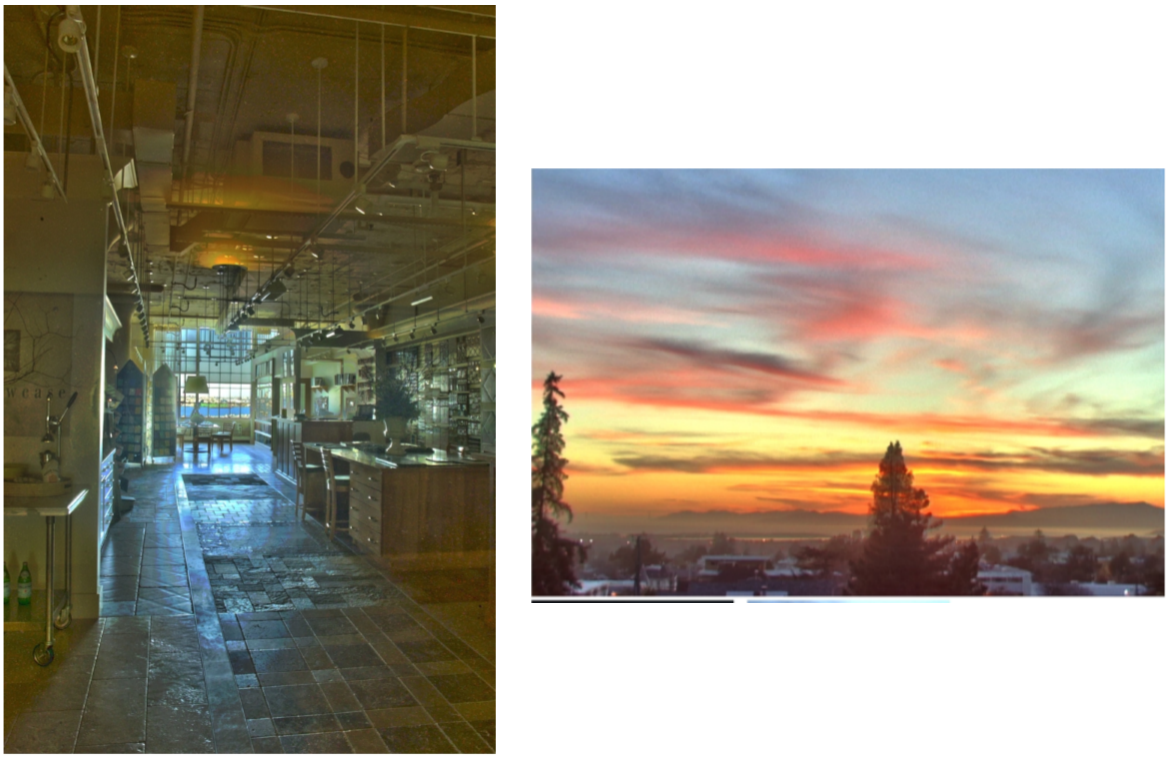


Рис. 3.19. Примеры работы алгоритма.

Рис. 3.20. Примеры работы градиентного тонального оператора.

HDRI увеличивает магнитуду градиента яркости для некоторого масштаба. Четкие детали, такие так текстура, соответсвуют градиентам меньшей магнитуды. Идея в том, чтобы идентифицироваь большие изменения градиента больших масштабов и увеличить их магнитуду, сохранив их направление. Уменьшенное HDRI реконструируется из измененного поля градиентов. Вычисления производятся с логарифмом освещенности.

Другие пространственно-зависимые операторы

Мы рассмотрели всего лишь несколько методов. Здесь лишь будут упомянуты некоторые другие значительные идеи и статьи. Schlick [18]– пошел путем увеличения эффективности вычислений и упрощения параметров для настройки. Он использовал полином первой степени для тональной компрессии и предложил 3 метода для моделирования локальной адаптации.

Jobson et al. [19] презентовали метод сжатия динамического диапазона на основе многокомпонентой версии теории цветного видения “retinex”. Ретинекс рассчитывает отражательную компоненту как отношение изображения к его профильтрованной низко-частотным фильтром версии. Таким образом в статье была представлена многомасштабная версия алгоритма для одновременного сжатия диамического диапазона и сохранения цвета. Но для проверки они использовали обычные изображения узкого динамического диапазона. Также были проблемы для сцен с доминантным цветом, так как ретинекс использует предположение о «сером мире».

Pattanaik, Ferwerda, Fairchild and Greenberg [20] изобрели технику многомасштабного представления паттерна, яркости и цвета в зрительной системе человека и адресовали проблему сжатия и восприятия сцен на пороговом и надпороговом уровне. Они обеспечили вычислительную модель адаптации и пространственного зрения для реалистичного тон-маппинга. В этой модели есть 2 основные части: модель зрения, обрабатывающая входное изображение для кодирования воспринимаемого контраста для для хроматических и ахроматического каналов и их частотный механизм; и модель дисплея, которая принимает закодированную входную информацию и реконструирует изображение. Модель зрительной системы из этой статьи может быть использована в других областях, таких как метрики качества изображений, методы сжатия и основанные на восприятии алгоритмы синтеза.

В 1999 Tumblin and Turk [21] предложили метод LCIS(Low Curvature Image Simplifier). Метод разделяет изображение на крупные черты и мелкие детали. Идея проистекает из искусства, где начальный набросок описывает общую структуру картины, а детали и освещение заполняются позднее. The LCIS использует вариант анизотропной диффузии для определения четких деталей (границ) и плавного затенения. Эта техника позволяет значительно сжать динамический диапазон, сохраняя четкие детали. При этом недостаток в том, что детали иногда слишком сильно выражены, а также в методе более 8 параметров, поэтому сложно их подобрать правильно.

Exposure Fusion

Мы рассмотрели алгоритмы синтеза и визуализации HDR-изображений. При последовательном применении алгоритмов синтеза HDR-изображения и алгоритмов тональной компрессии из набора LDR-изображений с разной выдержкой можно получить другое LDR-изображение, максимально приближенное к тому, что видит человек.

Существует также техника техника объединения изображений с разной выдержкой минуя непосредственно построение HDR –изображения. Она носит название смешивания экспозиций (Exposure Fusion) [Mertens, Tom; Kautz, Jan; Van Reeth, Frank (2007), "Exposure Fusion Pacific Graphics, retrieved 2011-01-21].

При этом не делается никаких предположений о времени выдержки каждого кадра, не требуется калибровка кривой отклика камеры. Недостатком алгоритма является узкая область его применимости по сравнению с техниками тональной-компрессии. Его можно использовать для визуализации фотографий, но нельзя использовать при расчетах или при синтезе виртуальной сцены.

В основе техники смешивания экспозиций лежит мера насыщенности или контраста, в соответствии с которой взвешиваются значения яркости в изображениях с разной выдержкой. Каждому пикселю p каждого элемента i в наборе N изображений сопоставляется весовое значение W_I^p. Результат получается согласно следующей формуле:

$$R^p = \frac{\sum_{i=1}^{N} W_i^p * I_i^p}{\sum_{i=1}^{N} W_i^p}$$

Однако попиксельное смешивание может привести к артефактам на финальном изображении, поэтому обычно используются некоторые модификации, например, весовые функции слегка размываются. Однако это приводит к ореолам. Более хороший подход - смешивание коэффициентов изображения в пирамиде лапласиан.

Пирамида Гаусса – набор уменьшенных копий одного и того же изображения, каждое последующее в 2 раза меньше предыдущего по ширине и высоте. Пирамида Лапласа – набор разностей исходного изображения и его уменьшенной, а затем вновь увеличенной копии (см. рисунок 3.21). Каждый новый уровень пирамиды содержит более низкие частоты, чем предыдущий. Для весовых функций строится пирамида Гаусса, для исходных изображений – пирамида Лапласа. Затем на каждом уровне элементы пирамиды Лапласа изображений смешиваются при помощи элементов пирамид Гаусса весовых функций. Таким образом, получается избежать как неравномерности итогового результата, так и артефактов размытия и ореолов, которые получаются в результате размытия весовых функций.

Пример работы алгоритма смешения экспозиций представлен на рисунке 3.22.

3.4.4. Отображение на дисплеях с широким динамическим диапазоном

Идея дисплея с широкии динамическим диапазоном в том, чтобы осветить каждый маленький участок изображения, отображаемый на LCD источником LED со специальной излучательностью для этого участка. Это значит, что если сцена содержит черные детали, нужно выключить LED для получения истинно черного цвета. Если же яркость области высока, то нужно включить подсчетку на максимальную мощность. Серые оттенки получается с промежуточными интенсивностями LED.

Теоретический базис для работы с таким дисплеем дается в работе Cohen, Tchou, Hawkins and Debevec [22]. Авторы решают проблему отображения хранением и визуализацией текстурных карт в реальном времени с использованием аппаратных возможностей. В их методе текстурные карты хранятся

Рис. 3.21. Иллюстрация пирамиды Гауссиа (слева) и пирамиды Лапласа (справа). Яркость элементов пирамиды Гаусса усилена в 10 раз.

Рис. 3.22. Слева - набор изображений с разной выдержкой, справа - результат алгоритма смешения экспозиций.

в форме двух различных восьмибитных текстурных карт, одна предсталяет высокие интенсивности, а другая – низкие. В процессе отображения, они рекомбинируются для динамической настройки экспозиции.

Таким образом текстурная карта состоит из:

- изображения для передачи LCD
- изображения для передачи LED

Производители мониторов уверяют, что результат чрезвычайно высоко контрастен для наблюдателя, с высоко интенсивным белым и полностью темным черным. Однако пока такие дисплеи не дошли до массового потребителя. Методы тональной компрессии, напротив, сильно развились. Существует множество программ, позволяющих конвертировать изображения с широким динамическим диапазоном в изображения с узким динамическим диапазоном. Кроме того, печатный диапазон по-прежнему ограничен, поэтому полезно иметь представление об описанных выше алгоримах.

Глава 4.

Метод излучательности

В физике излучательность (Radiosity) по определению - это энергия, покидающая поверхность на единицу площади и в единицу времени. Изучательность представляет собой отношение полного излучения участка поверхности к площади данного участка. Фактически это единица, обратная к освещенности, т.е. это плотность полного потока, покидающего поверхность. Данная величина обозначается буквой B (в русской литературе иногда обозначается как M и называется Энергетической светимостью), и измеряется в Вт/м^2.

Алгоритм расчёта освещения, называемый также словом - «излучательность» исходит в своей основе из закона сохранения энергии и предположения «энергетического равновесия». Термин «энергетическое равновесие в системе» 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. Можно сказать, что все современные методы вычисляющие глобальное освещение исходят из предположения энергетического равновесия.

Излучательность - один из старейших алгоритмов решения проблемы глобальной освещённости. Однако и на сегодняшний день он остаётся актуальным методом при приближённом расчёте глобального освещения для интерактивных приложений и компьютерных игр (рис. 4.1), успешно конкурируя с более новыми методами расчёта глобальной освещенности - такими как Light Propagation Volumes (LPV) [15] и Voxel Cone Tracing (VCT) [16].

Рис. 4.1. Использование излучательности в игровом движке Frost Byte 2. Изображение A показывает расчитанное освещение на площадках (*патчей*) и упрощённой геометрии. Изображение B - финальный результат комбинирования освещения и детализированную геометрию.

При определённых условиях (упрощение геометри до нескольких тысяч площадок) и/или правильной реализации (иерархическое представление площадок), излучательность позволяет рассчитать вторичную освещённость с достаточно высокой точностью всего за несколько миллионов сложений и обращений в память. Ни один другой метод расчёта глобальной освещённости не может похвастаться такой дешевизной в вычислительном плане.

4.1. Основная идея метода излучательности

Основная идея метода излучательности заключается в том, чтобы перейти от непрерывного решения уравнения глобальной освещённости к дискретному, и вычислять освещённость, рассматривая взаимодействие относительно-небольшого числа площадок конечного размера (4.2).

При этом можно показать (приложение A), насколько сильно упростится задача расчета интеграла при сведении решения данного уравнения с интегралом к решению системы линейных уравнений. *Алгоритм излучательности создает набор линейных уравнений, связывающих исходящую энергии каждого патча с энергией, приходящей на каждый патч.*

Метод излучательности не решает непосредственно задачу нахождения цвета пикселя, его главная цель - *решение задачи нахождения цветов поверхностей обьектов в пространстве* (при этом требуется нахождение цветов даже тех поверхностей, которые не будут видны на синтезированном изображении).После расчета излучательности получаемую картинку можно восстановить методом обратной трассировки или методом растеризации (проекций площадок на экранную плоскость).

Рис. 4.2. Основная идея метода излучательности [17]. Слева показано разбиение геометрии на площадки. Справа - финальный результат расчёта освещенности. Только вторичная освещенность была рассчитана при помощи метода излучательности на данном изображении. Обратите внимание, что достаточно чёткая тень от мусорного ведра на правом изображении представлена всего 4 площадками на левом изображении.

Чистая трассировка лучей, как правило, дает резкие тени от объектов, что может быть как правильным, так и ошибочным. Метод излучательности практически всегда выдает на выход изображение с мягкими тенями (рисунок 4.3), что также может быть ошибочно (например, тень от стула должна быть скорее резкой, чем мягкой). Поэтому излучательность стараются применять в основном при расчёте вторичной освещенности, которая почти всега меняется плавно и лишена резких переходов. Прямую освещённость и отражения можно считать при помощи трассировки лучей. Такой метод применяется в рендере-по-умолчанию (Default Scan-Line Render) для Autodesk 3D Studio Max.

Рис. 4.3. Слева - изображение, синтезированное методом трассировки лучей, справа - изображение, синтезированное с применением метода излучательности.

4.2. Излучательности «на пальцах»

Рассмотрим простой пример (комнату из 4 стен в 2D, рисунок 4.4). Предположим, что одна из стен излучила некоторое количество энергии. 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.

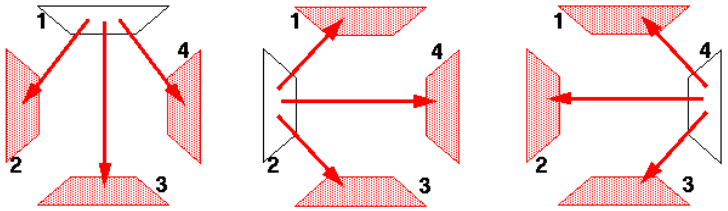


Рис. 4.4. Простая 2D комната из 4 стен.

Для корректного расчета итогового цветного изображения необходимо знать спектры поглощения всех материалов (поверхностей). В простейшем случае, необходимо повторить все расчёты для 3 классических цветовых компонент - RGB.

Стоит подчеркнуть, что несмотря на достаточно объёмный материал в данной главе метод излучательности чрезвычайно прост. Руководствуясь описанными выше соображениями не трудно прийти к алгоритму из раздела 4.4. Для того чтобы это сделать осталось только ввести *форм-факторы* - коэффициенты, указывающие какая часть энергии при излучении с одной площадки переходит на другую. *Форм-Фактор (F_{ij}) - это часть всей энергии, покинувшей площадку P_i и пришедшей на площадку P_j*. Алгоритм из раздела 4.4, состоящий всего из 3 вложенных циклов, просто моделирует описанный выше процесс.

4.3. Ограничения метода излучательности

Метод излучательности основывается на следующих ограничениях:

1) Все поверхности ламбертовы - т.е. идеально диффузные, равномерно отражающие энергию по всем направлениям. Излучательность рассматривает только Ламбертовскую модель отражения.

2) Все поверхности разбиваются на площадки (патчи) с конечной площадью.

3) Все точки одной площадки имеют одинаковую яркость.

4) Расчет переноса энергии происходит для замкнутой системы, т.е. вся энергия ушедшая с одной площадки должна без потерь распределиться между остальными площадками сцены.

Например, при моделировании комнаты с окном, согласно данному предположению, окно представляется набором площадок с заданной начальной излучательностью и нулевыми отражающими свойствами (чёрный ламбертовский материал). Если свет через окно не поступает (ночью), начальная излучательность площадок окна также будет равна нулю. Нулевые отражающие свойства моделирует процесс ухода света из комнаты через окно в открытое пространство, не нарушая требования замкнутости системы.

Перечисленные нами предположения, как уже было сказано, позволяют преобразовать уравнение освещенности к линейному виду. Алгоритм излучательности через СЛАУ рассмотрен в приложении А. Далее рассмотрим более простой в реализации прямой алгоритм рассчёта.

4.4. Прямой расчёт излучательности

Алгоритм прямого расчёта достаточно прост. На каждой итерации для каждой площадки собирается вся входящая энергия. В конце итерации она умножается на коэфифциент отражения и превращается в исходящую энергию для следующей итерации. На данном этапе мы считаем что все форм-факторы известны, то есть для каждой площадки известно, в каком соотношении излучённая с неё энергия распределяется между остальными площадками.

```
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);

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

for (int pass = 0; pass < a_numPasses; pass++) // do radiosity
{
  // each patch collects light from the scene
  for (int i = 0; i < patchesR.size(); j++)
  {
    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;
  }
}
```

Листинг 4.1. Алгоритм прямого расчёта излучательности

4.5. Решение СЛАУ

Имея матрицу форм-факторов (формула 4.1), можно перейти (приложение А) к СЛАУ (формула 4.2).

$$\begin{pmatrix} F_{11} & F_{12} & \cdots & F_{1,N-1} & F_{1N} \\ F_{21} & F_{22} & \cdots & F_{2,N-1} & F_{2N} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ F_{N1} & F_{N2} & \cdots & F_{N,N-1} & F_{NN} \end{pmatrix} \tag{4.1}$$

$$\begin{pmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ -\rho_N F_{N1} & \cdots & \cdots & 1-\rho_N F_{NN} \end{pmatrix} \underbrace{\begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{pmatrix}}_{indirect} = \underbrace{\begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{pmatrix}}_{emission} \tag{4.2}$$

В приведённом уравнении ρ_i - цвет i-ой площадки. Отметим, что в отличие от прямого расчёта, решение СЛАУ позволяет получить все переотражения сразу. Чтобы эффективно решать СЛАУ, можно использовать LU-разложение, получая, таким образом, решение за $n^2 + O(n)$ арифметических операций.

Альтернативным методом, позволяющим получить решение уравнения излучательности за $n^2+O(n)$ является предварительный расчёт цветной матрицы форм-факторов F' такой что:

$$\underbrace{\begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{pmatrix}}_{indirect} = \underbrace{\begin{pmatrix} F'_{11} & F'_{12} & \cdots & F'_{1N} \\ F'_{21} & F'_{22} & \cdots & F'_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ F'_{N1} & F'_{N2} & \cdots & F'_{NN} \end{pmatrix}}_{F'} \cdot \underbrace{\begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{pmatrix}}_{emission} \tag{4.3}$$

Это можно сделать на основе цветной матрицы форм-факторов F^C, состоящей из элементов $F^C_{ij} = F_{ij}\rho_j$, пред-рассчитав F' как матричный полином [18]:

$$F' \approx F^C \cdot emission + F^C \cdot F^C \cdot emission + \cdots + \left(F^C\right)^k \cdot emission$$

4.6. Расчет форм-факторов

4.6.1. Определение Форм-Фактора

Мы уже встречались с термином форм-фактора выше. Введем понятие форм-фактора, необходимого для наших расчетов. Напомним, что *Форм-Фактор (F_{ij}) - это часть всей энергии, покинувшей площадку P_i и пришедшей на площадку P_j*. Сумма всех F_{ij} в строчке или столбце должна быть равна единице, т.к. энергия покидающая некоторую площадку i перераспределяется

между всеми остальными площадками. Аналогично, вся приходящая энергия на некоторую площадку складывается из энергий, излучённых другими площадками.

Отметим, что значение форм-фактора зависит исключительно от взаимного расположения площадок. Процесс вычисления форм-факторов может проходить отдельно от процесса рендеринга в совокупности с расчетом геометрии.

4.6.2. Дифференциальный Форм-Фактор

При рассмотрении двух бесконечно малых дифференциальных площадок (будем считать, что площадки обладают плоскостью и ориентацией, но не имеют конкретной формы), Форм-Фактор будет включать в себя $\cos\theta_i$ - косинус угла между нормалью к одной площадке и направлением на другую площадку, $\cos\theta_j$ - аналогично, косинус угла между направлением на площадку i и нормалью к площадке j, r^2 - вектор направления между двумя дифференциальными площадками.

4.6.3. Взаимное расположение патчей

Дифференциальный Форм-Фактор для двух площадок i и j (рисунок 4.5) записывается следующим образом:

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

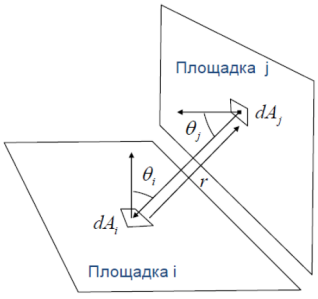


Рис. 4.5. Взаимное расположение площадок.

4.6.4. Полный Форм-Фактор

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:

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

Таким образом, мы сделали переход от всех точек сцены к форм-фактору между двумя конкретными площадками.

4.6.5. Свойства форм-фактора

Перечислим основные свойства форм-фактора:

- **Зависимость только 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.
- **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.
- **Сумма всех возможных Форм-Факторов для каждого патча сцены равна единице.** Вся энергия, покидающая патч, должна прийти на некоторый другой патч.

4.6.6. Расчёт форм-факторов. Метод площадной дискретизации (бросание лучей)

Метод площадной дискретизации представляет собой численное вычисление интеграла по поверхности патча (рисунок 4.5) при помощи метода Монте-Карло. Мы рассмотрим его более подробно в 8 главе, а сейчас приведём выражения для вычисления одномерного определённого интеграла (выражение

4.4) и двуменрого площадного интеграла стоящего в выражении форм фактора (выражение 4.5, рисунок 4.5)

$$\int_a^b f(x)dx \approx \frac{b-a}{N}\sum_{i=1}^{N} f(u_i) \tag{4.4}$$

$$\int\limits_{A_i}\int\limits_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} V_{ij} dA_i dA_j \approx \frac{A_i A_j}{N\pi}\sum_{k=1}^{N} \frac{\cos\theta_i(u_k)\cos\theta_j(u_k)V(u_k)}{(r(u_k))^2} \tag{4.5}$$

Можно представить процесс вычисления двойного итеграла следующим образом: площадки A_i и A_j разбивается на множество небольших участков, каждый из которых считается дифференциальным. Затем из каждого дифференциального участка dA_i бросается 1 луч в дифференциальный участок dA_j (рис. 4.5). В случае попадания (отсутствия пересечений луча с другими объектами, расположенными между площадками dA_i и dA_j), данный дифференциальный форм-фактор добавляется в конечный форм-фактор (значение $V(u_k)$ не равно нулю). Повторяя данную процедуру для всех дифференциальных площадок A_j и A_i, можно получить полный Форм-Фактор для пары патчей A_i и A_j.

В действительности метод Монте-Карло работает немного проще. Для каждой Монте-Карло выборки (которых ровно N штук, выражение 4.5) необходимо случайно выбрать 2 точки - одну точку на площадке A_i, вторую на A_j. После необходимо соединить эти 2 точки лучом и вычислить выражение, стоящее под знаком суммы (выражение 4.5, рисунок 4.5).

Данный подход является одним из самых точных, но и наиболее вычислительно-трудоёмких методов. Проблема данного метода заключается в том, что для его реализации требуется явная трассировка лучей и большое число выборок (N от нескольких сотен до нескольких тысяч). Реализация данного подхода может быть целесообразной при наличии эффективного инструмента, позволяющего с высокой скоростью находить пересечения лучей с объектами сцены. Более быстрые методы рассмотрены в приложении В.

4.7. Метод излучательности, итоги

Подведем итоги рассмотрения классического метода расчета излучательности. Кратко перечислим этапы, из которых складывается процесс расчета излучательности сцены.

1) На первом этапе вся геометрия сцены разбивается на площадки. В случае, если поверхность в программе задана аналитически, необходимо преобразовать аналитическое представление в патчи (площадки).

2) После того, как геометрическая информация обо всех площадках сцены получена, запускается процесс расчета форм-факторов для каждой пары выбранных площадок.

3) На следующем этапе, после расчета всех Форм-Факторов, проводится решение записанной СЛАУ, которая в качестве решения выдаст излучательность каждой из рассматриваемых площадок. Вместо решения СЛАУ может применяться так называемый прямой расчёт (алгоритм из раздела 4.4).

4) На заключительном этапе метода происходит визуализация полученного решения на экране на основе рассчитанных излучательностей. Фактически на данном этапе нужно осуществить растеризацию площадок - проекцию всех площадок на экран с учётом их перекрытия их друг другом.

4.7.1. Недостатки классического метода излучательности

Отметим основные проблемы классического метода излучательности (naive radiosity) и основные способы их решения:

1) Потенциально большое число площадок; вследствие квадратичной сложности алгоритма расчёта освещения излучательность теряет своё главное преимущество - вычислительную простоту. Решение: Иерархическое подразбиение патчей, возможность рассматривать далёкие группы патчей как 1 большой патч.

2) Неэффективный расчёт освещения с резкими переходами (например резкие тени). Решение: использовать излучательность только для расчёта вторичного освещения, а первичное считать другими методами. Иерархическое подразбиение в местах быстрых изменений освещённости до некоторой степени позволяет решить проблему. Следует отметить, однако, что такое решение имеет другой недостаток. Если в классическом методе излучательности набор патчей был фиксирован и все форм-факторы посчитаны заранее, при подразбиении в местах резких перепадов освещённости форм-факторы придётся пересчитывать. Таким образом, уходит главное преимущество излучательности - перенос вычислительно сложной части на этап пред-расчёта.

3) Долгий расчёт форм-факторов. Несмотря на то, что расчёт форм-факторов может быть выполнен 1 раз для фиксированной геометрии сцены, пожалуй, эта проблема в излучательности стоит наиболее остро, поскольку в методе излучательности вся вычислительная сложность расчёта освещения переносится на этап предрасчёта форм-факторов. Решение: более быстрые методы вычисления форм-факторов, применение графической аппаратуры.

Глава 5.

Расчёт цвета пиксела. Основы фотографической оптики.

5.1. Введение

Для создания реалистичных изображений нужно не только уметь рассчитывать яркость точек трёхмерной сцены, но и понимать, как получается цвет пиксела из световой энергии. Что делать с энергией, которая попала в пиксел, как преобразоывавать её в цвет, в разные цветовые пространства? Промежуточный этап между попаданием энергии в оптическую систему фотоаппарата и этапом формирования изображения следует рассмотреть более подробно.

5.2. Уравнение измерений и уравнение освещенности

Вспомним материал предыдущих глав, и заодно перейдем к решению общей задачи рендеринга. Ранее уже приводилась формула, позволяющая рассчитать цвет конкретного пикселя для случая когда мы игнорируем дефокус (т.е. в предположении, что глубина резкости бесконечна, рис. 5.1):

$$M_j = \int_{380}^{780} \int_I W_{i,\lambda}(x) L_i(x) \frac{\pi}{4} n^2 dA_x d\lambda \tag{5.1}$$

$$n = \frac{f}{D} \tag{5.2}$$

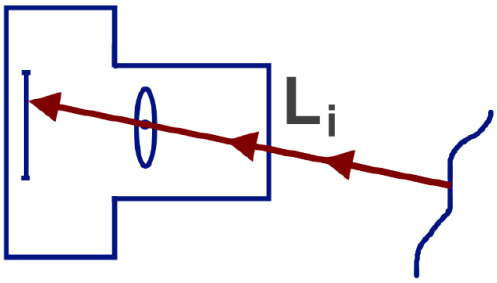


Рис. 5.1. Попадание изображения на диафрагму. Бесконечная глубина резкости.

Пусть у нас есть некоторая матрица, есть некоторая аппаратная система и есть некоторая оптическая система, обладающая диафрагмой, т.е. геометрией. Нам известно фокусное расстояние 5.10 данной системы, что позволяет нам узнать, каким образом данная система будет преобразовывать элементы видимой сцены, т.е. насколько сильно она будет масштабировать элемент виртуального пространства в проекцию на матрице.

Для определения данного преобразования рассматриваемой системы рассмотрим следующую ситуацию. Пусть есть наблюдатель в некоторой точке C и точка p излучает в его направлении энергию с некоторой яркостью, воспринимаемой наблюдателем. Данную яркость обозначим за L_i.

Также известны некоторые кривые устройства измерения, определяющие чувствительность данного устройства к разным длинам волн (рис. 5.3). Согласно уже упомянутому в предыдущей главе, при использовании виртуальной камеры данные кривые можно взять равными кривым для цветового пространства XYZ или для пространства RGB и с их помощью рассчитать M_j - вектор из трех цветовых компонент. Множитель n_2 под знаком интеграла нужен для коррекции изменения яркости, возникающего из-за размера оптической системы. Чем меньше n, тем темнее будет результат (итоговое изображение). Интегрирование происходит по площади нашего пикселя и по видимым длинам волн (рис. 5.2).

Проблема подобного расчета состоит в следующем: не принимается во внимание тот факт, что на освещенность некоторого пиксела свето-чувствительной матрицы может влиять не только точка 3D сцены, идеально сфокусированая на данном пикселе, но и соседние, не сфокусированные точки трёхмерной сцены. В частности, если рассматривать некоторую часть лучей, которая фокусируется не в той точке, которую мы хотим увидеть, подобная система создаст несфокусированное пятно на матрице, т.е. освещенность каждого пикселя будет задаваться не только теми точками, которые на него сфокусировались (5.2).

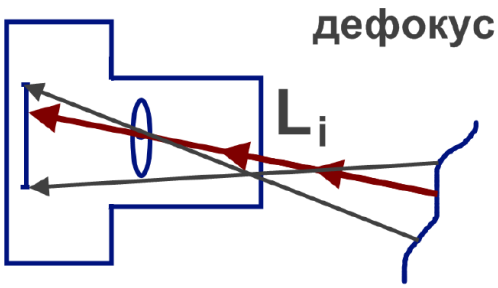


Рис. 5.2. Попадание изображения на диафрагму. Случай дефокуса.

5.3. Модель матрицы камеры

Рассмотрим следующую модель. Каждый пиксел – это некоторый датчик, который чувствителен ко всем длинам волн λ (колбочки-палочки в человеческом глазу – пример такого датчика). Датчик обладает некоторой площадью I. Мы имеем некоторое изображение конечного размера и можем считать, что оно представляет собой находящуюся где-то в пространстве чувствительную площадку, которая разбита на сектора-датчики с некоторой площадью. Датчик таков, что при облучении его всем спектром на выходе получится только одно скалярное значение. Он по-разному чувствителен в разных частях спектра, и это задается функцией спектральной чувствительности (рис. 5.3).

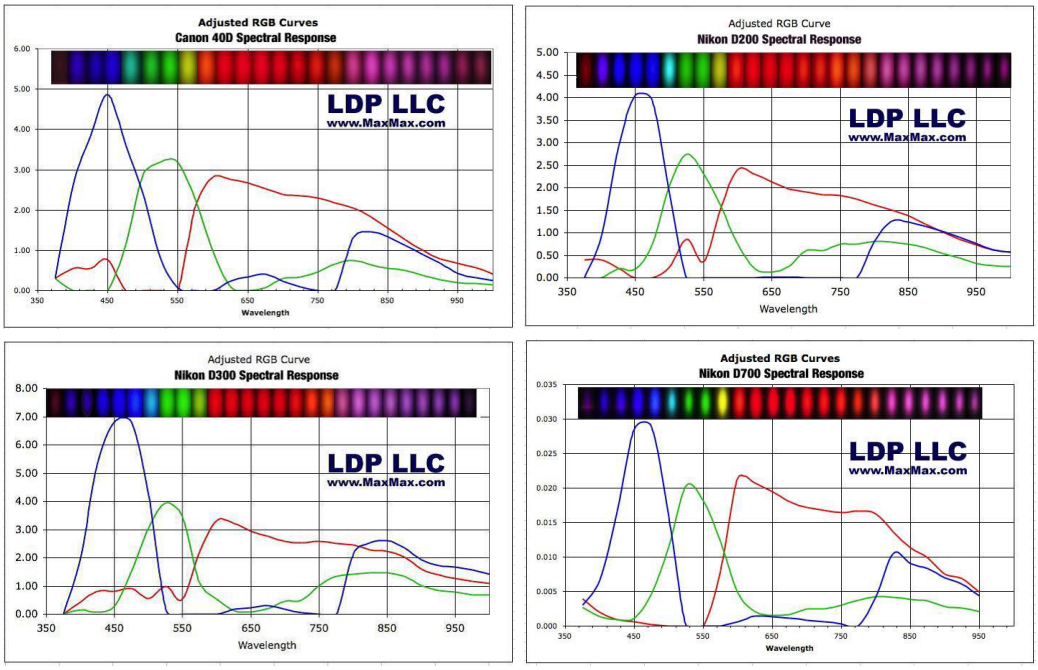


Рис. 5.3. Примеры кривых спектральной чувствительности для разных фотокамер.

Для каждой камеры на рисунке 5.3 показаны 3 кривые. Дело в том, что в фотокамерах установлен набор датчиков, которые по-разному реагируют на разные части спектра (по аналогии с человеческим зрением). Из графиков видно, что в разных камерах используются разные матрицы, которые обладают разной чувствительностью. Например, Nikon D200 и Nikon D700: видно, что они обладают похожей формой, но у одного из датчиков явно другая спектральная чувствительность. Каждый датчик по сути интегрирует произведение спектра всего попавшего на него излучения на свою кривую чувствительности и выдаёт получившееся скалярное значение в качестве результата. Излучение, попавшее на датчик, интегрируется по площади этого датчика.

В нашей модели отклик сенсора пропорционален степени его освещённости. На практике это действительно так. Кроме того, отклик зависит от времени облучения, но пока это не будет рассматриваться.

При условии линейной зависимости отклика датчиков от яркости, формируется некотрое линейное видимое цветовое пространство. Для каждого спектра мы получаем 3 числа, причем из свойства линейности следует, что для двух разных источника света, освещающих набор датчиков, можно сложить отклики соответствующих датчиков на каждый источник света в отдельности, и получить отклик на комбинацию этих источников света. Если мы уве-

личим в N раз мощность излучениия источника света (пропорционально на всех длинах волн), то в N раз увеличится полученный вектор откликов. Вектор откликов задаёт некоторый цвет (для камеры). Этот цвет может не иметь никакого отношения к тому, что видит наш глаз. Например, камера может видеть инфракрасный цвет, или какой-нибудь другой, это «псевдоцвета». Наша задача – сделать эти цвета такими, какими их видит человек. Поэтому в некотором смысле все эти датчики (см графики) должны быть максимально скоррелированы с учетом особенностей человеческого глаза. Они должны получать приблизительно ту же смесь цветов, которую видит человек.

В реальном фотоаппарате находится массив датчиков (рисунок 5.4). Разными цветами на рисунке ниже обозначены датчики, обладающие разной чувствительностью к разным частям спектра (так называемая ***Байесовская мозаика***).

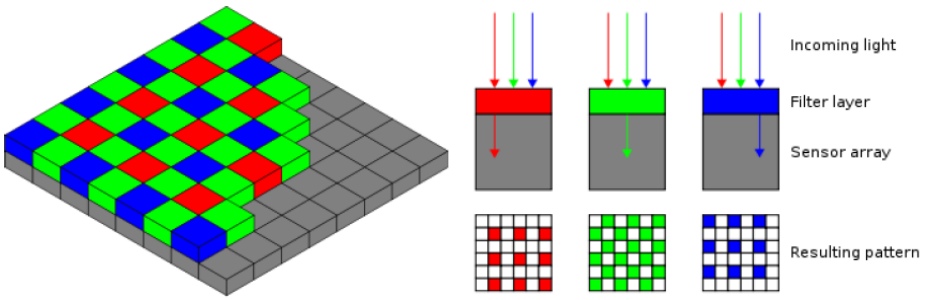
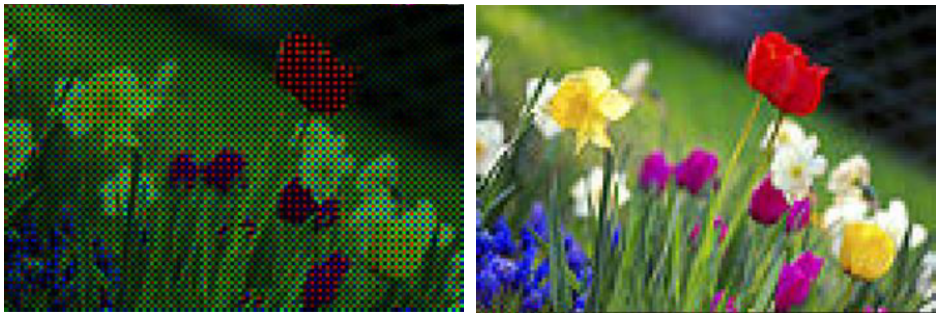


Рис. 5.4. Байесовская мозаика на массиве датчиков фотоаппарата.

Они реагируют на свет по-разному, создавая картинку на рисунке 5.5 слева. Это то изображение, которое дает на выходе матрица камеры.



изображений на матрице датчиков преобразованное изображение

Рис. 5.5. Изображение на матрице и преобразованное при помощи Байесовского фильтра.

Затем это изображение необходимо преобразовать в обычное изображение (рис. 5.5 справа) путем соответствующих фильтраций с помощью так называемого Байесовского фильтра. Есть разные алгоритмы того, как это сделать,

например, с помощью интерполяции между соседними элементами, или так, чтобы несколько соседних пикселов образовали в итоге один результирующий пиксел изображения.

Таким образом, три датчика дали нам картинку, где каждый канал – это один канал цветового пространства камеры. Затем этот вид преобразуется в пространство XYZ, его уже можно преобразовать к любому RGB-подобному пространству с помощью линейного преобразования. А это всё, что необходимо, чтобы сохранить данное изображение.

Если перед нами стоит задача рендеринга, т.е. никакой реальной сцены и исходного изображения с камеры нет, то мы моделируем каким-то образом процесс распределения света. При этом нам не нужно думать о каких-то физических кривых: эти кривые обусловлены возможностью физически сделать тот или иной сенсор. Математически мы можем задать любые кривые. Также не нужно моделировать байесовскую мозаику. В реальных сенсорах она обусловлена тем, что мы физически не можем для одного датчика получить три числа, не скалярный, а векторный выход. Поэтому мы можем считать, что один пиксел на выходе может сразу давать три значения, т.е. по сути сразу давать цвет. Таким образом, пиксел имеет сразу три кривые чувствительности, которые можно применять к одному и тому же спектру освещения данного пиксела. Например, в качестве этих кривых мы можем сразу взять кривые стандартного наблюдателя XYZ (рисунок 5.6).

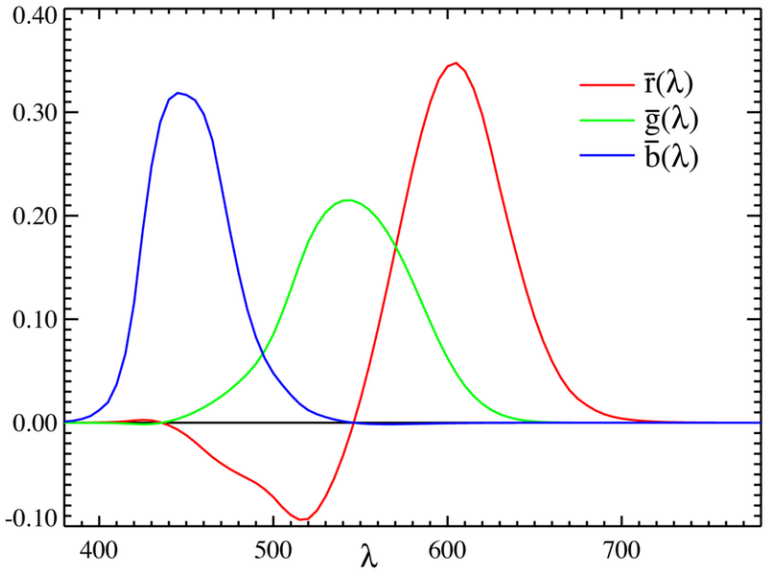


Рис. 5.6. Кривые стандартного наблюдателя XYZ.

$$X = \int_{380}^{780} \overline{x}(\lambda) E_j(\lambda) d\lambda \tag{5.3}$$

$$Y = \int_{380}^{780} \overline{y}(\lambda) E_j(\lambda) d\lambda \tag{5.4}$$

$$Z = \int_{380}^{780} \overline{z}(\lambda) E_j(\lambda) d\lambda \tag{5.5}$$

После применения формулы 5.1, которую мы несколько упростили, подставив сразу интегральную освещённость по всему пикселу и используя сразу подставленные кривые стандартного наблюдателя, мы можем сразу получить итоговый цвет пиксела в пространстве XYZ. Если использовать в качестве кривых в итегральной формуле кривые sRGB, например, то напрямую получится цвет. Стоит заметить, что здесь значение цвета не относительное, а абсолютное, и возможны значения цвета, большие единицы.

Зная освещённость, мы можем получить по этим формулам цвет. Поэтому главная задача – узнать освещённость пиксела. Если у нас есть реальный фотоаппарат, то он сделает всю работу за нас. Если мы строим систему рендеринга, то все величины нужно рассчитывать.

Нужно решить, как посчитать освещённость пиксела исходя из физических величин. Для этого нужно выяснить, как происходит формирование изображения. Важно понимать, какую освещённость создаёт в конкретном пикселе некоторая точка пространства, которая имеет яркость L . Об этом рассказывается в следующем разделе данной главы. Начнем с вывода закономерностей, а затем получим все необходимые формулы для того, чтобы, зная яркость точки в пространстве, посчитать, какую освещённость в пикселе она создает.

5.4. Формирование изображения

Необходимо сделать несколько замечаний о том, как строятся изображения. Возьмем две светящихся точки, красную и зеленую (рисунок 5.7).

Допустим, плоскость справа – это матрица фотоаппарата. Фотоаппарат будет освещен некоторой смесью красного и зеленого источников. Из схемы видно,

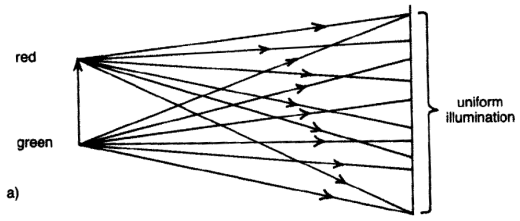


Рис. 5.7. распространение освещения от 2 светящихся точек.

что никакого изображения этих источников так, как мы хотели бы их получить, не выйдет. Источники светят в разные стороны, затем свет смешивается и равномерно освещает матрицу. В идеале, нужно пустить из источника один луч, и если мы этот луч отрежем, он сформирует четкое изображение на картинной плоскости (на приемнике), рисунок 5.8.

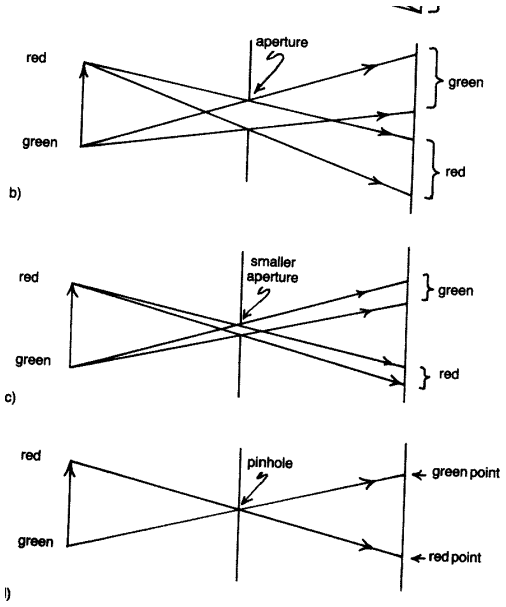


Рис. 5.8. В зависимости от размера диафрагмы точки на экране превращаются в протяженные элементы различного размера.

Существует два способа решения данной проблемы. Первый способ – отсекать часть лучей,оставлять один луч, и так формировать изображение. Второй способ – это собирать или фокусировать лучи в одной точке. Нужно сделать так, чтобы одна точка в пространстве сопоставлялась одной точке на картинной плоскости.

5.5. Камера-обскура

Если мы хотим ограничить лучи и за счет этого получить освещение, то переходим к идее камеры-обскуры (pinhole-camera). Пространство закрывается от картинной плоскости заслонкой с маленьким отверстием (рис. 5.9 слева). Получается так называемая ***апертура***, которая ограничивает пучок света, идущий от красного и зеленого. Этот пучок света создаёт уже более сформированное изображение, сответствующее данному источнику света, которое по-прежнему, однако, остаётся пятном. Чем меньше размер апертуры, тем более сформированное, чёткое изображение мы получаем. Таким образом, мы получили некую систему, которая формирует изображение. Заметим, что изображение получается перевернутым. Каждой точке в пространстве, ставится в соответствие точка на матрице. Далее в работу вступают сенсоры, чувствительные к свету.

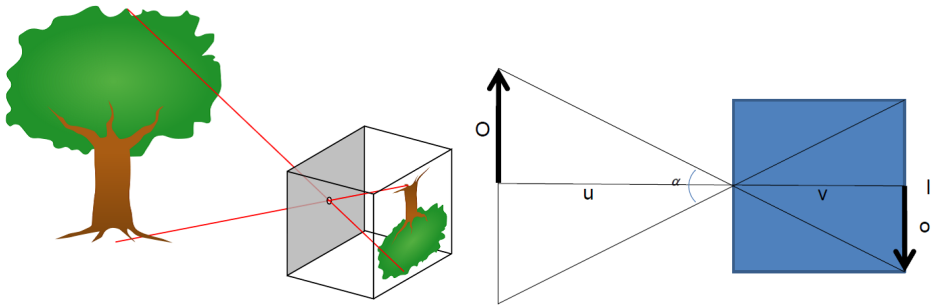


Рис. 5.9. Принцип работы камеры-обскура.

На рисунке 5.9 приведена камера-обскура (pinhole-camera). Она представляет собой ящик с отверстием. На задней стенке камеры мы получаем изображение. На ней может быть плёнка, или электронный датчик на основе фотоэлектрического эффекта, который позволяет рассматривать результирующее изображение.

С точки зрения геометрии, камера-обскура очень простая (рисунок 5.9, справа). Если мы в своей системе визуализации поддерживаем камеру-обскура, то, зная положение исходной точки в пространстве (условно на рисунке эта точка показана), легко из чисто геометрических соображений получить её координаты на картинной плоскости. Из трехмерных координат с помощью преобразования проекции получаем двумерные координаты. Камера-обскура определяется двумя параметрами: углом и расстоянием от центра проекции до задней стенки. Если мы знаем эти параметры, то можем задать такую камеру и строить проекции. Дополнительно нужно знать так называемые внешние параметры, т.е. положение и ориентацию камеры в пространстве.

Камера-обскура имеет некоторые проблемы. Они связаны с тем, что когда

для повышения резкости нужно уменьшить отверстие. Чем меньше оно становится, тем больше дифракция (рисунок 5.10).

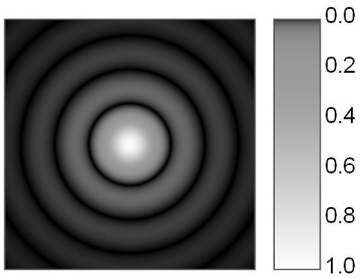


Рис. 5.10. Диффракция света на тонком отверстие.

Дифракция – волновой эффект, который проявляется, когда геометрические особенности среды имеют размер порядка длины световой волны. В результате, в определённый момент вместо ожидаемого повышения резкости картинки мы увидим соответствующий волновой эффект, и не получим ещё более резкую картинку.

Также при уменьшении отверстия через него проходит меньше света, тем более чувствительным должен быть сенсор. Чувствительность всегда идет в обмен на шум. Поэтому возникает проблема: мы не можем отобразить не очень яркие источники света, т.к. они приближаются к базовому шуму по яркости. Конечно, это зависит от качества приемника, но проблема все равно серьезная.

5.6. Камера с линзой

Общепринятым подходом к избежанию этой проблемы является использование линз. Т.е. переход от идеи ограничения потока света к идее его фокусировки. Вместо отверстия в заслонке ставится фокусирующая линза, которая спроецирует весь свет из некоторой точки пространства в соответствующую точку на картинной плоскости (рисунок 5.11).

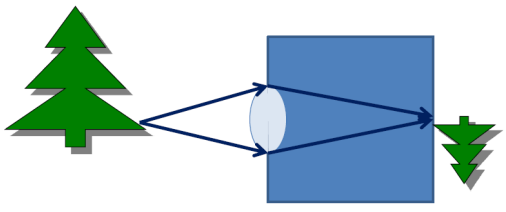


Рис. 5.11. Камера с линзой.

Таким образом, хотя и сейчас мы проецируем точку в точку, но света стало много, и теперь нет проблем с яркостью. Зато есть новая проблема: как сделать такую линзу, которая не вносила бы дополнительных искажений? Линзы применяются сейчас везде, в любой оптике, и когда мы будем писать систему рендеринга, мы должны будем уметь эти линзы использовать, и реализовывать систему не как ящик в камере-обскуре, а как фотоаппарат с некоторой оптикой.

5.7. Преломление света на сферической поверхности

Рассмотрим некоторые аспекты геометрии и оптики на линзах. Рассмотрим вначале одну сферическую поверхность, границу воздуха с преломляющим материалом (рисунок 5.12).

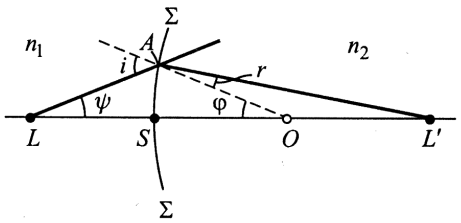


Рис. 5.12. Преломление луча на сферической поверхности.

Пусть есть точка L, источник света в пространстве. Есть некоторая сферическая поверхность, будем считать, это одна поверхность линзы. Центр сферической поверхности находится в точке O. Если мы проведем через центр этой сферической поверхности прямую, так называемую ***осевую*** или ***центральную линию*** (проходит через центр и через нашу точку). Имеем некоторый пучок света с осью OL и углом ϕ конической формы. Когда он попадает на поверхность линзы, то происходит преломление, луч LA преломлется и попадает в точку L'. Здесь проиходит фокусировка лучей, пришедших из точки L.

Для упрощения мы будем рассматривать так называемые ***параксиальные*** или ***приосевые пучки***. Мы предполагаем, что это LS и LA имеют приблизительно одинаковую длину. Это довольно сильное предположение, т.к. если бы они действительно имели одинаковую длину, угол между ними был бы равен нулю, и они бы совпали. Но, на самом деле, если считать, что рассстояние до объекта много больше, чем отклонение луча от оси, то угол будет очень маленьким, и наши рассуждения будут достаточно верны до тех пор, пока пучок

идет ровно через центр перпендикулярно касательной в точке пересечения с поверхностью. Основная задача - узнать, где этот пучок будет фокусироваться. Для этого нам нужно, зная расстояние LS, подсчитать расстояние SL'. Это делается достаточно просто, исходя из следующих соображений:

Рассмотрим LO и LA. Введем углы i и ϕ. Согласно теореме синусов, верно первое соотношение:

$$\frac{LO}{LA} = \frac{\sin i}{\sin \phi} \tag{5.6}$$

То же самое – с другой стороны нашей линзы: рассматриваем AL' и OL' и их соотношение тоже записываем по теореме синусов:

$$\frac{AL'}{OL'} = \frac{\sin \phi}{\sin r} \tag{5.7}$$

Перемножив эти равенства, получим, что произведение отношений длин равно, по закону Снелла, отношению показателей преломления соответствующих сред:

$$\frac{LO}{LA}\frac{AL'}{OL'} = \frac{\sin i}{\sin r} = \frac{n_1}{n_2} \tag{5.8}$$

Изначально мы знаем коэффициент преломления снаружи и коэффициент преломления внутри, а также OC – радиус сферической поверхности. Наши выкладки в итоге должны свестись к этим параметрам, что и получилось. Далее введём некоторые переопределения, необходимые для того, чтобы найти искомое SL'. Во-первых, введем знаковые величины $a_1 = LS$, $a_2 = SL'$. Обозначим радиус сферы $R = OS$. По логике прохода в одном направлении, когда мы идем слева направо, расстояние считается положительным, справа налево – отрицательным. Поэтому $SL = -a_1$. Подставляем эти обозначения, получаем:

$$\frac{-a_1 + R}{-a_1} \times \frac{a_2}{a_2 - R} = \frac{n_1}{n_2} \tag{5.9}$$

Перепишем это соотношение в классической форме:

$$\frac{n_1}{a_1} - \frac{n_2}{a_2} = \frac{n_1 - n_2}{R} \tag{5.10}$$

Надо иметь в виду, что если речь идёт об абсолютном расстоянии, то a_1 надо брать с минусом. Таким образом, зная коэффициенты преломления и радиус кривизны, мы можем связать между собой a_1 и a_2. Обратим внимание на две вещи. Во-первых, это справедливо только для параксиальных пучков, в силу нашего первоначального предположения о равенстве расстояний. Во-вторых, для параксиальных пучков у нас получлось, что a_2 зависит только от a_1. Это значит, что весь параксиальный пучок сойдется в одну точку. Это большой плюс для наших задач, т.к. это означает, что изображение точечного источника света сфокусируется сферической поверхностью в одну точку. К сожалению, в реальности это не совсем так. Пучки, как правило не параксиальные, и лучи в одну точку не фокусируются. Из-за этого в оптике возникает масса проблем с так называемыми ***аберрациями***. Для идеальной оптической системы, в которой мы рассматриваем исключительно параксиальные пучки, лучи будут сходиться в одну точку (рисунок 5.13).

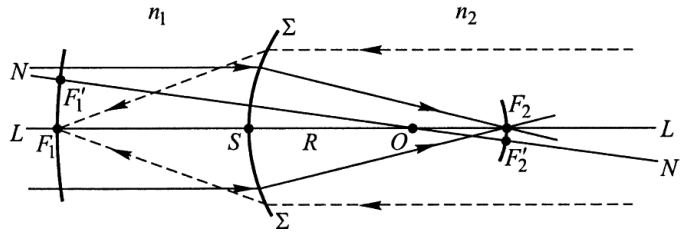


Рис. 5.13. Идеальная оптическая система.

Если источник излучения находится на расстоянии $a_1 = -\infty$, то линза сфокусирует свет в некоторой точке. Расстояние a_2 от центра S до точки фокусировки называется ***передним фокусным расстоянием***. В зависимости от того, с какой стороны поверхности приходят параллельные лучи, фокусируются они в одной из двух точек. Таким образом, мы имеем два фокусных расстояния.

5.8. Преломление света в линзе

Сейчас мы рассматривали одну поверхность, иначе говоря, мы предполагали (рисунок 5.12), что всё, что справа от поверхности, находится в стекле. Чтобы рассматривать настоящую линзу (5.14), нам нужно от одной поверхности перейти к двум.

Таким образом, настоящая линза представляет собой стекло, обработанное таким образом, что с обеих сторон линзы находятся две сферичские поверхности, возможно, разной кривизны. Мы рассматриваем центрированную оптическую систему (это означает, что центры всех оптических поверхностей

Рис. 5.14. Фотография настоящей линзы. Хорошо видно что линза имеет 2 поверхности и ненулевую толщину.

лежат на одной прямой). Линза – это такая центрированная оптическая система, которая состоит из двух сферических поверхностей, ограничивающих преломляющий материал.

Линзы бывают разной формы (рисунок 5.15).

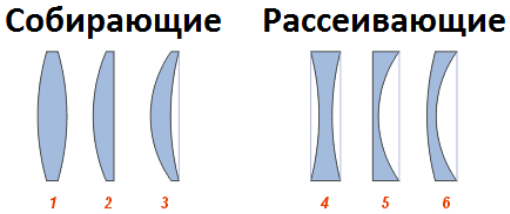


Рис. 5.15. Различные формы линз.

1) Двояковыпуклая
2) Выпукло-плоская
3) Выпукло-вогнутая (положительный мениск)
4) Двояковогнутая
5) Плоско-вогнутая
6) Выпукло-вогнутая (отрицательный мениск)

Нам, по сути, нужна некоторая абстрактная линза, которая будет характеризоваться своим радиусом кривизны.

5.9. Тонкие линзы

Рассмотрим так называемую ***тонкую линзу*** (рисунок **??**).

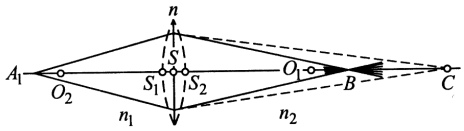


Рис. 5.16. Тонкая линза.

Тонкая линза характерна тем, что толщина линзы очень мала по сравнению с радиусом кривизны. То есть d – толщина линзы, расстояние между центрами сферических поверхностей – пренебрежимо мало по сравнению с радиусами кривизны сферических поверхностей (линза «плоская»). Это позволяет считать, что точки S_1 и S_2 сливаются. Мы можем считать, что у линзы есть некоторый средний радиус кривизны, который и будем использовать. Здесь очень много предположений, но на практике, для маленькой линзы в фотоаппарате или, тем более, хрусталика в глазу, и объекта на расстоянии 10 или 100 метров эти предположения работают с хорошей точностью. Таким образом, будем использовать вместо точек S_1 и S_2 точку S.

Как формируется изображение в данном случае? Первая поверхность даёт изображение на расстоянии a в точке C.

$$\frac{n_1}{a_1} - \frac{n}{a} = \frac{n_1 - n}{R_1} \tag{5.11}$$

Теперь у нас есть противоположная поверхность, но точка C находится внутри среды с коэффициентом преломления n_2. Получается, что это некоторый мнимый источник света, лучи от него преломляются через вторую сферическую поверхность. Она для этого источника света является фокусирующей и создает уже реальное изображение в точке B. (Получается, она вогнутая для мнимого источника, поэтому она является фокусирующей). Записываем такое же соотношение уже для второй поверхности и точки C.

$$\frac{n}{a} - \frac{n_1}{a_1} = \frac{n - n_1}{R_2} \tag{5.12}$$

Сложим эти две формулы.

$$n_1 \left(\frac{1}{a_2} - \frac{1}{a_1} \right) = (n - n_1) \left(\frac{1}{R_1} - \frac{1}{R_2} \right) \tag{5.13}$$

Введём относительный показатель преломления . Далее будем работать именно с относительным показателем преломления $N = \frac{n}{n_1}$. Запишем окончательное соотношение:

$$\frac{1}{a_2} - \frac{1}{a_1} = (N-1)\left(\frac{1}{R_1} - \frac{1}{R_2}\right) \tag{5.14}$$

5.10. Фокусное расстояние

Для данной линзы тоже можно найти фокус, как и для одной сферической поверхности. Фокусное расстояние – удобная характеристика. Посчитав это значение для данной линзы, можно не использовать радиусы и относительный коэффициент преломления. Фокусное расстояние можно, грубо говоря, назвать фокусирующей силой линзы. Для параксиальной оптики параллельные пучки, которые не параллельны главной оси, фокусируются на так называемых ***фокальных плоскостях***. Для непараксиальной оптики это не плоскости, а сферические поверхности, но нам это сейчас не важно.

Чему равно фокусное расстояние? Подставим бесконечность в качестве одного из расстояний до источника света a_1 или a_2 и получим соответствующее переднее и заднее фокусное расстояние. Как будто мы взяли готовую линзу, поставили где-то далеко от неё источник света и посмотрели, на каком расстоянии от центра линзы лучи сфокусировались. Это расстояние и будет равно фокусному расстоянию.

При $a_1 = \infty$ получаем $a_2 = f_2 = \frac{1}{(N-1)(1/R_1 - 1/R_2)}$

При $a_2 = \infty$ получаем $a_1 = f_1 = -\frac{1}{(N-1)(1/R_1 - 1/R_2)}$

Для приближения тонкой линзы оказывается, что эти фокусные расстояния равны по модулю, но отличаются по знаку. Фокусное расстояние становится характеристикой линзы, в которую входят радиусы кривизны и относительный коэффициент преломления.

5.11. Формула тонкой линзы

Для того, чтобы посчитать для любой точки пространства, где эта точка будет сфокусирована, достаточно знать только фокусное расстояние линзы. Мы приходим к известной всем формуле тонкой линзы:

$$\frac{1}{a_2} - \frac{1}{a_1} = \frac{1}{f} \tag{5.15}$$

Эта формула позволяет, зная фокусное расстояние линзы, связать между собой две точки: источник света и точку фокусировки света от него. Для реальных линз параметр f предоставляется производителем. Таким образом, для расчётов не нужны радиус кривизны и материал линзы. Более того, если сложная оптическая система приближается такой тонкой линзой, то для расчёта геометрии, опять же, требуется только фокусное расстояние.

5.12. Изображение в линзе

Рис. 5.17. Изображение в линзе переворачивается.

На рисунке 5.17 показана одна линза, дающая перевернутое изображение. Зная фокусное расстояние линзы и расстояние от свечи до линзы, в соответствии с формулой тонкой линзы можно поставить плоскость, где формируется изображение, на правильном расстоянии от линзы (рисунок 5.18). Это знание понадобится нам при разработке алгоритмов глобального освещения.

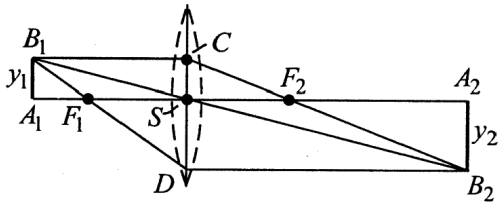


Рис. 5.18. Изображение в линзе переворачивается.

Полезным параметром линзы является увеличение. Это соотношение размеров образа и объекта.

$$V = \frac{a_2}{a_1} = \frac{f}{a_1 + f} = \frac{1}{a_1/f + 1} \tag{5.16}$$

Видно, что увеличение может быть разное в зависимости от того, где находится данный объект. Также увеличение зависит от фокусного расстояния. Чем больше фокусное расстояние, тем больше результирующее увеличение.

5.13. Диафрагмирование

Нужно разобраться, что происходит с теми лучами, которые попали в оптическую систему, но при этом не находятся на расстоянии от главной оси, идеальном для того, чтобы сфокусироваться в точку. Они будут портить нам картинку. Чтобы бороться с этим, используется диафрагмирование. В чем-то его смысл аналогичен отсечению пучков для камеры-обскуры. ***Диафрагма*** – это некоторый экран с отверстием. Функцию этого экрана выполняет оправа линзы, чаще – специальные отдельно стоящие элементы внутри оптической системы (рисунок 5.19).

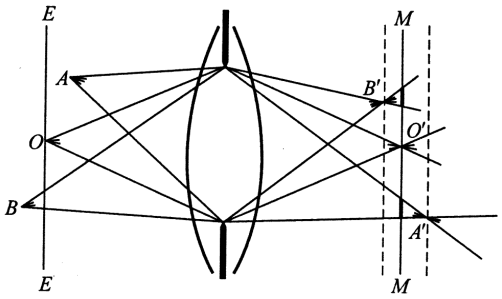


Рис. 5.19. Диафрагмирование.

К чему это приводит закрытие части линзы? Точка O перешла в точку O', идеально сфокусировавшись. Точка A находится ближе, чем та плоскость, для которой в данной линзе происходит фокусировка. Расчёт показывает, что она сфокусировалась позади приемника. Это значит, что на приемнике в соответствующем месте мы увидим размытый образ точки A. То же самое с точкой B, которая стояла за точкой O, она сфокусировалась перед приёмником. Наблюдаем тот же эффект, который возникает в связи со слабым зрением в результате вытянутого глазного яблока, или не совсем правильной работы хрусталика. На сетчатке формируется местами размытое изображение. Чем сильнее мы закрываем диафрагму, тем уже становится этот пучок и уже становится образ на приёмнике, чётче изображение. В случае, когда размер образа одной точки пространства на матрице меньше или сравним с размером пиксела, будем считать, что эта точка находится в резкости. Это значит, что появляется некоторая интересная зависимость: если сделать отверстие диафрагмы достаточно маленьким, то получится, что в резкости

находится не одна плоскость, а большой диапазон. На практике это может зависеть от расстояния, на котором мы сфокусировались. Этот диапазон может иметь разный размер: несколько метров, или несколько сантиметров, в зависимости от оптической силы линзы и от расстояния до объекта. Важно то, что он есть, и мы можем получить резкое изображение достаточно большой глубины.

Ограничение размера пучков – это результат совместной работы различных элементов внутри оптической системы. Как правило, она сложнее, чем одна линза. Можно не учитывать все ограничители, а выделить одну наименьшую диафрагму, поскольку именно она будет максимально блокировать распространение света, и считать, что остальные вообще не ограничивают ход лучей. Такая диафрагма называется апертурной или просто ***апертурой***. Иногда, если мы говорим про однолинзовую диафрагму, здесь же она является апертурой, потому что она единственная.

5.14. Виртуальный фотоаппарат

С точки зрения теории вышесказанного достаточно, чтобы нарисовать виртуальный фотоаппарат (рисунок 5.20).

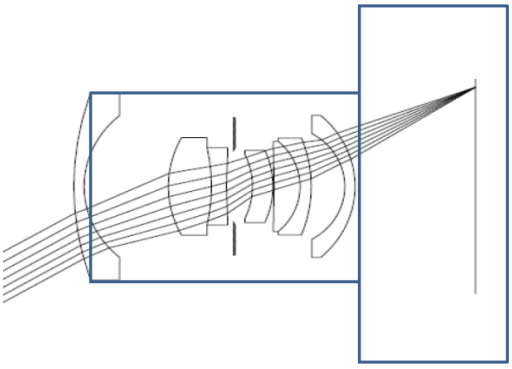


Рис. 5.20. Виртуальный фотоаппарат.

У нас есть объектив – это система, включающая в себя набор линз и апертуру, которая позволяет определенным образом фокусировать поток излучения. Все точки фокусировки расположены в картинной плоскости, это плоскость формирования изображений. Она находится внутри фотоаппарата, на ней может быть расположена пленка, или матрица, как и в камере-обскуре. Наш виртуальный фотоаппарат будет иметь точно такую же структуру: оптическая система и картинная плоскость. Чтобы правильно с этим работать, нужно разобрать модель оптической системы. Она может быть как простой, так и

сложной, вплоть до того, что можно промоделировать все линзы и получить идеально совпадающую с некоторым реальным фотоаппаратом картинку, или ограничиться достаточно простой схемой тонкой линзы, например.

Что мы знаем на практике про фотоаппарат? Для того или иного фотоаппарата всегда известно фокусное расстояние f т.к. именно оно влияет на то, какое изображение будет строиться. Обычно оно выражается в миллиметрах: 15, 20, 30 мм. В принципе, знания расстояния f должно быть достаточно для того, чтобы построить модель тонкой линзы. Модель тонкой линзы будет верна тогда, когда характерное расстояние, с которого мы снимаем объект, будет намного больше фокусного расстояния. К примеру, если фокусное расстояние оптической системы – 50 мм, то нескольких десятков метров до объекта будет достаточно, чтобы использовать модель тонкой линзы. Если мы занимаемся макросъемкой, то нужны более сложные модели, иначе мы не учтём изменения в геометрии. Что мы еще не знаем? Расстояние до картинной плоскости . В реальности матрица находится в фотоаппарате, ведь от этого зависит то, на каком расстоянии находящиеся объекты будут в резкости, и мы не знаем размеров внутри фотоаппарата. Но их можно ввести, или найти. Обычно можно считать, что для линз, сфокусированных на бесконечности, расстояние до картинной плоскости будет равно фокусному расстоянию. Но здесь возникают дополнительные сложности. Например, путь луча нужно считать в зависимости от объектива, и центральная точка, которая нужна, чтобы моделировать сложную оптическую систему одной линзой, может двигаться в зависимости от точки фокусировки объектива и фокусного расстояния, но таким образом, чтобы фокус всегда приходился по расстоянию на картинную плоскость.

5.14.1. Виртуальный фотоаппарат на основе тонкой линзы

Взяв на вооружение формулу тонкой лизны можно смоделировать эффект глубины резкости (Depth Of Filed, DOF) в трассировщике лучей (рис. 5.21) или трёхмерном движке с растеризацией на графический процессорах (рис. 5.22).

```
float  tFocus   = A / (-rayDir.z);
float3 focusPos = rayPos + rayDir*tFocus;
float2 offsets  = A * UniformDisc(-1,1);

rayPos.x += offsets.x;
rayPos.y += offsets.y;

rayDir = normalize(focusPos - ray_pos);
```

Листинг 5.1. Алгоритм изменения луча проходящего через тонкую линзу для DOF.

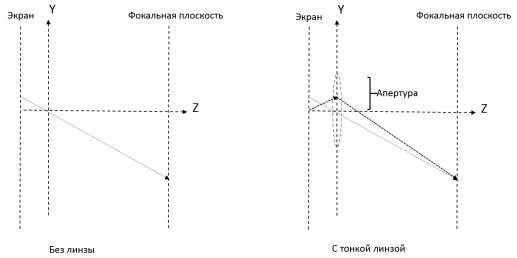


Рис. 5.21. Моделирование эффекта глубины резкости при помощи трассировки лучей. Лучи начинают свой путь не из одной точки на сенсоре, а со случайными смещениями по сенсору в пределах некоторого круга. Радиус этого круга для трассировки лучей задаётся как параметр и называется радиусом линзы или **апертурой** (см. раздел 5.13). Расчёт направления луча записан в листинге 5.1 [19].

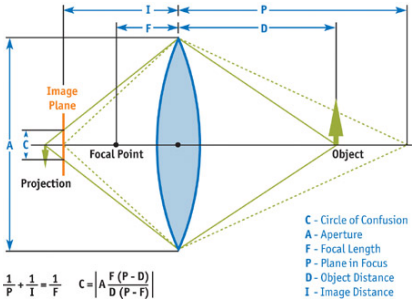


Рис. 5.22. Вычисление радиуса «Circle of Confusion» для растеризации. Зная фокусное расстояние линзы F и расстояние до фокальной плоскости D, мы можем вычислить радиус так называемого круга размытия «Circle of Confusion». Этот круг моделирует тот факт, что свет из одной точки может попадать на разные пиксели (и из разных точек попадать в 1 пиксел).

5.15. Расчёт освещённости матрицы

На практике все это важно, если мы делаем некоторый виртуальный фотоаппарат. Чтобы получить проекцию в виртуальном фотоаппарате, нужно знать фокусное расстояние, центр проекции и размер картинной плоскости.

На рисунке 5.23 показан условный фотаппарат и некоторая точка поверхности, площадка площади a. Допустим, мы все знаем про неё. При помощи алгоритмов глобального освещения мы посчитали то, как эта площадка излучает в том или ином направлении. Нас интересует, какая освещённость будет на соответствующей плодадке в соответствующем пикселе картинной плоскости. Это основная задача. Зная это, по формулам, которые были приведены в начале, можно вычислить цвет.

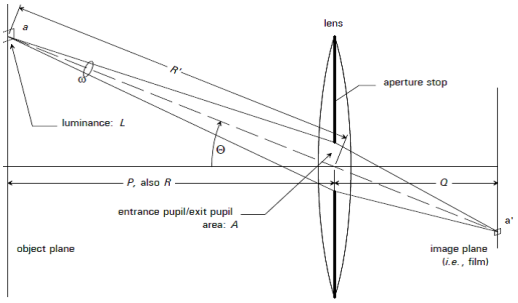


Рис. 5.23. Условный фотоаппарат.

Рассмотрим тонкую линзу. Есть апертура определённой площади A, она будет определять, сколько света проходит на матрицу. Апертурой ограничен некоторый телесный угол ω. Площадка a фокусируется линзой в некоторой точке плоскости a'. Нужно посчитать освещённость, которая будет здесь возникать.

Обратим внимание на то, что энергия, входящая в телесный угол ω, исходит не с исходной площади a, а с проекцией площадки, которая стоит перпендикулярно направлению излучения. Угол Θ определяет наклон плодадки. Если площадка стоит под большим углом к проекции, то через диафрагму будет проходить небольшое количество света. Поэтому нас интересует площадь проекции на P, она будет равна $A_p = A\cos\theta$.

Если мы для простоты предположим, что поверхность объекта ламбертова, то его яркость L будет постоянна для любых углов наблюдения. Таким образом, если взять силу света, излучаемую с этой площадки суммарно в направлении центра апертуры, то она будет равна $I = La_p$. Сила света – это, по сути, яркость точки, но с учетом площади, которой она излучается. Чтобы перейти к силе света, нужно яркость домножить на эту площадь.

Площадь A_pсоответствует некоторой площадке, которая находится не на единичной сфере. Чтобы получить явно телесный угол, нам нужно произвести такое преобразование:

$$R' = \frac{R}{\cos\Theta} \tag{5.17}$$

$$\omega = \frac{A_p}{R'^2} \tag{5.18}$$

Подставим выраженные величины, получим:

$$\omega = \frac{A \cos^3 \Theta}{R^2} \tag{5.19}$$

Энергитический поток от площадки a в угол ω равен $\Phi = I\omega$ по определению. Подставляем сюда найденные значения I, ω и получаем:

$$\Phi = \frac{La \cos^4 \Theta}{R^2} \tag{5.20}$$

Заметим, что весь этот поток оказался сфокусирован на площадке a'. Вся яркость площадки a перешла в a'. Нужно применить соотношение для увеличения a' относительно a.

$$m = \frac{Q}{R} \tag{5.21}$$

Площади относятся как линейные размеры в квадрате.

$$a' = m^2 a \tag{5.22}$$

$$a' = \frac{aQ^2}{R^2} \tag{5.23}$$

Чему равна освещённость? Освещённость – это плотность потока. Это отношение потока, который проходит через дифференциальную площадку, к её площади.

$$E = \frac{\Phi}{a'} = \frac{LA \cos^4 \Theta}{Q^2} \tag{5.24}$$

Чем дальше находится объект, тем меньше освещённость. Чем больше угол Θ, тем меньше освещённость. Чем больше яркость L исходной точки в направлении линзы, тем больше освещённость. Чем больше апертура, тем больше освещённость.

Для линз, сфокусированных на бесконечности, $Q = F$ по определению. Поэтому можно сюда подставить R, помня, что это будет справедливо лишь в случае, когда объект находится достаточно далеко. Площадь A можно выразить через размер диафрагмы: $A = \frac{\pi}{4} D^2$.

Обозначим $n = \frac{f}{D}$. Эта величина называется ***диафрагменным числом*** (или ***f-числом***) и является характеристикой оптической системы с точки зрения пропускания ей света. Это число также называют относительным отверстием, или светосилой, или скоростью объектива. Зная это число, мы можем рассчитать освещённость. Чем меньше диафрагменное число, тем больше света будет попадать на матрицу. Если увеличивается фокусное расстояние, то уменьшается количество света, попадающего на матрицу. Подставляем диафрагменное число в уравнение для E, получаем:

$$E(x') = L\frac{\pi \cos^4 \Theta}{4n^2} \tag{5.25}$$

Если нужно увеличить общую чувствительность фотоаппарата, то надо увеличивать или диафрагму, или экспозицию, или чувствительность сенсора.

От $\cos^4 \Theta$ можно избавиться. Если мы рассматриваем вариант параксиальной оптики, этот косинус стремится к единице (если угол между лучами и главной осью стремится к нулю). Таким образом, оставаясь в рамках параксиальной оптики, мы его можем убрать и считать, что он всегда равен единице. Однако косинус в четвертой степени довольно сильно убывает. Согласно расчётам, на реальных фотографиях мы должны наблюдать существенное падение освещённости к краям кадра. Но производители фотоаппаратов и оптических систем борются с этим, чтобы мы не видели этого падения освещённости. Для этого применяются так называемые микролинзовые системы, которые специальным образом фокусируют свет перед матрицей, чтобы избавиться от зависимости и создать более равномерное освещение по всему кадру. Таким образом, если мы делаем некоторую виртуальную систему, которая должна получить то же изображение, что и фотоаппарат, мы можем об этом не задумываться (конечно, если задачей не является очень точное моделирование конкретной оптической системы).

5.16. Заключение

Итак, чтобы рассчитать цвет пиксела, нужно знать 2 вещи: кривые отклика, освещённость сенсора. Если у нас нет задачи смоделировать конкретную оптическую систему, мы берем кривые стандартного наблюдателя CIE, которые нужны, чтобы преобразовать произвольный спектр в цвет. Применение этих крвиых позволит сразу получить цвет в нужном пространстве по спектру распределения освещённости данного сенсора.

Освещённость сенсора в точке мы можем рассчитать из оптической системы. Для этого, в свою очередь, нужно знать яркость всех точек, влияющих на

яркость изображения, в направлении диафрагмы. Если мы отдельно хотим построить геометрическое расположение пикселя, нужно знать, какой луч в пространстве внёс в него свой вклад. Соответствующая точкам пространства проекция вычисляется по формуле тонкой линзы. Осталось посчитать яркость для любой точки сцены, вносящей вклад в формирование изображения.

Глава 6.

Трассировка лучей

6.1. Обратная трассировка лучей

Алгоритм трассировки лучей был впервые продемонстрирован Turner Whitted-ом в 1980-ом году [20]. Этот алгоритм часто называют обратной трассировкой лучей или 'Whitted-style' трассировкой. Он позволяет получить такие эффекты как тени, отражения и преломления.

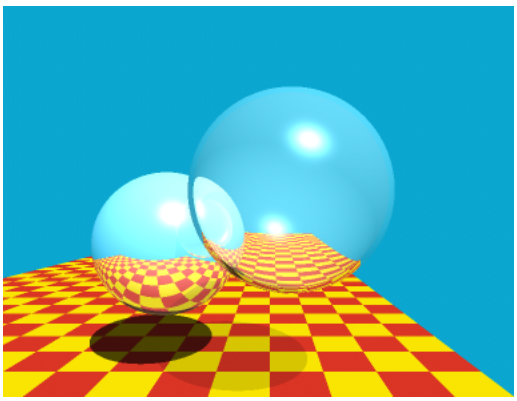


Рис. 6.1. Whitted-style трассировка лучей.

Обратная трассировка лучей выглядит следующим образом: из виртуального глаза через каждый пиксел изображения испускается луч и находится точка его пересечения с поверхностью сцены (для упрощения изложения объемные эффекты вроде тумана не рассматриваются). Лучи, выпущенные из глаза называют первичными. Допустим, первичный луч пересекает некий объект в точке H1 (рис. 6.2).

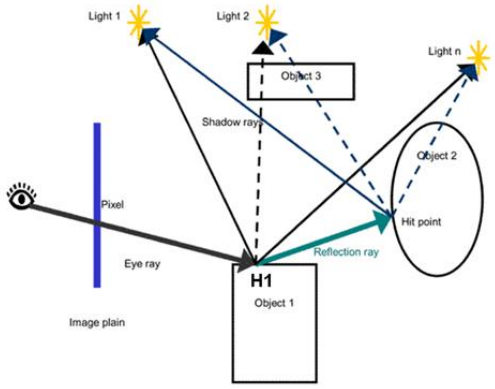


Рис. 6.2. Схема обратной трассировки лучей.

Для расчета тени необходимо определить для каждого источника освещения, видна ли из него эта точка. Предположим пока, что все источники света точечные (т.е. имеют пренебрежимо малый размер). Тогда для каждого точечного источника света до него испускается теневой луч из точки H1 и проверяется, пересекает ли луч какие-то объекты на своем пути. Если теневой луч находит пересечение с другими объектами, расположенными ближе чем источник света, значит, точка H1 находится в тени от этого источника и освещать ее не надо. Иначе, считаем освещение по некоторой локальной модели (Модель Фонга, Блина-Фонга, Кука-Торранса и.т.д.).

При учете вклада освещения от точечного источника важно делить интенсивность источника света на квадрат расстояния до него. Этот прием позволяет корректно учитывать вклад источников и его объяснение будет дано позже, при рассмотрении алгоритма трассировки путей.

Освещение со всех видимых (из точки H1) источников света просто складывается. Далее, если материал объекта 1 имеет отражающие свойства, из точки H1 испускается отраженный луч и для него вся процедура трассировки рекурсивно повторяется. Аналогичные действия должны быть выполнены, если материал имеет преломляющие свойства.

В трассировке лучей для представления самого луча, как правило, используется параметрическое уравнения прямой 6.1.

$$\overline{X(t)} = \overline{O} + \overline{D} * t \tag{6.1}$$

В выражении 6.1 $\overline{X(t)}$ обозначает некоторую точку на прямой. $\overline{O}$ - точка начала (сокращение от Origin) луча, или точка вылета. $\overline{D}$ - нормализованный (имеющий единичную длинну) вектор направления луча (сокращение от

Direction). t - скалярная величина, большая или равная нулю. Таким образом для задания луча достаточно хранить 2 вектора - $\overline{O}$ и $\overline{D}$. Задания значения параметра t позволит получить любую точку лежащую на этом луче.

Важным моментом при вычислении отраженного и преломленного лучей является необходимость добавлять небольшое смещение к позиции луча. В случае отраженного луча нужно прибавлять смещение $\varepsilon * \overline{n}$, в случае преломленного - $(-1) * sign(dot(\overline{n}, \overline{D})) * \varepsilon * \overline{n}$. Это делается для того, чтобы отраженный (или преломленный) луч не встретил пересечение на следующем шаге трассировки с той же самой поверхностью. В выражениях использованы следующие обозначения:

1) $\overline{n}$ - вектор нормали к поверхности.

2) $\overline{D}$ - вектор направления луча.

3) ε - некоторая малая величина, близкая к пределу машинной точности (но больше него) в отношении сложения. Для стандатра IEEE754 плавающей точки с одинарной точностью рекомендуется значение 1e-5.

4) *sign* - функция, возвращающая 1 если ее аргумет больше или равен 0 и -1 в противном случае.

5) *dot* - скалярное произведение двух векторов.

6.1.1. Генерация луча

Как уже было сказано, на начальном этапе алгоритма трассировки лучей через каждый пиксел экрана необходимо выпустить луч. Это можно сделать руководствуясь несложными геометрическими соображениями, рассчитав направления лучей в пространстве камеры (выражения 6.2). Пространство камеры - это такое пространство, где камера находится в точке (0,0,0) и смотрит в положительном направлении оси z.

$$\begin{aligned} fov &:= \frac{\pi}{2} \\ D.x &:= x + 0.5 - w/2 \\ D.y &:= y + 0.5 - h/2 \\ D.z &:= -w/tan(\frac{fov}{2}) \\ D &:= normalize(D) \end{aligned} \tag{6.2}$$

Где:

1) $\overline{x}$ - горизонтальная координата пиксела в пространстве изображения.

2) $\overline{x}$ - вертикальная координата пиксела в пространстве изображения.

3) $\overline{w}$ - размер изображения по горизонтали.

4) $\overline{h}$ - размер изображения по вертикали.

5) fov - Угол обзора камеры. Обычно равен $\frac{\pi}{2}$.

6) $normalize$ - функция, осуществляющая нормализацию вектора (приведение к вектору с единичной длинной).

Что касается начала луча $\overline{O}$, то в пространстве камеры это точка (0,0,0).

Данный подход моделирует так называемую камеру-обскура. Однако он обладает одним недостатком. При создании системы визуализации, в которой сцена может отрисовываться как трассировкой лучей, так и растеризацией (например при помощи библиотеки OpenGL), могут возникнуть трудности с получением геометрически-совпадающей картинки для растеризатора и трассировщика лучей. Причина этого заключается в том, что во всех современных системах, основанных на растеризации (и не только), используются понятия видовой матрицы (mView) и матрицы проекции (mProj). Видовая матрица переводит мир из мирового пространства в пространство камеры (т.е. преобразовывает пространство так что, камера перемещается в (0,0,0) и смотрит в положительном направлении оси z). Эта матрица не вызывает осложнений, т.к. всегда можно преобразовать луч при помощи обратной матрицы, чтобы эмитировать перемещение камеры. Однако матрицу проекции необходимо пересчитать. Следующая формула используется для вычисления матрицы перспективной проекции в OpenGL [21]:

$$mProj = \begin{bmatrix} e & 0 & 0 & 0 \\ 0 & e/a & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Где:

- e = 1/tan(FOV/2)
- FOV - Filed Of View; угол обзора. Обычно равен $\frac{\pi}{2}$.
- a - aspect ratio; a = h/w;

- f - far clip plane; дальняя плоскость отсечения.
- n - near clip plane; ближняя плоскость отсечения.

В случае камеры-обскура, f стремится к бесконечности, а n - к нулю. Таким образом, получаем:

$$mProj = \begin{bmatrix} e & 0 & 0 & 0 \\ 0 & e/a & 0 & 0 \\ 0 & 0 & -1 & -2n \\ 0 & 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} e & 0 & 0 & 0 \\ 0 & e/a & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

В целях обеспечения совместимости с существующими приложениями, где матрица проекции уже задана (или задается извне и менять её нельзя), необходимо использовать другой алгоритм генерации луча (выражение 6.3):

$$\begin{aligned} P1.x &:= 2 * (x + 0.5)/w - 1 \\ P1.y &:= -2 * (y + 0.5)/h + 1 \\ P1.z &:= 0 \\ P1.w &:= 1 \\ P2 &:= (mView * mProj)^{-1} * P1 \\ P2 &:= P2 * (1/P2.w) \\ P2.y &:= -P2.y \\ D &:= normalize(P2) \end{aligned} \tag{6.3}$$

В выражение 6.3 используется распространенная в трехмерной компьютерной графике однородная система координат [21], где вводится четвертая координата w в целях расширения множества преобразований, которые можно записать при помощи матриц. В соответствии с этим, матрицы $mView$ и $mProj$ имеют размер 4x4.

6.1.2. Устранение ступенчастости

Рассмотрим причину появления ступенчастости. Изображение хранится в памяти как двумерная матрица пикселей. Однако считается, что изображение предметов реального мира это не просто набор пикселей. Авторы [19] склонны рассматривать изображение как непрерывную двумерную функцию. То,

что мы видим на экране – приближение этой функции, ее дискретизованное в заданном разрешение представление. Ступенчатость - результат дискретизации.

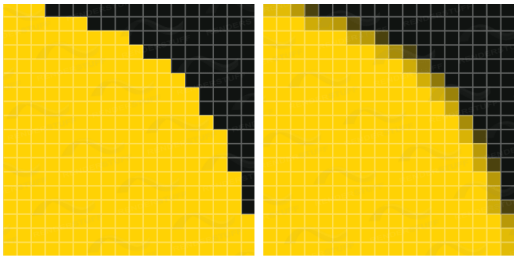


Рис. 6.3. Устранение ступенчастости.

Трассировка лучей в самом общем понимании - это метод, позволяющий восстановить значения функции изображения с помощью точечных выборок, то есть значений этой функции в точках. Самый простой способ устранения ступенчатости - трассировать больше чем 1 лучей на пиксел с некоторыми смещениями внутри самого пиксела, а полученный результат усреднять. Однако для того чтобы уменьшить ступенчатость, достаточно часто не только трассируют больше чем 1 луч на пиксел, но и применяют фильтрацию. Рассмотренное выше усреднение соответствует так называемому box-фильтру. Достаточно простым, но более эффективным методом является билинейная фильтрация, при которой вклад от луча распределяется по 4 ближайшим пикселам.

Как правило процесс вычисления значения функции стараются отделить от способа хранения изображения. По этой причине во многих системах визуализации и рассчета освещения присутствуют такие классы как Sampler (отвечает за фильтрацию) и Integrator (отвечает собственно за рассчет отсвещения).

6.1.3. Прозрачные объекты и тени

При расчете тени, можно использовать более сложную модель. Если если есть вероятность того, что один объект перекрывается другим прозрачным объектом, можно трассировть теневой луч сквозь прозрачный объект, расчитывая по закону Бугера-Ламберта-Бэра затухание в прозрачной среде в зависимости от пройденного расстояния (этот же закон нужно использовать и при учете затухания внутри прозрачных объектов для обычных лучей). При таком алгоритме тень становится цветной. Разумеется, тени, рассчитанные таким образом корректны, только если прозрачный объект, отбрасывающий тень, имеет близкий к единице коэффициент преломления (считаем что коэффициент преломления воздуха равен 1). Если это не так, то под прозрачным объектом образуется сложная картина, называемая каустиком. Типичный пример

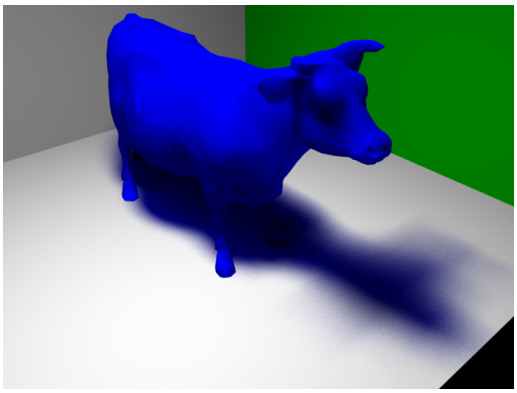


Рис. 6.4. Цветная тень.

каустика – солнечный зайчик от стакана воды, когда через него просвечивает солнце. Корректные способы расчета каустиков будут рассмотрены далее в разделе про вычисление интеграла освещенности.

6.1.4. Корректный рассчет преломлений

В рассмотренном ранее алгоритме не обсуждалась возможность получения реалистичных изображения преломляющих объектов. Здесь необходимо отметить 2 основных момента. Во-первых существует эффект так называемого полного внутреннего отражения при котором выходя из более плотной среды в менее плотную, при большом угле падения, луч полностью отражается внутрь более плотной среды и преломления не происходит. Во-вторых, учета только закона Снэллиуса для описания преломляющий объектов недостаточно. Поведение света для преломляющих объектов с достаточно-высокой степенью реалистичности описывается при помощи формул Френеля, которые устанавливают зависимость между долей отраженного и преломленного света в зависимости от угла падения луча и показателя преломления материала [22]. Хотя следует отметить, что и этого может быть недостаточно для реалистичной визуализации некоторых кристаллов, где необходимо учитывать такие эффекты как двулучепреломление, поляризацию и интерференцию (на тонкой пленке оксида или другого покрытия кристалла) [23].

6.1.5. Поиск пересечений

Поиск пересечений луча с геометрическими объектами сводится к нахождению всех общих точек луча и объекта. Объекты в компьютерной графике,

как правило представляются набором примитивов. Далее будут рассмотрены 2 наиболее важных вида примитовов - трегольник и прямоугольный паралеллипипед. Треугольник важен, поскольку в современной компьютерной графике поверхности в подавляющем большинстве случаев задаются именно при помощи треугольников, а прямоугольный паралеллипипед необходим для реализации эффективного поиска пересечений на основе BVH деревьев (которые будут рассмотрены далее).

Треугольник, барицентрический тест

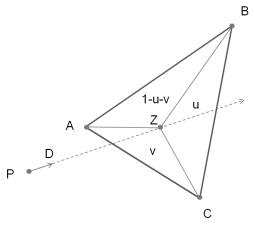


Рис. 6.5. Барицентрический тест треугольника и луча.

Введем следующие обозначения:

- P - точка из который вылетает луч;
- D - напраление луча;
- A,B,C - вершины треугольника;
- Тройка (u, v, 1-u-v) - барицентрические координаты. Они представляют собой отношения площадей маленьких треугольников к большому треугольнику. То есть u = S(ZCB)/S(ABC), v = S(ZAC)/S(ABC), 1-u-v = S(ZAB)/S(ABC);

$$\begin{aligned} Z(u,v) &= u*A + v*B + (1-u-v)*C \\ Z(u,v) &= P + t*D \\ P + t*D &= u*A + v*B + (1-u-v)*C \end{aligned} \tag{6.4}$$

Имея 3 точки на плоскости, можно выразить любую другую точку через ее барицентричечкие координаты. Первое уравнение получается из определения барицентрических координат, выражая точку пересечения Z. С другой стороны, эта же точка Z лежит на прямой. Второе уравнение таким образом,

это параметрическое уравнение прямой. Приравняв правые части уравнений 1 и 2 получаем третье уравнение, которое, по сути, является системой 3-х уравнений (P,D,A,B,C - векторы) с 3-мя неизвестными (u,v,t). Проведя алгебраические преобразования получим ответ в следующем виде:

$$\begin{aligned} E1 &:= B - A \\ E2 &:= C - A \\ T &:= P - A \\ P &:= cross(D, E2) \\ Q &:= cross(T, E1) \end{aligned} \tag{6.5}$$

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{dot(P, E1)} * \begin{bmatrix} dot(Q, E2) \\ dot(P, T) \\ dot(Q, D) \end{bmatrix}$$

Треугольник, юнит-тест (Woops unit test)

Основная идея данного алгоритма заключается в том, чтобы посчитать матрицу преобразования треугольника в некий единичный треугольник (отсюда название) с вершинами (1,0,0); (0,1,0); (0,0,0); и нормалью (0,0,1). Во время подсчета пересечения с треугольником луч преобразуется этой матрицей в пространство, где треугольник имеет единичное представление. Назовем такое проебразование T_Δ. После преобразования вычислить пересечение намного проще, так как нужно считать пересечение с заранее известным треугольником - (1,0,0); (0,1,0); (0,0,0).

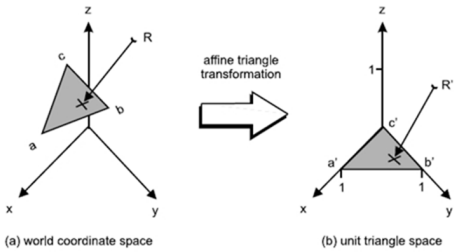


Рис. 6.6. Преобразование T_Δ.

Пусть задан треугольник с вершинами A,B,C. Рассмотрим преобразование T_Δ^{-1}:

$$T_{\Delta}^{-1} = \begin{bmatrix} A_x - C_x & B_x - C_x & N_x - C_x \\ A_y - C_y & B_y - C_y & N_y - C_y \\ A_z - C_z & B_z - C_z & N_z - C_z \end{bmatrix} * X + \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} \tag{6.6}$$

Заметим, что применив данное преобразование к треугольнику (1,0,0); (0,1,0); (0,0,0) - получим исходный треугольник (A,B,C). То есть T_{Δ}^{-1} в действительности является обратным преобзазованием к T_{Δ}

$$T_{\Delta}^{-1} * \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = A \quad T_{\Delta}^{-1} * \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = B \tag{6.7}$$

$$T_{\Delta}^{-1} * \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = C \quad T_{\Delta}^{-1} * \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = N \tag{6.8}$$

Для того чтобы найти нужное нам преобразование необходимо дополнить матрицу T_{Δ}^{-1} до 4x4 и (добавить еще одну строчку (0,0,0,1)) и найти обратную матрицу. Это будет соответствовать преобразованию T_{Δ}.

Для каждого треугольника матрицу T_{Δ} нужно рассчитать один раз и сохранить. Пересечение луча и треугольника в юнит-тесте вычисляется следующим образом (здесь M - матрица преобразования T_{Δ}).

$$\begin{aligned} \overline{O} &:= mul3x4(M, \overline{O}) \\ \overline{D} &:= mul3x3(M, \overline{D}) \\ t &:= -o.z/d.z; \\ u &:= o.x + t * d.x \\ v &:= o.y + t * d.y \end{aligned} \tag{6.9}$$

Функция mul3x4 выполняет умножение подматрицы 3x3 на трехмерный вектор и добавляет к результату последний столбец (3 его компоненты). Функция mul3x3 просто умножает подматрицу 3x3 на трехмерный вектор.

Прямоугольный паралеллипипед (AABB)

Прямоугольный парлеллипипед (по англ. Axis Aligned Bounding Box - или AABB) обычно представляется в виде координат 2 точек. Нижнего левого

угла (третье измерение опущено) - boxMin и правого верхнего угла - boxMax. Все точки, координаты которыйх находятся в интервале [boxMin, boxMax] по всем 3 измерениям находятся внутри паралеллипипеда. Для того чтобы вычислить пересечение луча и прямоугольного паралеллипипеда сначала вычисляют точки пересечения луча со всеми 6 плоскостями паралеллипипеда (получая значения координат t в уравнении луча), после чего из каждой пары плоскойтей (для x,y и z) выделяют минимум и максимум (рис. 6.7). Результирующие координаты ищут как максимальный из минимумов (t_{min}) и минимальный из максимумов (t_{max}) - выражение 6.10.

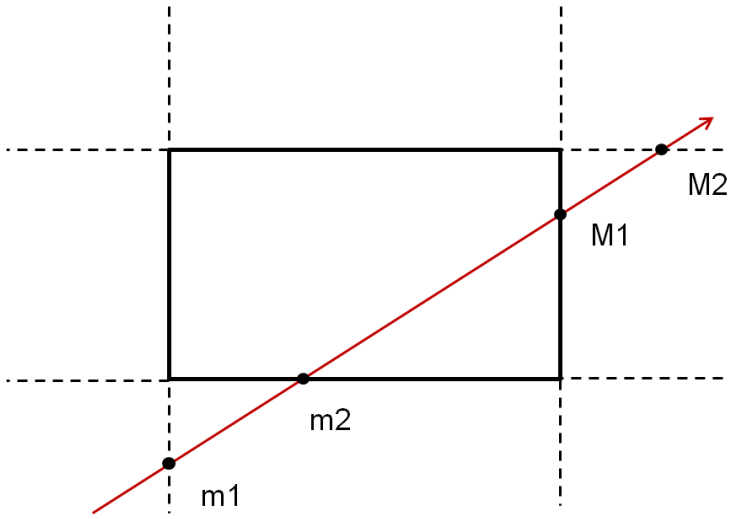


Рис. 6.7. Пересечение луча и AABB. Точки m1 и m2 - минимумы пересечения луча с парой плоскостей. M1 и M2 - максимумы.

$$
\begin{aligned}
l_x &:= (1/D.x) * (boxMin.x - O.x);\\
h_x &:= (1/D.x) * (boxMax.x - O.x);\\
l_y &:= (1/D.y) * (boxMin.y - O.y);\\
h_y &:= (1/D.y) * (boxMax.y - O.y);\\
l_z &:= (1/D.z) * (boxMin.z - O.z);\\
h_z &:= (1/D.z) * (boxMax.z - O.z);\\
t_{min} &:= max(min(l_x, h_x), min(l_y, h_y), min(l_z, h_z))\\
t_{max} &:= min(max(l_x, h_x), max(l_y, h_y), max(l_z, h_z))
\end{aligned}
\tag{6.10}
$$

6.1.6. Ускорение поиска пересечений

Алгоритмы ускорения поиска пересечений для больших сцен рассмотрены в приложении C.

Глава 7.

Методы расчета интеграла освещенности на основе трассировки лучей

7.1. Проблема глобальной освещенности

Освещение принято разделять на локальное и глобальное. Локальным освещением называют освещение вызванное прямым попаданием света от источника на поверхность. Глобальным освещением называют освещение вызванное прямым и непрямым попаданием света на поверхность. Непрямое попадание света, в свою очередь, вызвано переотражениями световой энергии от различных объектов трехмерной сцены и является трудоемким в плане вычислений.

Один из способов получения фотореалистичных изображений - моделирование оптических процессов с целью точного вычисления освещенности каждой точки сцены. Как следствие, появляется необходимость в вычислении глобальной освещенности. Трассировка лучей, описанная выше, позволила бы рассчитать изображение корректно, если бы поверхности объектов реального мира были гладкими и отражали свет строго по закону 'угол падения равен углу отражения'. Однако, из-за наличия микрорельефа поверхностей это не так, и свет приходящий строго с одного направления, распределяется по полусфере вокруг нормали к поверхности в соответствии со свойствами материала.

Для моделирования свойств материала вводят Двунаправленную Функцию Отражения (ДФО) [24]. В английской терминологии обозначается как BRDF

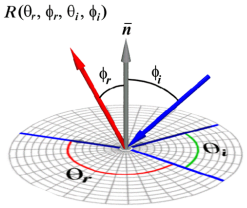


Рис. 7.1. Двунаправленная Функция Отражения.

(Bidirectional Reflectance Distribution Function). ДФО связывает по некоторому закону интенсивность и угол падающего света с интенсивностью и углом отраженного поверхностью света. Зная ДФО материала, можно вычислить освещенность точки поверхности при помощи интеграла освещенности [25]:

$$I(\phi_r, \theta_r) = \int\limits_{\phi_i \theta_i} L(\phi_i, \theta_i) R(\phi_i, \theta_i, \phi_r, \theta_r) cos(n, l_{\phi_i, \theta_i}) \mathrm{d}\phi_i \mathrm{d}\theta_i \tag{7.1}$$

В уравнении 7.1 функция $L(\phi_i, \theta_i)$ задает яркость света, падающего с направления задаваемого вектором l_{ϕ_i,θ_i}. $R(\phi_i, \theta_i, \phi_r, \theta_r)$ - ДФО. n - вектор нормали к поверхности. Вычисляемая интегрированием интенсивность освещения в точке $I(\phi_r, \theta_r)$ является не числом, а функцией. Другими словами, она дает значения интенсивности света, отражаемой поверхностью под разными углами. Таким образом, выполняя интегрирование по полусфере приходящего (падающего) в точку освещения L с учетом свойств поверхности R, получаем новую функцию распределения освещения в пространстве I, обусловленную свойствами отражения поверхности (материала) в некоторой точке x. Если зафиксировать конкретный луч (например выпущенный для конкретного пиксела и виртуальной камеры), направление ϕ_r, θ_r было бы фиксированно и задавалось бы этим лучом.

Формула 7.1 берется их тех соображений, что освещенность в точке поверхности складывается из света, падающего на поверхность со всех направлений и отражающегося в заданном направлении по определенному закону (который задает ДФО). Эта формула не более чем математическая модель и она верна не всегда. Например, в случае стекла нужно учитывать свет, приходящий из под поверхности и проводить интегрирование по полной сфере. В случае кожи ситуация еще сложнее, так как необходимо учитывать такой эффект как подповерхностное рассеивание [26, 27].

Размерность интеграла освещенности (формула 7.1) зависит от количества учитываемых переотражений света, поскольку функция L в некоторой точке x будет зависеть от функции I в некотрой другой точке y. Из-за боль-

шой размерности интеграла для его вычисления обычно используют метод Монте-Карло, поскольку считается, что его скорость сходимости не зависит от размерности интеграла. В действительности это не совсем так (чуть ниже, в разделе 7.1.2 «Интегралы большой размерности» мы увидим почему), но все остальные методы на большой размерности справляются ещё хуже.

7.1.1. Метод Монте-Карло

Пусть u - случайная величина, равномерно распределенная на отрезке [a,b]. Тогда f(u) - тоже случайная величина. Исходя из определения, ее математическое ожидание равно:

$$Ef(u) = \int_a^b f(x)p(x)dx$$
$$p(x) = \frac{1}{b-a} \tag{7.2}$$

Где $p(x)$ - плотность вероятности равномерного распределения. Выражая $\int_a^b f(x)dx$ из формулы 7.2 получаем:

$$\int_a^b f(x)dx = (b-a)Ef(u) \approx \frac{b-a}{N}\sum_{i=1}^{N} f(u_i) \tag{7.3}$$

В применении к интегралу освещенности:

$$I(\phi_r, \theta_r) \approx \frac{\sum_{i=1}^{N} L(\phi_i, \theta_i)R(\phi_i, \theta_i, \phi_r, \theta_r)cos(n, l_{\phi_i,\theta_i})}{N} \tag{7.4}$$

Итак, мы можем оценивать значения интеграла при помощи вычисления суммы достаточно большого числа значений под-интегральной функции [28]. Далее будут рассмотрены некоторые определения и понятия математической статистики.

Выборка по значимости (Importance Sampling).

Эффективным способом улучшения сходимости метода Монте-Карло является выборка по значимости (метод существенной выборки, importance sampling) [28]. Данный метод предполагает использование случайной величины не с равномерным распределением, а с таким распределением, которое пропорционально модулю вычисляемой функции. Идея выборки по значимости базируется на том, что некоторые значения случайной величины в процессе моделирования имеют большую значимость для оцениваемой функции, чем другие. Если эти 'более вероятные' значения будут появляться в процессе выбора случайной величины чаще, дисперсия оцениваемой функции уменьшится и скорость сходимости, таким образом, увеличится [28]. При этом, для того чтобы метод давал корректный результат, необходимо учитывать получаемые значения с весами, обратно-пропорциональными плотности вероятности случайной величины. Таким образом, если распределение случайных величин не равномерное, оценка для метода Монте-Карло запишется в более общем виде [28]:

$$\int_a^b f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(u_i)}{p(u_i)} \tag{7.5}$$

где $p(u_i)$ - функция плотности вероятности случайной величины u. Выборка по значимости - один из ключевых механизмов, позволяющий ускорить интегрирование освещенности.

Смещенные и несмещенные оценки и методы.

В данном разделе мы кратко рассмотрим понятие смещенных и несмещенных оценок в применении к вычислению интеграла освещенности чтобы пояснить такие часто встречающиеся англо-язычные термины как biased renderer и unbiased renderer.

Пусть $X_1..X_n$ случайная выборка из распределения, зависящего от параметра $\theta \in \Theta$

Статистикой называется любая измеримая функция от выборки θ_i, не зависящая от θ (важно, что значение статистики можно вычислить при каждой реализации выборки, не зная значения θ). В случае метода Монте-карло подинтегральное выражение может называться статистикой.

Точечной оценкой называется любая статистика, принимающая некоторые значения на области определения Θ.

Оценка называется несмещенной (unbiased) если ее математическое ожидание равно оцениваемому параметру. В противном случае оценка называется смещенной (biased).

Оценка называется состоятельной (consistent) если она сходится по вероятности (сходимость почти наверное) к оцениваемому параметру.

Будем называть метод или алгоритм вычисления интеграла освещенности несмещенным, если он позволяет получить несмещенную оценку интеграла освещенности для индивидуальной точки или набора точек трехмерного пространства. В противном случае будем называть метод смещенным.

Будем называть метод вычисления интеграла освещенности состоятельным, если он позволяет получить состоятельную оценку интеграла освещенности для индивидуальной точки или набора точек трехмерного пространства.

Таким образом, если программное обеспечение для вычисления освещенности и фотореалистичной визуализации (англ. renderer) использует в том или ином виде метод Монте-Карло и при этом позволяет при бесконечно-большом времени расчета получить абсолютно точное решение, говорят что это несмещенный рендерер (unbiased renderer). К таким программам, например, можно отнести все виды трассировщиков путей. Если программа использует метод Монте-Карло, но при этом в пределе (при бесконечно-большом времени расчета) не позволяет получить абсолютно точное решение, говорят, что такой рендерер является смещенным (например программы, использующие фотонные карты и/или кэш освещенности). К программам, не использующим метод Монте-Карло понятия смещенный/несмещенный неприменимы. Например, некорректно использовать указанные термины по отношению к программе, использующей метод излучательности [29]. Смещенность алгоритма еще не означает, что он вычисляет интеграл с низкой точностью. На практике смещение (bias) проявляется в виде цветных пятен. Если смещение небольшое, пятна незаметны для глаза. Как правило, смещенные методы на начальном этапе работы (при небольшом числе выборок или небольшом времени работы) дают более приемлимую для человечесткого глаза оценку, чем несмещенные. Однако в пределе, при длительном рассчете несмещенные методы обычно дают более качественное изображение.

7.1.2. Интегралы большой размерности

Когда говорят, что скорость сходимости метода монте-карло не зависит от размерности, имеют ввиду, что какая-бы у интеграла не была размерность,

нет необходимости выписывать и считать рекурсивные суммы (сумму от суммы, от суммы и.т.д), поскольку такую последовательность можно заменить одной суммой, - и сходимость всегда будет **пропорциональна** $\sqrt{N}$ где N - число выборок. Ключевое слово здесь - пропорциональна. Да, - рекурсивные суммы можно заменить на одну, - но при этом никто не говорит о количестве выборок для достижения той же самой ошибки. Да, сходимость **пропорциональна** $\sqrt{N}$, но никто не говорит, - насколько большой ошибка может быть в начале расчёта.

Проведём мысленный эксперимент - попробуем оценить методом Монте-Карло объём N-мерной сферы. Хорошо известна формула по которой вычисляется этот объём и соответствующий ей график функции (рис. 7.2). При увеличении размерности объем сферы сначала растёт, а затем очень быстро уменьшается. Нетрудно вычислить, что при размерности 20 объем уже будет около одной миллионной. При таком незначительном объёме слишком малое число выборок будет попадать внутрь сферы, - или вообще ни одной.

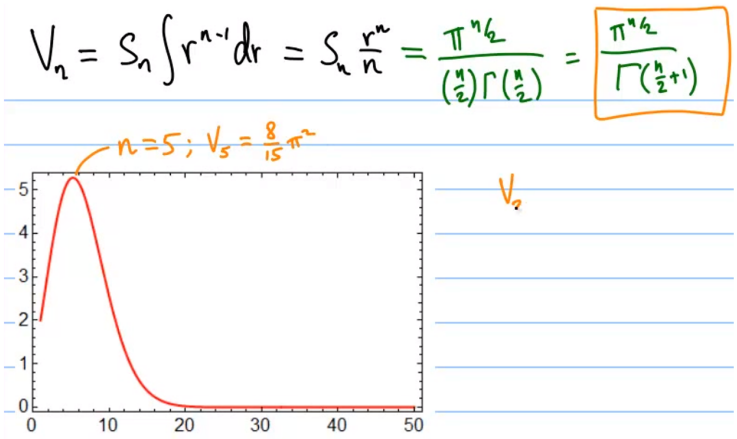


Рис. 7.2. Зависимость объема многомерной сферы от размерности.

Можно резонно возразить, - что раз объем сферы маленький, он нас и не интересует с высокой относительной точностью, т.к. абсолютная ошибка будет маленькой. Но в действительности метод Монте-Карло используют не только для оценки объема, а в общем случае для оценки произвольных функций, - для которых значения внутри сферы могут быть огромными. К сожалению, именно этот случай достаточно часто встречается в стохастической трассировке лучей/путей. Далее по тексту и в разделе 7.5 мы расскажем о том, как с этим можно бороться.

7.2. Стохастическая трассировка лучей

Стохастическая трассировка лучей (также называемая Монте-Карло трассировкой) вычисляет значение интеграла освещенности (для фиксировнаного направления ϕ_r, θ_r) напрямую
по формуле 7.4. Метод был предложен James-ом Kajiya в 1986 году [25].

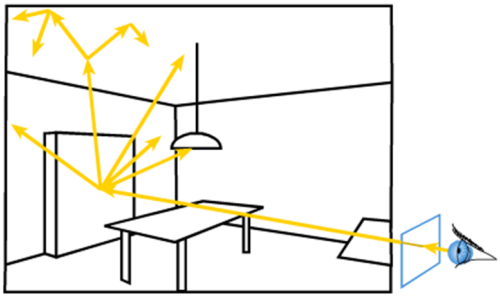


Рис. 7.3. Иллюстрация монте-карло трассировки лучей.

Из виртуальной камеры испускается луч, который трассируется в сцену. В точке пересечения с поверхностью выпускается некоторое число лучей по полусфере и для каждого луча процедура выполняется рекурсивно (рис. 7.3). Полученные значения яркости на каждом уровне рекурсии складываются с учетом формулы 7.4.

В дальнейшем будем рассматривать модификацию стохастической трассировки лучей, при которой отраженный луч всегда один (рис 7.4). Такое упрощение немного замедляет сходимость метода, но позволяет упростить реализацию и при желании избавиться от рекурсии в реализации алгоритма.

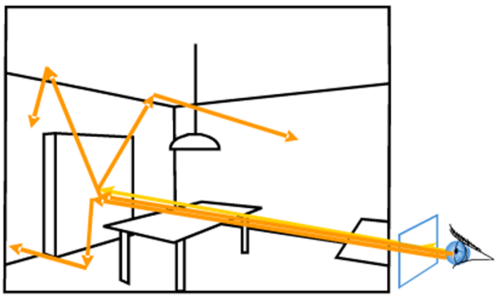


Рис. 7.4. Иллюстрация монте-карло трассировки путей.

Для начала будем считать, что в сцене все источники света имеют ненулевой размер и заданы в виде геометрических объектов, поверхность которых излучает свет. В целях упрощения изложения также будем полагать пока, что все

материалы в сцене Ламбертовы (т.е. равномернорассеивают свет во се стороны). Тогда следующий алгоритм позволит вычислить интеграл освещености:

Исходные параметры: ray - Луч, depth - глубина рекурсии
Результат: Монте-Карло выборка значения освещенности
Function TracePath(ray: Ray; depth : Integer) : Color **is**
    if depth == MAX_DEPTH **then**
        return Black;
    end
    $hit \leftarrow RaySceneIntersection(ray)$
    if not hit.exists **then**
        return Black;
    end
    $m \leftarrow hit.material$
    if m.IsLight **then**
        return m.emittance;
    end
    $newRay.O \leftarrow hit.pos$
    $newRay.D \leftarrow RandomUnitVectorInHemisphereOf(hit.norm)$
    $cosTheta \leftarrow dot(newRay.D, hit.norm)$
    $BRDF \leftarrow m.reflectance/\pi$
    $PDF \leftarrow 1/\pi$ // for RandomUnitVectorInHemisphereOf
    return (BRDF/PDF)*cosTheta*TracePath(newRay, depth + 1) ;
end

Алгоритм 1: Простейший вид Монте-Карло трассировки путей. В данном псевдокоде emittance обозначает яркость. π в знаменателе берется из определения BRDF Лабмерта.

Данный алгоритм также называют обратной трассировкой путей. Для генерации случайного вектора по полусфере необходимо реализовать функцию $RandomUnitVectorInHemisphereOf$, отображающую случайные числа r_1 и r_2 с равномерным распределением из интервала [0,1] на сферу. Такое отображение можно реализовать использую формулы 7.6 (результирующие координаты - (x, y, z)) [22].

$$\begin{aligned} \phi &= 2 * \pi * r_1 \\ h &= 2 * r_2 - 1 \\ x &= sin(\phi) * \sqrt{1 - h^2} \\ y &= cos(\phi) * \sqrt{1 - h^2} \\ z &= h \end{aligned} \tag{7.6}$$

Рис. 7.5. Иллюстрация алгоритма трассировки путей. Корректный перенос света моделируется путем просчета большого числа путей.

7.2.1. Прогрессивное вычисление интеграла

Для того чтобы иметь возможность визуализировать промежуточный результат в процессе расчета изображения, удобно использовать прогрессивную схему вычисления интеграла. Будем полагать, что мы используем простейший бокс-фильтр, усредняющий все пути для каждого пиксела. То есть цвет пиксела вычисляется по следующей формуле:

$$C = \sum_{i=1}^{N} \frac{P_i}{N} \tag{7.7}$$

Где:

1) P_i - значение Монте-Карло выборки на i-ом шаге.

2) N - число Монте-Карло выборок.

Будем вычислять цвет пиксела в виде сходящейся к C последовательности $\{C_i\}$ Пусть далее C_i - цвет пиксела изображения на i-ом проходе. Тогда он может быть вычислен прогрессивно по следующей схеме:

$$\begin{aligned} C_0 &= P_0 \\ C_1 &= \frac{1}{2}C_0 + \frac{1}{2}P_1 \\ C_2 &= \frac{2}{3}C_1 + \frac{1}{3}P_2 \\ C_3 &= \frac{3}{4}C_2 + \frac{1}{4}P_3 \\ &\ldots \\ C_n &= \frac{n}{n+1}C_{n-1} + \frac{1}{n+1}P_n \end{aligned} \tag{7.8}$$

Значение освещенности, таким образом, хранится в пикселах изображения, а выпускание случайных лучей через пикселы позволяет бороться с алиасингом. Данная схема позволяет пользователю наблюдать постепенно улучшающееся изображение (с постепенно снижающимся уровнем шума). Если пользователя устраивает полученное изображение на шаге i, он может остановить расчет.

Критерий остановки

Однако ручное управление не всегда возможно применить. Например, пользователь не может контролировать каждый отдельный пиксель изображения. Одним из наиболее стабильных критериев, позволяющих оценить точность полученного решения (и осуществить автоматическую остановку расчета) является *оценка* дисперсия D и получаемое на ее основе оценка стандартного отклонения err. Дисперсию и стандартное отклонение можно оценивать по определению (7.9):

$$D = \frac{\sum_{i=1}^{n} X_i^2}{n} - (\frac{\sum_{i=1}^{n} X_i}{n})^2 \quad (7.9)$$

$$err = \sqrt{D} \quad (7.10)$$

Для реализации формулы 7.9 достаточно накапливать прогрессивно (по схеме 7.8) сумму цветов и сумму квадратов цветов.

7.2.2. Применение выборки по значимости

Известно, что для ламбретовых поверностей наибольший вклад дают лучи близкие к направлению нормали, а наименьший - лучи почти перпендекулярные нормали. Это следует из того, что значение падающей освещенности умножается на косинус между нормалью и направлением падающей освещенности ($cosTheta$ в алгоритме 1).

Чтобы применить выборку по значимости, необходимо генерировать такой вектор отражения, который соотвествует косинусоидальному распределению вокруг направления нормали (рис. 7.6). Это можно сделать эффективно используя соотношения 7.11, которые позволяют отобразить 2 равномерно-распределенных числа на сферу с косинусоидальным распределением [22]. В соотношениях

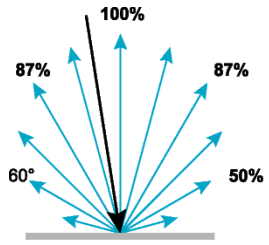


Рис. 7.6. Выборка по значимости для Ламбертовского отражения.

7.11 переменная e задает степень косинуса в косинусоидальном распределении.

$$\begin{aligned} e &= 1 \\ \phi &= 2 * \pi * r_1 \\ cosTheta &= (1 - r_2)^{1/(e+1)} \\ sinTheta &= \sqrt{1 - cosTheta^2} \\ x &= sinTheta * cos(\phi) \\ y &= sinTheta * sin(\phi) \\ z &= cosTheta \end{aligned} \tag{7.11}$$

Тогда для того, чтобы результат не изменился, необходимо делить значение BRDF (или приходящей яркости умноженной на BRDF) на плотность вероятности выбранного распределения, то есть $cosTheta$. В результате, при применении выборки по значимости для Ламбертовских материалов функция $RandomUnitVectorInHemisphereOf$ заменяется на функцию $RandomCosineVectorOf$ и алгоритм 1 переходит в алгоритм 2.

Исходные параметры: ray - Луч, depth - глубина рекурсии
Результат: Монте-Карло выборка значения освещенности
Function TracePath(ray: Ray; depth : Integer) : Color **is**
    if depth == MAX_DEPTH **then**
        return Black;
    end
    $hit \leftarrow RaySceneIntersection(ray)$
    if not hit.exists **then**
        return Black;
    end
    $m \leftarrow hit.material$
    if m.IsLight **then**
        return m.emittance;
    end
    $newRay.O \leftarrow hit.pos$
    $newRay.D \leftarrow RandomCosineVectorOf(hit.norm)$
    $cosTheta \leftarrow dot(newRay.D, hit.norm)$
    $PDF \leftarrow cosTheta/\pi$ // for RandomCosineVectorOf
    $BRDF \leftarrow m.reflectance/\pi$;
    return (BRDF/PDF)*cosTheta*TracePath(newRay, depth + 1) ;
end

Алгоритм 2: Трассировка путей с учетом косинусоидального распределения отраженных лучей.

Обратите что мы могли бы сократить член $cosTheta/\pi$ но мы оставляем его в алгоритме явно, поскольку в дальнейшем выражения для вычисление pdf и brdf будут меняться (π в знаменателе берется из определения BRDF Лабмерта).

Результат работы алгоритма (с учетом реализации идеального зеркала и преломлений с формулами френеля) представлен на рисунке 7.7. Алгоритм позволяет корректно вычислить интеграл освещенности и получить весь спектр эффектов геометричекой оптики (рис. 7.7).

7.2.3. Учет сложных материалов

Ранее был рассмотрен алгоритм трассировки путей (алгоритмы 1 и 2), и делалось предположение, что все материалы в сцене Ламбертовы. Такие материалы отражают свет равномерно во все стороны. Однако, большинство материалов реального мира обладают намного более сложными ДФО, а также могут не только отражать но и пропускать свет (рис. 7.8).

Идеально-зеркальное(S) и матовое(G) отражения моделируются при помощи единого механизма (который будет рассмотрен ниже), поэтому в дальнейшем

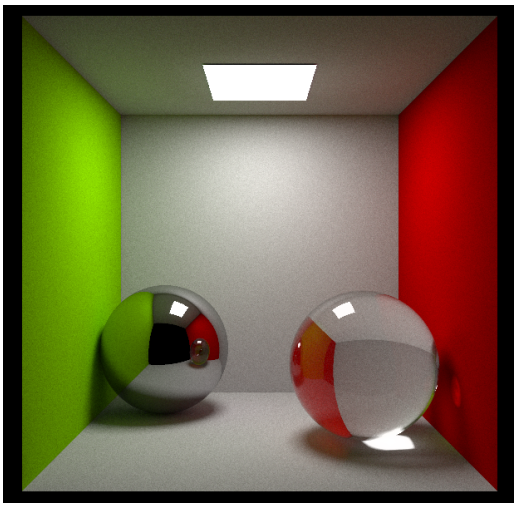


Рис. 7.7. Пример работы алгоритма трассировки путей на классической сцене Cornell Box.

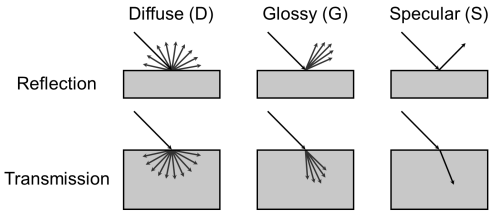


Рис. 7.8. Различные виды взаимодесйствий света с материалом. Diffuse(D) - Ламбертовское отражения. Specular(S) - зеркальное. Glossy(G) - матовое.

будем рассматривать взаимодействия G и S совместно и называть такой тип отражения зеркальным (S).

Сложные ДФО часто представляются в виде композиции более простых компонент. Классической композицией является композиция ДФО Ламберта и Фонга (или любой другой ДФО, моделирующей зеркальное и/или матовое (glossy) отражение). В этом случае ДФО получается как линейная комбинация (с коэффициентами kd и ks) отдельных компонент. Помимо этого, для прозрачных объектов необходимо учитывать пропускание (по аналогии с ДФО вводят функцию ДПФ или BTDF) с коэффициентом kt. Для того чтобы учитывать такие композитные свойства материала, используют следующий механизм для выбора отраженного/преломленного луча:

$$\Sigma = kd + ks + kt$$
$$\xi = rnd(0, 1)$$
$$\xi \in [0, \frac{kd}{\Sigma}] \Rightarrow diffuse$$
$$\xi \in [\frac{kd}{\Sigma}, \frac{kd + ks}{\Sigma}] \Rightarrow specular$$
$$\xi \in [\frac{kd + ks}{\Sigma}, 1] \Rightarrow transmittion$$

kd - коэффициент диффузного отражения. ks - коэффииент зеркального отражения. kt - коэффициент пропускания. Важно отметить, что при использовании модели выбора компоненты ДФО указанной выше, полученное значение яркости не нужно умножать на значения коэффициентов (kd, ks или kt) т.к. эти коэфициенты получаются стохастически из самой схемы (важное условие при этом чтобы $\Sigma = 1$).

Нетрудно заметить, что описанная выше схема - это снова метод Монте-Карло. Причем, если сумма $kd + ks + kt = 1$, то коэффициенты kd, ks и kt в точности равны плотностям вероятности случайной величины при выборе соответствующей компоненты. По этой причине мы не должны умножать яркость луча на соответствующий коэффициент после его выбора. Если необходимо учитывать цвет (или сумма не равна 1), схема усложняется:

$$Pd = max(kd.r, kd.g, kd.b)$$
$$Ps = max(ks.r, ks.g, ks.b)$$
$$Pt = max(kt.r, kt.g, kt.b)$$
$$\Sigma = Pd + Ps + Pt$$
$$\xi = rnd(0, 1)$$
$$\xi \in [0, \frac{Pd}{\Sigma}] \Rightarrow diffuse$$
$$\xi \in [\frac{Pd}{\Sigma}, \frac{Pd + Ps}{\Sigma}] \Rightarrow specular$$
$$\xi \in [\frac{Pd + Ps}{\Sigma}, 1] \Rightarrow transmittion$$

Однако это еще не все. Теперь полученную яркость необходимо делить на значение плотности вероятности распределения выбранной компоненты. На-

пример, если было выбрано зеркальное отражение, яркость луча необходимо пересчитать по следующей формуле:

$$color.r = color.r/(\frac{Ps}{\Sigma})$$
$$color.g = color.g/(\frac{Ps}{\Sigma})$$
$$color.b = color.b/(\frac{Ps}{\Sigma})$$

Для того чтобы применять выборку по значимости к ДФО с учетом матовых (glossy) ДФО, необходимо генерировать лучи со степенным косинусоидальным распределением в направлении отраженного вектора. Такое распределение представляет собой лепесток (cosine lobe) вокруг вектора отражения [22]. Чтобы его реализовать, необходимо изменить значение переменной e в соотношениях 7.11 присвоив ему соответствующую степень.

Отметим также, что при реализации модели Фонга, для соблюдения физической корректности рекомендуется использовать так называемую 'исправленную' модель Фонга [30]. Для двух равномерно-распределенных случайных величин ξ_1, ξ_2, в тангенциальном базисе по отношению к направлению идеального отражения (в таком базисе, что вектор идеального отражения $r = (0, 1, 0)$) получим [30]:

$$\begin{aligned} h &= \sqrt{1 - \xi_1^{\frac{2}{n+1}}} \\ (x, y, z) &= (h * cos(2\pi\xi_2), h * sin(2\pi\xi_2), \xi_1^{\frac{1}{n+1}}) \\ cos(\theta) &= dot((0, 1, 0), (x, y, z)) = y \\ brdf &= \frac{n+2}{2\pi} cos(\theta)^n \\ pdf &= \frac{n+1}{2\pi} cos(\theta)^n \end{aligned}$$

Где n - степень косинуса в оригинальной модели Фонга. (x,y,z) - новое направление отражения, распределенное в соответствии с 'исправленной моделью' вокруг вектора направления идеального отражения.

7.2.4. Теневые лучи

Представленный ранее алгоритм работает чрезвычайно долго, причем его скорость сильно зависит от размера источника освещения; алгоритм сходится тем дольше, чем меньше размер источника. Причина такой зависимости заключается в том, что вероятность случайного луча попасть в источник света зависит от его размера и расстояния до источника. Из этого следует, что трассировка путей в том виде, в котором она была рассмотрена ранее, не применима для сцен с точечными источниками. Чтобы устранить зависимость времени расчета от размера источника и добавить поддержку точечных источников света, вводят так называемые теневые лучи. Идея заключается в том, чтобы генерировать выборки в соответствие положением источника света, а не формой ДФО (рис. 7.9).

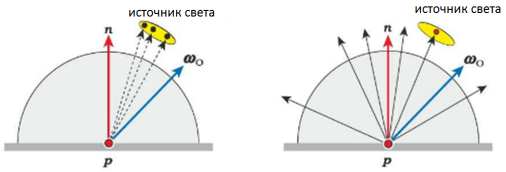


Рис. 7.9. Сэмплирование (генерация выборок) по ДФО (справа) и по источнику света (слева).Первую стратегию генерации выборок принято называть неявной. Вторую - явной.

При реализации теневых лучей следует иметь ввиду 2 важных момента. Во-первых, чтобы не учитывать источник света 2 раза, случайные лучи, полученные в результате сэмплирования ДФО при попадании в источник света должны терминироваться и возвращать черный цвет. На рисунке 7.10 изображена ситуация, при которой случайный отраженный луч попадает в источник света. Такой луч необходимо терминировать и вернуть 0 в качестве яркости от источника света.

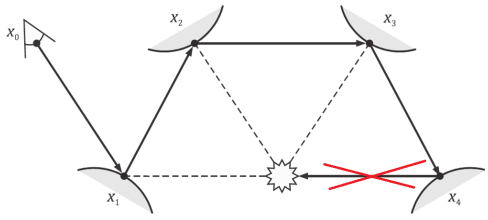


Рис. 7.10. Теневые лучи обозначены пунктиром. Отраженный луч, попавший в источник зачеркнут пеерекрестной красной линией. При возникновении такие лучи терминируются с тем чтобы не учитывать источник 2 раза.

Во-вторых, для случайного (например по полусфере) луча вероятность по-

пасть в источник света зависит от его площади поверхности и расстояния до него а для теневых лучей вероятность попадания луча в источник света равна единице по определению (поскольку такие лучи всегда испускаются точно в источник). Чтобы получить ту же самую яркость, теневые лучи необходимо учитывать специальным образом. Фактически, теневые лучи - это еще один способ сэмплирования, то есть генерации выборок в методе Монте-Карло. Если спроецировать источник света (вернее, случайно выбранную точку на нем) на полусферу вокруг точки интегрирования p (рис 7.9), для такой точки можно вычислить плотность вероятности случайной величины в сферических координатах, как если бы мы сгенерировали выборку традиционным образом - случайно по полусфере. На эту плотность вероятности в дальнейшем необходимо поделить значение яркости, пришедшее с направления теневого луча (Алгоритм 3, переменная $lgtPdf$).

Исходные параметры: ray - Луч, depth - глубина рекурсии
Результат: Монте-Карло выборка значения освещенности
Function TracePath(ray: Ray; depth : Integer) : Color **is**
    if depth == MAX_DEPTH **then**
        return Black;
    end
    $hit \leftarrow RaySceneIntersection(ray)$
    if not hit.exists **then**
        return Black;
    end
    $m \leftarrow hit.material$
    if m.IsLight **then**
        return Black;
    end
    $lpos \leftarrow LightSample(light)$
    $shadow \leftarrow Visibility(hit.pos, lpos)$
    $R \leftarrow dist(hit.pos, lpos)$
    $sdir \leftarrow normalize(lpos - hit.pos)$
    $cosTheta1 \leftarrow -dot(sdir, light.norm)$
    $cosTheta2 \leftarrow dot(sdir, hit.norm)$
    $lgtPdf \leftarrow (1.0/light.surfaceArea) * R * R/cosTheta1$
    $lgtVal \leftarrow light.intensity * (m.reflectance * cosTheta2/\pi)$;
    $explicitColor \leftarrow lgtVal/lgtPdf$;
    $newRay.O \leftarrow hit.pos$
    $newRay.D \leftarrow RandomCosineVectorOf(hit.norm)$
    $cosTheta \leftarrow dot(newRay.D, hit.norm)$
    $pdf \leftarrow cosTheta/\pi$
    $BRDF \leftarrow m.reflectance/\pi$;
    return explicitColor + (BRDF/pdf)*cosTheta*TracePath(newRay,
     depth+1);
end

Алгоритм 3: Трассировка путей с теневыми лучами. Для ламбертовых материалов и площадного источника света.

Значения light.intensity и m.emittance задают яркость точки поверхности и измеряются в канделлах на квадратный метр (cd/m^2). Чем больше источник про размеру, тем больше его площадь и тем сильнее он светит при той же яркости его поверхности. Чтобы избежать деления на ноль, для точечных источников света можно положить площадь равную 1 а вместо $light.intensity$ задавать не яркость а силу света источника (cd). Напомним, что канделла эквивалентна Ватт/Стерадиан и задает силу света источника.

Реализация трассировки путей с теневыми лучами значительно быстрее сходится (рис. 7.12) и позволяет обрабатывать источники любых размеров. Од-

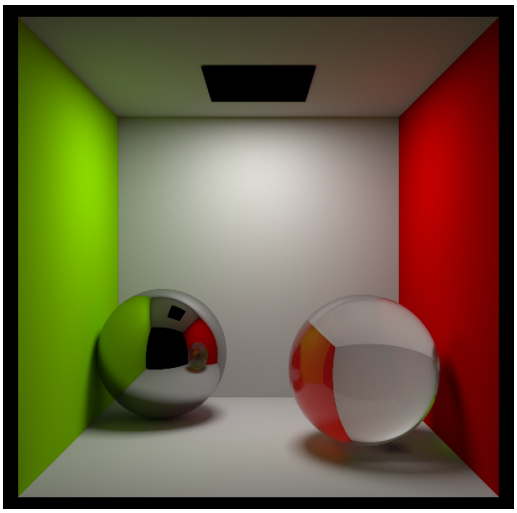


Рис. 7.11. Пример работы алгоритма трассировки путей с теневыми лучами (алгоритм 3) на классической сцене Cornell Box.

нако, как нетрудно заметить (рис. 7.11), источник света стал черным и на изображении исчезают каустики. Первую проблему (черный источник света) легко исправить если возвращать цвет источника вместо черного при отсутствии диффузных и матовых (glossy) переотражений. Это всегда можно делать поскольку теневые лучи применяются только для учета матовой и диффузной компонент ДФО.

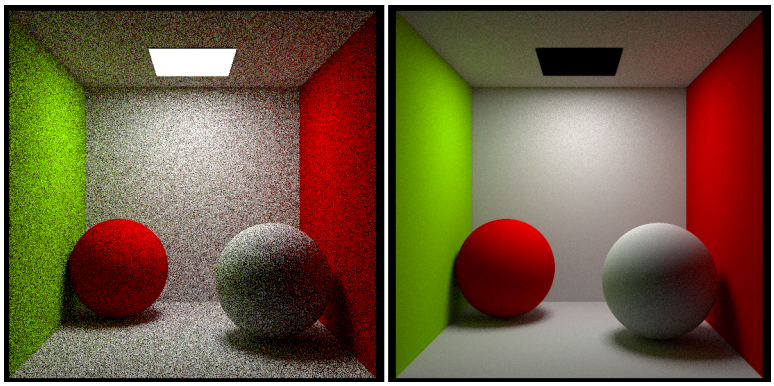


Рис. 7.12. Сравнение простого алгоритма трассировки путей (64 путя на пиксел, слева на изображении) и алгоритма трассировки путей использующего теневые лучи (32 путя на пиксел, справа на изображении). Оба изображения получены за примерно одинаковое время.

Вторая проблема - отсутствие каустик. Чтобы вернуть каустики, определим для начала в каких условиях они возникают. Каустик - яркое пятно на диффузной поверхности, вызванное непрямым освещением посредством зеркаль-

ных отражений (или преломлений) света, излученного из источника. Таким образом, если в процессе трассироки путей после диффузного переотражения встретилось зеркальное, на месте упомянутого диффузного отражения может возникнуть каустик. Добавить каустики в трассировку путей с теневыми лучами, таким образом, достаточно просто:

1) Трассируем путь из глаза с применением теневых лучей до тех пор пока не встретилось зеркальное переотражение после диффузного.

2) Как только указанное условие выполнилось, превращаем оставшуюся часть пути в 'каустический путь'. Такой путь не использует теневые лучи и строит оставшуюся его часть руководствуется только формой ДФО.

В результате получем изображение на рисунке 7.7.

7.2.5. Две стратегии сэмплирования

Ранее были рассмотрены стратегими сэмплирования (стратегиями генерации выборок) - сэмплирование по ДФО и сэмплирование по источнику света (рис. 7.9). Будем называть сэмплирование по ДФО неявной стратегией, а сэмплироание по источнику - явной.

Для того чтобы иметь корректный результат, мы могли использовать явную и неявную стратегии (превращая оставшуюся часть пути в 'каустический' путь), но не могли смешивать их. То есть если сгенерированный по неявной стратегии путь попадал в источник света из той-же точки, в которой уже был учтен теневой луч по явной стратегии, неявный луч приходилось исключать из расчетов. Однако, в этом случае эффективность расчета падает, особенно если неявных лучей, попадющих в источник, много. Далее рассмотрим, как можно смешивать явную и неявную стратегии чтобы учитывать их одновременно и избежать избыточных вычислений.

7.2.6. Многократная выборка по значимости

Для того чтобы корректно комбинировать явную и неявную стратегии, Эриком Вичем был предложен метод многократной выборки по значимости (англ. Multiple Importance Sampling, MIS) [31], который позволяет скомбинировать результаты нескольких способов выборки (нескольких стратегий сэмплирования) в одну несмещенную оценку. На практике реализация многократной выборки по значимости имеет некоторые неочевидные, но существенные детали. Мы рассмотрим их далее.

Пусть мы имеем сцену с ламбертовскими материалами и площадным источника света (с площадью A) с равномерным распределением. Пусть мы находимся в некоторой точке X на поверхности. И пусть для явной стратегии мы выбираем на источники точку L_{pos}. Пусть L_e - яркость пришедшая от источника света в направлении теневого луча, а L_i яркость в точке источника, куда попал случайно отраженный луч.

Тогда для ламбертовского материала плотность вероятности случайной величины, распределенной косинусоидально, будет $p_i = cos(\theta)/\pi$ где θ - угол между нормалью и отраженным лучом. А плотность вероятности случайной величины при сэмплировании источника света будет $p_e = \frac{d^2}{cos(\phi)*A}$. Где d - расстояние от точки на поверхности до точки пересечения луча и поверхности источника лучом при сэмплировании явной стратегией ($d = dist(L_{pos}, X)$), а $cos(\phi)$ - угол между нормалью источника и направлением луча (при сэмплировании неявной стратегией) умноженного на -1. Следует отметить, что поскольку мы выбираем на источнике точку L_{pos} равномерно, изначальная плотность вероятности при сэмплировании площадного источника равна $\frac{1}{A}$. Множитель $\frac{d^2}{cos(\phi)}$ переводит плотность вероятности в радиальную систему координат - в ту же самую систему, в которой задана плотность вероятности $p_i = cos(\theta)/\pi$. Итак, имеем:

$$\begin{aligned} p_i &= cos(\theta)/\pi \\ p_e &= \frac{d^2}{cos(\phi) * A} \\ w_e &= \frac{p_e^\beta}{p_i^\beta + p_e^\beta} \\ w_i &= \frac{p_i^\beta}{p_i^\beta + p_e^\beta} \\ \beta &= 2 \end{aligned}$$

w_i - вес для учета яркости полученной по неявной стратегии. w_e - вес для учета яркости полученной по явной стратегии. Схема многократной выборки по значимости записывается по формуле 7.12.

$$L_k = w_e \frac{L_e}{p_e} + w_i \frac{L_i}{p_i} \tag{7.12}$$

Оптимальность такого взвешивания показана в [31].

Детали реализации

Казалось бы, все переменные известны и мы готовы к применению многократной выборки по значимости. Достаточно лишь хранить раздельно яркости L_e, L_i и соответствующие плотности вероятности. Если случайно-выстреленный луч попал в источник, используем многократную выборку по значимости. Если не попал - не используем и просто складываем L_e и L_i. Однако, это приведет к неправильному результату. Причина этого заключается в невыполнении некоторого важного условия MIS. Сумма весов для обоих стратегий должна быть стохастически равна 1.

Рассмотрим, что это означает. Пусть мы имеем серию выборок (или сэмплов, экспериментов) $f(x_1)$ с плотностью вероятности случайной величины x_1 равной p_1. Пусть также мы имеем другую серию экспериментов $f(x_2)$ с плотностью вероятности случайной величины x_2 равной p_2. Имея две серии экспериментов мы можем оценивать значения интеграла при помощи метода Монте-Карло (уравнение 7.13):

$$I_1 = \frac{1}{N1}\sum_{k=0}^{N1}\frac{f(x_{1,k})}{p_{1,k}} \tag{7.13}$$

$$I_2 = \frac{1}{N2}\sum_{k=0}^{N2}\frac{f(x_{2,k})}{p_{2,k}} \tag{7.14}$$

Поскольку обе оценки в пределе сходятся к одному и тому же значению, имеем право смешивать их любыми коэффициентами, сумма которых равна 1 - например так $I_{new} = 0.25 * I_1 + 0.75 * I_2$. Обратите внимание, что весовые коэффициенты вышли за знак суммы. Это и есть условие, обсуждаемое ранее - сумма весов для обоих стратегий должна быть стохастически равна 1. То есть:

$$\frac{1}{N1}\sum_{k=0}^{N1} w_{i,k} + \frac{1}{N2}\sum_{k=0}^{N2} w_{e,k} = 1 \tag{7.15}$$

$$\tag{7.16}$$

При этом следует отметить, что сумма весов на каждой итерации k вовсе не обязана быть равна единице, более того - число итераций для разных стратегий может быть разным ($N1$ и $N2$). Теперь рассмотрим рисунок 7.13 и алгоритм, правильно вычисляющий веса w_i и w_e.

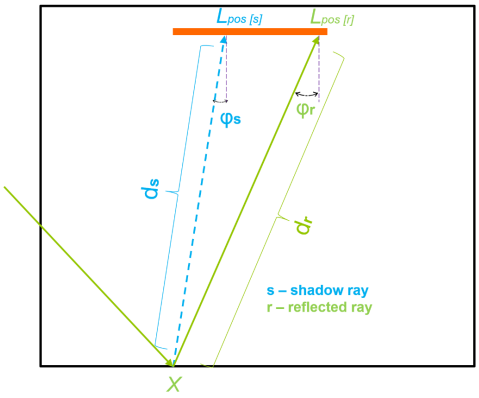


Рис. 7.13. Сэмплирование источника двумя различными стратегиями. Теневой луч (синий, суффикс s) отвечает за явную стратегию. Случайно-отраженный (зеленый, суффикс r) - за неявную стратегию.

Для того чтобы правильно вычислить вес w_i вам необходимо знать не только плотность вероятности явной стратегии для данной точки ($L_{pos[s]}$), но и плотность вероятности для неявной стратегии - как если бы вы попали в точности в ту же самую точку ($L_{pos[s]}$). Аналогично и для неявной стратегии. Когда вы попадаете в источник света, необходимо вычислять плотность вероятности попадания в ту же самую точку $L_{pos[r]}$ теневым лучом. Сопоставим обозначения на рисунке 7.13 и в алгоритме 4:

- D - d_r
- R - d_s
- $cosTheta1$ - косинус угла ϕ_s
- $dot(light.norm, -ray.direction)$ - косинус угла ϕ_r
- $lpos$ - $L_{pos[s]}$
- $prevHit$ - позиция и нормаль к поверхности в точке X.

Исходные параметры: ray - Луч, depth - глубина рекурсии
Результат: Монте-Карло выборка значения освещенности
Function TracePath(ray: Ray; depth : Integer) : Color **is**
    if depth == MAX_DEPTH **then**
        return Black;
    end
    $hit \leftarrow RaySceneIntersection(ray)$
    if not hit.exists **then**
        return Black;
    end
    $m \leftarrow hit.material$
    if m.IsLight **then**
        $D \leftarrow dist(prevHit.position, hit.position)$
        $p_e \leftarrow D^2/(light.surfaceArea * dot(light.norm, -ray.direction))$
        $p_i \leftarrow dot(prevHit.norm, ray.direction)/\pi$
        $w_i \leftarrow p_i^2/(p_e^2 + p_i^2)$
        return light.intensity*w_i ;
    end
    $lpos \leftarrow LightSample(light)$
    $shadow \leftarrow Visibility(hit.pos, lpos)$
    $R \leftarrow dist(hit.pos, lpos)$
    $sdir \leftarrow normalize(lpos - hit.pos)$
    $cosTheta1 \leftarrow -dot(sdir, light.norm)$
    $cosTheta2 \leftarrow dot(sdir, hit.norm)$
    $lgtPdf \leftarrow (1.0/light.surfaceArea) * R * R/cosTheta1$
    $lgtVal \leftarrow light.intensity * (m.reflectance * cosTheta2/\pi)$;
    $explicitColor \leftarrow lgtVal/lgtPdf$;
    $newRay.O \leftarrow hit.pos$
    $newRay.D \leftarrow RandomCosineVectorOf(hit.norm)$
    $cosTheta \leftarrow dot(newRay.D, hit.norm)$
    $p_e \leftarrow R^2/(light.surfaceArea * cosTheta1)$
    $p_i \leftarrow cosTheta/\pi$
    $w_e \leftarrow p_e^2/(p_e^2 + p_i^2)$
    $BRDF \leftarrow m.reflectance/\pi$;
    return explicitColor*w_e + (BRDF/pdf)*cosTheta*TracePath(newRay, depth+1);
end

Алгоритм 4: Трассировка путей с многократной выборкой по значимости (MIS). Для ламбертовских материалов и площадного источника света.

Хорошим примером реализации техники многократной выборки по значимости может служить соответствующи пример в [32] и реализация в прилагаемых к книге исходных кодах простого трассировщика путей [33]. Ситуация, в которой многократная выборка по значимости действительно дает суще-

ственный выигрыш изображена на рис 7.14.

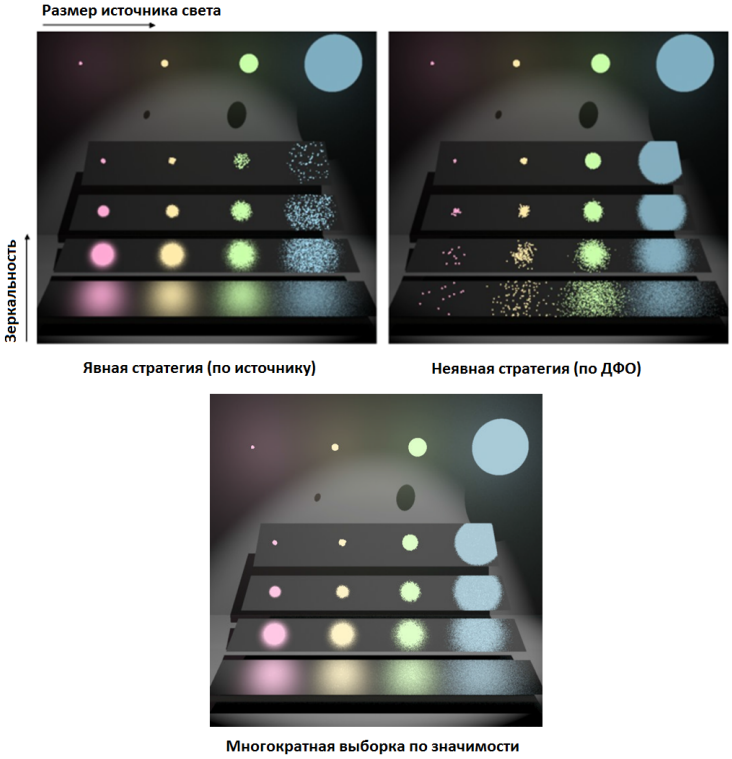


Рис. 7.14. Сравнение различных стратегий генерации выборки для сцены с матовым (glossy) отражением. Явная стратегия дает лучший результат, если источник находится далеко от повехности или имеет малый размер. Неявная стратегия дает лучший результат, если источник имеет большой размер или находится близко к поверхности. Многократная выборка по значимости использует преимущества обоих пододов.

Обобщим алгоритм 4 на случай произвольных материалов и источников света:

Исходные параметры: ray - Луч, prevSample - сэмпл BRDF на предыдущем уровне рекурсии, depth - глубина рекурсии
Результат: Монте-Карло выборка значения освещенности
Function TracePath(ray: Ray; prevSample : BRDFSample depth : Integer) : Color **is**

```
if depth == MAX_DEPTH then
    return Black;
end
hit ← RaySceneIntersection(ray)

if not hit.exists then
    return Black;
end

m ← hit.material
if m.IsLight then
    D ← dist(prevHit.position, hit.position)
    p_e ← light.EvalPDF(ray.O, ray.D, D) //abstract
    p_i ← prevSample.pdf
    w_i ← p_i^2/(p_e^2 + p_i^2)
    return light.intensity*w_i ;
end

lightSam ← light.Sample(hit.position) //abstract
shadow ← Visibility(hit.pos, lightSam.shadowRayD)
shadeColor ← m.EvalBRDF(lightSam.shadowRayD, hit.normal, ...)
brdfPdf ← m.EvalPDF(lightSam.shadowRayD, hit.normal, ...)
explicitColor ← shadow * (lightSam.intensity/lightSam.pdf) * shadeColor;

brdfSam ← m.Sample(ray.direction, hit.normal, ...)
newRay.O ← hit.pos
newRay.D ← brdfSam.rayDir
p_e ← lightSam.pdf
p_i ← brdfPdf
w_e ← p_e^2/(p_e^2 + p_i^2)
cosTheta ← dot(newRay.D, hit.norm)
return explicitColor*w_e +
 (brdfSam.val/brdfSam.pdf)*cosTheta*TracePath(newRay, brdfSam,
 depth+1);
```

end

Алгоритм 5: Трассировка путей с многократной выборкой по значимости (MIS). С поддержкой различных типов источников и материалов.

Обратите внимания на 5 абстрактных (виртуальных) функции в алгоритме 5.
Это $light.Sample$, $light.EvalPDF$, $m.Sample$, $m.EvalPDF$ и $m.EvalBRDF$. Рассмотрми их более подробно:

1) $light.Sample$ - генерирует Монте-Карло выборку (сэмпл) при помощи явной стратегии для источника света $light$ и заданной позиции освещаемой им точки $hit.position$. $lightSam.shadowRayD$ - направление теневого луча, вылетающего из точки $hit.position$ в некоторую точку на источнике. $lightSam.intensity$ - значение выборки яркости источника, $lightSam.pdf$ - плотность вероятности для этой выборки в сферических координатах. Таким образом, перевод плотности вероятности в сферические координаты ложится на плечи методов источника и для каждого источника, вообще говоря, должен быть реализован по своему.

2) $light.EvalPDF$ - при попадании случайного луча (получаемого при помощи неявной стратегии) в источник света вычисляет плотность вероятности, равную плотности вероятности явной стратегии для данного луча (как если бы этот же луч был получен при помощи явной стратегии). Данная может быть использована также внутри $light.Sample$ при расчете $lightSam.pdf$.

3) $m.Sample$ Генерирует Монте-Карло выборку при помощи неявной стратегии, сэмплируя материал. Возможно, более правильное название для данной функции $SampleAndEvalBRDF$.

4) $m.EvalPDF$ - используется при вычислении вклада теневого луча и вычисляет плотность вероятности для теневого луча, как если бы он был получен при помощи неявной стратегии.

5) $m.EvalBRDF$ - вычисляет значение BRDF для заданного теневого луча.

7.2.7. Русская рулетка и глубина трассировки

Одной из причин низкой эффективности трассировки путей путей может являться завышенное значение максимальной глубины трассировки. Стеклянные объекты могут требовать значительного числа переотражений для корректной визуализации. В то же время, лучи попавшие на диффузные объекты, как правило, уже после нескольких переотражений теряют значимость. Одним из простых решений является терменирование пути при достижении определенного числа диффузных отскоков и/или низкого 'коэффициента пропускания' (произведение коэффициентов отражения во всех точках пути).

Параметр терменирования при этом требует ручной настройки. Однако, такой метод вносит смещение в получаемое решение и может дать неверный результат на сцене с очень яркими источниками и сложным перераспределением световой энергии [34].

Одно из решений, позволяющих автоматически вбирать глубину терминирования пути - стохастическое терминирования пути на освнове механизма русской рулетки (алгоритм 6).

Фактически, русская рулетка позволяет трассировать меньше путей на большой глубине. Однако, она присваивает им большую значимость, за счет чего в пределе получается верный результат.

Исходные параметры: ray - Луч, depth - глубина рекурсии
Результат: Монте-Карло выборка значения освещенности

```
Function TracePath(ray: Ray; depth : Integer) : Color is
    pabsorb ← terminationProbability(depth, ...)
    if random(0.0, 1.0) < pabsorb then
        return Black;
    end
    ...;
    return result*(1/(1-pabsorb)) ;
end
```

Алгоритм 6: Трассировка путей с русской рулеткой.

Механихм русской рулетки, вообще говоря, увеличивает дисперсию в оцениваемом решении [35], поэтому должен применяться осторожно - только тогда, когда ожидаемый вклад от текущего пути достаточно мал. В этом случае:

1) Дисперсия повысится незначительно, так как в среднем вклад от путей, для которых применяется русская рулетка относительно мал.

2) По сравнению с простым терминированием, время трассировки не увеличится, т.к. число лучей, трассируемых на большую глубину, невелико.

3) Решение станет более точным за счет учета большего числа переотражений.

7.2.8. Резюме по обратной трассировке путей

Рассмотренный алгоритм обратной трассировки путей с применением теневых лучей и многократной выборки по значимости является несмещенным и достаточно эффективным методом для вычисления интеграла освещенности;

за исключением каустик, которые вычисляются этим алгоритмом довольно медленно. Для эффективного вычисления каустик далее будет рассмотрен алгоритмы прямой трассировки путей (Light Tracing) и алгоритм фотонных карт.

7.2.9. Грамматика путей

Для классификации различных ситуаций, возникающих в процессе трассировки путей часто используют понятие грамматики путей [36]. Этот способ классифицирует пути при помощи строк и регулярных выражений. Каждый символ в строке обозначает определенное событие, произошеднее с лучом (путем) в процессе трассировки.

1) S - (Specular); зеркальное отражение/преломление

2) D - (Diffuse); диффузное отражение

3) G - (Glossy); матовое отражение/преломление

4) V - (Volume); объемное рассеивание

5) L - (Light); источник света

6) E - (Eye); глаз

При этом под символами S и G понимается не только отражение но и преломление света. Грамматика удобна не только для описания различных ситуаций, возникающих в трассировке путей, но она позволяет также классифицировать видимые эффекты, однозначно ставя им в соответствие классы путей которые эти эффекты вызывают (для упрощения изложения материала объемное рассеивание V мы далее не рассматриваем). Например, EDL - прямое освещение диффузной поверхности. $E(S|G)^+L$ - яркий блик. $EDSL$ - каустик, видимый из камеры напрямую, обусловленный одним зеркальным переотражением света (например солнечный зайчик от зеркала). EDS^+L - каустик, видимый из камеры напрямую, вызванный одни или более зеркальным переотражением. ES^+DS^+L - каустик, видимый через стекло или зеркало, вызванный одним или более зеркальным переотражением.

При помощи грамматики путей удобно классифицировать алгоритмы по типу освещения, которые они способны расчитывать. Например, Трассировка лучей Уиттеда, рассмотренная нами в самом начале, строит пути ES^*L. Обратная трассировка путей строит пути вида $E(S|D|G)^*L$.

7.3. Прямая трассировка (Light Tracing)

В противоположность обратной трассировки путей, рассмотренной выше, алгоритм прямой трассировки путей начинает свою работу из источника света и строит пути вида $L(S|D|G)^*E$. При каждом соударении с поверхностью создается аналог теневого луча, направленный в камеру. Если луч успешно доходит до камеры, значение яркости записывается в пиксель, соответствующий проекции начала луча на экранную плоскость. Для реализации операции проекции вы можете использовать матрицу $mProj$, которая была рассмотрена нами в разделе 'Генерация луча'.

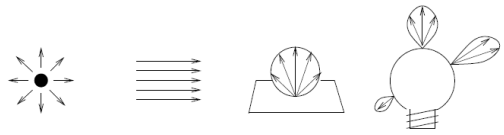


Рис. 7.15. Иллюстрация различных видов распределения световой энергии для различных источников света.

Пути создаваемые на источнике должны быть сгенерированы в соответствии с распределением световой энергии для источника света (рис. 7.15). Обратите внимание, что если каждая точка поверхности излучает свет равномерно во все стороны (например площадный источник из рис. 7.7), то исходя из геометрических соображений и применяя выборку по значимости, необходимо использовать косинусоидальное распределение для расчета направления пути точно так же, как мы вычисляли ранее диффузное отражение. В обратной трассировке путей данный косинус получался стохастически, за счет того что число лучей попавших в источник зависит от направления, с которого они были выпущены.

Существующий миф о крайне низкой скорости прямой трассировки путей (или лучей) далеко не всегда соответствует действительности. Такие эффекты как каустики чрезвычайно быстро могут быть расчитаны именно при помощи прямой трассировки. В связи с этим, возникает вопрос о том, можно ли комбинировать прямую и обратную трассировки путей для того, чтобы исключить недостатки обоих алгоритмов.

7.3.1. Усечённая двунаправленная трассировка путей

Метод, реализующий эту идею существует и называется «усеченная двунаправленная трассировка путей» [34]. Идея этого метода в упрощенном представлении изложена ниже:

1) Будем производить обратную трассировку путей с теневыми лучами. При этом отключим расчет каустиков. Результат сохраняем в изображение с номером 1. На изображении 1 мы полчим картинку без каустиков.

2) Будем производить прямую трассировку из источника света, но на этот раз будем сохранять вклад только от тех путей, для которых первое переотражение было зеркальным. Результат сохраняем в изображение с номером 2. На изображении 2 мы получим только каустики.

3) Финальное изображение можно получить простым сложением изображений 1 и 2.

Усеченная двунаправленная трассировка путей дает почти корректный результат, поскольку изображения 1 и 2 будут нести в себе освещение, обусловленное различными типами переотражений. Слово 'почти' означает, что при помощи такого метода не удасться эффективно рассчитать каустики, не видимые камерой напрямую. То есть не удасться эффективно вычислять освещение, вызванное путями вида ES^+DS^+L. Время расчета таких эфектов будет сопостовимо со временем их рассчета обыкновенной трассировкой путей. Предлагаемая в работе [37] модификация метода усеченной двунаправленной трассировки путей на основе регуляризации позволяет устранить недостаток метода и сделать его эффективным при рассчете путей содержащих S^+DS^+ фрагменты.

Мы рассмотрели идею «усеченной двунаправленной трассировка путей» в упрощённом изложении. В действительности усечённая двунаправленная трассировка путей предлагает не просто складывать вклад от различных стратегий, а учитывать вклад при помощи многократной выборки по значимости. В отличие от обычной трассировки путей (где используется две стратегии - явная и неявная) здесь добавляется ещё одна, - прямая. Это стратегия для сформированного от источника пути. Далее в разделе 7.4.1 мы увидим, что «усеченная двунаправленная трассировка путей» имеет такое название потому что при использовании MIS она является частным случаем двунаправленной трассировки путей с более сложной схемой применения MIS.

Если в обычной трассировке путей MIS использует 2 стратегии (неявную и явную), то в «усеченной двунаправленной трассировка путей» к ним добавляется ещё одна - «световая стратегия» [34]. Это стратегия, формируемая прямой трассировкой от источника. Далее мы увидим, что в двунаправленной трассировке путей таких стратегий может быть много.

7.4. Двунаправленная трассировка путей

Идея объеденить преимущества прямой и обратной трассировки привела к созданию алгоритма двунаправленной трассировки путей (Bidirectaional Path Tracing, BPT) [31]. Двунаправленной трассировки путей базируется на 2 основных идеях:

1) Соединять теневыми лучами не только источник с точками пути (как в обратной трассировки) или камеру с точками путя (как в прямой трассировки), но и различные точки путей друг с другом (рис. 7.16). В действительности, в BPT точки от двух путей соединяются по принципу **каждый с каждым** (рис. 7.17) в целях переиспольвования уже посчитанных вершин.

2) Применить многократную выборку по значимости не только для сэмплирования источника (как было рассмотрено нами ранее), но на всех уровнях переотражений.

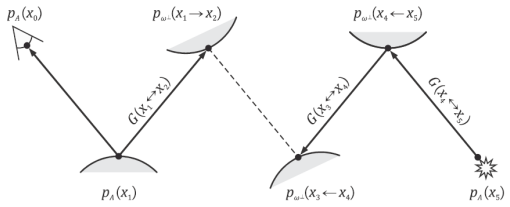


Рис. 7.16. Идея двунаправленной трассировки путей. На данном рисунке показан упрощённый вариант соединения конечных точек, в то время как в оригинальном BPT все точки одного пути должны соединяться со всеми точками другого пути 7.17.

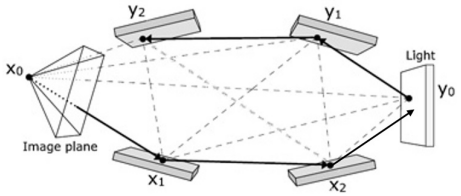


Рис. 7.17. Соединение по принципу каждый с каждым. Случайное попадание луча в камеру на данном рисунке не учитывается, хотя в общем случае (если диафрагма имеет существенный размер), может появится ещё один дополнительный луч $y_2 \to x_0$.

Как уже было сказано, в оригинальном методе двунаправленной трассировке путей используются соединения каждый с каждым – формируя, таким обра-

зом, сразу несколько итоговых путей света от источника до камеры. На рис. 7.16 изображён только один такой итоговый путь. Не ограничивая общности далее будем рассматривать только итоговые пути без относительно того как именно они были сформированы (по сехме на рис. 7.16 или как один из итоговых путей на рис. 7.17).

Двунаправленная трассировка путей обладает лучшей сходимостью (по сравнению с прямой или обратной трассирокой) на сценах со сложными условиями освещения (преобладающее вторичное освещение) только при реализации многократной выборки по значимости. В противном случае, изображение получается сильно зашумленным. Однако даже при реализации многократной выборки по значимости, двунаправленная трассировка путей по прежнему не позволяет эффективно рассчитывать пути вида ES^+DS^+L (каустики, видимые через зеркало или стекло). Такие пути могут быть эффективно вычислены при помощи фотонных карт, переноса света Метрополиса и метода соединения вершин (vertex merging)[38].

Поскольку теневые лучи в двунаправленной трасировки могут соединять произвольные точки различных путей, реализация многократной выборки по значимости значительно усложняется. Если ранее для вычисления весов достаточно было рассмотреть только 2 стратегии, то теперь число рассматриваемых вариантов значительно увеличивается.

7.4.1. Многократная выборка по значимости в BPT

Абстрактный пример

Расмотрим сначала абстрактный пример, не вдаваясь в детали. Пусть имеем некоторый путь $x_0 \dots x_3$, где x_0 - точка на источнике, а x_3 - точка на поверхности сенсора камеры. Для такой последовательности точек существует несколько способов построения этого пути - т.е. несколько результирующих стратегий. Соединение теневым лучём обозначено вертикальной чертой, трассировка от источника и камеры обозначены стрелками:

$$
\begin{aligned}
x_0 \to x_1 \to x_2 \to x_3 & \\
x_0 \to x_1 \to x_2 \quad & | \quad x_3 \\
x_0 \to x_1 \quad & | \quad x_2 \leftarrow x_3 \\
x_0 \quad & | \quad x_1 \leftarrow x_2 \leftarrow x_3 \\
& | \quad x_0 \leftarrow x_1 \leftarrow x_2 \leftarrow x_3
\end{aligned}
$$

При вычислении весов для какого-либо из случаев, многократная выборка по значимости должна учитывать все остальные случаи. Формула 7.17 применяется для вычисления веса конкретного пути.

$$w_i = \frac{p_i^\beta}{\sum_{j=1}^{N} p_j^\beta} \tag{7.17}$$

Например, для случая $x_0 \to x_1 \to x_2 | x_3$ i равно 3 т.к. это третий по счету вариант; N равняется 5 т.к. всего вариантов 5. Причем, p_3 - произведение всех плотностей вероятности на протяжении всего пути. В данном случае $p_3 = p(x_0 \to x_1)_{implicit} * p(x_1 \to x_2)_{implicit} * p(x_2 \to x_3)_{explicit}$. Таким образом, для указанного случая из 4 точек $x_0..x_3$ необходимо просчитать 5 плотностей вероятности для 5 возможных стратегий:

$$\begin{aligned}
PDF_4 &= p(x_0 \to x_1)_{implicit} * p(x_1 \to x_2)_{implicit} * p(x_2 \to x_3)_{implicit} \\
PDF_3 &= p(x_0 \to x_1)_{implicit} * p(x_1 \to x_2)_{implicit} * p(x_2 \to x_3)_{explicit} \\
PDF_2 &= p(x_0 \to x_1)_{implicit} * p(x_1 \to x_2)_{explicit} * p(x_2 \leftarrow x_3)_{implicit} \\
PDF_1 &= p(x_0 \to x_1)_{explicit} * p(x_1 \leftarrow x_2)_{implicit} * p(x_2 \leftarrow x_3)_{implicit} \\
PDF_0 &= p(x_0 \leftarrow x_1)_{implicit} * p(x_1 \leftarrow x_2)_{implicit} * p(x_2 \leftarrow x_3)_{implicit}
\end{aligned}$$

Обычно PDF_4 полагают равной нулю, поскольку вероятность (и соотв. плотность $p(x_2 \to x_3)_{implicit}$) попасть случайным лучом в диафрагму камеру считается очень малой. Далее мы также будем рассматривать только 4 стратегии $PDF_0..PDF_3$.

Конкретный пример

В действительности одной из самых больших проблем при реализации двунаправленной трассировки путей является рассчёт плотностей вероятности для каждого случайного события. Чтобы итоговое произведение плотностей имело физический смысл (иначе вес будет просто неправильным), рекомендуется вычислять все плотности вероятности в единой системе измерений, причём вполне конкретной - площадной [19]. Таким образом, плотность вероятности представляет из себя «вероятность/м^2». Следует уделить особое внимание при вычислении площадной плотности вероятности, поскольку типичные реализации в рендерах считают плотность вероятности по телесному углу P^W.

Далее плотность вероятности по спроецированному телесному углу $P^{WP} = P^W/cos(\theta_{here})$ [34], а площадная плотность вероятности P^A вычисляется как $P^A = (P^{WP} * G) = (P^W/cos(\theta_{here})) * G$ где G - геометрический терм [31], [34]. Поскольку $cos(\theta_{here})$ присутствует внутри G, во многих рендер-системах (например в SmallVCM [39]) этот косинус (рис. 7.18) сокращают. Однако мы не рекомендуем это делать т.к. это может служить источником потенциальных ошибок, когда при переходе от P^W к P^A из-за того что будет взят не тот косинус.

Рассмотрим теперь более подробно как вычисляются веса и соответсвующие плотности вероятности для 3 переотражений (соответствущих 4 вершинам). Для данного случая (длинна пути равна 3) выпишем плотности вероятности для различных событий:

- $P_0^A = \frac{1}{A}$; Выбор случайной точки на источнике (lightSample.pdfA = 1/light.surfaceArea); A - площадь источника.

- $\overrightarrow{P_L}$ – Выбор случайного направления. Равен (lightSample.pdfW/sample.cosTheta) где lightSample.pdfW - плотность вероятности при выборе направления по телесному углу (для ламбертовских источников она равна sample.cosTheta/PI, т.е. косинус сокращается и в итоге $\overrightarrow{P_L} = \frac{1}{\pi}$ есть плотность вероятности по спроецированному телесному углу).

- $\overrightarrow{P_1}$ – (matSam.pdf/cosNext) или (reversePdfW/cosCurr); Плотность вероятности по спроецированному телесному углу. Выбираем случайное направление когда мы сэмплируем материал на первом переотражени со стороны источника. При этом matSam.pdf - плотность вероятности по телесному углу при сэмплировании материала; reversePdfW - плотность вероятности сэмплирования материала в *обратном* (от камеры к источнику) направлении по телесному углу.

- $\overrightarrow{P_2}$ – аналогично $\overrightarrow{P_1}$, но уже для второго переотражения со стороны источника.

- $\overleftarrow{P_2}$ – (matSam.pdf/cosNext) или (forwardPdfW/cosCurr); Плотность вероятности по спроецированному телесному углу. Выбираем случайное направление когда мы сэмплируем материал на первом переотражени со стороны камеры. При этом matSam.pdf - плотность вероятности по телесному углу при сэмплировании материала; forwardPdfW - плотность вероятности сэмплирования материала в *прямом* (от источника к камере) направлении по телесному углу.

- $\overleftarrow{P_1}$ – аналогично $\overleftarrow{P_2}$, но уже для второго переотражения со стороны камеры.

- P^A_{cam} – (imageToSurfaceFactor/N); Плотность вероятности при выборе случайной точки на сэнсоре, сконвертированная в площадную меру для первой точки пересечения луча из камеры и поверхности. Как она вычисляется можно посмотерть в [34] или [39]. N - число сэмплов (обычно полагается равным разрешению изображения width*height).

- G_1, G_2 - Геометрические термы [31]. $G = \frac{cos(\theta_{here})cos(\theta_{prev})}{R^2}$.

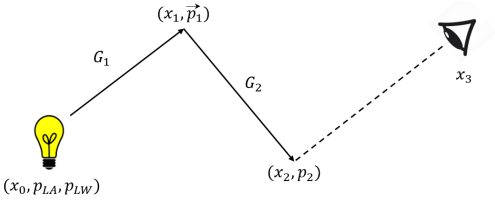
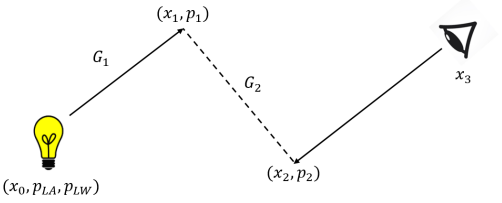
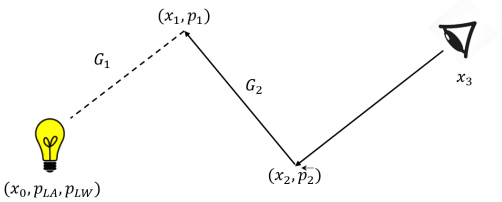
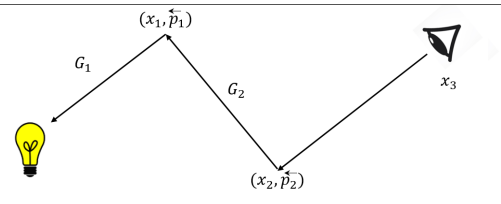
	(s=3,t=0) $P_{acc0} = P_0^A * (\overrightarrow{P_L} * G_1) * (\overrightarrow{P_1} * G_2) * 1$
	(s=2,t=1) $P_{acc1} = P_0^A * (\overrightarrow{P_L} * G_1) * 1 * P^A_{cam}$
	(s=1,t=2) $P_{acc2} = P_0^A * 1 * (\overleftarrow{P_2} * G_2) * P^A_{cam}$
	(s=0,t=3) $P_{acc3} = 1 * (\overleftarrow{P_1} * G_1) * (\overleftarrow{P_2} * G_2) * P^A_{cam}$

Таблица 7.1. Различные стратегии построения пути глубиной 3 в BPT.

$$
\begin{aligned}
P_{acc0} &= P_0^A * (\overrightarrow{P_0} * G_1) * (\overrightarrow{P_1} * G_2) * 1 \\
P_{acc1} &= P_0^A * (\overrightarrow{P_0} * G_1) * 1 * P_{cam}^A \\
P_{acc2} &= P_0^A * 1 * (\overleftarrow{P_2} * G_2) * P_{cam}^A \\
P_{acc3} &= 1 * (\overleftarrow{P_1} * G_1) * (\overleftarrow{P_2} * G_2) * P_{cam}^A
\end{aligned}
$$

Для того чтобы вычислять плотности вероятности итоговых стратегий для произвольной глубины мы предлагаем ввести два массива - $\overrightarrow{Array}$ и $\overleftarrow{Array}$. Во время трассировки в эти массивы записываются площадные плотности вероятности для каждого отражения:

$$
\begin{aligned}
\overrightarrow{Array} &= \{P_0^A, (\overrightarrow{P_0} * G_1), (\overrightarrow{P_1} * G_2), 1\} \\
\overleftarrow{Array} &= \{1, (\overleftarrow{P_1} * G_1), (\overleftarrow{P_2} * G_2), P_{cam}^A\}
\end{aligned}
$$

Нулевой элемент массивов $\overrightarrow{Array}$ и $\overleftarrow{Array}$ хранит плотность вероятности события выбора точки на источнике. Обратите внимание что $\overleftarrow{Array}[0] \equiv 1$ поскольку рассматривая трассировку в обратном направлении (от глаза к источнику) попадание в источник происходит неявно – т.е. когда мы попали в источник, мы *уже* попали в него. Элемент в $\overrightarrow{Array}[1]$ хранит площадную плотность вероятности для события выбора направления на источнике. Первый элемент в $\overleftarrow{Array}[1]$ хранит площадную плотность вероятности для события выбора случайного направления при трассировке по тому же лучу в обратном направлении.

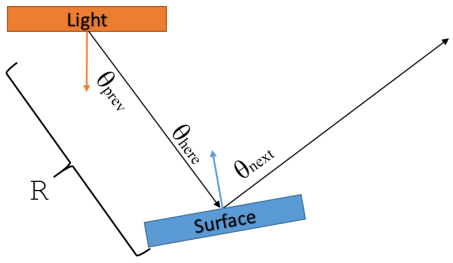


Рис. 7.18. Иллюстрация условных обозначения для углов и расстояния R. Для случая прямой трассировки (от источника к камере).

Обратите внимание на рис. 7.20, 7.21. На них хорошо видно, что многократная выборка по значимости зануляет вклад от стратегии в местах, где изоб-

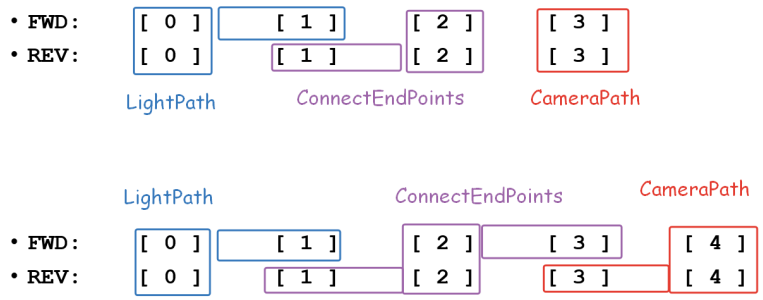
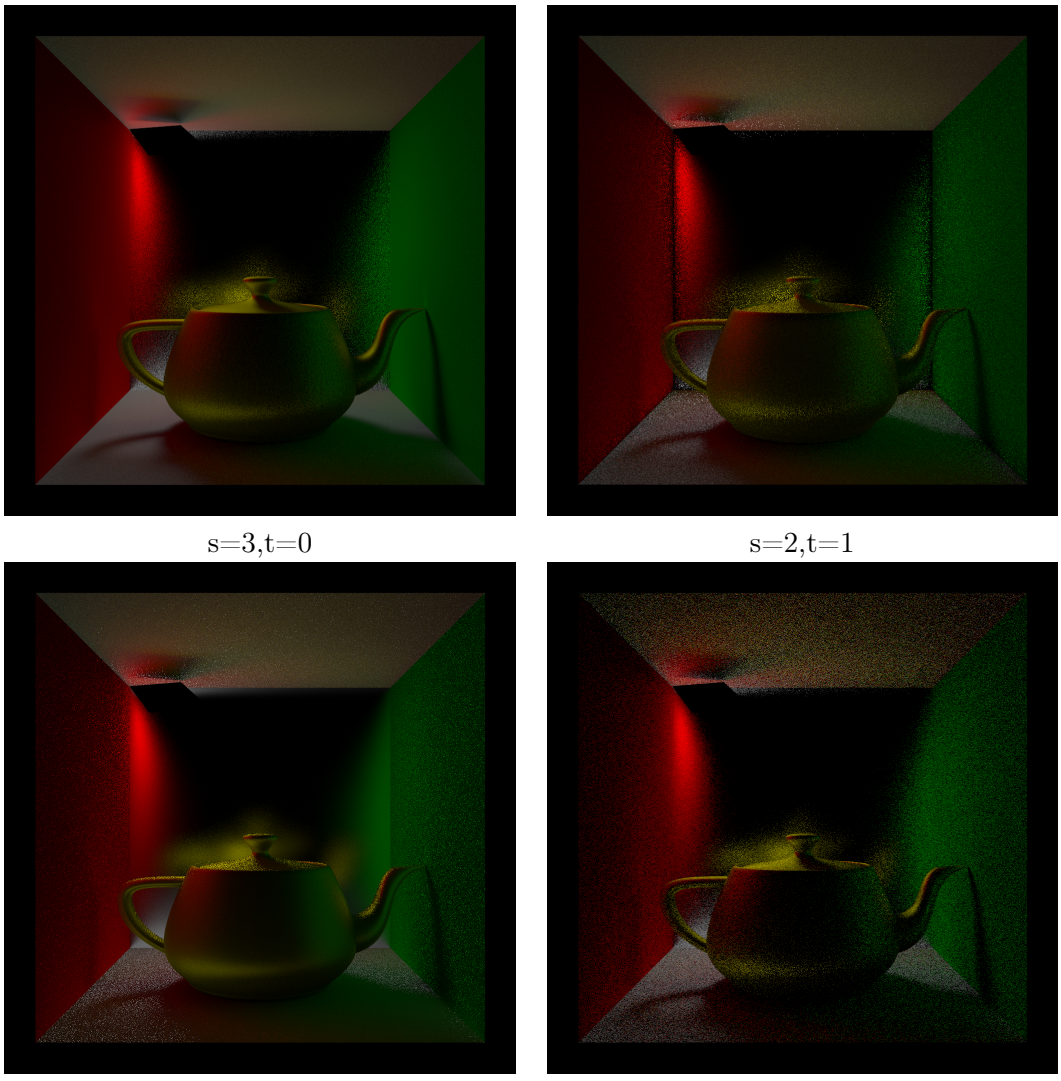


Рис. 7.19. Схема вычисления прямых и обратных плотностей вероятности различными участками алгоритма двунаправленной трассировки путей для пути длинны 3 и 4.

ражение, полученное данной стратегией шумное. Отметим, что многократная выборка по значимости работает в многомерном пространстве, *учитывая каждую выборку отдельно*, а не в пространстве изображения над уже просуммированными выборками.

```
float pdfThisWay = 1.0f;
float pdfSumm    = 0.0f;
assert(d == s+t);
for (int split = 0; split <= d; split++)
{
  const int s1 = split;
  const int t1 = d - split;

  float pdfOtherWay = specularBounceAt(split) ? 0.0f : 1.0f;
  if (split == d)
    pdfOtherWay = pdfFwd[d];

  for (int i = 0;      i < s1; i++) pdfOtherWay *= pdfFwd[i];
  for (int i = s1 + 1; i <= d; i++) pdfOtherWay *= pdfRev[i];

  if (s1 == s && t1 == t)
    pdfThisWay = pdfOtherWay;

  pdfSumm += pdfOtherWay;
}

misWeight = pdfThisWay / pdfSumm;
```

Листинг 7.1. Алгоритм вычисления MIS весов из двух массивов плотностей вероятностей pdfFwd и pdfRev.



s=3,t=0 s=2,t=1

s=1,t=2 s=0,t=3

Рис. 7.20. Изображение непрямой освещённости для рассматриваемого случая с глубиной равной 3 (4 вершины) и различными стратегиями генерации пути. До применения MIS весов.

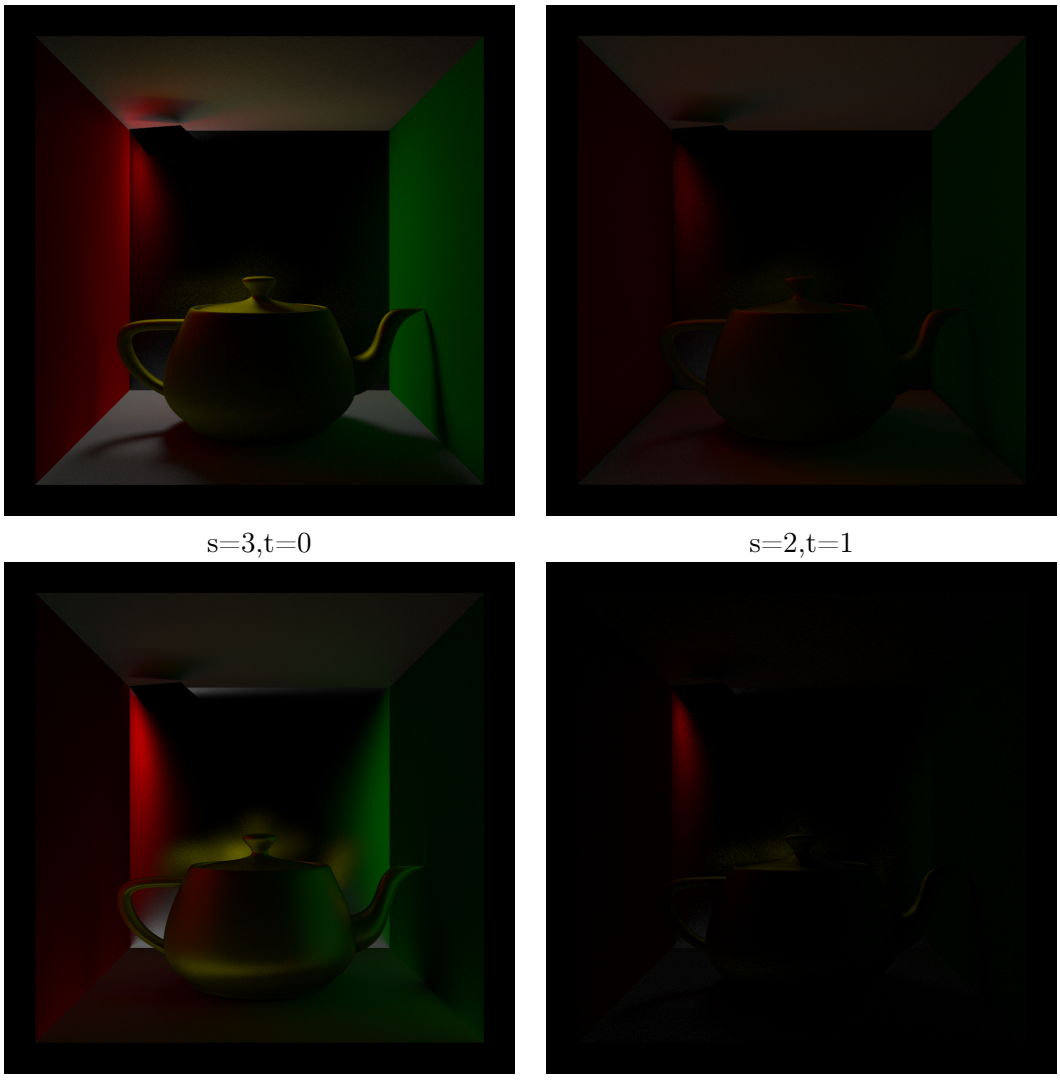


Рис. 7.21. Изображение непрямой освещённости для рассматриваемого случая с глубиной равной 3 (4 вершины) и различными стратегиями генерации пути. После применения MIS весов.

7.4.2. Эффективность MIS в BPT

Многократная выборка по значимости (MIS) на практике увеличивает даже не скорость расчёта освещения, а скорее устойчивость алгоритма. Она гарантирует отсутствие выбросов в виде ярких точек, возникающих в случаях, когда происходит какое-то очень редкое событие с высокой значимость. Например, вторичный луч случайно попал в яркую область сцены, не являющуюся источником. MIS зануляет такие выбросы своим весом, поскольку она считает что такой путь должен обрабатываться другой стратегией. Такой стратегией, где эта область как бы становится источником. И к ней считаются явные теневые лучи.

Однако, следует обратить внимание на то, что на типичных сценах (особенно архитектурных) двунаправленная трассировка путей может сильно проиграть в производительности обычной обратной трассировке путей. Причина этого в том, что BPT делает огромное число соединений по схеме «каждый с каждым». А уже после того как все эти соединения были посчитаны, при помощи MIS веса фактически зануляет большую часть из них. Кроме того, многие соединения могут найти преграду на своём пути и внесут, таким образом, нулевой вклад. Это довольно частая проблема при расчёте дневного освещения в помещении, когда BPT соединяет огромное количество точек через крышу и стены.

Таким образом, многократная выборка по значимости увеличивает устойчивость, но может значительно уменьшить скорость, т.к. из 10-20 посчитанных теневых лучей в среднем только 2-3 вносят существенный вклад. Остальные либо зануляются MIS весом, либо могут встретить преграду и внести нулевой вклад. В обычной трассировке путей, где число стратегий равно 2, эффективность не может упасть ниже 50%. В «усеченной двунаправленной трассировка путей», в которой количество втратегий равно 3, она никогда не падает ниже 33%. В полноценной же двунаправленой трасисровке путей эффективность может упасть значительно ниже (хотя при этом повышается устойчивость алгоритма к выбросам).

7.5. Metropolis light transport (MLT)

Трассировка путей обладает достаточно низкой скоростью на сценах со сложными условиями освещения. Сценами со сложными условиями освещения будем называть 2 вида сцен:

1) Сцены, на которых преобладает вторичное освещение вызванное узкими и яркими световыми пятнами (рис. 7.22).

2) Сцены, на которых ES^+DS^+L пути вносят значительный вклад (рис. 7.23).

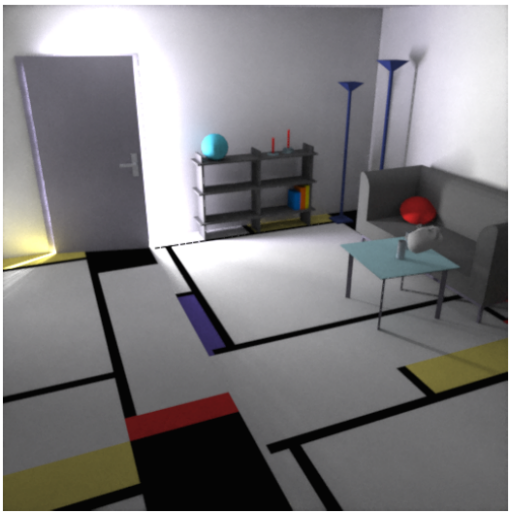


Рис. 7.22. Пример сцены со сложными условиями освещения.

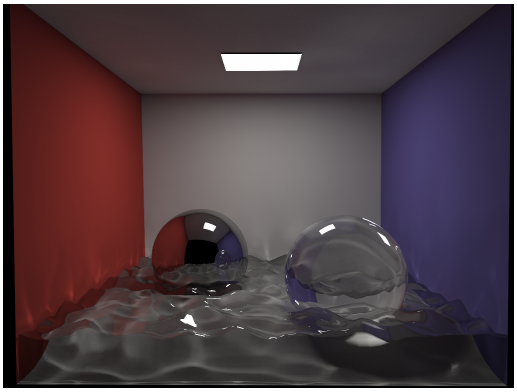


Рис. 7.23. Пример сцены с ES^+DS^+L путями. Каустики, под водой (рис. 7.22).

Проблема на таких сценах заключается в том, что в большинстве мест на сцене фунция падающего освещения имеет один или более резких максимума. Для того чтобы интеграл от такой функции вычислить с выской точностью методом Монте-Карло с равномерным распределением случайной величины, нужно большое число выборок. Выборка по значимости (в том числе многократная выборка по значимости - MIS) не всегда может помочь, поскольку эти методы основываются на априорной информации о функции падающего освещения, которой может не быть вовсе. Рисунок 7.22 является классическим примером такого случая. Вся сцена покрыта диффузным материалом и поэтому выборка по значимости на основе BRDF ничего не дает. Выборка по

значимости на основе сэмплирования источника света также ничего не дает, т.к. весь свет в сцене непрямой.

$$L(\phi_i, \theta_i)R(\phi_i, \theta_i, \phi_r, \theta_r)cos(n, l_{\phi_i,\theta_i}) \tag{7.18}$$

Для улучшения скорости сходимости на таких сценах Э. Вичем и Л. Гибсом был разработан алгоритм переноса света Метрополиса (Metropolis Light Transport, MLT) [40]. Алгоритм метрополиса позволяет сгенерировать выборки более оптимально за счет стохастического Марковского процесса, не имея изначально какой-либо априорной информации о функции падающего освещение (как это было в случае простой или многократной выборки по значимости). Metropolis Light Transport относится к так называемым методам Монте-Карло по схеме Марковских цепей (Markov Chain Monte Carlo, MCMC).

7.5.1. Алгоритм Метрополиса

Идея этого метода аналогична применению выборки по значимости, однако имеет совершенно иную природу: в действительности алгоритм метрополиса создаёт распределение выборок пропорционально даже не под-интегральному выражению 7.18 (что обычно пытается сделать многократная выборка по значимости), а пропорционально интегралу от него! То есть MLT генерирует выборки пропорционально искомому ответу - конечному изображению. На самом деле в случае MLT интеграл и под-интегральное выражение, - это одно и то же, потому что интегрирование осуществляется домножением на константу. Алгоритм Метрополиса-Гастингса (Metropolis-Hastings), из которого образовался MLT, позволяет сгенерировать распределение пропорционально неизвестной функции сложной формы, умея лишь вычислять значения этой функции в точках и широко используется в физическом моделировании. Рассмотрим работу оригинального алгоритма Метрополиса.

Предположим что вы хотите сгенерировать распределение X пропорционально некоторой функции $f(x)$. Запишем это следующим образом: $X \sim f(x)$. Предположим также что мы хотим построить гистограмму нашей функции $f(x)$ с некоторым ограниченным разрешением (рис. 7.24) на отрезке от a до b. Если мы построим такую гистограмму (или хотя бы гистограмму, пропорциональную этой, что и делает алгоритм Метрополиса), это и будет искомым распределением на этом отрезке.

Алгоритм Метрополиса делает это необычным образом. Пусть у нас есть точка x_0 в одномерном пространтсве, лежащая внутри отразке $[a, b]$. Далее,

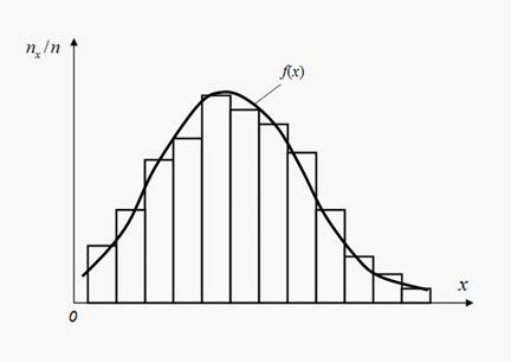


Рис. 7.24. Гистограмма для некоторой одномерной функции $f(x)$.

представьте себе броуновское движение. Пусть наша точка может путешествовать по отрезку некоторым случайным образом, перепрыгивая в новые позиции и формируя последовательность своих положений: x_1, x_2, ... x_n. Для начала - не важно как именно. Она может прыгать в полностью случайную позицию на отрезке или перемещаться в некоторую новую позицию, находящуюся близко от своей предыдущей позиции (например выбирая смещение в пределах $1/10$ от длинны отрезка $[a, b]$). Условимся так же, что для любых двух позиций x_i и x_{i+1} переход от x_i к x_{i+1} и обратно равновероятный.

Если точка посетила какой-то определённый мини-отрезок, соответствующий столбику в гистограмме, мы прибавляем в этот столбик 1. Таким образом, мы будем строить гистограмму основываясь на том как часто наша точка побывала в том или ином столбике гистограммы. В конце мы поделим значение каждого столбика на число шагов. Если движение будет полностью случайным, мы получим равномерный шум. Что будет, если движение будет не совсем случайным?

Пусть мы находимся в некоторой позиции x_i. Выражаясь метафорой, из этой точки x_i мы «бросим камень» в некоторую новую позицию y_i. Но пока не будем переходить туда, а вычислим вероятность перехода следующим образом [41]:

$$a(x_i \to y_i) = min\{1, \frac{f(y_i)}{f(x_i)}\} \tag{7.19}$$

Затем, с вероятностью $a(x_i \to y_i)$ мы переходим из x_i в y_i. Если переход был произведён, точка y_i становится для нас новой позицией: $x_{i+1} = y_i$. Если переход не был произведён, мы остаёмся в точке x_i и продолжаем «бросать камни» из неё (рис. 7.25).

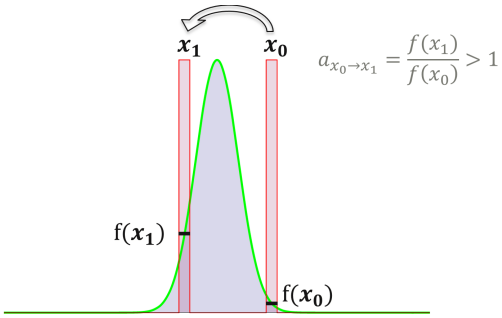


Рис. 7.25. Иллюстрация алгоритма Метрополиса. В данном случае вероятность перехода равна 1.

Таким образом, мы всегда переходим в более значимые области $f(x)$. А в менее значимые мы переходим с некоторой вероятностью, меньшей 1. Причём, каждый следующий шаг по сути масштабирует гистограмму, поскольку в конце мы делим значение каждого столбика на общее число шагов (общее число бросаний камней, а не число переходов). Это важно, поскольку если мы не выполняем переход, то увеличиваем значимость текущей позиции. А если выполняем, оставляем значимость текущей позиции без изменений и переходим к новой (рис. 7.26).

Оказывается, что описанный выше алгоритм позволяет решить искомую проблему - построить гистограмму пропорционально $f(x)$. При этом, чем больше значение функции в некотором столбике, тем чаще мы в него попадаем. Важным свойством алгоритма Метрополиса является то, что он строит гистограмму пропорционально $f(x)$ даже если значения $f(x)$ могут быть вычислены только стохастически, при помощи выборок. Это именно та ситуация, которая присутствует в трассировке путей.

Константа нормализации

Чтобы получить искомую $f(x)$ из нашей гистограммы, её ещё нужно масштабировать (просто умножить) не которую константу. В общем случае неизвестную. Алгоритм метрополиса сам по себе эту константу никак получать не умеет (он строит гистограмму пропорциональную $f(x)$ но не равную такой же гистограмме посчитанной напрямую из $f(x)$!). Для оценки этой константы как правило используют обычный Монте-Карло. Далее мы увидим, как это можно сделать для алгоритма MLT.

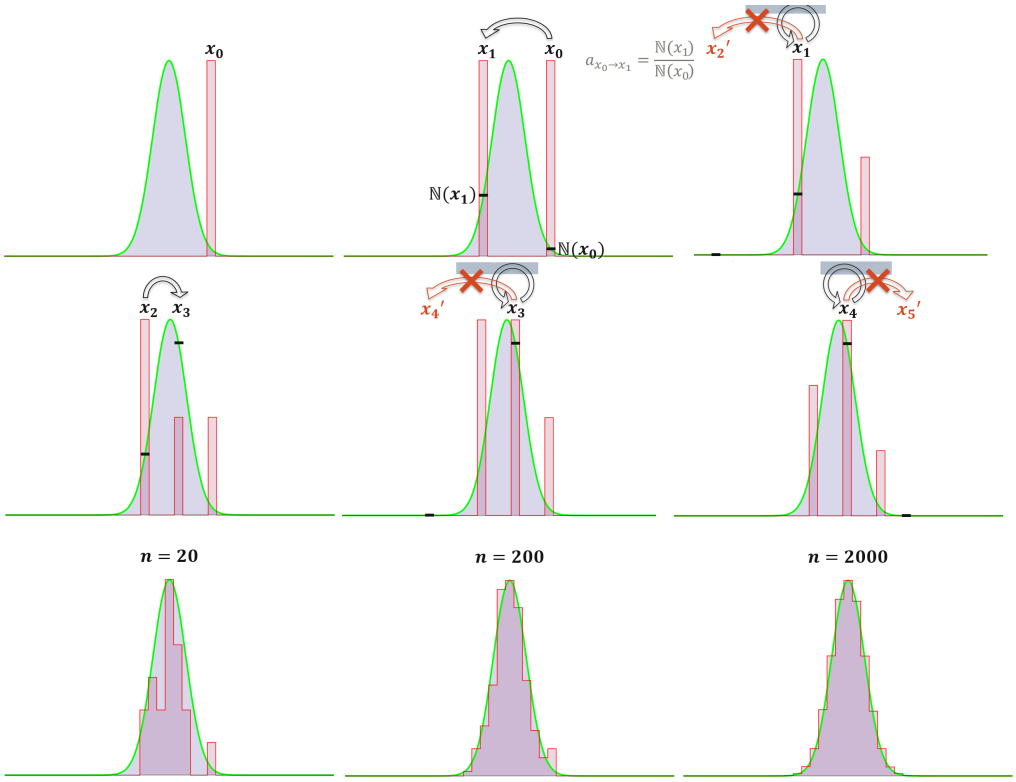


Рис. 7.26. Иллюстрация алгоритма Метрополиса по шагам. Если мы не выполняем переход, то увеличиваем значимость текущей позиции. Если выполняем, оставляем значимость текущей позиции без изменений и переходим к новой.

7.5.2. Алгоритм Метрополиса-Гастингса

Гастингс обобщил алгоритм Метрополиса на случай, когда вероятности переходов (вернее, плотности вероятностей, поскольку вероятность попасть в точку обычно равна нулю) из x_i в y_i и обратно не равны. В этом случае Вам необходимо уметь считать эти плотности вероятностей и модифицировать выражение для вычисление вероятности принятия мутации следующим образом [41]:

$$a(x_i \to y_i) = min\{1, \frac{f(y_i)T(x_i \leftarrow y_i)}{f(x_i)T(x_i \to y_i)}\} \tag{7.20}$$

Где:

- $T(x_i \to y_i)$ – плотность вероятности перехода из x_i в y_i.
- $T(x_i \leftarrow y_i)$ – плотность вероятности перехода из y_i в x_i.

Для реализации MLT вы можете использовать в качестве базы как просто Метрополиса, и Метрополиса-Гастингса. Последнее является в некотором смысле аналогом выборки по значимости для Метрополиса. Основываясь на текущем значении функции $f(x)$ или некоторой дополнительной информации, вы можете делать переходы из x_i в y_i и из y_i в x_i не равновероятными и, таким образом, быстрее сходиться к желаемому распределению.

7.5.3. Принцип работы Markov Chain Monte Carlo

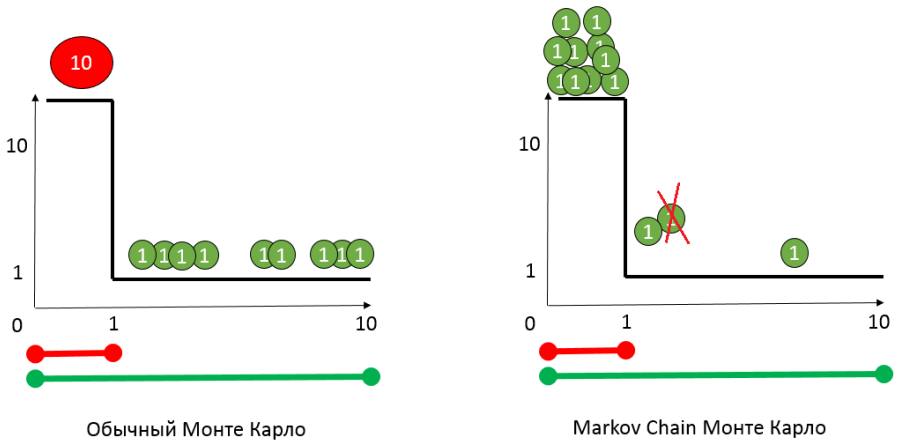


Рис. 7.27. Иллюстрация различий в работе обычного Монте-Карло метода и Монте-Карло по схеме марковских цепей (MCMC). Обычный Монте-Карло метод сохраняет значения функции, в то время как MCMC набирает столбики в гистограмме только из единичек. Цветные кружочки с цифрами иллюстрируют отдельные выборки. Перечёркнутый кружочек условно обозначает отвергнутый переход, значение которого было выброшено (в самом просто варианте отвергнутые переходы не учитываются совсем). Обратите внимание что даже там где функция принимает значение равное 10, MCMC все-равно сохраняет 1.

Идея Markov Chain Monte Carlo на самом деле достаточно проста. Рассмотрим задачу построения гистограммы простой функции на интервале от 0 до 10 (например с шагом в 1). На интервале от 0 до 1 функция принимает значение равное 10, а в остальной части области определения она равна 1 (рис. 7.27).

Обычный Монте-Карло будет генерировать выборки с равномерным распределением (мы заранее не знаем природу функции) и сохранять в гистограмму приходящие значения. В большинстве случаев это будут единички, но в 10% случаев (когда мы попали в начало интервала) мы получим выброс равный 10 (рис. 7.27). В противоположность этому, MCMC будет всегда сохранять единички. Даже когда он находится в начале интервала, где значение функции равно 10 (рис. 7.27). Но это значит, что для того чтобы «набрать десятку», он должен 10 раз побывать в начале интервала, каждый раз прибавляя 1 в начальный столбик. Именно это и происходит - значения функции как таковые в гистограмму не сохраняются вообще. Они используются только для того чтобы принять или отвергнуть переход.

Проблема начального смещения (Start up Bias)

Из рассмотренного выше примера нетрудно заметить, что MCMC в начале работы может просто не успеть «набрать десятку» суммируя единички, в результате чего гистограмма получается слишком однородной и «малоконтрастной».

Если вы делаете MLT на CPU, этой проблемы не существует потому что на CPU за один «проход» одна цепь успеет сделать сотни тысяч шагов. И, забегая немного вперёд, в любой локальной окрестности изображения контраст будет правильный, хотя одна цепь разумеется может не успеть обойти всё изображение. Однако ситуация поменяется, если вы будете запускать сотни тысяч потоков параллельно на GPU, каждый из которых будет условно за 1 «проход» делать лишь небольшое число шагов. В этом случаев в начале работы алгоритма картинка будет выглядеть неправильно. С течением времени (когда все цепи сделают достаточно большое число шагов) изображение станет корректным (рис. 7.28).

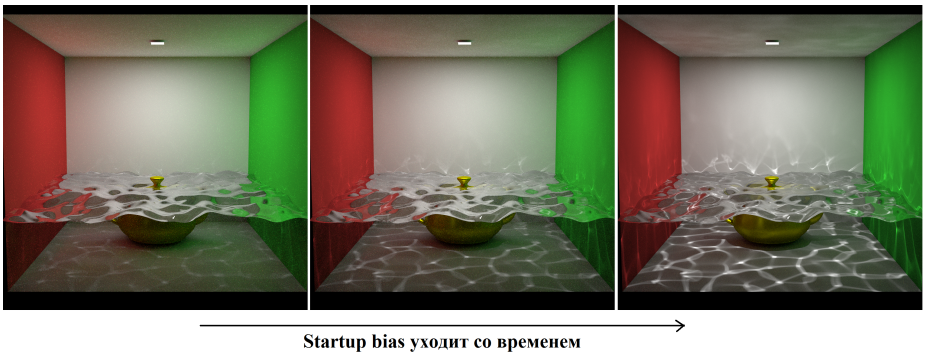


Рис. 7.28. При запуске большого числа потоков параллельно на GPU start up bias значительно увеличивается. Каждая цепть должна сделать достаточно большое число шагов (хотя бы несколько десятков тысяч) чтобы изображение выглядело корректно. Одно из решений решений (т. н. «прожиг») заключается в использовании обыкновенного Монте-Карло для отбора начальных состояний цепей Маркова [42].

7.5.4. Собственно MLT

В случае с интегрированием освещенности, алгоритм Метрополиса будет строить некоторое распределение, пропорциональное изображению которое мы хотим посчитать. Однако, этого ещё не достаточно для того чтобы получить изображение.

Пожалуй самой важной и неочевидной особенностью MLT является то, что этот алгоритм работает сразу для всего изображения, т.е. мутации лучей в экранной плоскости являются обязательным элементом. MLT невозможно применить независимо для отдельных пикселей, т.к. константа нормализации в этом случае для каждого пиксела будет своя. Но есть и другие вещи, на которые необходимо обратить внимание.

1) Во-первых, мы должны научиться вычислять константу нормализации. Это можно сделать, оценивая среднюю яркость всех пикселей при помощи обычного Монте-Карло. Т.к. независимо от метода расчёта средняя яркость (по всем пикселам изображений) должна быть одинаковой, можно оценить её при помощи обычной монте-карло трассировки путей, а затем подобрать константу нормализации нашей MLT гистограммы так, чтобы получить ту же самую среднюю яркость.

2) Во-вторых, не стоит пытаться считать прямое освещение при помощи MLT. Это не будет работать, поскольку алгоритм просто уткнётся в источник или другую ярко-освещёную область, и будет из них выходить крайне редко. Из этого следует что прямое освещение нужно считать

в отдельном изображении, а затем уже складывать это изображение с непрямым, которое можно считать при помощи MLT (рис. 7.30).

3) В третьих нужно подумать над пространством в котором мы будем работать и над стратегиями мутаций. Или, выражаясь использованным ранее понятием, нужно научиться «метко бросать камни». Простой и неплохой стратегией является гауссианна в превичном пространстве путей с периодическими попытками слезть с пика. Первичное пространство путей - это многомерный единичный куб, в котором генерируются случайные числа для метода Монте-Карло. Мутации в этом пространстве хороши тем, что они не зависят от характера сцены [43]. Рисунок 7.29 иллюстрирует эту идею.

4) В целях повышения эффективности нам бы хотелось учитывать не только принятые переходы, но и отвергнутые. Также нам необходимо вычислять цветное изображение. Алгоритм 7.2 показывает как это можно сделать.

5) Наконец, для MLT следует более внимательно подходить к выбору генератора случайных чисел, чем при реализации обычной Монте-Карло трассировки путей. Одно из необходимых условий для корректной работы Markov Chain Monte Carlo (MCMC) - это отсутствие «повторяемости» состояний. Обязательно нужно использовать 2 разных генератора случайных чисел. Один для мутаций («бросания камней»), второй для самого алгоритма метрополиса (алгоритм 7.2).

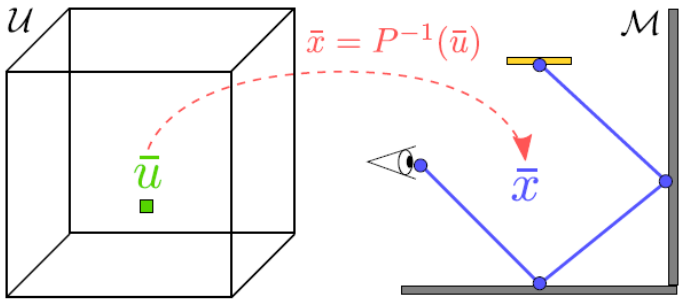


Рис. 7.29. Идея алгоритма PSSMLT. Цепь Маркова работает в многомерном единичном кубе U, точки $\overline{u}$ которого используются далее вместо случайных чисел в трассировщике путей. Путь $\overline{x}$ получается, таким образом, как некоторая функция $P^{-1}(\overline{u})$, называемая в англоязычной литератре «cumulative distribution function».

Отметим также, что вопреки наличию схожих черт между MLT и генетическим алгоритмом, эти алгоритмы в действительности имеют совершенно раз-

личную природу. Генетический алгоритм позволяет найти максимум функции, в то время как MLT - построить пропорциональное (самой функции) распределение выборок $p(x)$. Чтобы избежать путанницы, при описании алгоритма Метрополиса мы специально не использовали традиционный термин «мутации» а воспользовались метафорой с «бросанием камней». За дополнительными подробностями реализации алгоритма MLT рекомендуется обратиться к [41], [44] и [40]. Рисунок 7.31 демонстрирует применение MLT для расчёта каустиков и сопоставляет его эффективность с обычной трассировкой путей.

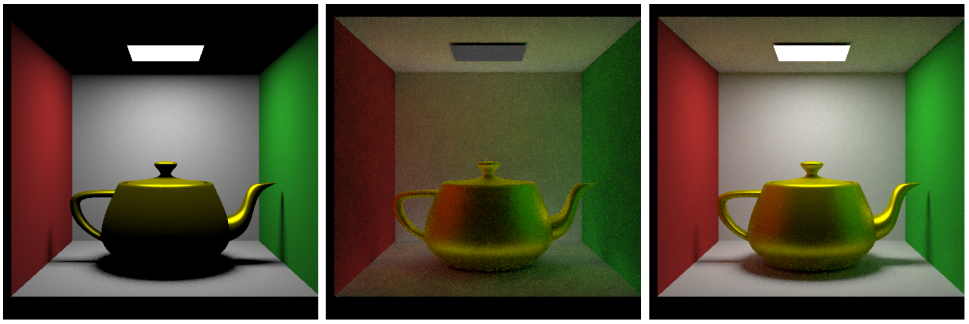


Рис. 7.30. Применение MLT для расчёта изображений. Прямой свет был посчитан при помощи обычной Монте-Карло трассировки лучей (Path Tracing, PT, слева). Metropolis Light Transport был использован только для расчёта вторичного освещения (посередине). Результат сложения двух изображений справа.

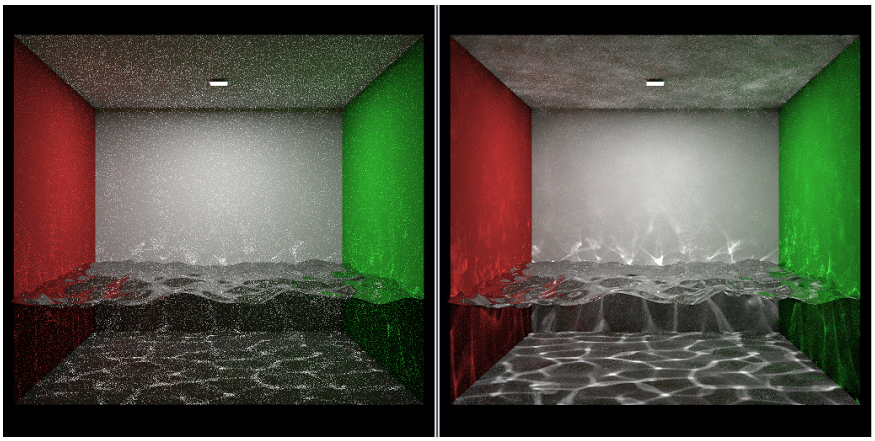


Рис. 7.31. Сравнение обычной монте-карло трассировки путей (слева) и MLT (справа) за одно и то же время. В данном случае использовался однонаправленный PSSMLT.

7.5.5. PSSMLT и Path Space MLT

В настоящий момент существует довольно много различных вариаций MLT. В предыдущем разделе мы рассмотрели MLT в первичном пространстве путей (Primary Sample Space MLT или PSSMLT, 7.29). Это самый простой, но далеко единственный вариант. Существует довольно большой класс т. н. Path Space методов, которые работают в пространстве путей, напрямую внося изменения в координаты вершин в мировом пространстве. Самая первая реализация MLT, придуманная Вичем была как раз Path Space вариантом [40].

Наиболее существенным недостатком MLT в пространстве путей является то, что под каждый феномен освещённости необходимо не только тщательно проектировать специфические стратегии предложений переходов («бросания камней» в терминологии, используемой нами ранее), но и уметь вычислять плотности вероятности переходов – $T(x \to y), T(x \leftarrow y)$ для правила Метрополиса (формула 7.20). Например, работа [45] предлагает специфические стратегии предложений переходов (называемых пертурбациями) для зеркальных отражений, а работа [46] - для глянцевых (glossy) поверхностей.

Современные системы синтеза реалистичных изображений содержат десятки различных типов материалов и источников освещения, что приводит к сотням всевозможных типов взаимодействий (феноменам освещённости). Это чрезвычайно усложняет процесс разработки стратегий «бросания камней» и процесса вычисления плотностей вероятности $T(x \to y)$ и $T(x \leftarrow y)$, поэтому MLT в пространстве путей практически не применяется на практике, а используется в основном в исследовательских и демонстрационных приложениях. В настоящий момент известен только один «общий» (т. е. не использующий специализированные стратегии предложений перехода для отдельных феноменов освещённости) Path Space метод. Это Hessian-Hamiltonian Monte Carlo (HHMC) [47], использующий Монте-Карло с Гамильтоновой механикой (HMC) [48]. Мы не станем углубляться в Path Space методы из-за их высокой сложности и большого количества нюансов, без рассмотрения которых описывать Path Space методы на наш взгляд не имеет смысла, учитывая практическую направленность данной книги. Вместо этого сосредоточимся на простых и универсальных методах интегрирования освещённости — методах, работающих в первичном пространстве путей.

В PSSMLT, прямая и обратная вероятности (вернее плотности вероятности) одинаковы. Из этого следует, что $T(x \to y)$ и $T(x \leftarrow y)$ в формуле 7.20, сокращаются, - и следовательно их не нужно вычислять. PSSMLT можно реализовать как для однонаправленной, так и для двунаправленной трассировки путей (BPT). Алгоритм, который мы рассмотрим далее является развитием PSSMLT для двунаправленной трассировки путей.

7.5.6. Multiplexed Metropolis Light Transport

Алгоритм Multiplexed Metropolis Light Transport (MMLT) [49] является следующим шагом на пути развития MLT в первичном пространстве путей (т.е. PSSMLT). В работе [49] было замечено, что многократная выборка по значимости и алгоритм Метрополиса в некоторой степени всё же конкурируют друг с другом, особенно если используется двунаправленная трассировка: MLT увеличивает число путей в BPT, где существенен вклад только от одной стратегии, а остальные зануляются MIS весом. Для того чтобы комбинировать алгоритм Метрополиса и многократную выборку по значимости более эффективно, в [49] было предложено использовать марковскую цепь в том числе и для выбора способа построения пути в BPT (рис. 7.32). Благодаря этому, алгоритм Метрополиса автоматически перераспределяет вычислительные ресурсы таким образом, что малозначимые стратегии и соединения в BPT считаются редко. Причём, это происходит в том числе и с учётом функции видимости, т.к. алгоритм Метрополиса строит распределение пропорционально итоговому ответу.

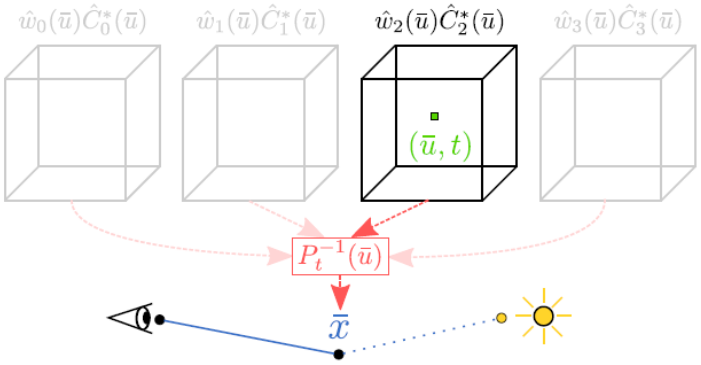


Рис. 7.32. Идея алгоритма MMLT для глубины в 2 переотражения. Дополнительный варьируемый параметр t (который вносится **внутрь** вектора $\overline{u}$) позволяет Марковской цепи выбирать способ отображения $\overline{x} = P^{-1}(\overline{u})$ среди четырёх возможных $P_i^{-1}(\overline{u})$ благодаря представлению $C_i(\overline{u})$ как произведение $\hat{w}_i(\overline{u})\hat{C}_i^*(\overline{u})$. Многократная выборка по значимости в BPT автоматически обеспечивает оптимальность такого выбора благодаря тому что зануляет неудачные способы отображения MIS весом $\hat{w}_i(\overline{u})$.

Чтобы понять идею MMLT, вернёмся к рассмотрению примера с вычислением MIS весов в двунаправленной трассировке путей в разделе 7.4.1, где мы описали 4 возможных стратегии генерации пути с 3 переотражениями. Прежде всего отметим, что в MMLT соединения делаются по упрощённой схеме — соединяются только конечные точки, как изображено на рис 7.17. Таким образом, для глубины 3 мы будем иметь 4 возможных способа построения пути,

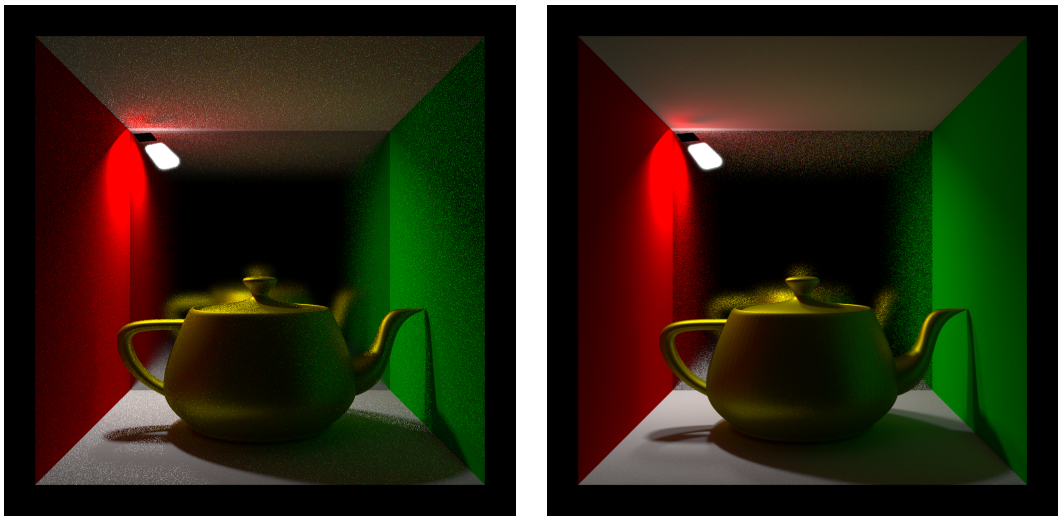
изображённых в таблице 7.1.

Далее, пусть **для i-ой стратегии** построения пути в BPT $C_i(\overline{u}) = \hat{w}_i(\overline{u})\hat{C}_i^*(\overline{u})$ — функция вклада («contribution function»). Мы сохраняем обозначения из работы [49]. Эта функция (вернее её часть без MIS веса) может вычисляться, например, как взвешенное среднее по 3 цветовым каналам:
$\hat{C}_i^*(\overline{u}) = lum(PathColor(\overline{u}))$. В соответствие с механизмом многократной выборкой по значимости итоговая функция вклада в изображение будет вычисляться как взвешенная сумма (выражение 7.21).

$$C(\overline{u}) = \hat{w}_1(\overline{u})\hat{C}_1^*(\overline{u}) + \hat{w}_2(\overline{u})\hat{C}_2^*(\overline{u}) + \hat{w}_3(\overline{u})\hat{C}_3^*(\overline{u}) + \hat{w}_4(\overline{u})\hat{C}_4^*(\overline{u}) \tag{7.21}$$

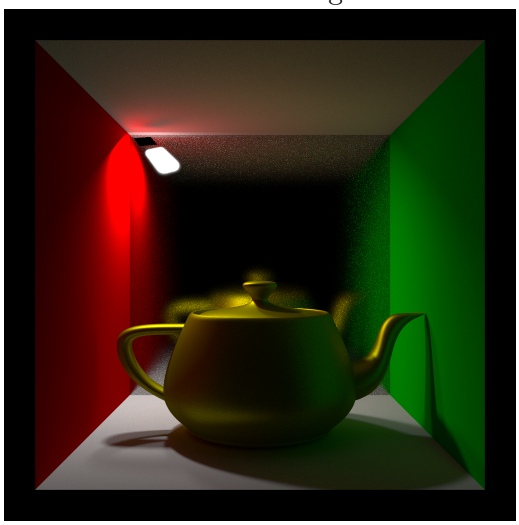
Обычный BPT будет использовать все четыре способа $\hat{C}_1^*(\overline{u})..\hat{C}_4^*(\overline{u})$ равновероятно, зануляя выбросы MIS весом $\hat{w}_i(\overline{u})$. В MMLT ситуация меняется. Благодаря тому что алгоритм Метрополиса строит распределение выборок *уже* пропорционально взвешенной функции $C(\overline{u}) = \sum_{i=1}^{4} w_i\hat{(\overline{u})}C_i^{*}\hat{(\overline{u})}$, стратегии, которые были бы занулены в BPT апостериорно, в данном случае будут реже выбираться. Причём, поскольку алгоритм Метрополиса строит распределение пропорционально итоговому ответу, видимость (которая находится где-то внутри функции $\hat{C}_i^*(\overline{u})$) также будет учитываться. То есть MMLT будет делать меньше соединений через стены и потолок чем классическая двунаправленная трассировка путей, эффективно генерируя выборки в многомерном пространстве.

На сегодняшний день MMLT можно считать одним из наиболее эффективных и прогрессивных алгоритмов для расчёта освещённости на основе цепей Маркова. Далее мы продемонстрируем несколько сравнений, полученных в результате наших собственных экспериментов.

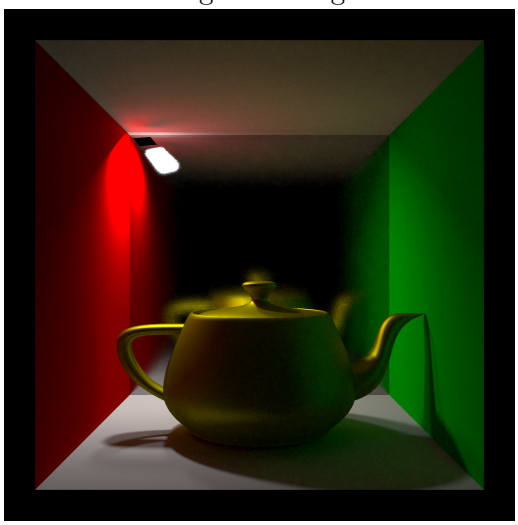


Path Tracing

Light Tracing

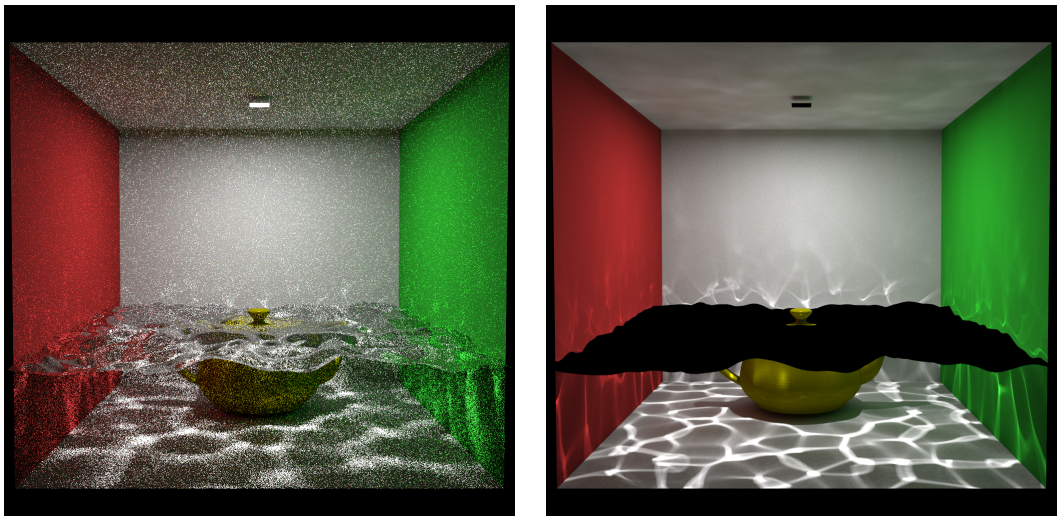


IBPT



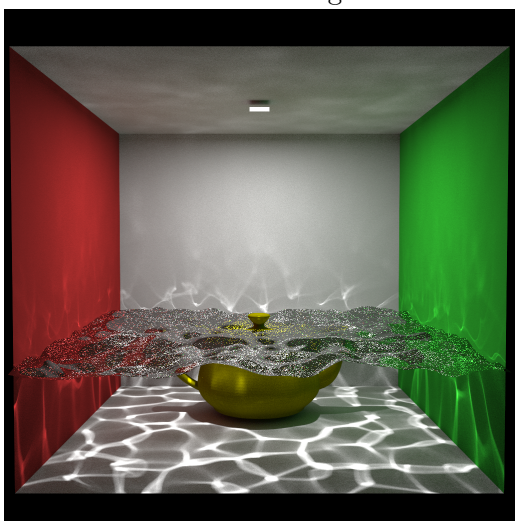
MMLT

Рис. 7.33. Сравнение различных алгоритмов интегрирования освещённости. Время рендеринга — 10 минут на процессоре Intel Core i7 3770, 3.4 GHz. IBPT — усечённая двунаправленная трассировка путей.

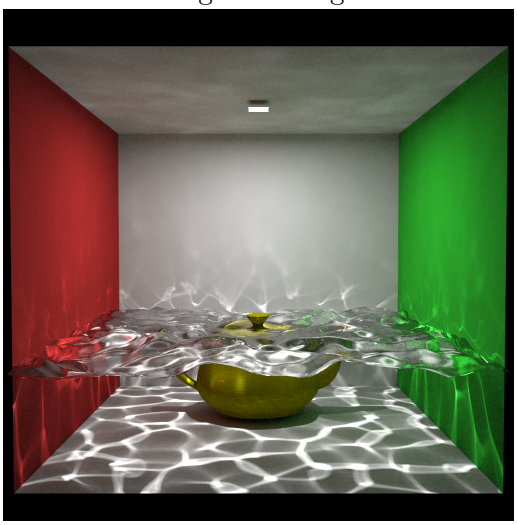


Path Tracing

Light Tracing



IBPT



MMLT

Рис. 7.34. Сравнение различных алгоритмов интегрирования освещённости. Время рендеринга — 10 минут на процессоре Intel Core i7 3770, 3.4 GHz. IBPT — усечённая двунаправленная трассировка путей.

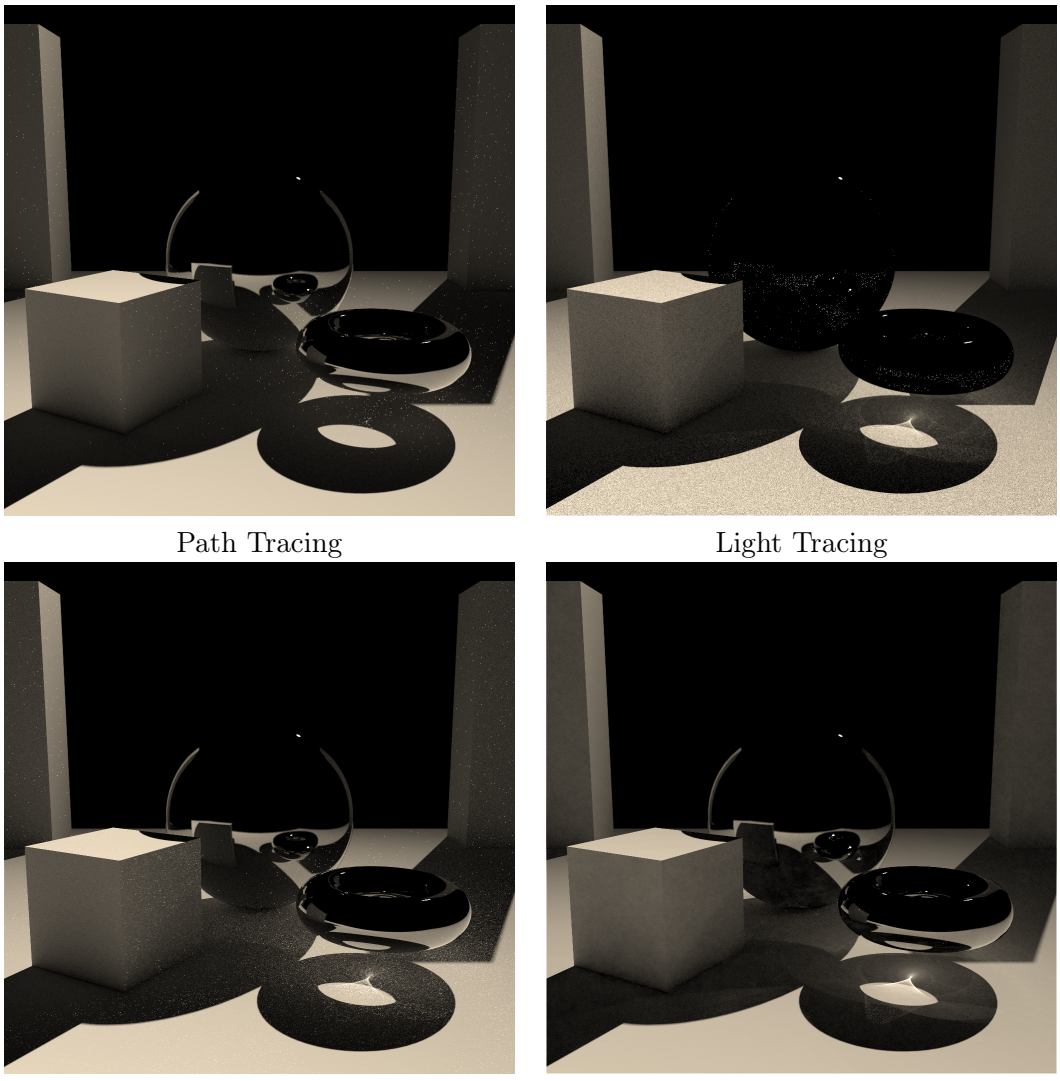
Path Tracing

Light Tracing

IBPT

MMLT

Рис. 7.35. Сравнение различных алгоритмов интегрирования освещённости. Время рендеринга — 60 минут на процессоре Intel Core i7 3770, 3.4 GHz. IBPT — усечённая двунаправленная трассировка путей.



Path Tracing | Light Tracing

IBPT | MMLT

Рис. 7.36. Сравнение различных алгоритмов интегрирования освещённости. Время рендеринга — 5 минут на процессоре Intel Core i7 3770, 3.4 GHz. IBPT — усечённая двунаправленная трассировка путей.

7.5.7. Недостатки MLT

Первый и основной недостаток MLT вытекает из его главного принципа. Алгоритм работает сразу для всего изображения и свободно перемещается между пикселами. При этом для отдельно взятого пиксела невозможно гарантировать какую-бы то ни было фиксированную точность. MLT гарантирует только что всё изображение целиком получится несмещённым. Отдельно взятые пикселы могут вообще не получить ни одной выборки (или получить их очень мало).

Второй недостаток - это startup bias, который в некоторой степени ограничивает параллеллизм. Metropolis Light Transport можно распараллелить на произвольное количество потоков (CPU или GPU - неважно), но каждый поток при этом обязан сделать достаточно большое число шагов (хотя решение в данном случае существует [42]).

Наконец, третий существенный недостаток — видимая человеческим глазом структура шума в виде траекторий движения, которая проявляется т.к. сэмплы в MCMC коррелированы.

Первый и третий недостатки могут быть в некоторой степени амортизированны применением фильтрации к результатам работы MCMC (например, медианный фильтр для устреанения выбросов где встречается т. н. «застревание цепи» [50] и Non Local Means для удаления оставшегося шума). На наш взгляд применение фильтра к результатам работы MCMC более естественно, чем применение фильтров к результатам работы обычного Монте-Карло, т.к. MCMC всегда сэмплируют группы пикселей.

```
vector<float> x = initialSample_PS();
float3 yColor   = F(x);
float  y        = luminance(yColor);

for (int mId = 0; mId < mutationsNum; mId++)
{
  vector<float> xOld = x;
  vector<float> xNew = mutatePimarySpace(x);

  float  yOld      = y;
  float3 yOldColor = yColor;

  float3 yNewColor = F(xNew);
  float  yNew      = luminance(yNewColor);

  float a = (yOld == 0.0f) ? 1.0f : fmin(1.0, yNew / yOld);

  if (rnd(0,1) <= a) // accept
  {
    x = xNew;
    y = yNew;
    yColor = yNewColor;
  }

  hyst(xOld[0], xOld[1]) += yOldColor*(1.0 / yOld)*(1.0f - a);
  hyst(xNew[0], xNew[1]) += yNewColor*(1.0 / yNew)*a;
}
```

Листинг 7.2. Алгоритм Metropolis Light Transport. F(x) трассирует единичный путь и вычисляет его яркость (в RGB).

7.6. Фотонные карты

Рассмотренные ранее несмещенные методы на основе трассировки путей работают в терминах яркости. Каждый путь в них отвечал за перенос яркости от источника света к камере. Метод фотонных карт работает в терминах потока. Идея этого метода заключается в том, чтобы вычислить распределение световой энергии по сцене при помощи трассировки множества частиц, переносящих порцию световой энергии. После чего можно оценивать интеграл освещенности, выполняя на поверхности поиск ближайших фотонов (рис. 7.40). Рассмотрим интеграл освещенности (формула 7.22). Выразим функцию падающей освещенности в терминах потока. Подставим $L(\phi_i, \theta_i)$ в интеграл и получим:

$$I(\phi_r, \theta_r) = \iint\limits_{\phi_i \theta_i} L(\phi_i, \theta_i) R(\phi_i, \theta_i, \phi_r, \theta_r) cos(n, l_{\phi_i, \theta_i}) \mathrm{d}\phi_i \mathrm{d}\theta_i \tag{7.22}$$

$$L(\phi_i, \theta_i) = \frac{d^2\Phi}{cos(n, l_{\phi_i, \theta_i}) d\phi_i d\theta_i dA} \tag{7.23}$$

(7.24)

$$I(\phi_r, \theta_r) = \iint\limits_{\phi_i \theta_i} \frac{d^2\Phi}{cos(n, l_{\phi_i, \theta_i}) d\phi_i d\theta_i dA} R(\phi_i, \theta_i, \phi_r, \theta_r) cos(n, l_{\phi_i, \theta_i}) \mathrm{d}\phi_i \mathrm{d}\theta_i$$

$$I(\phi_r, \theta_r) = \iint\limits_{\phi_i \theta_i} \frac{d^2\Phi}{dA} R(\phi_i, \theta_i, \phi_r, \theta_r)$$

То есть дифференциальный поток можно аппроксимировать суммой энергий фотонов в окрестности. Таким образом, в результате преобразований получаем сумму K ближайших фотонов.

$$I(\phi_r, \theta_r) = \iint\limits_{\phi_i \theta_i} \frac{d^2\Phi}{dA} R(\phi_i, \theta_i, \phi_r, \theta_r) \tag{7.25}$$

$$I(\phi_r, \theta_r) \approx \sum_{i=1}^{K} R(\phi_i, \theta_i, \phi_r, \theta_r) \frac{\Delta\Phi(\phi_i, \theta_i)}{\Delta A} = \frac{1}{\pi r^2} \sum_{i=1}^{K} R(\phi_i, \theta_i, \phi_r, \theta_r) \Delta\Phi(\phi_i, \theta_i)$$

Переменная r отвечает за радиус диска на поверхности, в пределах которого выполняется поиск ближайших фотонов. Метод фотонных карт, таким образом, состоит из 4 шагов:

1) Испускание фотонов из источников света

2) Трассировка фотонов

3) Построение фотонной карты

4) Сбор освещенности

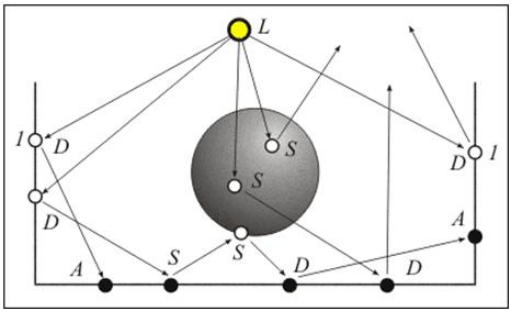


Рис. 7.37. Процесс трассировки и сохранения фотонов на поверхностях.

Испускание фотонов. Фотоны в данном методе – это частицы, переносящие некоторую небольшую порцию световой энергии. На начальном этапе фотоны испускаются из источника света в соответствие с распределением световой энергии для данного источника. Например известно, что точечный или сферический источник света (такой как солнце) испускают свет равномерно во всех направлениях. Площадные источники света имеют косинусоидальное распределение, имеющее максимум по направлению, совпадающему с нормалью к плоскости источника. Стадия испускания фотонов не отличается от стадии испускания путей вида $L(S|D|G)^*E$ в прямой трассировки путей.

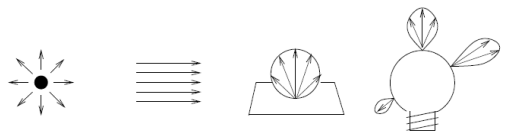


Рис. 7.38. Иллюстрация различных видов распределения световой энергии для различных источников света.

Трассировка фотонов. Русская рулетка. Трассировка фотонов происходит похожим на прямую трассировку путей образом за исключением одного важного отличия - механизма русской рулетки (этот механизм можно также применять и в трассировке путей [34], но исторически он появился именно в методе фотонных карт) [51]. Легче всего продемонстрировать механизм русской рулетки на примере. Допустим мы имеем ламбертовский материал с коэффициентом отражения 0.25 и фотон падающий на эту поверхность с энергией, равной 1. Очевидным решением является модифицировать энергию фотона, умножив ее на 0.25 и продолжить трассировку фотона. Механизм русской рулетки поступает по другому. Вместо того чтобы модифицировать энергию фотона, его энергия сохраняется, но с вероятностью 0.25 фотон отражается и продолжает трассировку, а с вероятностью 0.75 - погибает (рис. 7.39). Таким образом, если на поверхность падает 1000 фотонов, 250 отразится, сохранив энергию, а остальные 750 - погибнут. За счет такого механизма повышается точность решения, поколько в местах с высокой освещенностью будет сохраняться больше фотонов, чем в местах с низкой освещенностью; в неосвещенных областях фотоны будут отсутствовать. Нетрудно заметить, что

Рис. 7.39. Русская рулетка.

при использовании русской рулетки в чистом виде все фотоны будут иметь единичную энергию. То есть, энергию фотона хранить не обязательно, а для того чтобы оценить значение освещенности, достаточно либо посчитать число фотонов в заданном радиусе сбора, либо найти радиус, в котором лежит K ближайших фотонов. Мы вернемся к этому вопросу позже, когда будем рассматривать процесс сбора освещености. При этом для того чтобы выполнять правильно сбор освещенности, нам потребуется хранить позицию фотона и нормаль к поверхности в точке удара о поверхность.

Сохранение фотонов в фотонной карте. Вопрос сохранения фотонов на поверхностях (в фотонной карте) заслуживает отдельного рассмотрения. Фотоны сохраняются на всех диффузных (ламбертовых) поверхностях, которые

они ударяют. Как правило фотоны используются только при вычислении части интеграла, обусловленной диффузной компонентой. Зеркальная компонента не может быть вычислена при помощи фотонов по той же причине, по которой прямая трассировка путей (Ligh Tracing) отображает зеркала и стекла черными: вероятность встретить фотон (или путь от источника), отразившийся в строго заданном направлении стремится к нулю. Из этого следует, что компонента, обусловленная матовым отражением (glossy reflection) все же может быть вычислена при помощи фотонных карт, однако эффективность такого расчета будет тем ниже, чем более зеркальный характер приобретает ДФО. Следует подчеркнуть, что фотон в течении трассировки может быть сохранен несколько раз - на всех диффузных поверхностях, которые он ударяет.

Итак, фотоны сохраняются на всех диффузных поверхностях, которые они ударяют. Однако, одним из часто-применяемых трюков является в буквальном смысле 'пропуск' первого отскока для процедуры сохранения фотона. Фотоны в этом случае сохраняются только после первого отскока, а освещение, получаемое в результате интегрирования фотонной карты - целиком вторичное. Этот метод значительно снижает число фотонов, необходимое для расчета вторичного освещения. Первичное освещение гораздо быстрее вычисляется при помощи трассировки лучей/путей, поэтому данный прием имеет важное значение для ускорения процесса рендеринга.

Построение фотонной карты. В самом простом случае можно хранить фотонную карту в виде обыкновенного массива. Однако это не позволит в последствие выполнять эффективный поиск ближайших фотонов. Поэтому под построением фотонной карты мы будем понимать построение ускоряющей структуры, позволяющей выполнять эффективный поиск ближайших фотонов. Существует довольно большое число способов, как делать это эффективно. Способы могут различаться для поиска k-ближайших или всех фотонов в заданном радиусе. Среди наиболее эффективны структур следует отметить пространственные хэш-таблицы и деревья, построенные при помощи эвристики VVH [52] аналогично эвристике SAH, рассмотренной нами ранее.

Сбор освещенности. После того как фотонная карта построена, наступает стадия сбора освещенности. Из виртуальной камеры испускаются лучи и выполняется обратная или стохастическая трассировка лучей (или трассировка путей). В местах соударений луча и поверхности производится сбор освещенности. Причем, для стохастической трассировки лучей (или путей), диффузные переотражения не учитываются. Компонента, обусловленная диффузными переотражениями получается при интегрировании фотонной карты (фомула 7.26). Тем не менее на практике такой метод (за исключением расчета

каустиков) дает не очень хороший результат и приводит к появлению пятен, которые достаточно трудно удалить (фактически только увеличением числа фотонов). Довольно часто диффузные переотражения в процессе обратной трассировки лучей/путей вычисляются методом Финального Сбора (будет рассмотрен позже), в процессе которого сбор из фотонной карты фактически откладывается на 1 переотражение.

$$I(\phi_r, \theta_r) \approx \frac{1}{\pi r^2} \sum_{i=1}^{K} R(\phi_i, \theta_i, \phi_r, \theta_r) \Delta\Phi(\phi_i, \theta_i) \tag{7.26}$$

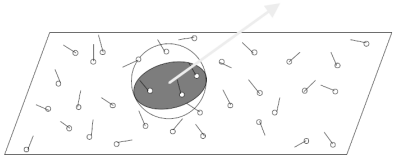


Рис. 7.40. Иллюстрация к формуле 7.26. Сбор освещенности или интегрирование фотонной карты. В данной формуле r - радиус диска, полученного при проекции сферы на поверхность. πr^2 - площадь диска.

Существует 2 основные стратегии сбора освещенности:

1) Сбор k-ближаших фотонов (динамический радиус сбора), в соответствии с формулой 7.26. Рисунок 7.41.

2) Сбор всех фотонов в заданном радиусе (фиксированный радиус сбора). Данная стратегия упрощает процесс сбора и дает корректный результат в пределе (при стремлении общего числа фотонов к бесконечности), однако не обеспечивает постоянное значение K в формуле 7.26.

Поиск ближайших фотонов, как правило, использует специальную структуру 'max-heap' для упорядочивания списка фотонов [51]. Однако, поиск k-ближайших заключает в своей основе более простой поиск всех фотонов в заданном радиусе, поскольку даже при поиске k-ближайших необходимо начинать с некоторого радиуса и уметь находить все фотоны в нем.

7.6.1. Финальный сбор

Один из артефактов, возникающих при реализации сбора освещенности - темные края поверхностей(рис 7.43, 7.44). Данный артефакт возникает по

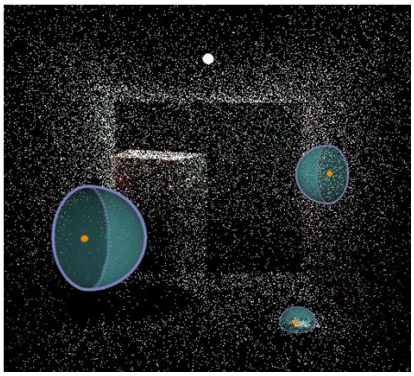


Рис. 7.41. Сбор k-ближайших фотонов. Радиус сбора выбирается динамически (чтобы ограничить ровно k фотонов) в зависимости от плотности фотонной карты.

причине того, что на границах поверхностей освещение собирается только с половины диска. При этом площадь, вычисляемая как πr^2 для целого диска остается прежней. Один из способов, позволяющий убрать темные края заключается в том, чтобы найти реальную площать элемента поверхности на котором происходил сбор при помощи вычисления площади пересечение сферы сбора и всех треугольников с нормалью, близкой к нормали в точке сбора (то есть вычислить честную проекцию сферы на поверхность). Второй метод называется Финальный Сбор (Final Gathering, FG). Вместо непосредственного сбора освещенности с фотонов, в методе финального сбора из заданной точки испускается некоторое число лучей по полусфере и освещенность собирается уже в тех местах, куда попали лучи (рисунок 7.42).

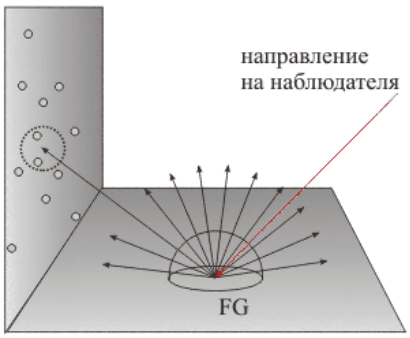


Рис. 7.42. Иллюстрация метода финального сбора.

При использовании финального сбора сами фотонные карты, таким образом, задействуются только для вычисления апроксимации вторичной освещенности. Помимо избавления от темных краев, финальный сбор значительно повышает точность решения, позволяя обрабатывать значительно число

фотонов. Хотя скорость финального сбора в среднем сравнима со скоростью обычной трассировки путей, финальный сбор, как правило, дает меньше шума (особенно для сцен со сложными условиями освещения) чем обычная трассировка путей и значительно ускоряет расчет при наличии большого числа источников света поскольку сбор фотонов становится в этом случае быстрее чем стохастическая трассировка лучей. Метод финального сбора наиболее часто применяется совместно с кэшем освещенности (будет рассмотрен далее), поскольку для кэша освещенности важна даже не столько точность решения, сколько отсутствие шума в нем.

Карты светимости (radiosity maps). Процесс поиска ближайших фотонов является, как правило, узким местом финального сбора. Поскольку в данном случае при помощи самих фотонных карт достаточно получить лишь грубую оценку освещенности, существуют различные методы ускорения финального сбора, использующие этот факт. К таким методам относятся карты светимости [53] или октанные текстуры [54]. Их смысл заключается в том, чтобы предрасчитать светимость в точках поверхности [53] (или в объеме [54]) и запомнить ее в виде карты светимости - двумерной или трехмерной (возможно разряженной) текстуры. При попадании луча финального сбора в определенную точку поверхности поиск ближайших фотонов не выполняется. Вместо этого производится быстрая выборка значения из карты светимости.

Рис. 7.43. Сбор освещенности с 10M фотонов (слева). Финальный сбор с 1M фотонов (справа).

7.6.2. Прогрессивные фотонные карты

Для фотонных карт в классической реализации серьезной проблемой является объем потребляемой памяти. Для того, чтобы получить изображение высокого качества может потребоваться до нескольких миллиардов фотонов. Хранить все фотоны в памяти в этом случае не представляется разумным, поскольку даже если хранить фотоны в соответствии с компактным представлением в 16 байт, один миллиард фотонов займет $16 * 10^9 \approx 15Gb$.

Рис. 7.44. Сбор освещенности с 10M фотонов (слева). Финальный сбор с 1M фотонов (справа).

Для того чтобы иметь возможноть обрабатывать произвольное число фотонов имея фиксированный объем памяти был разработан метод прогрессивных фотонных карт [55]. Основная идея метода прогрессивных фотонных карт заключается в том, чтобы повторять все 4 шага (испускание фотонов из источника света, трассировка, построение фотонной карты и сбор освещенности) несколько раз для порций фотонов фиксированного размера (например, по 1 миллиону фотонов). При этом, при обработке каждой следующей порции фотонов радиус сбора и аккумулируемый поток уменьшаются в соответствии с формулой 7.27. Такой метод называется Прогрессивными фотонными картами (Progressive Photon Mapping, PPM [55]). Если выбрана стратегия с фиксированым радиусом сбора, то для каждой следующей порции фотонов радиус сбора уменьшается в соответствие с соотношением 7.28. Такой алгоритм называется Стохастическими Прогрессивными Фотонными Картами (Stochastic Progressive Photon Mapping, SPPM [56], [57]). Параметр α варьируется в пределах от 0 до 1. Рекомендуемое значение - больше или равно 2/3.

$$\frac{r_{i+1}^2}{r_i^2} = \frac{N_i + \alpha M_i}{N_i + M_i} \tag{7.27}$$

$$\frac{r_{i+1}^2}{r_i^2} = \frac{i + \alpha}{i + 1} \tag{7.28}$$

В соотношении 7.27 N_i - число уже собранных фотонов. M_i - число вновь добавленных фотонов (на очередном проходе).

Метод стохастических прогрессивных карт значительно проще в реализации, чем обыкновенные прогрессивные фотонные карты, поскольку он не фикси-

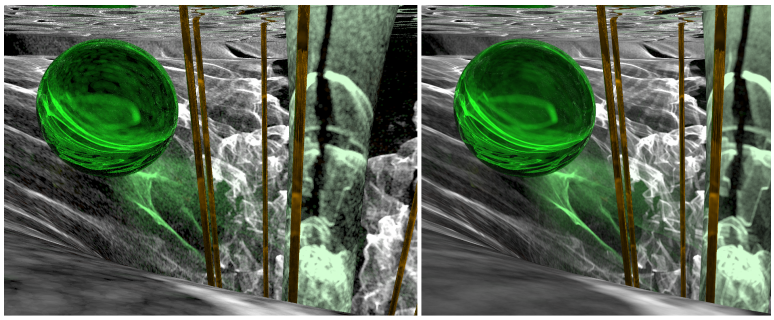


Рис. 7.45. Стохастические прогрессивные фотонные карты. 1M фотонов слева. 10M (10 проходов) - справа.

рует точки сбора (что необходимо делать в методе прогрессивных фотонных карт)[56], [57]. Причем, в [56] было показано, что стохастические прогрессивные фотонные карты обладают лучшей сходимостью. Результаты для различных порций фотонов могут быть скомбинированы простым усреднением, как и в обыкновенной трассировке путей (с.м. 'прогрессивное вычисление интеграла'). Алгоритм стохастических прогрессивных фотонных карт выглядит следующим образом:

1) Заранее выделяем некоторый объем памяти для хранения достаточно большой порции фотонов (например 1 миллион фотонов).

2) Трассируем фотоны из источника света до тех пор пока в фотонной карте не накопится указанное число фотонов. Важно отметить при этом, что при расчете доли световой энергии, которую переносит один фотон, необходимо в качестве суммарного числа фотонов использовать количество выпущенных из источника света (а не количество аккумулированных) фотонов.

3) Строим ускоряющие структуры для быстрого поиска ближайших фотонов.

4) Трассируем пути из виртуальной камеры. В точках пересечения с диффузными поверхностями собираем освещенность по формуле 7.26. При этом используем фиксированный радиус сбора r_i. Цвет, как и при обыкновенной трассировке путей накапливается в пикселах.

5) Уменьшаем радиус в соответствии с формулой 7.28. Переходим к шагу 2.

7.6.3. Некоторые важные оптимизации

Три фотонные карты В целях эффективности (повышения точности) в [51] предлагается хранить 3 раздельные фотонные карты:

1) Глобальная фотонная карта. Данная карта содержит фотоны, прошедшие путями вида
$L(S|D|G|V)^+D$ и используется для вычисления диффузного освещения.

2) Каустическая фотонная карта. Данная карта содержит фотоны, прошедшие путями вида LS^+D и используется для вычисления каустиков.

3) Объемная фотонная карта. Данная карта содержит фотоны, прошедшие путями вида
$L(S|D|G|V)^+V$ и используется для вычисления объемных эффектов.

Разделение фотонных карт позволяет обрабатывать каустики и диффузное освещение (а также объемное рассеивание) при помощи различных порций фотонов. Это особенно важно при использовании финального сбора, поскольку в этом случае для диффузного освещения с финальным сбором нужно гораздо меньше фотонов чем для каустиков.

Определение видимости. Рассмотрим изображение сцены классной комнаты на рис. 7.46. Для рассчета освещения на этой сцене фотоны испускались снаружи. При этом, лишь небольшая часть фотонов проникла внутрь комнаты через окно. Остальные фотоны сохраняются с обратной стороны стен и потолка, приводя к нэффективному расходованию памяти. Для того чтобы не сохранять фотоны на стенах снаружи комнаты, необходимо перед тем, как производить трассировку фотонов, пометить некоторым образом поверхности, для которых фотоны сохраняются в фотонной карте.

Простейший способ заключается в трассировке некоторого небольшого числа путей для каждого пиксела из виртуальной камеры и прямой пометки пересекаемых треугольников. При этом, каждую сторону треугольника необходимо помечать отдельно. Чтобы идентифицировать сторону треугольника, можно использовать векторное произведение для вычисления нормали. Поскольку вершины треугольника хранятся в памяти в заданном порядке, такой способ однозначно позволяет идентифицировать сторону по нормали. Недостаток данного метода заключаетс в том, что некоторые тонкие примитивы (треугольники) могут быть пропущены алгоритмом. Хотя, если используется финальный сбор, это не имеет значения посколько такие примитивы вносят почти нулевой вклад в результирующее значение интеграла. Т.е. чем тоньше

треугольник, тем больше вероятность не попасть в него в процессе трассировки путей из виртуальной камеры. Но также уменьшается вероятность попасть в данный треугольник в процессе сбора освещенности.

Еще один способ заключается в трассировке так называемых частиц видимости [58]. Частицы видимости представляет из себя фотоны, испускаемые из камеры. Таким образом, сначала строится фотонная карта видимости, в которой все фотоны трассируются из виртуальной камеры, а затем, используя карту видимости, строится фотонная карта освещенности.

Рис. 7.46. Комната, освещенная солнцем через окно. Лишь $\frac{1}{7}$ часть выпущенных из источника света фотонов попадает внутрь комнаты.

Карты проекций. При оптимизированной реализации построения ускоряющей структуры и сбора освещенности, одним из наиболее ресурсоемких этапов в фотонных картах становится трассировка фотонов. Причина низкой скорости трассировки заключается не том, что медленно происходит сам процесс трассировки одиночного фотона, а в том, что на сложных сценах значительная часть испускаемых из источника света фотонов погибает в процесе трассировки поскольку фотоны не достигают видимых областей. Это особенно верно при реализации определения видимости, рассмотренном выше и при трассировки каустических фотонов. На сцене из рисунка 7.46 только $\frac{1}{7}$ часть фотонов выпущенных из источника света попала внутрь комнаты. На классической сцене 'cornell box' (рис. 7.3) только $\frac{1}{10}$ часть каустических фотонов, выпущенных из источника света, попала в зеркальный или стеклянный шары, образуя каустики. Русская рулетка дополнительно усугубляет ситуацию, стохастически убивая фотоны.

Карты проекций [51] - это метод, позволяющий не выпускать фотоны в те области, где они гарантированно погибают, не сохраняясь ни на одной поверхности. Идея этого метода заключается в том, чтобы спроецировать сцену на

источник света (например при помощи растеризации в кубическую текстурную карту) и пометить на этой проекции 'мертвые области'. После чего в эти мертвые области фотоны не трассируются совсем, погибая еще при рождении на источнике света.

Приминение фильтрации. Довольно простым и эффективным способом, позволяющим улучшить четкость каустиков при помощи метода фотонных карт является применение фильтра на основе ядра Епанечникова [59]. При таком подходе освещенность при сборе домножается на определенный вес в зависимости от расстояния до фотона (формула 7.29, нормирующий множитель '2' вынесен за сумму).

$$w_i = 1 - d^2/r^2 \tag{7.29}$$

$$I(\phi_r, \theta_r) \approx \frac{2}{\pi r^2} \sum_{i=1}^{K} R(\phi_i, \theta_i, \phi_r, \theta_r) \Delta\Phi(\phi_i, \theta_i) w_i \tag{7.30}$$

Возможно использование и других типов фильтров. Например в [51] предлагается использовать 'cone filter' и фильтра Гаусса. Однако, фильтрация на основе ядра Епанечникова эффективнее в вычислительно плане, поскольку не требует вычисления квадратного корня и оперирует отношениями квадратов расстояний (а $1/r^2$ можно вычислить заранее).

7.6.4. Двунаправленные фотонные карты

Рассмотренный метод фотонных карт позволяет эффективно рассчитывать каустики и диффузное вторичное освещение. В комбинации с обратной стохастической трассировкой лучей или путей можно получить практичное для большинства ситуаций решение. Достаточно лишь выбирать на каждом переотражении - продолжать ли трассировку лучей/путей либо остановиться и произвести процедуру сбора освещённости из фотонной карты [60].

Однако, остаётся одно слабое место - это многократные глосси отражения и каустики от таких отражений. Одним из способов решения этой проблемы считается метод двунаправленных фотонных карт, использующий вместо простого выбора между трассировкой лучей и сбором освещённости оба этих метода и многократную выборку по значимости для их комбинации [61]. Такой метод называется двунаправленными фотонными картами или Bidirectional Photon Mapping (BDPM).

Следует отметить, однако, что на практике использование многократной выборки по значимости не всегда оправданно. Дело в том, что в алгоритме BDPM сбор освещённости - чрезвычайно дорогая операция, вычисляющая функцию ДФО на каждый фотон. Поэтому метод из работы [60] может оказаться эффективнее в некоторых случаях, хотя BDPM при этом более устойчив к выбросам.

7.6.5. Объемные фотонные карты

При помощи алгоритма фотонных карт можно достаточно эффективно вычислять эффекты основанные на объемном рассеянии света. В этом случае фотоны сохраняются прямо в объеме, а сбор освещенности происходит при помощи 'маршрирования по лучу' (ray marching).

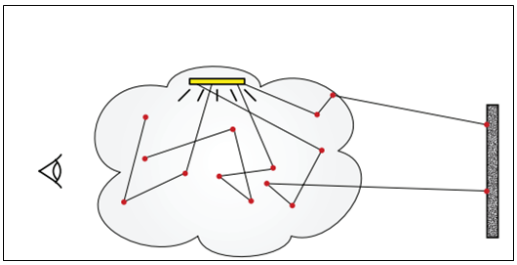


Рис. 7.47. Трассировка фотонов внутри объема [57].

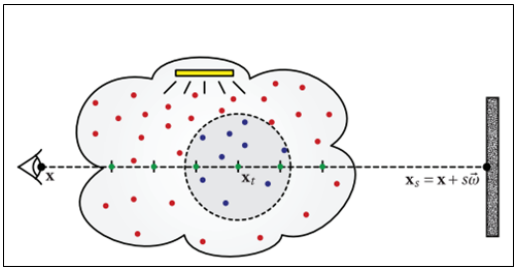


Рис. 7.48. Сбор освещенности при помощи маршрирования вдоль луча [57].

$$\frac{r_{i+1}^3}{r_i^3} = \frac{i+\alpha}{i+1} \tag{7.31}$$

Для стохастических прогрессивных объемных фотонных карт соотношение 7.28 модифицируется в 7.31 с учетом того что процесс происходит в объеме.

7.6.6. Резюме по фотонным картам

Фотонные карты - метод дающий смещенную оценку (смещенный метод). Прогрессивные фотонные карты и стохастические прогрессивные фотонные карты - также методы смещенные, но при этом состоятельные (сходимость по вероятности) [55]). Это выливается в меньшую скорость сходимости по сравнению с методами, имеющими несмещённые оценки - $O(\frac{1}{\sqrt[3]{N}})$ против $O(\frac{1}{\sqrt{N}})$ [62]. На практике смещение проявляется на изображении в виде цветных пятен (как и в кэше освещенности). В фотонных картах цветные пятна достаточно долго не удаляются с изображения в процессе расчета и изображение лишенное пятен может потребовать обработки нескольких миллиардов фотонов. При таком большом числе фотонов прямая и/или обратная трассировка путей во многих случаях даст более точное и приемлимое для человеческого глаза решение.

Финальный сбор позволяет снизить число необходимых фотонов примерно на порядок. Однако, он может быть использован только для вычисления диффузного вторичного освещения и его скорость сравнима со скоростью обыкновеной трассировки путей. Для эффективной реализации финального сбора рекоммендуется использовать карты светимости [53] или октанные текстуры [54]. Последний метод предпочтительнее, поскольку для него не нужно реализовывать отображение поверхностей трехмерных объектов на двумерную текстуру (что в общем случае является нетривиальной задачей).

Тем не менее, фотонные карты наряду с методом соединения вершин [38] являются одним из самых эффективных алгоритмов для вычисления путей вида S^+DS^+ (каустики, видимые через стекло или зеркало). Такие пути, как правило, значительно хуже вычисляются несмещёнными методами.

7.7. Кэш Освещенности (Irradiace Cache)

Кэш Освещенности (Iradiance Cache, IC) – это не способ вычисления интеграла освещенности. Это лишь способ ускорения вычисления этого интеграла на множестве точек. Алгоритм был впервые представлен в работе [63]. Основная идея заключается в том, что вторичное освещение разделяется на две компоненты – низкочастотную (диффузную) и высокочастотную (отражающую). Низкочастотная компонента на изображении и в трехмерном пространстве меняется плавно, поэтому ее можно вычислить каким-либо из методов лишь в очень небольшом числе точек, а в остальных - интерполировать [64]. Рисунки 1 и 2 демонстрируют пример использования кэша освещенности.

Высокочастотная компонента обычно обусловлена резкими максимумами BRDF, поэтому она может быть вычислена сэмплированием с помощью относительно небольшого числа (что конечно не всегда так) лучей только в максимумах BRDF (importance sampling).

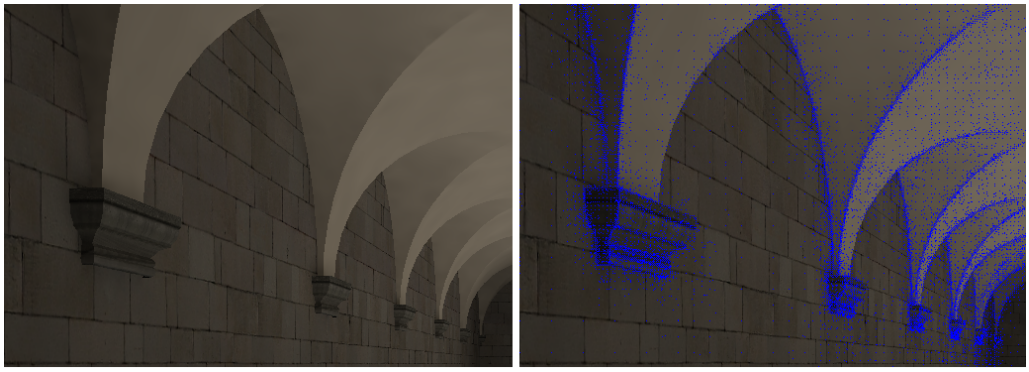


Рис. 7.49. Освещение вычисленное при помощи кэша освещенности (слева). Визуализированные точки кэша (справа).

Различают кэш освещенности в трехмерном пространстве сцены (world space) и в экранной плоскости (screen space).Реализация кэша в экранной плоскости проще и эффективнее. Интерполяция также производится в пространстве экрана. Однако, кэш освещенности в пространстве экрана может быть использован по очевидным причинам только для первично видимых поверхностей.

Как правило, кэш освещености вычисляется на лету. Каждый раз при вычислении вторичного диффузного освещения делается попытка выборки из кэша. Если значение может быть выбрано так, что ошибка меньше допустимой величины, оно получается как результат интерполяции ближайших точек кэша. Если ошибка больше допустимой величины, значение честно вычисляется с помощью трассировки лучей по всем направлениям (или финального сбора из фотонной карты) и полученная точка сохраняется в КЭШе [64].

Исходные параметры: p - точка на поверхности, n - нормаль к поверхности в точке p
Результат: Значение освещенности
if InterpolationIsPossibe(p,n) **then**
  return IrradiacneCacheLookUp(p,n) ;
else
  $I \leftarrow EvalIntegral(p, n)$;
  $R \leftarrow CalcValidityRadius(p, n)$;
  $InsertRecordInCache(I, R, p, n)$;
  return I;
end

Алгоритм 7: Алгоритм кэширования освещенности

Алгоритм кэширования вторичного освещения содержит 2 узких места - расчет собственно вторичного освещения в точках кэша и финальный алгоритм интерполяции освещения во всех остальных точках.

7.7.1. Вычисление вторичного освещения

Стандартным способом вычисления вторичного освещения в точках кэша является монте-карло трассировка лучей или финальный сбор в сочетании с фотонными картами. Однако в [64] предлагается вычислять вторичную освещенность в точках с помощью растеризации в кубическую текстурную карту. Данная операция может производиться очень быстро т.к. имеет аппаратную поддержку даже в довольно старых видеокартах (обычно используется для симуляции отражений на объектах).

Отдельного внимания заслуживает вопрос точности вычисления освещенности в точках кэша. Допустим для вычисления освещенности при помощи трассировки путей в пикселах на некоторой сцене будет достаточно около 1000 путей на пиксел. Тогда для вычисления освещенности в точках кэша потребуется может потребоваться большая точность (порядка 4000 путей). Недостаточная точность вычисления для попиксельной трассировки путей приводит к шуму на изображении, который в некоторой степени фильтруется глазом и мало заметен. В случае кэша освещенности, недостаточная точность приводит к появлени цветных пятен, которые гораздо более заметны для человеческого глаза, чем шум.

7.7.2. Вычисление радиуса валидности

Одним весьма неочевидным моментом при реализации кэша освещенности является вычисление радиуса валидности точки кэша (переменная R, алгоритм 7). Подразумевается, что каждая точка кэша освещенности может быть

использована для интерполяции только в некоторой определенной области. Есть по крайней мере 2 причины, по которым радиус валидности точки нужно ограничивать. Первая причина - стремление избежать артефактов. Вторая - скорость выборки из кэша. Если слишком много точек будут затрагиваться при каждой выборке, интерполяция станет медленной.

Обычно область валидности ограничивают сферой с центром в точке кэша и некоторым радиусом, называемым радиусом валидности (хотя есть работы где для этой цели используются эллипсоиды [65]). В [64] для вычислениия радиуса валидности было использовано 5 различных эвристик. Каждая из них имеет недостатки, но все вместе они дают удовлетворительный результат.

Из самых простых - используется эвристика расстояния до ближайших поверхностей. Эта эвристика вычисляется исключительно из геометрических соображений. При оценке освещенности, для всех трассируемых лучей запоминается расстояние, на котором они ударились о поверхность и из этих расстояний берется наименьшее. Однако при использовании этого подхода, в процессе нахождения минимума важно не принимать в расчет лучи, имеющие маленький угол с тангент-плоскостью (рис. 7.50).

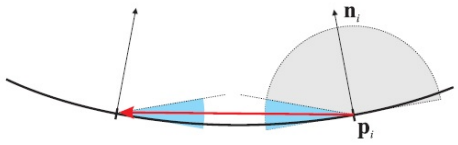


Рис. 7.50. Точки расположенные на вогнутой поверхности. Если принимать в расчет лучи, идущие перпендикулярно нормали, радиус валидности получится слишком маленьким.

В противном случае на вогнутых криволинейных поверхностях радиус валидности точки всегда будет равным нулю. Альтернативный метод реализации эвристики расстояния до поверхностей заключается в том, чтобы не искать минимальное расстояние, а вычислить так называемое 'среднее грамоническое расстояние' (формула 7.32).

$$R_i^{HMD} = \frac{N}{\sum_{i=1}^{N} \frac{1}{r_i}} \tag{7.32}$$

Число N в формуле 7.32 - это количество лучей используемых при сэмплировании полусферы. r_i - расстояние от точки кэша до поверхности в которую ударился соответствующий луч. Формула 7.32 для среднего гармонического расстояния характерна тем, что чем меньше расстояние r_i тем больше оно

влияет на финальный результат. Дополнительно размер радиуса валидности рекомендуется ограничивать снизу размером 2-4 пикселей (спроецированных на сцену в мировое пространство) и сверху размером 20-64 пикселей. Конкретные числа могут варьироваться.

7.7.3. Структуры данных и интерполяция

Для хранения точек кэша освещенности обычно используется специальное окто-дерево со множественными ссылками (multiple reference octree) [64]. Записи кэша освещенности в таком окто-дереве хранятся только в листьях в виде сфер таким образом, что каждый лист хранит список всех сфер, которые он пересекает. Основной плюс такого окто-дерева в том, что возможно осуществить простой нерекурсивный поиск от корня к листу с последующим перебором всего лишь нескольких записей кэша освещенности.

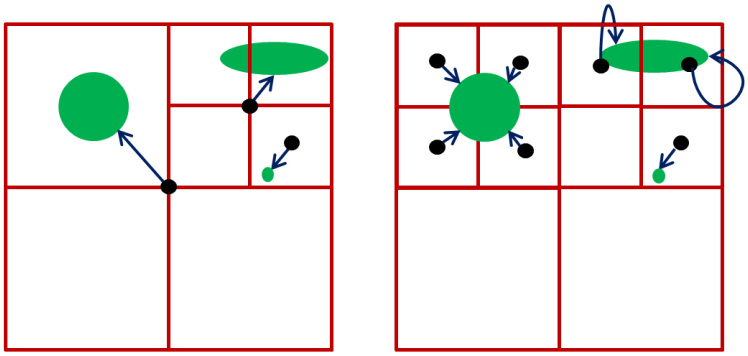


Рис. 7.51. Октодерево с одиночными ссылками (слева) и окто-дерево с множественными ссылками (справа). Стрелки обозначают ссылки.

Интерполяция. В заданной точке трехмерного пространства, при помощи поиска в октодереве необходимо найти все сферы (с центрами в точках иррадианс кэша и радиусами равными радиусам валидности записей кэша), которые пересекают данную точку. Затем, пройдя в цикле по всем точкам, нужно собрать с них освещение. Одна из практичных формул была предложена Tabellion-ом и Lamorlette в [53].

7.7.4. Резюме по кэшу освещенности

Кэширование вторичного освещения - весьма эффективная техника ускорения вычисления глобального освещения, позволяющая снизить количество

необходимых вычислений практически на 3 порядка (для больших разрешений). Допустим имеется изображение размером 1024x1024 пиксела. Если вычислять вторичное освещение в каждой точке, то потребуется суммарно протрассировать порядка 10^9 лучей только для вычисления вторичной освещенности. При использовании кэша освещенности количество точек, в которых необходимо вычислить вторичную освещенность сокращается с миллиона до нескольких тысяч. Соответственно, число лучей падает с 10^9 до 10^6 - 10^7.

Однако, кэширование освещенности имеет определенные недостатки:

1) Кэш освещенности хорошо работает только на сценах с преимущественно гладкими поверхностями. На сценах с большим количеством геометрических деталей (например карты нормалей для имитации микрорельефа) кэш освещености не даст преимущества и может даже замедлить рендеринг.

2) Кэш освещенности достаточно сложно сделать параллельным [66].

3) Кэш освещенности дает мерцание при анимации. Если стохастическая трассировка путей дает шум, который эффективно удаляется при помощи размытия в движении, то пятна вызванные кэшированием освещенности удалить гораздо труднее. Для того чтобы избежать мерцания при анимаци потребуется дополнительно проводить интерполяцию во времени и фиксировать положения записей кэша освещенности [67].

4) Кэш освещенности все же снижает точность решения и является смещенным методом. Чем ниже точность, тем больше выигрыш в скорости, но тем более заметны артефакты.

Appendices

Приложение А.

Вывод линейного уравнения излучательности

Запишем уравнение освещенности:

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int\limits_{\Omega} f_r(x, \omega_o, \omega) L_i(x, \omega) \cos\theta d\omega$$

Рассмотрим данное уравнение. Можно заметить, что под интегралом всегда присутствует величина $L_o(x, \omega_o)$ - собственное излучения объекта в точке в направлении ω_o, не зависящее от его освещения другими объектами. Данная величина всегда присутствует в интеграле освещенности для излучающих объектов. Это означает, что любой источник света для уравнения излучательности представляется в вышенаписанной форме (в силу того, что любой источник света по определению будет излучающим объектом).

Рассмотрим идеально диффузную поверхность. Вычислим, как будет выглядеть уравнение рендеринга при работе с идеально диффузными поверхностями. От бесконечно малой точки x мы перейдем к патчу (участку постоянной интенсивности). Наша цель - в конечном итоге преобразовать уравнение освещенности к системе линейных уравнений.

Рассмотрим $B(x)$ - интеграл по всем направлениям, в которых объект может излучать свет с точки x по всей полусфере. Данный интеграл вычисляется для точки x с учетом углов по всем направлениям излучения. Домножение на косинус данных углов позволяет корректно учесть тот факт, что энергия, измеряемая под углом, собирается с некоторой большей площади (что следует из определения понятия излучательности как отношения излученной энергии

к площади излучаемой поверхности). В силу того, что мы рассматриваем диффузный объект, L будет являться константой (объекты с более сложной Двулучевой Функцией отражения мы в данном случае не рассматриваем).

$$B(x) = \int_{\Omega} L_o(x) \cos\theta d\omega_o = \pi L_o(x)$$

В силу того, что L постоянна, она легко выносится за знак интеграла. Таким образом, мы получаем интеграл по полусфере от косинусов. Заменяя данный интеграл по полусфере на повторный интеграл и преобразуя его, легко вычислить, что данный интеграл будет равен π.

Таким образом, мы получили связь между излучательностью B и яркостью L точки рассматриваемой поверхности. Яркость L - это та величина, которую нам нужно в конечном итоге найти для того, чтобы посчитать итоговое изображение. Величина B будет далее участвовать в процессе переноса энергии между излучающими поверхностями. В силу зависимости, установленной между данными величинами, можно легко переходить от одной величины к другой.

Двулучевая функция отражения идеально диффузной поверхности была рассмотрена в предыдущих разделах и представляет собой отношение исходящей яркости к освещенности данной поверхности в некоторой точке. Мы выяснили, что для диффузных объектов L_0 является константой. Из этого следует, что Двулучевая функция отражения поверхности $f_r(x, \omega_o, \omega)$ (см. выражение под интегралом) также будет константой, равной $\frac{\rho}{\pi}$, где ρ - коэффициент поглощения поверхности.

A.1. Перенос энергии - упрощения формулы глобальной освещенности

Поставим формулу для двулучевой функции отражения в исходный интеграл и домножим обе части полученного уравнения на π:

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{\Omega} f_r(x, \omega_o, \omega) L_i(x, \omega) \cos\theta d\omega$$

$$\pi L_o(x, \omega_o) = \pi L_e(x, \omega_o) + \pi \int_{\Omega} f_r(x, \omega_o, \omega) L_i(x, \omega) \cos\theta d\omega$$

Получаем, что слева будет стоять выражение $\pi L_o(x, \omega_o)$, что, согласно нашим выводам, эквивалентно излучательности, а справа записана полная светимость ламбертовой поверхности по рассматриваемой полусфере. Плотность поглощения поверхности также постоянна в силу ее диффузности, поэтому величину ρ (не зависящую от других величин) можно вынести за знак интеграла, а полученное уравнение сократить на π. В итоге получаем следующий интеграл:

$$B(x) = E(x) + \rho(x) \int_{\Omega} L_i(x, \omega) \cos\theta d\omega$$

В данном выражении отсутствует Двулучевая функция отражения , однако по-прежнему присутствует L под знаком интеграла.

A.2. Изменение области определения

Чтобы избавиться от данной величины L под знаком интеграла, нужно в первую очередь изменить форму, с которой мы работаем. В данный момент работа идет с интегралом по полусфере, т.е. с интегралом в полусферической форме. Данный интеграл возможно переписать в несколько другой форме, преобразуя его в интеграл по точкам поверхностей сцены. Величину $d\omega$ можно выразить через площадь поверхности сцены.

Пусть у нас есть точка p некоторой поверхности и из всей полусферы (из всего пространства Ω углов по полусфере) мы рассматриваем телесный угол $d\omega$, соответствующий некоторому небольшому участку на полусфере. Наша задача - найти площадь данного участка. Будем считать, что рассматриваемая сфера является единичной. Найдя способ расчета данной площади, мы перейдем к другой системе счисления, которая упростит исходную задачу.

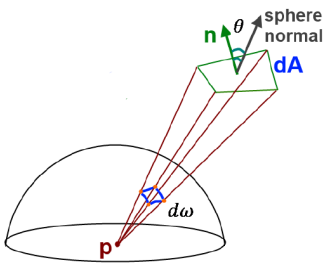


Рис. A.1. Смена области определения Форм-Фактора.

Рассмотрим некоторый объект произвольной сцены. Площадь $d\omega$ вычисляется как площадь того участка поверхности объекта, с которым пересекся рассматриваемый луч (фактически - как площадь пересечения рассматриваемого телесного угла со сферой с центром в точке P). Обозначим данную дифференциальную площадь как dA. Площадь данного участка поверхности dA, а также расстояние r считаются известными.

Пусть рассматриваемая площадка находится под некоторым углом к поверхности на расстоянии R от центра сферы и требуется найти площадь поверхности пересечения данного луча с полусферой. Фактически это задача расчета площади для поверхности криволинейного объекта под заданным углом. В таком случае площадь данного участка будет вычисляться как произведение площади dA на косинус угла между нормалью к площадке и нормалью к бесконечно малому участку dA на сфере, поделенное на r^2 (согласно тому факту, что площадь сферы и любой ее части с увеличением размера сферы растет пропорционально ее радиусу).

$$d\omega = \frac{\cos\theta dA}{r^2}$$

Искомая величина $d\omega$, таким образом, может быть выражена через приведенные площадки в сцене. Просуммировав $d\omega$ по всей полусфере (т.е. рассчитав с некоторым приближением $d\omega$ по всем площадкам сцены), можно вычислить таким образом приближенное значение интеграла. Основная проблема состоит в том, что площадки в принципе могут перекрываться, и между двумя произвольно взятыми площадками может располагаться еще одна площадка. Данное обстоятельство делает расчет переноса энергии между двумя данными площадками физически некорректным. Для более корректного рассмотрения данных двух случаев вводится специальная функция от двух аргументов x и y, связывающая точки двух произвольных площадок, из которых дифференциально построена рассматриваемая поверхность:

$$V(x,y) = \begin{cases} 1, & \text{если x и y взаимно видны} \\ 0, & \text{иначе} \end{cases} \tag{A.1}$$

С учетом введенной функции и вышезаписанных формулировок интеграл преобразуется к следующему виду:

$$V(x,y) = E(x) = \rho(x) \int\limits_{y \in S} B(y) \frac{\cos\theta \cos\theta\prime}{\pi r^2} V(x,y) dA$$

Напомним, что для произвольной точки сцены мы рассматриваем остальные точки сцены, из которых она произвольно видна, и таким образом собираем оттуда всю энергию.

A.3. Переход от непрерывного решения к дискретному

До текущего момента мы рассматривали способы расчета излучательности для случая непрерывных переменных. Для быстрого расчета интеграла нам требуется перейти от данных непрерывных переменных в дискретное пространство. Поэтому рассматриваемую поверхность требуется разбить на несколько частей (квадратов или так называемых *патчей*) B_i с площадью A_i каждого патча (A.2).

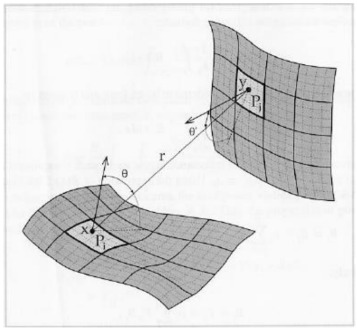


Рис. A.2. Разбиение поверхности на патчи.

Будем считать, что в пределах одного патча излучательность не изменяется (равна константе) и что нам требуется посчитать не излучательность всех точек поверхности, а среднюю излучательность поверхности по всем точкам патча. Фактически мы рассчитываем излучательность в каждой точке p и рассчитываем ее в точке x, принадлежащей данному патчу. В дальнейшем будем рассматривать именно B_i, т.е. среднюю излучательность всего патча целиком.

$$B_i = \frac{1}{A_i} \int\limits_{x \in i_j} B(x) dx$$

Таким образом, от интеграла по всем точкам сцены мы переходим к сумме интегралов, каждый из которых представляет собой интеграл по всем точкам конкретного патча. Иначе говоря, интеграл по всем точкам рассматриваемой

сцены (которых бесконечное количество) преобразуется к интегралу по всем патчам поверхностей сцены (число которых ограниченно и, как правило, не велико).

$$B(x) = E(x) + \rho(x) \sum_{j=1}^{N} \int\limits_{y \in p_j} B(y) \frac{\cos\theta \cos\theta\prime}{\pi r^2} V(x,y) dA$$

После этого дополнительно интегрируем обе части нашего уравнения по всем точкам некоторого i-го патча . Слева получаем B_i по определению, справа - сумма среднего самосвечение данного патча $E(x)$ и умноженной на коэффициент ρ_i - среднему по патчу коэффициент поглощения и некоторой суммы.

$$B(x) = E(x) + \rho(x) \int\limits_{y \in S} B(y) \frac{\cos\theta \cos\theta\prime}{\pi r^2} V(x,y) dA$$

$$\frac{1}{A_i} \int\limits_{y \in p_j} B(x) dx = \frac{1}{A_i} \int\limits_{y \in p_j} [E(x) + \rho(x) \sum_{j=1}^{N} \int\limits_{y \in p_j} B(y) \frac{\cos\theta \cos\theta\prime}{\pi r^2} V(x,y) dy]$$

Стоит обратить внимание, что, несмотря на кажущееся усложнение (мы фактически перешли от одинарного интеграла к двойному), в получившемся двойном интеграле отсутствует L и P. В нем присутствуют косинусы, квадраты (расстояния между патчами) и функция светимости. Это значит, что *данный интеграл не зависит от наблюдателя и от источника света в сцене.* Таким образом, получившийся интеграл фактически зависит только от взаимного расположения патчей x,y сцены, по которым проходит интегрирование.

A.4. Уравнение дискретной излучательности

Будем считать, что форм-факторы для всех возможных пар площадок ij подсчитаны и подставлены под знак интеграла. Получаем следующую систему:

$$B_i = E_i + \rho_i \sum j = 1N B_j F_{ij}$$

Правая часть уравнения, как можно заметить, представляет собой некоторую сумму (интеграл находится под знаком суммы, будем считать, что он уже подсчитан). Получаем:

$$B = E + BF$$

где F - это в данном случае некоторая матрица с элементами F_ij, состоящая из взаимных Форм-Факторов для рассматриваемых патчей. Переписывая данное выражение, получаем классический вид СЛАУ:

$$E = MB, M = 1 - F$$

Величина E представляет собой вектор-столбец, содержащий самосвечение каждого патча , M - разность единичной матрицы и матрицы F, домноженная на коэффициент поглощения поверхности патча и B - вектор-столбец, включающий в себя излучательности каждого патча. По известным значениям E и M предстоит вычислить B. Таким образом, задача нахождения излучательности для каждого патча расматриваемой сцены, состоящей из N патчей, сводится к решению СЛАУ с матрицей M размера NxN.

Приложение В.

Дополнительные способы расчёта форм-факторов

В.1. Расчёт форм-факторов. Аналогия Нуссельта.

Запишем выражение для произвольного Форм-Фактора:

$$F_{ij} = \frac{1}{A_i} \int\limits_{A_i} \int\limits_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} V_{ij} dA_i dA_j$$

Данная величина представляет собой два площадных интеграла по двум площадкам (площадке i и площадке j. Видимость заданного патча при этом определяется из геометрических соображений.

Вычисление подобных двойных интегралов, как правило, занимает много времени. Данные вычисления можно упростить: существуют аналогии, позволяющие оптимизировать процесс вычисления интеграла. Приемы для упрощения подобных вычислений могут осуществляться как в счет некоторой потери точности, так и без потерь точности.

Аналогия Нуссельта представляет собой чистого вида интегрирование в особой геометрической форме.

Рассмотрим вычисление форм-фактора между дифференциальной прощадкой D_i и интегральной площадкой A_j. Немецким инженером Джоном Нуссельтом был разработан универсальный алгоритм, позволяющий вычислять

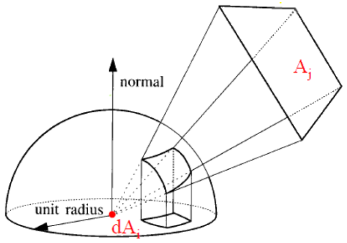


Рис. B.1. Аналогия Нуссельта.

значение дифференциального форм-фактора между поверхностью и точкой на другой поверхности.

Суть аналогии Нуссельта заключается в проецировании интегральной площадки A_j на поверхность единичной сферы с центром в D_i (рисунок B.1). На первом этапе площадка A_j проецируется на плоскость, перпендикулярную нормали к сфере. На втором этапе полученная проекция, в свою очередь, проецируется на единичную сферу. Получив проекции площалки A_j вдоль нормали к сфере на поверхность сферы , можно вычислить площадь искаженного квадрата на сфере. Данную операцию требуется произвести для всех точек поверхности A_i (т. е . для любого дифференциального участка dA_i и просуммировать полученный результат. Таким образом, расчет Форм-Фактора с использованием аналогии Нуссельта сводится от вычисления двойного интеграла к вычислению одного интеграла, а вложенный интеграл вычисляется при помощи проецирования на единичную сферу.

Аналогия Нуссельта достаточно сложна технически, несмотря на то, что существенно упрощает вычисления.

B.2. Расчёт форм-факторов. Метод полукуба.

B.2.1. Классический метод полукуба

Во многих случаях в качестве оптимизации возможно произвести аппроксимацию между дифференциальной точкой поверхности dA_i и некоторым полигоном полигональной площадки A_j.

Идея состоит в отказе от абсолютной точности и замене рассматриваемой полусферы на полукуб (рисунок B.2). Для нахождения приближенного значения форм-фактора нужно спроецировать на все стороны полукуба полигон, соответствующий патчу, для которого ищется форм-фактор. Как только

будет получена площадь полигонов-проекций, можно будет выяснить, какое количество полигонов исходной модели находится из данной стороны куба. В стандартном методе полукуба в качестве грубой аппроксимации единичной полусферы используется полукуб со стороной 1. Данный полукуб разбивается на множество ячеек (например, 512x512 ячеек для полной, т.е. верхней грани куба).

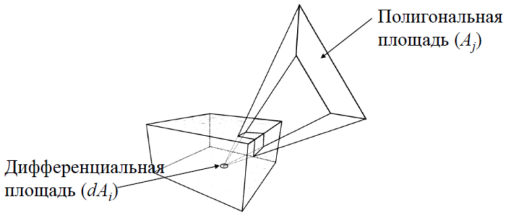


Рис. B.2. Метод полукуба: полигональная и дифференциальная площади при расчете форм-фактора.

Рассмотрим, как происходит работа с полукубом (рисунок B.3). Сначала некоторый полигон (или совокупность полигонов некоторого объекта) сцены проецируется на грани полукуба. Предполагается, что полукубу соответствует множество ячеек поверхности его граней. Для всех ячеек полукуба хранится информация о том, в какие пиксели куба попали те или иные полигоны сцены, а также информация о площади этих объектов. Проецируя полигоны(т.е. фактически патчи) объекта на грани полукуба, можно получить информацию о форм-факторе данного объекта.

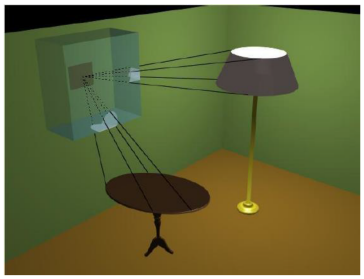


Рис. B.3. Пример работы метода полукуба для сцены.

Для вычисления видимости треугольника можно ипользовать обыкновенный z-буфер. Фактически, после каждой проекции для каждой точки полукуба нужно вычислять, не перекрывает ли эту точку другой объект своей проекцией, находясь ближе к поверхности воображаемого полукуба. В случае перекрытия тот патч, который находится дальше от грани полукуба, не визуализируется.

B.2.2. Достоинства и недостатки метода полукуба

Метод полукуба обеспечивает быстрое вычисление форм-факторов за счет использования аппроксимации, позволяющей очень быстро свести расчеты к вычислению линейных проекций. Одним из быстрых методов (например, методом растеризации) производится проецирование патчей на поверхность полукуба, а параллельно проводится одновременное вычисление форм-факторов для нескольких точек сцены.

В то же время, поскольку метод полукуба позволяет осуществить вычисление дифференциально-конечного форм-фактора, требуется дополнительно отодвигать центр проекции вокруг некоторой точки, что будет в несколько раз увеличивать вычислительную сложность метода. Другой вариант - считать, что все объекты находятся достаточно далеко для того чтобы разница между удаленными пластинками была небольшой, чтобы можно было предположить (что удобно), что на всей пластинке будет одинаковая видимость для всех ее точек. Тем самым можно обощить дифференциально-конечный форм-фактор на конечный форм-фактор.

Еще одним существенным недостатком метода полукуба является возникновение алиасинга: из-за дискретности измерений участков поверхности полукуба возникает эффект блочности (B.4).

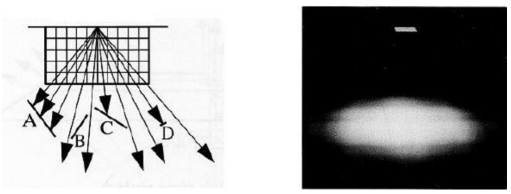


Рис. B.4. Эффект алиасинга при работе метода полукуба.

Приложение C.

Ускорение поиска пересечений

C.1. Регулярные сетки

Ускорение поиска пересечений в трассировке лучей - довольно обширная тема.

Если сцена представлена набором примитивных объектов (сфера, AABB, треугольник), то ускорять процесс поиска пересечений при помощи структур пространственного разбиения имеет смысл, как правило, если число объектов больше одного-двух десятков. В противном случае, стоимость поиска может быть сравнима со стоимостью перебора всех объектов вслепую. Как правило, это происходит из-за большего количеста ошибок предсказателя ветвлений при выполнении сложного кода (особенно верно для иерархического поиска в дереве).

Данная ускоряющая структура регулярно разбивает пространство на N^3 вокселей, где N - разрешение сетки. При поиске в регулярной сетке проверяются только те объекты, которые оказались внутри вокселей на пути луча.

Алгоритм построения сетки сводится к нахождению геометрических пересечений между вокселами и объектами сцены. Каждый воксел должен содержать список всех указателей или индексов объектов с которыми он имеет пересечение.

Алгоритм поиска носит фамилию своего автора Фуджимото [68]. Функция поиска состоит из 2 частей. Инициализация и перебор вокселей. Инициализирующая часть весьма объемная, однако, она выполняется для каждого луча 1 раз. Алгоритм 8 и рисунок C.2 демонстрируют поиск Фуджимото.

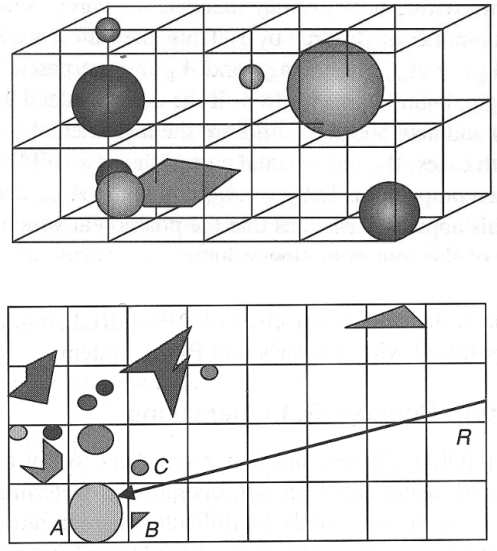


Рис. C.1. Регулярное разбиение пространства.

Исходные параметры: Луч - ray, ограничивающий куб - box
Результат: Перебор вокселей, лежащих на пути луча
Инициализирующая часть алгоритма:
$(tMaxX, tDeltaX, stepX) \leftarrow$ ВычислитьЗначения(ray,box);
$(tMaxY, tDeltaY, stepY) \leftarrow$ ВычислитьЗначения(ray,box);
Основная часть алгоритма:
while (x < N) and (x >= 0) and (y < N) and (y >= 0) **do**
  ОбработатьВоксел(x,y);
  if $tMaxX < tMaxY$ **then**
    $tMaxX \leftarrow tMaxX + tDeltaX$;
    $x \leftarrow x + stepX$;
  else
    $tMaxY \leftarrow tMaxY + tDeltaY$;
    $y \leftarrow y + stepY$;
  end
end

Алгоритм 8: Обход регулярной сетки Fujimoto

Несмотря на простой алгоритм перебора вокселей, регулярная сетка - не очень хорошая ускоряющая структура. Причин этому несколько:

1) Высокие затраты памяти, пропорционально N^3.

2) Отсутствие адаптивности или проблема чайника на стадионе. Регулярная сетка одинаково разбивает все пространство независимо от того,

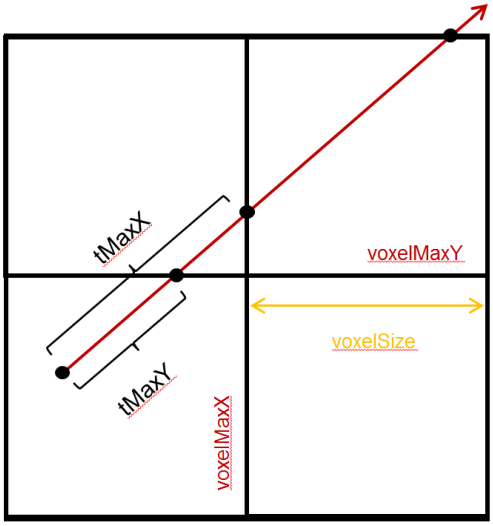


Рис. C.2. Поиск в регулярной сетке.

какая геометрия и как расположена на сцене. Размер воксела выбирается исходя из некоторого среднего размера объектов. При этом если выбрать размер сетки большим, поиск пересечений значительно замедлится для небольших высокополигональных объектов. А если размер воксела выбрать маленьким, не только значительно возрастут затраты памяти но и замедлится трассировка на всей остальной сцене, т.к. в процессе трассировки луча придется перебирать большее число вокселей.

3) Проблема повторных пересечений. Эта проблема также усугубляется с ростом разрешения сетки. Объекты, пересекающие несколько вокселей при поиске пересечений будут встречаться несколько раз. Следовательно, при трассировке в регулярной сетке луч будет вычислять пересечение с такими примитивами более чем один раз. Проблему можно частично амортизировать сохраняя идентификатор луча в примитивах, с которым пересечение уже было посчитано. Если в данный момент времени идентификатор луча равен идентификатору сохраненному в примитиве, вычислять пересечение не нужно. Однако этот подход трудно применять при параллельной реализации трассировки лучей.

C.2. Окто-деревья и иерархические сетки

Перейдя к иерархическому представлению, можно в некоторой степени решить первые 2 проблемы регулярной сетки (затраты памяти и неадаптивность). Иерархическая сетка представляет из себя дерево (обычно неболь-

шой глубины, но зато очень широкое), в узлах которого лежат относительно небольшие регулярные сетки. Если размер сетки положить равным 2x2x2 получаем окто-дерево. Таким образом, окто-дерево представляет из себя иерархическое разбиение пространства на 8 частей (рис. C.3).

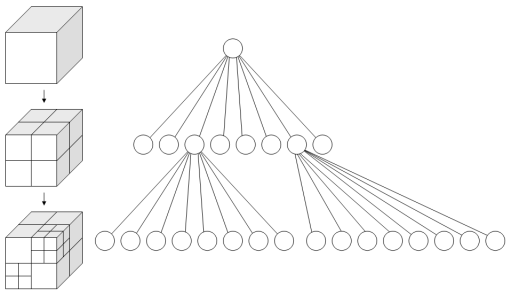


Рис. C.3. Окто-дерево.

Окто-деревья широко используются в компьютерной графике для различных задач пространственного поиска, но они довольно плохо подходят для ускорения трассировки лучей. Алгоритм обхода окто-деревьев достаточно сложен. При этом, адаптивность у окто-дерева низкая, так как чтобы ограничить небольшой объект (чайник на стадионе) нужно много уровней подразбиения. Окто-дерево, таким образом, обычно содержит множество пустых узлов и сложные регионы обходятся медленно (рис. C.4). При этом, проблема повторных пересечений по прежнему остается актуальной для окто-деревьев, поскольку при разбиение один объект очень часто попадает сразу в несколько дочерних узлов.

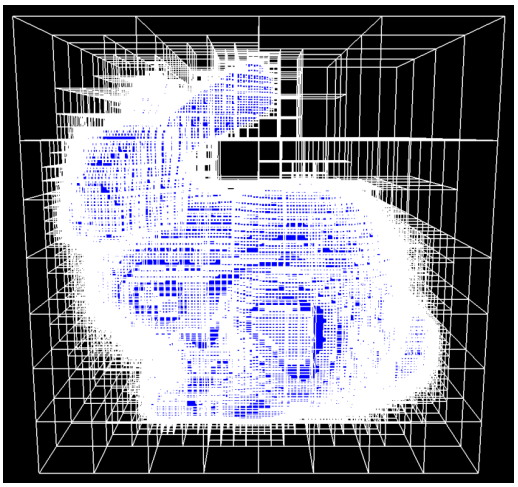


Рис. C.4. Модель стэнфордского кролика в окто-дереве. Количество белого цвета на изображении визуально позволяет оценить сложность трассировки.

C.3. kd-деревья

Рассмотрим структуру бинарного пространственного разбиения, называемую kd-дерево (аббревиатура kd расшифровывается как k-dimentional). Эта структура представляет собой бинарное дерево ограничивающих параллелепипедов, вложенных друг в друга. Каждый параллелепипед в kd-дереве разбивается плоскостью, перпендикулярной одной из осей координат на два дочерних параллелепипеда.

Вся сцена целиком содержится внутри корневого параллелепипеда, но, продолжая рекурсивное разбиение параллелепипедов, можно прийти к тому, что в каждом листовом параллелепипеде будет содержаться лишь небольшое число примитивов. Таким образом, kd-дерево позволяет использовать бинарный поиск для нахождения примитива, пересекаемого лучом.

Если плоскость, разбивающую пространство выбирать каждый раз по середине паралеллипипеда (при этом текущий узел можно разбивать всегда по оси, в которой он имеет максимальный размер), kd-дерево будет эквивалентно окто-дереву и наследует все его недостатки (за исключением сложного алгоритма обхода). Однако, основное преимущество kd-дерева над окто-деревом заключается как раз в том, что плоскость разбиения можно ставить не только по середине разбиваемого узла, а в любом месте.

Алгоритм построения kd-дерева можно представить следующим образом (будем называть прямоугольный параллелепипед сокращенно AABB).

1) 'Добавить' все примитивы в ограничивающий AABB. Т.е построить ограничивающий все примитивы AABB, который будет соответствовать корневому узлу дерева.

2) Если примитивов в узле мало или достигнут предел глубины дерева, завершить построение.

3) Выбрать плоскость разбиения, которая делит данный узел на два дочерних. Будем называть их правым и левым узлами дерева.

4) Добавить примитивы, пересекающиеся с AABB левого узла в левый узел, примитивы, пересекающиеся с AABB правого узла в правый.

5) Для каждого из узлов рекурсивно выполнить данный алгоритм начиная с шага 2.

На рисунке C.5 изображен процесс построения kd-дерева с учетом некоего оптимального алгоритма выбора разбивающей плоскости. Самым сложным в построении kd-дерева является 3-ий шаг. От него напрямую зависит эффективность ускоряющей структуры. Существует несколько способов выбора плоскости разбиения, рассмотрим их по порядку.

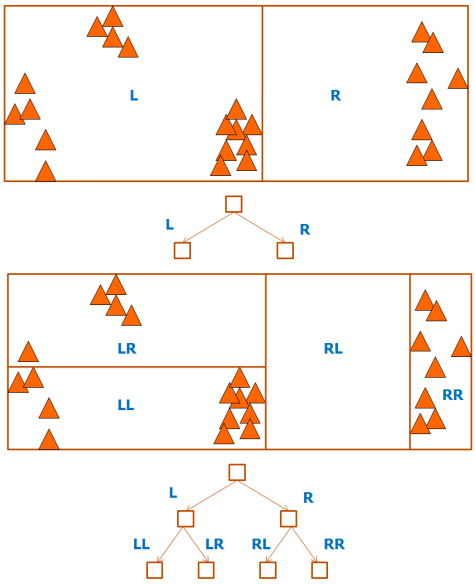


Рис. C.5. Построение kd-дерева. L - обозначает левое поддерево. R - правое.

C.3.1. Разбиение по середине.

Самый простой способ - выбирать плоскость разбиения по центру. Сначала выбираем ось (x, y или z), в которой AABB имеет макимальный размер, затем разбиваем его по центру. Ранее обсуждалось, что kd-дерево в этом случае имеет плохую адаптивность и эквивалентно окто-дереву.

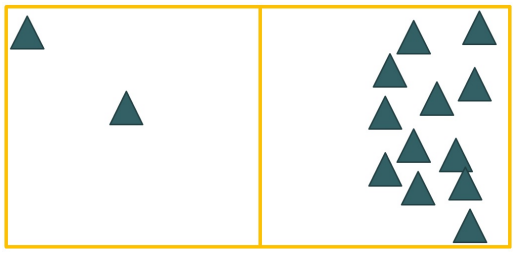


Рис. C.6. Разбиение по середине.

C.3.2. Разбиение по медиане.

Если разбивать узел на два дочерних таким образом, чтобы в правом и левом поддереве количество примитивов было одинаково, будет построено сба-

лансированное дерево. Это не очень удачная идея. Все дело в том, что сбалансированные деревья могут помочь только если искомый элемент каждый раз находится в случайном узле, то есть если распределение лучей по узлам во время поиска будет равномерно. В действительности, это не так. Лучей в среднем пойдет больше в тот узел, который больше по своей площади поверхности, а при медианном разбиении эти площади у узлов могут быть разные.

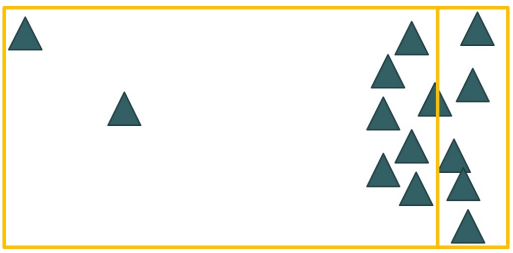


Рис. C.7. Разбиение по медиане.

C.3.3. Разбиение на основе оптимизации функции стоимости.

Каковы же критерии хорошо-построенного kd-дерева? На интуитивном уровне такой критерий можно описать фразой 'как можно больше пустого пространство должно быть отброшено как можно быстрее'. Для решения поставленной задачи используем формальный подход. Введем функцию стоимости поиска, которая будет отражать, насколько дорого по вычислительным ресурсам производить поиск в данном узеле случайным набором лучей. Будем вычислять ее по следующей формуле [69]:

$$SAH(x) = CostEmpty + SA(Left) * N(Left) + SA(Right) * N(Right)$$

В приведенной выше формуле SA(Left) и SA(Right) - площадь поверхности (SurfaceArea) соответственно левого и правого дочерних узлов, получающихся при разбиении. N(Left) и N(Right) - количество примитивов, попавших при разбиении соответственно в левое и правое поддерево. Аргумент функции x является одномерной координатой плоскости разбиения. На рисунке C.8 можно увидеть, что SAH сразу отбрасывает большие пустые пространства, плотно ограничивая геометрию.

Хорошими кандидатами на минимум SAH могут служить границы примитивов. Простой алгоритм посторения выглядит следующим образом: каждый

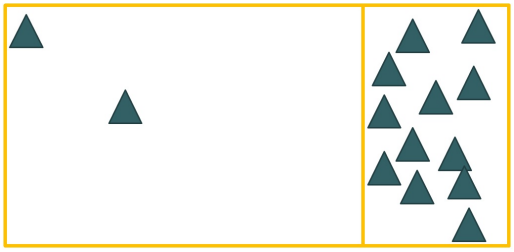


Рис. C.8. Разбиение с учетом оптимизации стоимости.

раз при выборе плоскости нужно перебрать всевозможные границы примитивов по трем измерениям, вычислить в них значение функции стоимости и найти минимум среди всех этих значений. Когда мы вычисляем SAH для каждой плоскости, то нам необходимо знать N(left) и N(right) - количества примитивов справа и слева от плоскости. Если вычислять N простым перебором, в итоге получится квадратичный по сложности алгоритм построения.

Действенным способом ускорения поиска минимума функции SAH является использование какого-либо метода минимизации одномерной функции с несколькими начальными приближениями. Например, метод золотого сечения. Это избавляет от необходимости полного квадратичного перебора. Также популярен метод называемый binning [70, 71], который ускоряет поиск за счет снижения точности поиска положения найденного минимума.

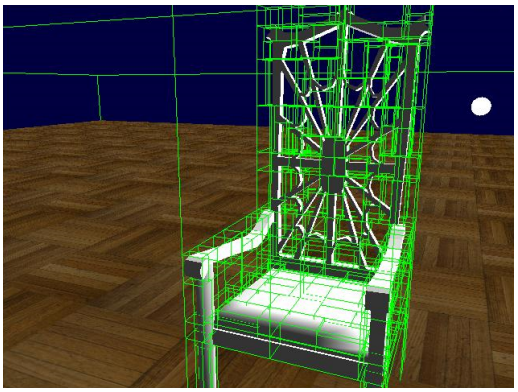


Рис. C.9. kd-дерево, построенное с учетом SAH.

C.3.4. Критерий остановки.

Одним из критериев остановки может служить сама функция SAH. Если оцениваемая функцией суммарная стоимость поиска в дочерних узлах больше

стоимости поиска в родительском узле, разбиение стоит остановить. Однако, весьма неочевидным фактом является то, что такой критерий остановки далеко не оптимален. Дело в том, что увеличение SAH при разбиении текущего уровня дерева еще не означает замедление трассировки для всего дерева. Рассмотрим цилиндр, составленный из полигонов на рисунке C.10. Узел kd-дерева, обозначенный буквой A является проблемным. Какая-бы плоскость разбиения не была выбрана, SAH полученных дочерних узлов будет больше чем SAH родительского узла. Таким образом, SAH в данном случае говорит о том, что подразбиение нужно остановить. Но это не всегда так.

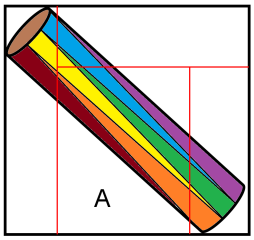


Рис. C.10. Цилиндр, составленный из треугольников. Какая бы плоскость разбиения не была выбирана для узла A, это не позволит уменьшить SAH.

Если представить, что цилиндр - высокополигональный объект, дальнейшее подразбиение (даже с учетом выбора плоскости по центру) приведет к тому, что в листьях окажется не более чем 1-2 примитива. Однако остается открытым вопрос - будет ли при этом быстрее трассировка лучей и если да, то на сколько. И очередной возникающий вопрос формулируется так: Если существуют такие плоскости разбиения, которые увеличиват суммарный SAH текущего узла (который может быть оценен как SAH(левого) + SAH(правого)), но при этом все-же позволяют ускорить трассировку, когда именно следует выбирать такие плоскости (т.е. на каком уровне дерева)?

Техника, позволяющая решить описанную выше проблему и ответить на поставленные вопросы называется в англоязычной литературе 'Early Split Clipping' [72]. Будем называть ее ранним подразбиение примитивов. Она говорит, что примитивы, которые плохо аппроксимируются при помощи AABB нужно подразбивать на несколько более мелких примитивов еще до построения дерева с тем, чтобы рассматривать такие 'сложные' плоскости на равне со всеми остальными в процессе оптимизации SAH. Подробнее данная техника будет рассмотрена при обсуждении BVH деревьев.

C.3.5. Поиск в kd-дереве.

Классический алгоритм бинарного поиска в kd деревьях (kd-tree traversal в англоязычной литературе), состоит в следующем: На первом шаге алгоритма

необходимо посчитать пересечение луча с ограничивающим сцену корневым параллелепипедом (AABB) и запомнить информацию о пересечении в виде двух координат (в пространстве луча) – t_near и t_far, обозначающих пересечение с ближней и дальней плоскостями соответственно. На каждом следующем шаге необходима информация только о текущем узле (его адрес) и этих двух координатах.

При этом нет необходимости вычислять пересечение луча и дочернего параллелепипеда, достаточно лишь узнать пересечение с разбивающей параллелепипед плоскостью (обозначим соответствующую координату как t_split).

В случае если t_split >= t_far (рис. C.11) или если t_split < t_near (рис. C.12) луч пересекает только один дочерний узел, поэтому можно просто отбросить правый (соответственно левый) узел и продолжить поиск пересечения в оставшемся узле.

В случае если луч пересекает оба дочерних узла (рис. C.13), необходимо сначала поискать пересечение в ближнем узле и если оно не найдено, искать его в дальнем. Так как в общем случае неизвестно, сколько раз произойдет последнее событие(отсутствие пересечения в ближнем узле), необходим стек. Каждый раз, когда луч пересекает оба дочерних узла, адрес дальнего узла, t_near и t_far помещаются в стек и поиск продолжается в ближнем. Если в ближнем узле пересечение не найдено, из стека достаются адрес дальнего узла, t_near, t_far и поиск продолжается в дальнем узле.

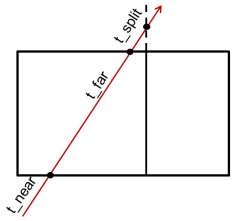


Рис. C.11. Поиск в kd-дереве. (t_split >= t_far).

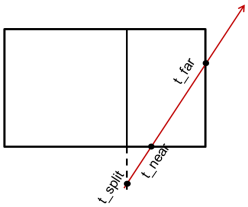


Рис. C.12. Поиск в kd-дереве. (t_split < t_near).

Почти все рассмотренные ранее проблемы регулярных сеток kd-дерево успешно решает. Оно обладает простым алгоритмом поиска, хорошей адаптивностью, быстро отбрасывая пустые пространства и значительно амортизирует

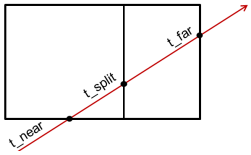


Рис. C.13. Поиск в kd-дереве. (t_near <= t_split <= t_far).

проблему повторных пересечений. Однако серьезный недостаток kd-дерева - сложный алгоритм построения. Дело здесь не только в необходимости вычисления функции стоимости SAH, но, что гораздо боле важно, перед началом построения kd-дерева невозможно сказать заранее, сколько памяти потребуется для его построения. Поскольку при разбиении узла все примитивы теоретически могут попасть как в левый, так и в правый дочерний узел, при разбиении узла с числом примитивов N необходимо иметь 2N свободной памяти.

C.4. BSP-деревья

В отличие от kd-деревьев, в которых плоскости разбиения имеют всего 3 возможных ориентации (XoY, XoZ, YoZ), в BSP (Binary Space Partion) деревьях плоскости разбиения могут быть ориентированны произвольным образом C.14. Это значительно усложняет выбор плоскости с учетом оптимизации SAH (и как следствие сам алгоритм построения), но частично решает проблему обозначенную на рисунке C.10. В работе [73] было показано преимущество BSP-деревьев в скорости поиска над kd-деревьями.

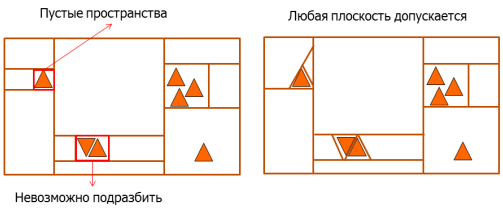


Рис. C.14. kd-дерево (слева); BSP-дерево (справа).

Тем не менее, даже BSP дерево не всегда гарантирует уменьшение числа примитивов в дочерних узлах (рис. C.14, узел с тремя треугольниками справа вверху). На практике из этого следует, что проблема с неизвестным количеством памяти, требующимся для построения дерева, все еще актуальна для BSP деревьев.

C.5. BVH-деревья

Этот вид ускоряющей структуры является наиболее практичным и часто используемым в современных трассировщиках лучей. BVH расшифровывается как Bounding Volume Hierarchy - Иерархия Ограничивающих Объемов. BVH-дерево состоит из вложенных друг в друга объемов, ограниченных некоторыми примитивами (рис. C.15). Такие деревья можно классифицировать по типу ограничивающего примитива.

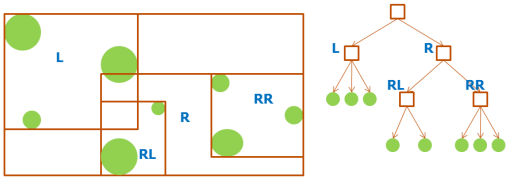


Рис. C.15. BVH-дерево. В отличие от других видов пространственных деревьев, узлы дерева могут пересекаться друг с другом.

В дальнейшем будем рассматривать только деревья, использующие прямугольные паралеллипипеды (Axis Aligned Bounding Box, AABB) в качестве ограничивающих примитивов и под BVH деревом понимать дерево состоящее из вложенных друг в друга прямоугольных паралеллипипедов, стороны которых выровнены по осям координат.

C.5.1. Две стратегии разбиения в BVH дереве

При построении дерева существует небольшая, но довольно существенная путаница в том, какие примитивы включать в дочерние узлы и как именно производить разбиение. Вопрос, который порождает путаницу звучит так: "Что делать, если примитив пересекает оба дочерних узла?". Существует 2 стратегии разбиения.

Первая стратегия называется разбиением по объектам (рис. C.16). При использовании этой стратегии необходимо найти некоторое разбиение нашего исходного множества примитивов на 2 подмножества. Затем достаточно лишь посчитать ограничивающий обьем для обоих подмножеств. В этом случае ответ на поставленные ранее вопрос звучит так: Такой примитив всегда идет в тот узел, в котором он содержится полностью. Такой узел всегда существует по построению указанного разбиения.

Вторая стратегия называется пространственным разбиением (рис. C.16). При использовании этой стратегии сначала выбирается некоторая плоскость, ко-

торая делит объем текущего узла на 2 части. Для полученных дочерних объемов ограничивающие AABB пересчитываются в соответствии с геометрией. В этом случае ответ на поставленный вопрос звучит так: "Такой примитив нужно добавлять в каждый из дочерних узлов, который он пересекает".

Рассмотренные стратегии порождают 2 принципиально разных вида BVH дерева. Стратегия разбиения по объектам позволяет строить такое дерево, в котором для каждого объекта хранится не более 1 ссылки. Будем называть такие дереьвя деревьями с одиночными ссылками (single reference trees).

В противоположность этому, стратегия пространственного разбиения может породить несколько ссылок на 1 объект. Будем называть такие деревья деревьями со множественными ссылками (multiple reference trees).

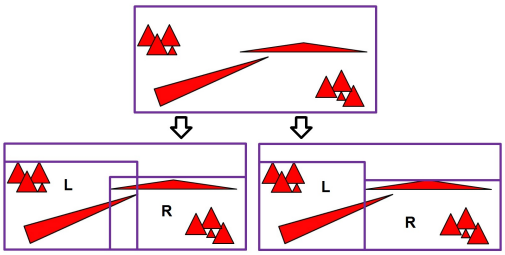


Рис. C.16. Разбиение по объектам (слева) и пространственное разбиение (справа). Тонкий длинный треугольник справа при стратегии разбиения по объектам попадет только в левый узел L. А при стратегии пространственного разбиения он попадет в оба дочерних узла L и R.

BVH деревья с одиночными ссылками позволяют значительно упростить алгоритм построения при относительно небольшой потери в эффективности поиска. В дальнейшем под BVH деревом будем понимать BVH дерево с одиночными ссылками, построенными при помощи стратегии разбиения по объектам.

C.5.2. Построение BVH дерева.

Алгоритм построения оптимального BVH дерева столь-же сложен, сколь и алгоритм построения оптимального kd-дерева. В теории даже сложнее. Когда мы строим kd-дерево, то на каждом шаге подразбиения минимизируем функцию стоимости SAH. В случае kd-дерева мы выбираем только плоскость разбиения и можем перебрать все возможные границы примитивов по трем измерениям (обычно SAH вычисляют на границах), чтобы найти такую плоскость, в которой функция SAH будет минимальна. То есть в худшем случае,

при отсутствие оптимизаций нам нужно вычислить SAH лишь 6*N раз (где N - число примитивов) и найти минимум.

В случае BVH всё сложнее, поскольку вариантов возможных разбиений не 6*N а $S(N,2) = 2^{N-1}-1$ (где S - число Стирлинга 2 рода). Перебрать их все не представляется возможным. В отличие от kd-дерева, BVH на каждом уровне вообще говоря может произвести разбиение на 2 совершенно произвольных подмножества примитивов. Всего таких подмножеств $2^{N-1} - 1$.

Однако можно использовать упрощение, не сильно ухудшающее качество дерева, но позволяющее значительно упростить процедуру его построения. Будем строить BVH дерево сверху вниз. При построении, в каждом узле дерева сортируем примитивы по минимумам (или центрам) их AABB и 3 осям координат. Таким образом, получаем 3 массива. Первый отсортирован по X координатам минимумов (или центров) AABB объектов, второй по их Y координатам и третий по их Z координатам. Затем для каждого из полученных массивов (длинной n) необходимо перебрать все разбиения на подмножества вида:

$$
\begin{aligned}
[0; 1..n] \\
[0..1; 2..n] \\
[0..3; 4..n] \\
... \\
[0..n-1; n]
\end{aligned}
$$

Для каждого из этих разбиений вычисляем значение SAH. Запоминая ограничивающие AABB в специальном массиве для получаемых подмножеств мы можем не пересчитывать эти AABB полностью, что сводит поиск минимума SAH всего к 2 проходам по массиву. С конца к началу и с начала к концу. Описанный выше подход используется в реализации трассировки лучей из работы [74].

Среди эффективных подходов, позволяющих значительно сократить время построения BVH дерева для анимированных сцен следует отметить подход из работы [75]. За счет кластеризации входных данных, в работе [75] достигнуто уменьшение объема обрабатываемых построителем примитивов и, как следствие, значительное ускорение построения BVH дерева.

C.5.3. Ранее подразбиение примитивов.

У AABB дерева (как и многих рассмотренных ранее структур пространственного поиска) есть один существенный недостаток: треугольники на самом деле плохо приближаются прямоугольными параллелипипедами. Даже те треугольники которые лежат в плоскости x,y или z треугольники обладают площадью поверхности, вдвое меньшей чем ограничивающих их бокс. Для остальных треугольников это соотношение еще больше. Рисунок C.17 демонстрирует.

Проблемма здесь в том, что через ограничивающий паралеллипипед треугольника проходит значительное количество лучей, которые на самом деле не пересекают этот треугольник. На рисунке C.17 черным цветом обозначены лучи которые не пересекли ни один треугольник (представьте эту картинку в 3D). Если в геометрии примутствует много длинных и тонких треугольников, получается BVH с огромным количеством самопересекающихся узлов, содержащих много пустого пространства. Чтобы побороть этот недостаток, можно отесселировать треугольник, подразбив его на более мелкие, но существует более элегеантное и экономное в плане памяти решение, называемое ранним подразбиенем примитивов ('Early Split Clipping' [72]). Ранее подразбиение, таким образом, изначально представляет треугольник не одним боксом а несколькими. То есть один длинный треугольник подается на вход BVH построителю в виде набора боксов, содержащих ссылки на этот треугольник (рисунок C.17, справа).

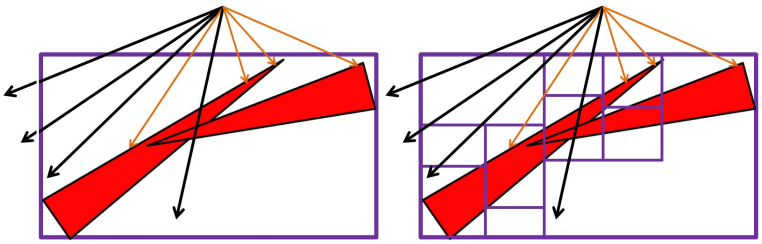


Рис. C.17. Обыкновенный узел BVH дерева (слева) и ранее подразбиение примитивов (справа).

C.6. Резюме по ускоряющим структурам

В сочетании с ранним подразбиением, BVH дерево на практике показало себя с наилучшей стороны [74]. Алгоритм построения BVH достаточно прост и занимает определенный объем памяти, пропорционально количеству входных примитивов.

Литература

[1] Большая Советская Энциклопедия. 1978.

[2] Wu Fukun, Zheng Changwen. Simulation of Wave Effects Based on Ray Tracing // 2013 International Conference on Computer-Aided Design and Computer Graphics. 2013. С. 9–15.

[3] Douglas V. Johnston Paul N. G. Tarjan. CS348b Final Project: Ray-tracing Interference and Diffraction. 2006.

[4] McCluney R. Introduction to Radiometry and Photometry. Artech House optoelectronics library. Artech House, 1994. URL: https://books.google.ru/books?id=3LJvQgAACAAJ.

[5] И.В. Валиев А.Г. Волобой Е.Ю. Денисов С.В. Ершов С.Г. Поздняков. Преобразование XYZ в спектр для задач моделирования распространения света // Математическое моделирование. 2016. Т. 28, № 9. С. 101–112.

[6] LH Hardy. Researches on normal and defective colour vision. // Archives of Ophthalmology. 1947. Т. 38, № 6. С. 850–851. URL: + http://dx.doi.org/10.1001/archopht.1947.00900010873016.

[7] Колесников Е.В. Абгарян К.К. SVD-разложение и его практические приложения: Master's thesis. 2015.

[8] N.J. M., NGAI P. The application of computer graphics in lighting design // Journal of the IES 14. 1984. oct. Vol. 14. P. 6–26.

[9] Shirley Peter, Marschner Steve. Fundamentals of Computer Graphics. 3rd изд. Natick, MA, USA: A. K. Peters, Ltd., 2009.

[10] J. T., H. R. Tone reproduction for computer generated images // IEEE Computer Graphics and Applications. 1993. nov. Vol. 13, no. 6. P. 42–48.

[11] Ward Greg. Graphics Gems IV / под ред. Paul S. Heckbert. San Diego, CA, USA: Academic Press Professional, Inc., 1994. С. 415–421. URL: http://dl.acm.org/citation.cfm?id=180895.180934.

[12] Ward Larson Gregory, Rushmeier Holly, Piatko Christine. A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes // ACM SIGGRAPH 97 Visual Proceedings: The Art and Interdisciplinary Programs of SIGGRAPH '97. SIGGRAPH '97. New York, NY, USA: ACM, 1997. С. 155–. URL: http://doi.acm.org/10.1145/259081.259242.

[13] Tumblin Jack, Hodgins Jessica K., Guenter Brian K. Two Methods for Display of High Contrast Images // ACM Trans. Graph. New York, NY, USA, 1999. jan. Т. 18, № 1. С. 56–94. URL: http://doi.acm.org/10.1145/300776.300783.

[14] Adaptive Logarithmic Mapping For Displaying High Contrast Scenes / F. Drago, K. Myszkowski, T. Annen [и др.] // Computer Graphics Forum. 2003.

[15] Kaplanyan Anton, Dachsbacher Carsten. Cascaded light propagation volumes for real-time indirect illumination // Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games. I3D '10. New York, NY, USA: ACM, 2010. С. 99–107. URL: http://doi.acm.org/10.1145/1730804.1730821.

[16] Interactive Indirect Illumination Using Voxel Cone Tracing / Cyril Crassin, Fabrice Neyret, Miguel Sainz [и др.] // Computer Graphics Forum (Proceedings of Pacific Graphics 2011). 2011. sep. Т. 30, № 7. URL: http://maverick.inria.fr/Publications/2011/CNSGE11b.

[17] of Engineering ANU College, Science Computer. Global Illumination Models. Physically Based Illumination, Ray Tracing and Radiosity. 2001. June. URL: http://escience.anu.edu.au/lecture/cg/GlobalIllumination/printCG.en.html.

[18] Shcherbakov A.S., Frolov V.A. Accelerating radiosity on GPUs // WSCG2017 25th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision.

[19] Pharr Matt, Humphreys Greg. Physically Based Rendering, Second Edition: From Theory To Implementation. 2nd изд. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2010.

[20] Whitted Turner. An improved illumination model for shaded display // Commun. ACM. New York, NY, USA, 1980. jun. Т. 23, № 6. С. 343–349. URL: http://doi.acm.org/10.1145/358876.358882.

[21] Wright Richard, Lipchak Benjamin, Haemel Nicholas. Opengl® Superbible: Comprehensive Tutorial and Reference, Fourth Edition. Fourth изд. Addison-Wesley Professional, 2007.

[22] Suffern Kevin. Ray Tracing from the Ground Up. 2007.

[23] Debelov V. A., Kozlov D. S. A Local Model of Light Interaction with Transparent Crystalline Media // IEEE Transactions on Visualization and Computer Graphics. Los Alamitos, CA, USA, 2013. Т. 19, № 8. С. 1274–1287.

[24] Nicodemus Fred E. Directional Reflectance and Emissivity of an Opaque Surface // Applied Optics. 2009. July. Т. 4, № 7. С. 767–773.

[25] Kajiya James T. The rendering equation // SIGGRAPH Comput. Graph. New York, NY, USA, 1986. aug. Т. 20, № 4. С. 143–150. URL: http://doi.acm.org/10.1145/15886.15902.

[26] Radiometry / F. E. Nicodemus, J. C. Richmond, J. J. Hsia [и др.] / под ред. Lawrence B. Wolff, Steven A. Shafer, Glenn Healey. USA: Jones and Bartlett Publishers, Inc., 1992. С. 94–145. URL: http://dl.acm.org/citation.cfm?id=136913.136929.

[27] Jensen Henrik Wann, Buhler Juan. A Rapid Hierarchical Rendering Technique for Translucent Materials // ACM Trans. Graph. New York, NY, USA, 2002. jul. Т. 21, № 3. С. 576–581. URL: http://doi.acm.org/10.1145/566654.566619.

[28] Соболь И.М. Численные методы Монте-Карло. 1 изд. М.: Nauka., 1973.

[29] Modeling the interaction of light between diffuse surfaces / Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg [и др.] // Proceedings of the 11th annual conference on Computer graphics and interactive techniques. SIGGRAPH '84. New York, NY, USA: ACM, 1984. С. 213–222. URL: http://doi.acm.org/10.1145/800031.808601.

[30] Lafortune Eric P., Willems Yves D. Using the Modified Phong Reflectance Model for Physically Based Rendering. 1994.

[31] Veach Eric. Robust monte carlo methods for light transport simulation. Ph.D. thesis. Stanford, CA, USA: Stanford University, 1998. AAI9837162.

[32] NVIDIA. NVIDIA OptiX ray tracing engine. 2014. URL: https://developer.nvidia.com/optix.

[33] Владимир Фролов. Образовательный проект AdaRT. Интегратор освещенности на основе трассировки путей на языке Ada. 2014. URL: http://ray-tracing.ru/articles229.html.

[34] Денис Боголепов. Методы глобального освещения для интерактивного синтеза изображений сложных сцен на графических процессорах. Ph.D. thesis. НГГУ им. Лобачевского, Нижний Новгород, Россия: Нижний Новгород, 2013.

[35] Szecsi Laszlo, Szirmay-Kalos Laszlo, Kelemen Csaba. Variance Reduction for Russian-roulette // WSCG. 2003.

[36] Heckbert Paul S. Adaptive radiosity textures for bidirectional ray tracing // SIGGRAPH Comput. Graph. New York, NY, USA, 1990. sep. Т. 24, № 4. С. 145–154. URL: http://doi.acm.org/10.1145/97880.97895.

[37] Bogolepov D.K., Turlapov V.E., D.Uljanov. Об одной реализации трассировки путей для графического процессора. НГГУ им. Лобачевского, Нижний Новгород, Россия: Н. Новгород: Изд-во Нижегор. гос. ун-та, 2013. С. 42–46.

[38] Georgiev Iliyan, Křivánek Jaroslav, Slusallek Philipp. Bidirectional light transport with vertex merging // SIGGRAPH Asia 2011 Sketches. SA '11. New York, NY, USA: ACM, 2011. С. 27:1–27:2. URL: http://doi.acm.org/10.1145/2077378.2077412.

[39] Iliyan Georgiev Jaroslav Křivánek Tomáš Davidovič, Slusallek Philipp. SmallVCM. A (not too) small physically based renderer. URL: http://www.smallvcm.com/.

[40] Veach Eric, Guibas Leonidas J. Metropolis light transport // Proceedings of the 24th annual conference on Computer graphics and interactive techniques. SIGGRAPH '97. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997. С. 65–76. URL: http://dx.doi.org/10.1145/258734.258775.

[41] Krauth Werner. Statistical Mechanics: Algorithms and Computations. 2015. Deceber. URL: https://class.coursera.org/smac-002.

[42] Frolov V. A., Galaktionov V. A. Memory-compact Metropolis light transport on GPUs // Programming and Computer Software. 2017. May. Т. 43, № 3. С. 196–203. URL: https://doi.org/10.1134/S0361768817030057.

[43] A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm / Csaba Kelemen, Laszlo Szirmay-Kalos, Gyorgy Antal [и др.] // Computer Graphics Forum. Т. 23. 2002. С. 531–540.

[44] Cline David, Egbert Parris. A Practical Introduction to Metropolis Light Transport: Tech. Rep.: : Brigham Young University, 2005.

[45] Jakob Wenzel, Marschner Steve. Manifold Exploration: A Markov Chain Monte Carlo Technique for Rendering Scenes with Difficult Specular Transport // ACM Trans. Graph. New York, NY, USA, 2012. jul. Т. 31, № 4. С. 58:1–58:13. URL: http://doi.acm.org/10.1145/2185520.2185554.

[46] Kaplanyan Anton S., Hanika Johannes, Dachsbacher Carsten. The Natural-constraint Representation of the Path Space for Efficient Light Transport Simulation // ACM Trans. Graph. New York, NY, USA, 2014. jul. Т. 33, № 4. С. 102:1–102:13. URL: http://doi.acm.org/10.1145/2601097.2601108.

[47] Anisotropic Gaussian Mutations for Metropolis Light Transport Through Hessian-Hamiltonian Dynamics / Tzu-Mao Li, Jaakko Lehtinen, Ravi Ramamoorthi [и др.] // ACM Trans. Graph. New York, NY, USA, 2015. oct. Т. 34, № 6. С. 209:1–209:13. URL: http://doi.acm.org/10.1145/2816795.2818084.

[48] Neal Radford M. MCMC Using Hamiltonian Dynamics // Handbook of Markov Chain Monte Carlo. 2010. Т. 54. С. 113–162.

[49] Hachisuka Toshiya, Kaplanyan Anton S., Dachsbacher Carsten. Multiplexed Metropolis Light Transport // ACM Trans. Graph. New York, NY, USA, 2014. jul. Т. 33, № 4. С. 100:1–100:10. URL: http://doi.acm.org/10.1145/2601097.2601138.

[50] Reversible Jump Metropolis Light Transport Using Inverse Mappings / Benedikt Bitterli, Wenzel Jakob, Jan Novák [и др.] // ACM Transactions on Graphics. 2017. October. Т. 37, № 1.

[51] Jensen Henrik Wann, Christensen Per. High quality rendering using ray tracing and photon mapping // ACM SIGGRAPH 2007 courses. SIGGRAPH '07. New York, NY, USA: ACM, 2007. URL: http://doi.acm.org/10.1145/1281500.1281593.

[52] Wald Ingo, Gunther Johannes, Slusallek Philipp. Balancing Considered Harmful – Faster Photon Mapping using the Voxel Volume Heuristic // Computer Graphics Forum. 2004. Т. 22, № 3. С. 595–603. Proceedings of Eurographics.

[53] Tabellion Eric, Lamorlette Arnauld. An approximate global illumination system for computer generated films // ACM SIGGRAPH 2004 Papers. SIGGRAPH '04. New York, NY, USA: ACM, 2004. С. 469–476. URL: http://doi.acm.org/10.1145/1186562.1015748.

[54] Востряков Константин. Глобальное освещение с помощью октанных текстур // Материалы 16-ой международной конференции Графикон'2006. GraphiCon'06. Новосибирск, Россия: Графикон, 2006.

[55] Hachisuka Toshiya, Ogaki Shinji, Jensen Henrik Wann. Progressive photon mapping // ACM Trans. Graph. New York, NY, USA, 2008. dec. Т. 27, № 5. С. 130:1–130:8. URL: http://doi.acm.org/10.1145/1409060.1409083.

[56] Hachisuka Toshiya, Jensen Henrik Wann. Stochastic progressive photon mapping // ACM Trans. Graph. New York, NY, USA, 2009. dec. Т. 28, № 5. С. 141:1–141:8. URL: http://doi.acm.org/10.1145/1618452.1618487.

[57] State of the art in photon density estimation / Toshiya Hachisuka, Wojciech Jarosz, Guillaume Bouchard [и др.] // ACM SIGGRAPH 2012 Courses. SIGGRAPH '12. New York, NY, USA: ACM, 2012. С. 6:1–6:469. URL: http://doi.acm.org/10.1145/2343483.2343489.

[58] Suykens Frank, Willems Yves D. Density Control for Photon Maps // Proceedings of the Eurographics Workshop on Rendering Techniques 2000. London, UK, UK: Springer-Verlag, 2000. С. 23–34. URL: http://dl.acm.org/citation.cfm?id=647652.732120.

[59] Photon differentials / Lars Schjøth, Jeppe Revall Frisvad, Kenny Erleben [и др.] // Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia. GRAPHITE '07. New York, NY, USA: ACM, 2007. С. 179–186. URL: http://doi.acm.org/10.1145/1321261.1321293.

[60] Жданов Д.Д. Ершов С.В. Волобой А.Г. Адаптивный выбор глубины трассировки обратного луча в методе двунаправленной стохастической трассировки лучей. // Юбилейная 25-я международная конференция Графикон 2015. Протвино: Институт физико-технической информатики., 2015. С. 44–49.

[61] Bidirectional photon mapping. Central European Seminar on Computer Graphics, 2011.

[62] Kaplanyan Anton S., Dachsbacher Carsten. Adaptive Progressive Photon Mapping // ACM Trans. Graph. New York, NY, USA, 2013. apr. Т. 32, № 2. С. 16:1–16:13. URL: http://doi.acm.org/10.1145/2451236.2451242.

[63] Ward Gregory J., Rubinstein Francis M., Clear Robert D. A ray tracing solution for diffuse interreflection // SIGGRAPH Comput. Graph. New York, NY, USA, 1988. jun. Т. 22, № 4. С. 85–92. URL: http://doi.acm.org/10.1145/378456.378490.

[64] Krivanek Jaroslav, Gautron Pascal. Practical Global Illumination with Irradiance Caching (Synthesis Lectures in Computer Graphics and Animation). Morgan and Claypool Publishers, 2009.

[65] Herzog Robert, Myszkowski Karol, Seidel Hans-Peter. Anisotropic Radiance-Cache Splatting for Efficiently Computing High-Quality Global Illumination with Lightcuts // Computer Graphics Forum (Proc. Eurographics) / под ред. Marc Stamminger, Philip Dutré. T. 28(2). München, Germany: Wiley-Blackwell, 2009. C. 259–268.

[66] Владимир Фролов. Методы решения проблемы глобальной освещённости на графических процессорах. Ph.D. thesis. Институт прикладной математики имени М.В. Келыша РАН, Россия: Москва, 2015. URL: http://keldysh.ru/council/1/2015-frolov/avtoref.pdf.

[67] Tabellion Eric. Irradiance Caching at DreamWorks // Practical Global Illumination with Irradiance Caching, SIGGRAPH 2008 Cource. 2008.

[68] Fujimoto A., Tanaka Takayuki, Iwata K. Tutorial: computer graphics; image synthesis / под ред. Kenneth I. Joy, Charles W. Grant, Nelson L. Max [и др.]. New York, NY, USA: Computer Science Press, Inc., 1988. C. 148–159. URL: http://dl.acm.org/citation.cfm?id=95075.95111.

[69] MacDonald David J., Booth Kellogg S. Heuristics for ray tracing using space subdivision // Vis. Comput. Secaucus, NJ, USA, 1990. may. T. 6, № 3. C. 153–166. URL: http://dx.doi.org/10.1007/BF01911006.

[70] Experiences with Streaming Construction of SAH KD-Trees / Stefan Popov, Johannes Günther, Hans-Peter Seidel [и др.] // Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing. 2006. sep. C. 89–94.

[71] Wald Ingo. On Fast Construction of SAH-based Bounding Volume Hierarchies // Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing. RT '07. Washington, DC, USA: IEEE Computer Society, 2007. C. 33–40. URL: http://dx.doi.org/10.1109/RT.2007.4342588.

[72] Ernst Manfred, Greiner Gunther. Early Split Clipping for Bounding Volume Hierarchies // Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing. RT '07. Washington, DC, USA: IEEE Computer Society, 2007. C. 73–78. URL: http://dx.doi.org/10.1109/RT.2007.4342593.

[73] Ize Thiago, Wald Ingo, Parker Steven G. Ray tracing with the BSP tree // Proceedings of the 2008 IEEE Symposium on Interactive Ray Tracing. 2008.

[74] Aila Timo, Laine Samuli. Understanding the efficiency of ray traversal on GPUs // Proceedings of the Conference on High Performance Graphics 2009. HPG '09. New York, NY, USA: ACM, 2009. C. 145–149. URL: http://doi.acm.org/10.1145/1572769.1572792.

[75] Garanzha Kirill. The Use of Precomputed Triangle Clusters for Accelerated Ray Tracing in Dynamic Scenes // Computer Graphics Forum. 2009. T. 28, № 4. C. 1199–1206.