

# Фотореалистичный рендеринг *(aka raytracing)*

Лекция 3: Радиометрия, распространение света,  
уравнение рендеринга, описание материалов,  
Монте-Карло интегрирование

---

2024

# Немного радиометрии

- Мы будем описывать связь между различными величинами, означающими *количество света*
- Полезно чётко разобраться, какие величины что означают
- Наука, занимающаяся этими величинами, называется **радиометрией (radiometry)**

## Энергия излучения (radiant energy)

- Центральный объект радиометрии – **энергия излучения** (**radiant energy**)  $Q$ : количество энергии, излучённое или отражённое объектом / прошедшее через среду / etc
- Единица измерения энергии – J (дюоули), как и в обычной физике

# Энергия излучения (radiant energy)

- Светящиеся объекты (напр. лампочка) излучают свет непрерывно, т.е. суммарная энергия их излучения постоянно растёт
- Другие объекты поглощают/отражают свет непрерывно, т.е. суммарная энергия поглощения/отражения тоже постоянно растёт
- Мы будем считать, что сцена статична и находится в эквилибриуме, т.е. яркость точек сцены не меняется со временем
- $\Rightarrow$  Для нас интереснее не общее количество энергии, а энергия за единицу времени

# Поток излучения (radiant flux)

- Поток излучения (radiant flux)  $\Phi$  – энергия излучения за единицу времени  $t$  (или производная энергии по времени)

$$\Phi = \frac{\partial Q}{\partial t}$$

- Например, поток, проходящий сквозь некоторую поверхность
- Единица измерения –  $J \cdot s^{-1} = W$  (дюоули в секунду = ватты)
- Примеры:
  - Поток излучения типичной светодиодной лампы  $\Phi = 10$  ватт
  - Поток излучения Солнца  $\Phi = 3.86 \times 10^{26}$  ватт

## Светимость / облучённость (radiant exitance / irradiance)

- Светимость (radiant exitance)  $M$  – поток излучения, излучённый единицей поверхности  $\sigma$  объекта

$$M = \frac{\partial \Phi}{\partial \sigma}$$

- Облучённость (irradiance)  $E$  – поток излучения, падающий на единицу поверхности  $\sigma$  объекта

$$E = \frac{\partial \Phi}{\partial \sigma}$$

- Единица измерения –  $\text{W} \cdot \text{m}^{-2}$  (ватты на квадратный метр)

## Светимость / облучённость (radiant exitance / irradiance)

- Например, если точечный источник излучает **поток излучения**  $\Phi$  ватт, и на расстоянии  $r$  от него находится поверхность, образующая угол  $\theta$  с направлением на источник, то **облучённость** этой поверхности составляет

$$\frac{\Phi \cos \theta}{4\pi r^2}$$

ватт на квадратный метр

## Сила излучения (radiant intensity)

- Сила излучения (radiant intensity)  $I$  – поток излучения, излучённый/принятый/отражённый поверхностью в некотором направлении  $\omega$

$$I = \frac{\partial \Phi}{\partial \omega}$$

- Единица измерения –  $\text{W} \cdot \text{sr}^{-1}$  (ватты на стерадиан)
- Например, если точечный источник излучает **поток излучения**  $\Phi$  ватт равномерно во всех направлениях, то его **сила излучения** равна  $\frac{\Phi}{4\pi}$  ватт на стерадиан

## Яркость (radiance)

- Яркость (radiance)  $L$  – поток излучения, излучённый/принятый/отражённый поверхностью в некотором направлении  $\omega$  на единицу площади  $\sigma \cos \theta$  проекции этой поверхности перпендикулярно направлению света

$$L = \frac{\partial^2 \Phi}{\partial \omega \partial \sigma \cos \theta}$$

- Единица измерения –  $\text{W} \cdot \text{m}^{-2} \cdot \text{sr}^{-1}$  (ватты на квадратный метр на стерadian)
- Самая важная для нас величина!

- Поток излучения (radiant flux)  $\Phi$  – световая энергия за единицу времени
- Светимость / облучённость (radiant exitance / irradiance)  $\Phi$  – на единицу поверхности (= в конкретной точке)
- Сила излучения (radiant intensity)  $\Phi$  – на единицу телесного угла (= в конкретном направлении)
- Яркость (radiance)  $\Phi$  – на единицу телесного угла на единицу поверхности (= в конкретном направлении в конкретной точке)

# Радиометрия

- В литературе по рендерингу часто встречаются радиометрические термины – полезно понимать, о чём идёт речь
- **N.B.:** В науке *фотометрии* есть абсолютно аналогичные термины, но не для энергии, а для воспринимаемой человеком яркости (усреднённой энергии света в видимом диапазоне)
- **N.B.:** Исторически, а также в других науках (напр. в астрономии) есть те же величины и те же названия, но соответствие величин и названий другое

## Радиометрия: ссылки

- [en.wikipedia.org/wiki/Radiant\\_energy](https://en.wikipedia.org/wiki/Radiant_energy)
- Слайды с объяснением радиометрических величин

# Что такое свет

- Свет – электромагнитная волна
- Пара векторных полей (электрическое  $\mathbf{E}$  и магнитное  $\mathbf{B}$ ), описываемых уравнениями Максвелла

$$\nabla \cdot \mathbf{E} = 4\pi\rho$$

$$\nabla \cdot \mathbf{B} = 0$$

$$\nabla \times \mathbf{E} = -\frac{1}{c} \frac{\partial \mathbf{B}}{\partial t}$$

$$\nabla \times \mathbf{B} = \frac{1}{c} \left( 4\pi \mathbf{J} + \frac{\partial \mathbf{E}}{\partial t} \right)$$

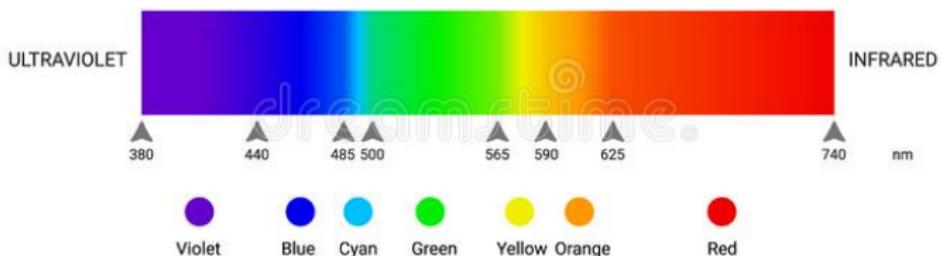
- Линейное волновое уравнение  $\Rightarrow$  раскладывается в сумму волн фиксированной длины волны  $\lambda$  (монохроматические волны)
- Много нетривиальных эффектов: интерференция, дифракция, поляризация, сложное взаимодействие с веществом

# Свет

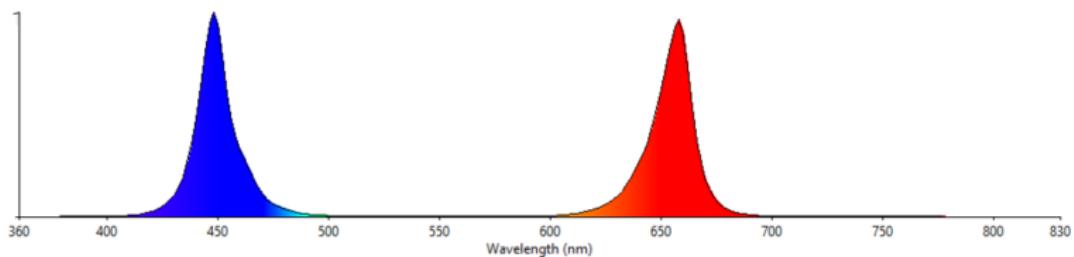
- В графике обычно достаточно геометрической оптики – предела при  $\lambda \rightarrow 0$
- Свет распространяется прямыми лучами (исключение – граница раздела сред или неоднородные среды)
- Свет распространяется бесконечно быстро ( $c \rightarrow \infty$ )
- Луч света разбивается в сумму монохроматических лучей, каждая имеет свою амплитуду (количество света)
- Интересуют только волны видимого спектра, который мы раскладываем в сумму **красного**, **зелёного** и **синего**

# Видимый спектр

## Visible spectrum



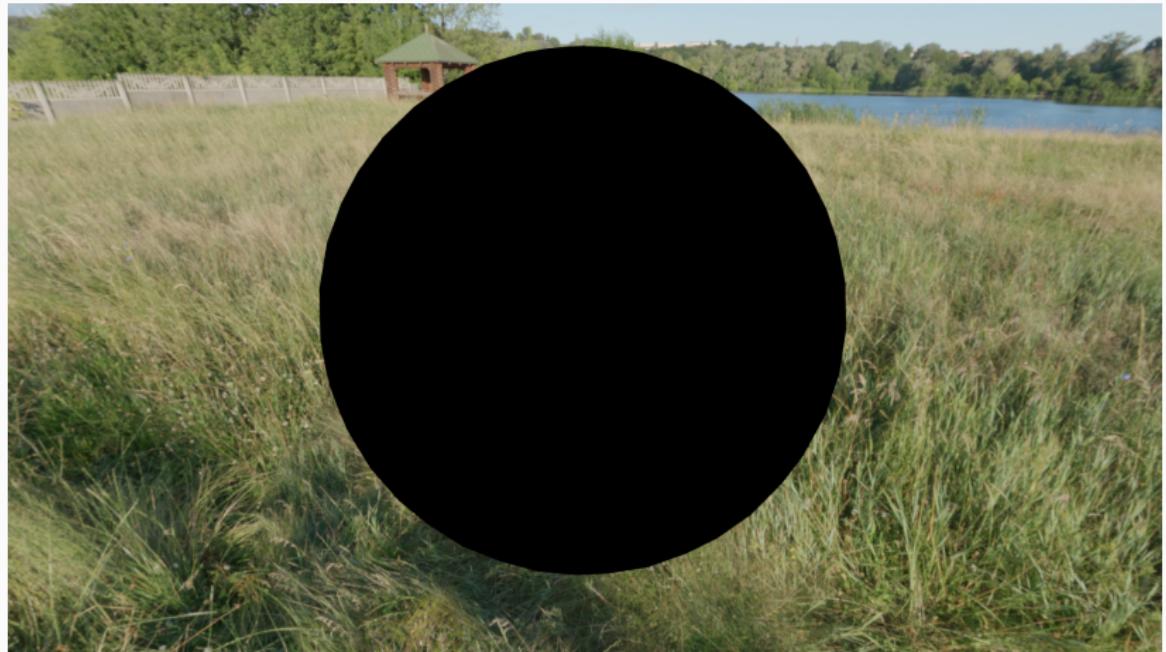
# Фиолетовый цвет



# Столкновение света с поверхностью

- Что происходит, когда луч света сталкивается с некой поверхностью?
- Зависит от её материала:
  - Абсолютно чёрное тело: поглощает весь свет
  - Зеркало: отражает свет в строго определённом направлении
  - Старое, плохо отполированное зеркало / лакированная поверхность: отражает свет примерно в определённом направлении
  - Диффузная (ламбертова) поверхность: отражает свет во все стороны равномерно
  - Диэлектрики (вода, стекло): частично отражает, частично преломляет (закон Френеля)
  - Воск: частично отражает, частично преломляет, но свет заходит неглубоко (subsurface scattering)

# Абсолютно чёрное тело



# Столкновение света с поверхностью

- Что происходит, когда луч света сталкивается с некой поверхностью?
- Зависит от её материала:
  - Абсолютно чёрное тело: поглощает весь свет
  - Зеркало: отражает свет в строго определённом направлении
  - Старое, плохо отполированное зеркало / лакированная поверхность: отражает свет примерно в определённом направлении
  - Диффузная (ламбертова) поверхность: отражает свет во все стороны равномерно
  - Диэлектрики (вода, стекло): частично отражает, частично преломляет (закон Френеля)
  - Воск: частично отражает, частично преломляет, но свет заходит неглубоко (subsurface scattering)

# Зеркало



# Столкновение света с поверхностью

- Что происходит, когда луч света сталкивается с некой поверхностью?
- Зависит от её материала:
  - Абсолютно чёрное тело: поглощает весь свет
  - Зеркало: отражает свет в строго определённом направлении
  - Старое, плохо отполированное зеркало / лакированная поверхность: отражает свет примерно в определённом направлении
  - Диффузная (ламбертова) поверхность: отражает свет во все стороны равномерно
  - Диэлектрики (вода, стекло): частично отражает, частично преломляет (закон Френеля)
  - Воск: частично отражает, частично преломляет, но свет заходит неглубоко (subsurface scattering)

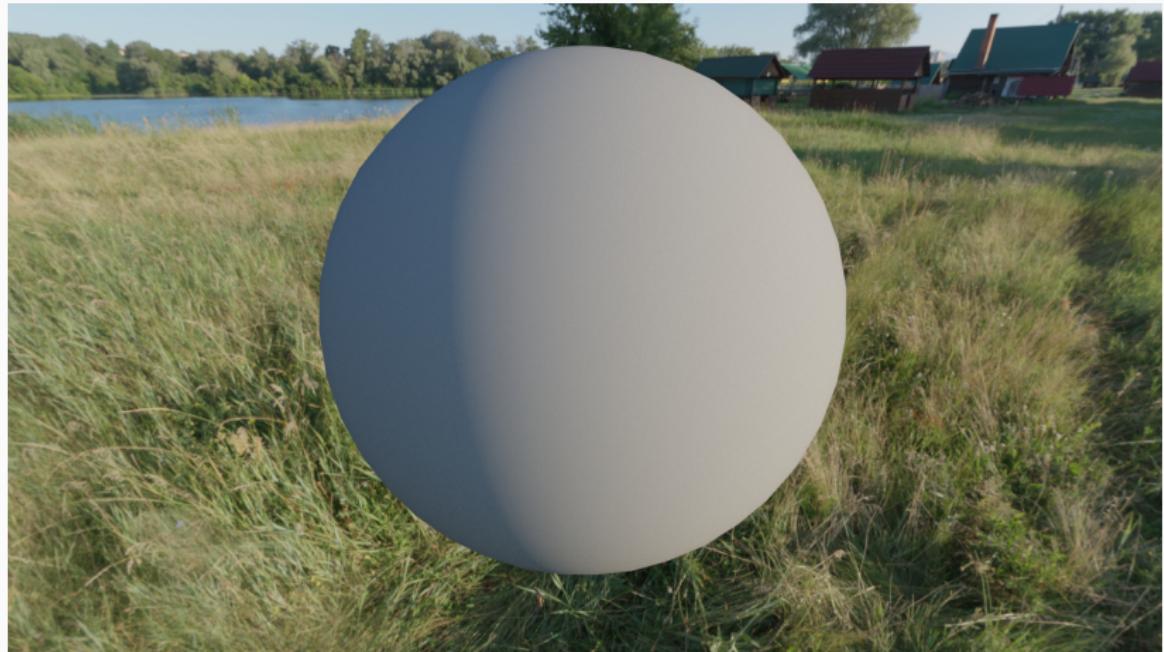
# Старое зеркало



# Столкновение света с поверхностью

- Что происходит, когда луч света сталкивается с некой поверхностью?
- Зависит от её материала:
  - Абсолютно чёрное тело: поглощает весь свет
  - Зеркало: отражает свет в строго определённом направлении
  - Старое, плохо отполированное зеркало / лакированная поверхность: отражает свет примерно в определённом направлении
  - Диффузная (ламбертова) поверхность: отражает свет во все стороны равномерно
  - Диэлектрики (вода, стекло): частично отражает, частично преломляет (закон Френеля)
  - Воск: частично отражает, частично преломляет, но свет заходит неглубоко (subsurface scattering)

# Диффузная поверхность



# Столкновение света с поверхностью

- Что происходит, когда луч света сталкивается с некой поверхностью?
- Зависит от её материала:
  - Абсолютно чёрное тело: поглощает весь свет
  - Зеркало: отражает свет в строго определённом направлении
  - Старое, плохо отполированное зеркало / лакированная поверхность: отражает свет примерно в определённом направлении
  - Диффузная (ламбертова) поверхность: отражает свет во все стороны равномерно
  - Диэлектрики (вода, стекло): частично отражает, частично преломляет (закон Френеля)
  - Воск: частично отражает, частично преломляет, но свет заходит неглубоко (subsurface scattering)

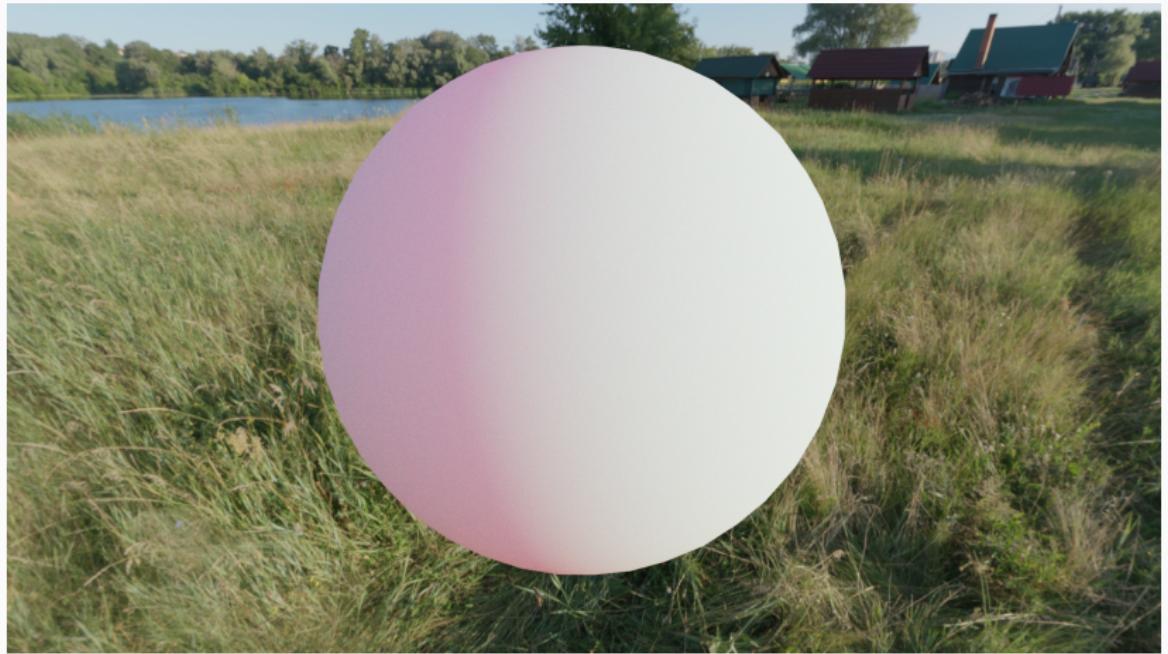
Стекло



# Столкновение света с поверхностью

- Что происходит, когда луч света сталкивается с некой поверхностью?
- Зависит от её материала:
  - Абсолютно чёрное тело: поглощает весь свет
  - Зеркало: отражает свет в строго определённом направлении
  - Старое, плохо отполированное зеркало / лакированная поверхность: отражает свет примерно в определённом направлении
  - Диффузная (ламбертова) поверхность: отражает свет во все стороны равномерно
  - Диэлектрики (вода, стекло): частично отражает, частично преломляет (закон Френеля)
  - Воск: частично отражает, частично преломляет, но свет заходит неглубоко (subsurface scattering)

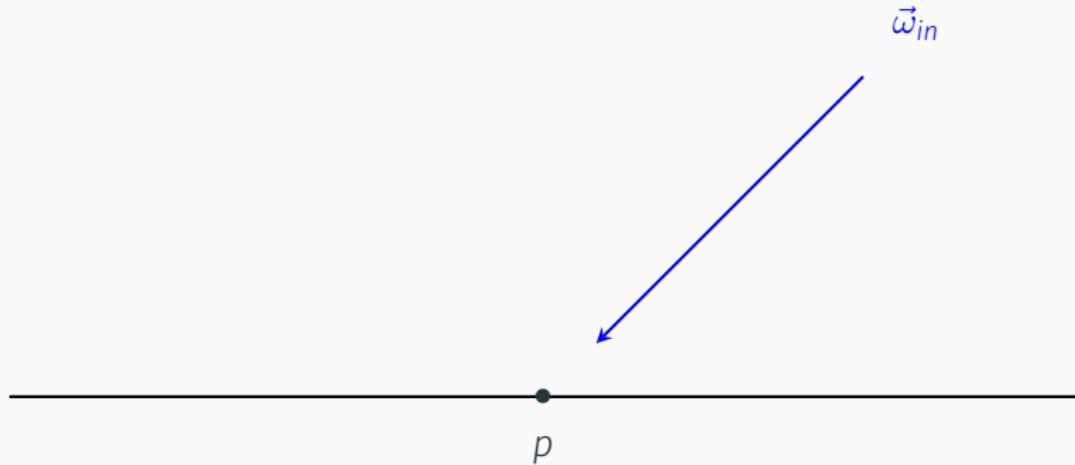
Bock



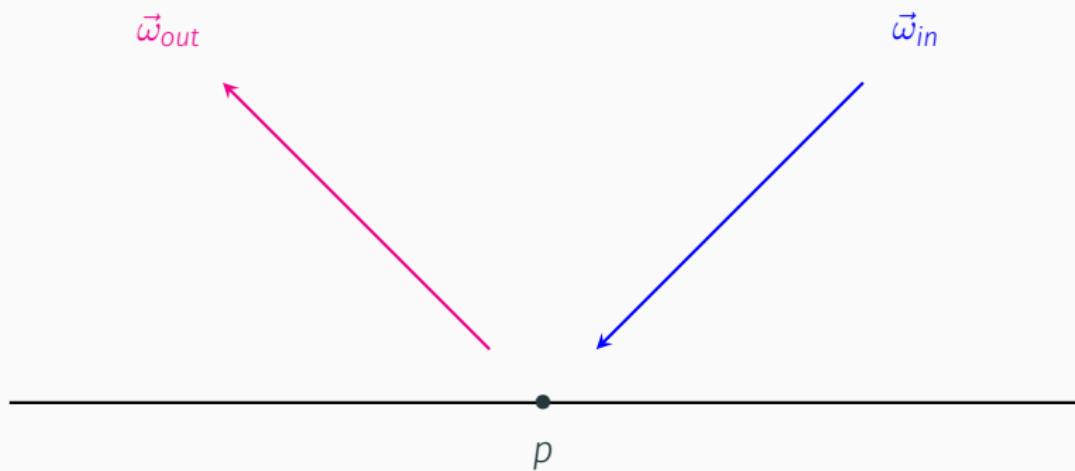
# Столкновение света с поверхностью

- Что происходит, когда луч света сталкивается с некой поверхностью?
- $\Rightarrow$  Свет, выходящий в каком-то направлении, складывается из света, пришедшего из *всех направлений*

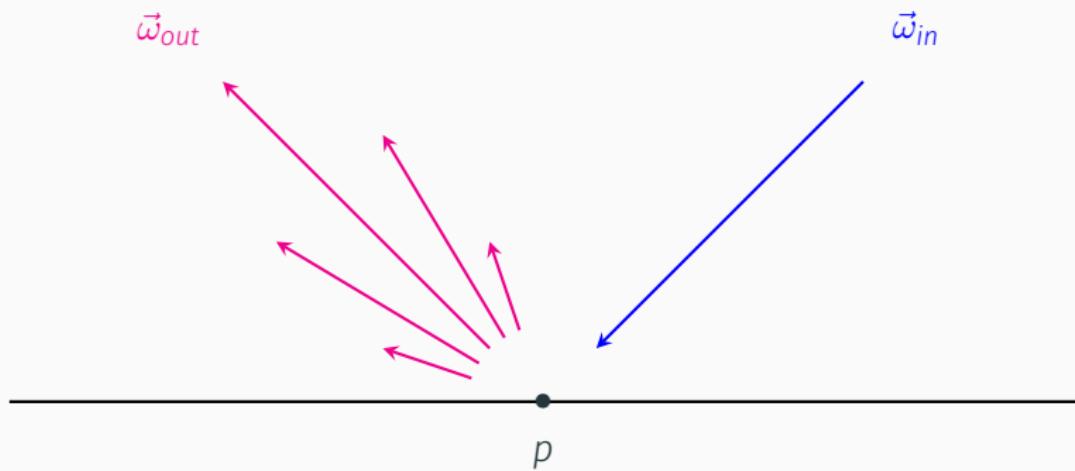
# Абсолютно чёрное тело



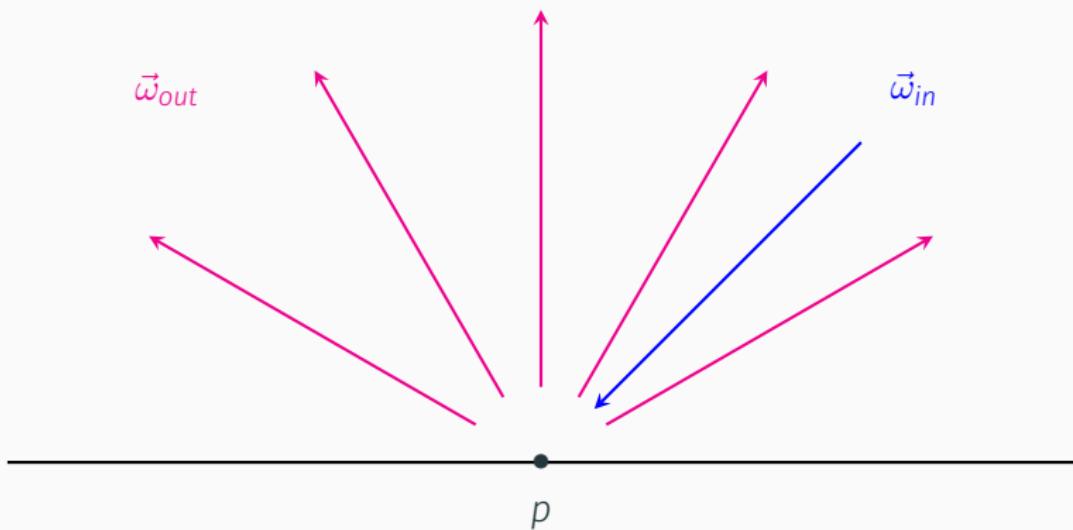
# Зеркало



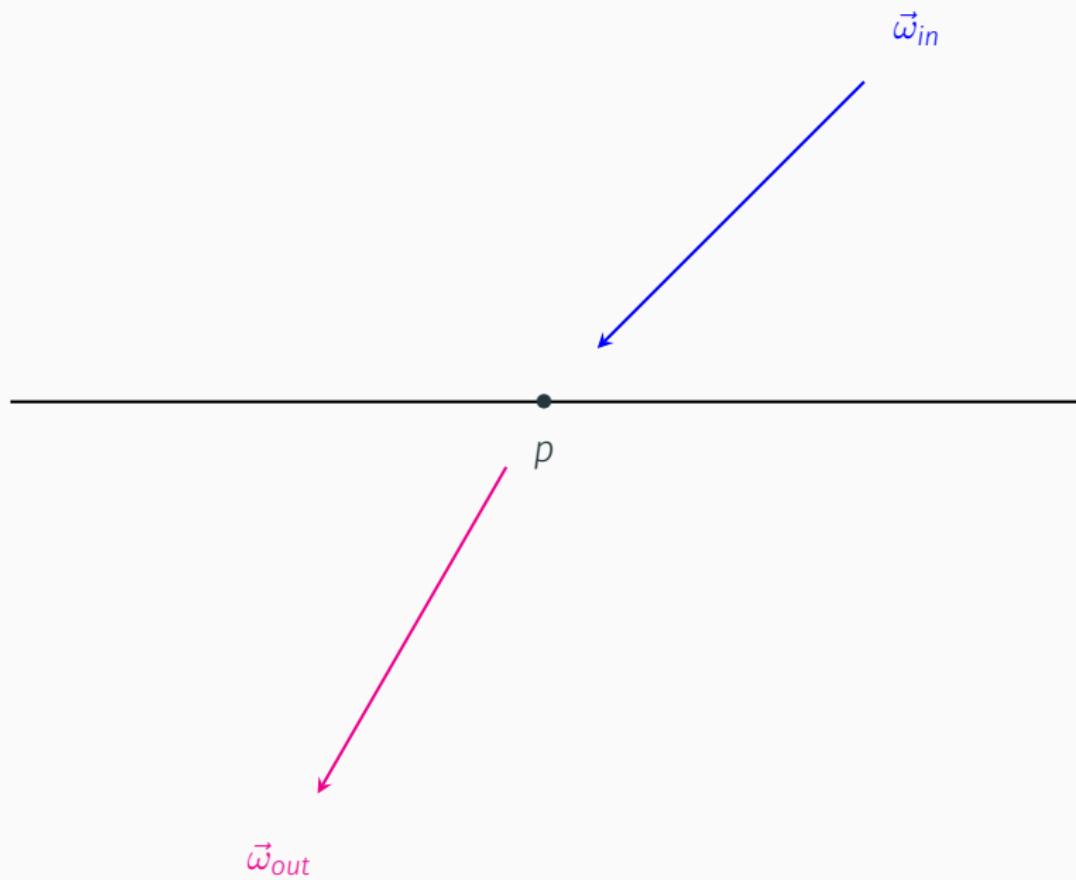
## Старое зеркало / лакированная поверхность



# Диффузная поверхность



# Стекло



## Уравнение рендеринга (Kajiya, 1986)

$$L_{out}(p, \vec{\omega}_{out}, \lambda) = L_e(p, \vec{\omega}_{out}, \lambda) + \int_{\mathbb{S}^2} L_{in}(p, \vec{\omega}_{in}, \lambda) \cdot f(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) \cdot (\vec{\omega}_{in} \cdot \vec{n}) d\vec{\omega}_{in}$$

- $L_{out}$  – яркость (radiance) света, выходящего из точки  $p$  в направлении  $\vec{\omega}_{out}$  с длиной волны  $\lambda$
- $L_e$  – яркость (radiance) света, излучаемого поверхностью из точки  $p$  в направлении  $\vec{\omega}_{out}$  с длиной волны  $\lambda$
- $L_{in}$  – яркость (radiance) света, приходящего в точку  $p$  из направления  $\vec{\omega}_{in}$  с длиной волны  $\lambda$
- $f$  – функция, определяющая, сколько света с длиной волны  $\lambda$ , пришедшего из направления  $\vec{\omega}_{in}$ , отразится в направлении  $\vec{\omega}_{out}$
- $\vec{n}$  – нормаль (перпендикуляр) к поверхности в точке  $p$

# Материал

- $f$  определяет материал объекта
- Абсолютно чёрное тело:  $f(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) = 0$
- Идеальное зеркало:  $f(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) = \delta(R_{\vec{n}}(\vec{\omega}_{in}) - \vec{\omega}_{out})$ 
  - $R$  – отраженный вектор:  $R_{\vec{n}}(\vec{\omega}) = 2\vec{n} \cdot (\vec{n} \cdot \vec{\omega}) - \vec{\omega}$
- Старое зеркало:  $f(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) = (R_{\vec{n}}(\vec{\omega}_{in}) \cdot \vec{\omega}_{out})^7$
- Диффузная поверхность:  $f(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) = 1$

# Цвет

- Зависимость  $f$  от  $\lambda$  обеспечивает цвет объектов
- Мы видим свет, отражённый объектом
- Объект синего цвета не отражает красный цвет (кажется чёрным при освещении красным светом)

## BRDF, BTDF, BSDF

- Если  $f$  только отражает свет, её называют **BRDF**: Bidirectional Reflectance Distribution Function
- Если  $f$  только преломляет свет, её называют **BTDF**: Bidirectional Transmittance Distribution Function
- В общем случае её называют **BSDF**: Bidirectional Scattering Distribution Function
- **BxDF** – обозначение для любого из вариантов выше

# BSDF

- BSDF неотрицательна:  $f(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) \geq 0$
- Обычно BSDF предполагается нормированной: тело не может отразить больше света, чем пришло

$$\int_{\mathbb{S}^2} f(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) (\vec{\omega}_{in} \cdot \vec{n}) d\vec{\omega}_{in} \leq 1$$

- Иногда нормированность нарушается – например, если BSDF имеет слишком сложную формулу, или задана неявно

- Helmholtz reciprocity:

$$\textcolor{green}{f}(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) = \textcolor{green}{f}(p, \vec{\omega}_{out}, \vec{\omega}_{in}, \lambda)$$

- Другими словами, если свет прошёл по какому-то пути в одну сторону (в т.ч. отразился от нескольких поверхностей), то он мог пройти и в обратную сторону вдоль такого же пути с такими же коэффициентами отражения
- Мы уже неявно используем этот принцип, пуская лучи в сторону, из которой мог прийти свет (вместо того, чтобы симулировать прямое распространение света)

# BSDF

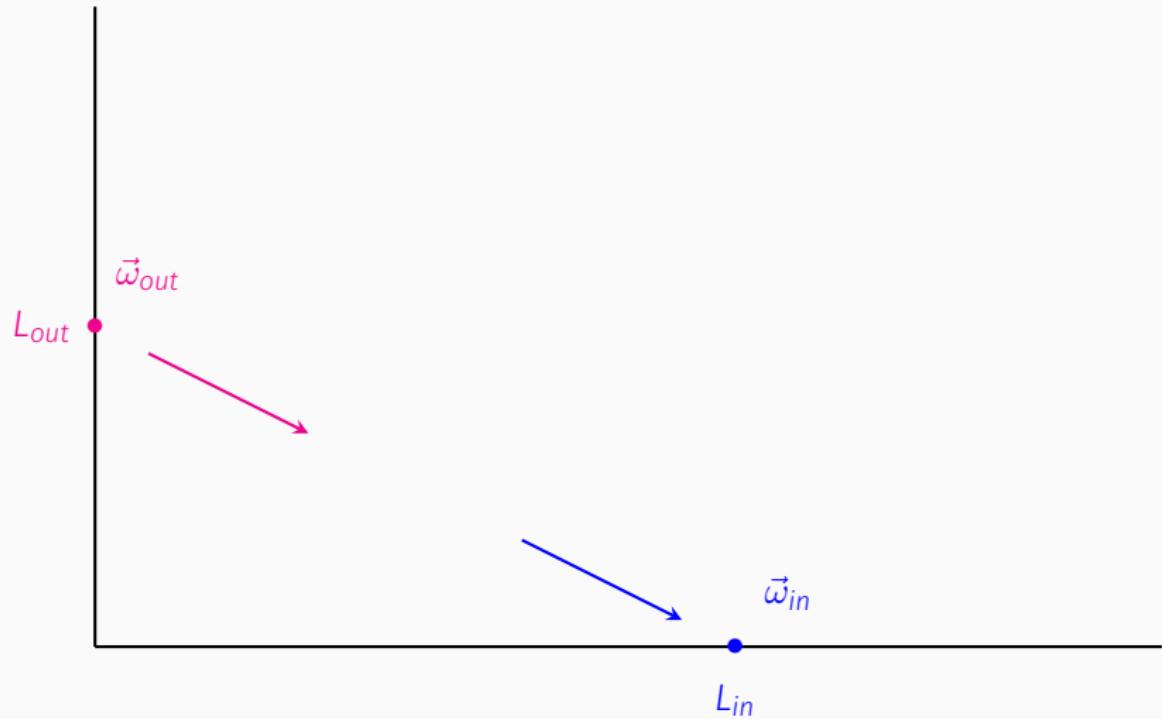
- Комбинация набора BSDF  $\{f_i\}$  – тоже BSDF:

$$\mu_i \geq 0, \sum \mu_i \leq 1 \Rightarrow \sum \mu_i f_i$$

- Часто материалы описываются, как комбинация более простых BSDF
- Некоторые BSDF невозможно описать функцией – например, идеальное зеркало или идеальное стекло, – в этом случае формально они описываются *обобщёнными функциями* в духе дельта-функции Дирака  $\delta(\omega)$

# Уравнение рендеринга

- Откуда взять  $L_{in}$ ? Ответ: это чей-то  $L_{out}$



# Уравнение рендеринга

- Интегральное уравнение для каждой точки каждой поверхности сцены
- Включает зависимость от материала объектов в каждой точке
- Геометрия сцены связывает уравнения для разных точек
- Обычно вместо всех возможных значений  $\lambda$  берут дискретный набор (**красный**, **зелёный**, **синий**)
- **N.B.:** *Spectral rendering* – область, в которой уравнение решают для произвольных распределений яркости по длинам волн
- В общем случае задача решения этого уравнения называется *Global Illumination* (GI)

# Как решать уравнение рендеринга?

- Real-time, low-quality подход: тонна дешёвых аппроксимаций
- Offline, high-quality подход: придётся что-то дискретизировать
  - Дискретизация пространства: алгоритм *radiosity*
  - Дискретизация домена интегрирования (сфера/полусфера) + трассировка лучей

## Алгоритм radiosity

- Взять в качестве переменных среднюю освещённость в каждой вершине сцены, интерполировать освещённость в треугольниках, свести к системе линейных уравнений
- Требует вычислить влияние каждого треугольника на каждый с учётом видимости (сложная геометрическая задача)
- Обычно решается итеративными методами (Якоби, Гаусса-Зейделя)
- Каждая итерация учитывает ещё одно возможное отражение света (bounce) перед попаданием в камеру
- Плохо подходит для динамических сцен, но позволяет предподсчитать GI для статической сцены
- Дискретизация пространства не позволяет получить картинку максимального качества

## Raytracing + дискретизация интеграла

- Пошлём луч из камеры в сторону пикселя, симулируя обратное распространение света (*raytracing*)
- В точке пересечения луча с объектом сцены будем аппроксимировать интеграл, рекурсивно пуская отражённые лучи в некотором наборе из  $N$  направлений
- Потенциально бесконечная рекурсия  $\implies$  ограничим количество отражений максимальной глубиной рекурсии  $D$
- Можем получать всё более и более качественные картинки, увеличивая  $N$  и  $D$

- Сложность алгоритма для вычисления одного пикселя:  $N^D$
- $\Rightarrow$  Увеличение  $N$  или  $D$  очень сильно увеличивает время выполнения
- Для интегрирования лучше подходит набор из равномерно распределённых направлений
- $\Rightarrow$  Нельзя остановить алгоритм посередине и ожидать разумный результат
- $\Rightarrow$  Нельзя продолжить алгоритм после его завершения (посчитать ещё чуть-чуть) – для каждого  $N$  нужно заново подбирать набор направлений
- **N.B.:** Последнее можно обойти с помощью *blue noise sequences*

# Raytracing + Монте-Карло интегрирование

- Вместо вычисления интеграла в наборе фиксированных направлений, давайте вычислять его в случайных направлениях
- На каждый пиксель посыпаем  $N$  лучей
- При пересечении луча со сценой посыпаем один отражённый луч в случайном направлении
- Ограничиваем рекурсию максимальной глубиной  $D$

## Raytracing + Монте-Карло интегрирование: плюсы

- Сложность вычисления цвета пикселя –  $N \cdot D$  – линейная по  $N$  и по  $D$
- $\Rightarrow$  Понятная и предсказуемая скорость выполнения
- Можно послать часть лучей, посмотреть на результат, при необходимости послать ещё лучей
- $\Rightarrow$  Удобно оценивать на предварительный результат
- $\Rightarrow$  Можно добавить ещё лучей при неудовлетворительном результате
- Основной алгоритм, использующийся сегодня для фотorealистичной графики

# Монте-Карло интегрирование

- Хотим вычислить интеграл

$$\int_a^b f(x)dx$$

- Сгенерируем набор из  $N$  случайных равномерно распределённых точек  $X_i \sim U(a, b)$  на отрезке  $[a, b]$
- Аппроксимируем интеграл как

$$\frac{b - a}{N} \sum f(X_i)$$

# Монте-Карло: происхождение названия

- Сам метод (и его более крутую вариацию – Markov chain Monte Carlo) разработали Джон фон Нейман и Станислав Улам в рамках проекта Манхэттен
- Для метода, как и для любой секретной разработки, нужно было кодовое имя
- Николас Метрополис (алгоритм Метрополиса-Гастингса) предложил назвать метод в честь казино Монте Карло в Монако, где часто растрачивал деньги дядя Станислава Улама

## Монте-Карло интегрирование: мат.ожидание

- Почему этот метод работает? Вычислим мат.ожидание аппроксимирующей величины

$$\begin{aligned}\mathbb{E} \left[ \frac{b-a}{N} \sum f(X_i) \right] &= \frac{b-a}{N} \sum \mathbb{E}[f(X_i)] = \\ &= (b-a)\mathbb{E}[f(X_1)] = (b-a) \int_a^b f(x)p(x)dx = \\ &= (b-a) \int_a^b f(x) \frac{1}{b-a} dx = \int_a^b f(x)dx\end{aligned}$$

- ⇒ В среднем мы получим искомую величину

## Монте-Карло интегрирование: дисперсия

- Почему метод сходится к искомой величине при  $N \rightarrow \infty$ ?  
Вычислим дисперсию аппроксимирующей величины

$$\begin{aligned}\text{Var} \left[ \frac{b-a}{N} \sum f(X_i) \right] &= \left( \frac{b-a}{N} \right)^2 \sum \text{Var}[f(X_i)] = \\ &= \left( \frac{b-a}{N} \right)^2 N \text{Var}[f(X_1)] = \frac{(b-a)^2}{N} \sigma^2\end{aligned}$$

- Здесь,  $\sigma^2 = \text{Var}[f(X_1)]$  не зависит от  $N$
- Среднеквадратичное отклонение:  $\frac{b-a}{\sqrt{N}} \sigma$
- $\implies$  Ошибка метода уменьшается как  $\frac{1}{\sqrt{N}}$

# Произвольное распределение

- Вместо равномерного распределения  $U(a, b)$  можно взять любое *абсолютно непрерывное* распределение  $p(x)$  на отрезке  $[a, b]$ , такое что  $f(x) \neq 0 \Rightarrow p(x) \neq 0$
- Тогда интеграл аппроксимируется как

$$\frac{1}{N} \sum \frac{f(X_i)}{p(X_i)}$$

- Доказательство сходимости почти не меняется
- Зачем это делать, обсудим на следующей лекции

## Дискретное распределение

- Дискретные распределения **не подходят** для Монте-Карло интегрирования: они дадут только набор значений в фиксированных точках, и не будут сходиться к искомому интегралу
- Даже смесь (аффинная комбинация) дискретного и непрерывного распределений тоже **не подходит**
- Тем не менее, иногда мы будем их использовать, когда наша BRDF сама является обобщённой функцией (например, для идеальных отражений)

## Произвольное множество

- Конечно, вместо отрезка  $[a, b]$  можно интегрировать по любому ограниченному множеству/многообразию, любой размерности
- Главное – уметь генерировать семплы  $X_i$  какого-то (лучше – равномерного) распределения вероятности  $p(x)$  на этом множестве
- Формула при этом вообще не меняется:

$$\frac{1}{N} \sum \frac{f(X_i)}{p(X_i)}$$

- (размер множества учитывается автоматически в  $p(x)$ )

# Random number generators (RNG)

- Обычно в основе генерации случайных значений лежит генератор случайных битов (в C++ это концепт `std::uniform_random_bit_generator`)
- Некоторые такие генераторы (напр. `std::mersenne_twister_engine` и `std::subtract_with_carry_engine` в C++) заточены под максимизацию случайности, и потому имеют большое состояние и сложные функции генерации (размер состояния `std::mt19937` составляет 5 килобайт)
- Некоторые, наоборот, имеют очень маленькое состояние и быструю генерацию, но менее случайный результат (например, `std::linear_congruential_engine` в C++)

## Random number generators (RNG)

- Для Монте-Карло интегрирования нам гораздо важнее скорость, а слишком большая случайность, наоборот, увеличивает variance (и ошибку интегрирования)
- Советую в C++ взять `std::minstd_rand`, или самим реализовать PCG генератор отсюда: [pcg-random.org/download.html](http://pcg-random.org/download.html) – его код умещается в 6 строк

## Генерация случайных семплов

- Будем считать, что мы умеем генерировать сэмплы  $U(0, 1)$  (равномерное) и  $\mathcal{N}(0, 1)$  (нормальное)
- Как сгенерировать произвольное распределение на произвольном множестве?
- Несколько способов:
  - Inversion sampling (CDF inversion)
  - Rejection sampling
  - Применение какого-нибудь преобразования

# Inversion sampling

- Хотим научиться генерировать семплы одномерной случайной величины  $Y$ , заданной функцией распределения  $F_Y(y)$
- Сгенерируем  $X \sim U(0, 1)$  и положим  $Y = F_Y^{-1}(X)$
- Доказательство того, что это работает:

$$P(y < Y) = P(y < F_Y^{-1}(X)) = P(F_Y(y) < X) = F_Y(y)$$

- Подходит, если  $F_Y^{-1}$  легко вычислить

## Inversion sampling: пример

- Равномерное распределение на  $[a, b]$ :

$$p_Y(y) = \frac{1}{b - a}$$

$$F_Y(y) = \frac{y - a}{b - a}$$

$$F_Y^{-1}(x) = a + (b - a)x$$

- Показательное распределение:

$$p_Y(y) = \lambda \exp(-\lambda y)$$

$$F_Y(y) = 1 - \exp(-\lambda y)$$

$$F_Y^{-1}(x) = -\frac{1}{\lambda} \ln(1 - x)$$

# Rejection sampling

- Хотим научиться генерировать семплы равномерно в некотором сложно устроенным множестве  $A \subset \mathbb{R}^n$
- Пусть, мы умеем проверять, что точка принадлежит множеству  $x \in A$
- Пусть, мы умеем генерировать семплы равномерно для некоторого объемлющего множества  $B \supset A$
- Алгоритм:
  - 1. Генерируем точку  $X \sim U(B)$
  - 2. Если  $X \in A$ , возвращаем  $X$ , иначе возвращаемся к пункту 1
- Точка из  $U(B)$  принадлежит  $A$  с вероятностью  $\frac{|A|}{|B|}$
- $\implies$  В среднем нам понадобится  $\frac{|B|}{|A|}$  итераций цикла

## Rejection sampling: пример

- Хотим сгенерировать точку внутри единичного круга  $B^2 \subset \mathbb{R}^2$
- Будем генерировать точку внутри квадрата  $(X, Y) \in [-1, 1]^2$ , каждая координата – равномерное на  $[-1, 1]$  распределение
- Точка лежит в круге, если  $X^2 + Y^2 \leq 1$
- Вероятность попасть в круг:  $\frac{|B^2|}{|[-1,1]^2|} = \frac{\pi}{4} \approx 0.78$
- Среднее число попыток:  $\frac{4}{\pi} \approx 1.27$
- **N.B.:** Метод работает и для  $B^n \subset \mathbb{R}^n$ , но вероятность попасть в шар  $\frac{\pi^{n/2}}{2^n \Gamma(\frac{n}{2} + 1)}$  стремится к нулю (и число попыток – к бесконечности)
- **N.B.:** Для  $B^3$  вероятность будет  $\frac{\pi}{6} \approx 0.52$

# Преобразование распределения

- Пусть у нас есть некоторая одномерная случайная величина  $X$  с распределением вероятности  $p_X(x)$
- Применим к нему некоторую биективную функцию  $G$  и получим новую величину  $Y = G(X)$
- Тогда её распределение вероятности равно

$$p_Y(y) = p_X(G^{-1}(y)) \left| \frac{d}{dy} G^{-1}(y) \right|$$

- **N.B.:** Модуль нужен, чтобы учесть убывающие функции
- **N.B.:** Если функция не биективна, нужно просуммировать по всем точкам прообраза  $G^{-1}(\{y\})$

## Преобразование распределения: пример

- Возьмём  $X \sim U(0, 1)$  и  $G(x) = x^2$
- Вычислим

$$G^{-1}(y) = \sqrt{y}$$

- Вычислим

$$\frac{d}{dy} G^{-1}(y) = \frac{1}{2\sqrt{y}}$$

- В итоге

$$p_Y(y) = p_X(\sqrt{y}) \frac{1}{2\sqrt{y}} = \frac{1}{2\sqrt{y}}$$

## Преобразование распределения: многомерный случай

- Пусть у нас есть некоторая случайная величина  $X$  заданная на подмножестве  $\mathbb{R}^n$  с распределением вероятности  $p_X(x)$
- Применим к ней некоторую биективную функцию  $G$  и получим новую величину  $Y = G(X)$
- Тогда её распределение вероятности равно

$$p_Y(y) = p_X(G^{-1}(y)) |\det J(G^{-1}(y))|$$

- Здесь,  $J$  – якобиан функции
- **N.B.**: Вместо определителя якобиана обратной функции, можно вычислять обратный определитель якобиана прямого преобразования:

$$\det J(G^{-1}) = \det (J G)^{-1} = (\det J G)^{-1}$$

## Преобразование распределения: многомерный пример

- Сгенерируем точку на двумерном единичном диске, взяв равномерные угол  $\Phi \sim U(0, 2\pi)$  и расстояние до центра  $R \sim U(0, 1)$
- Координаты точки на диске:

$$G(r, \phi) = (r \cos \phi, r \sin \phi)$$

- Якобиан  $J G$ :

$$\begin{pmatrix} \cos \phi & \sin \phi \\ -r \sin \phi & r \cos \phi \end{pmatrix}$$

- Определитель:  $\det J G = r$
- Определитель обратного преобразования:  $\det J(G^{-1}) = \frac{1}{r}$
- Итог:

$$p_Y(r, \phi) = p_X(G^{-1}(r, \phi)) \frac{1}{r} = \frac{1}{r}$$

## Преобразование распределения: многомерный пример

- Итог:
- При таком способе получается *неравномерное* распределение на диске: точки, близкие к центру, имеют большую плотность вероятности

## Преобразование распределения: многомерный пример №2

- Сгенерируем точку на двумерном единичном диске, взяв равномерный угол  $\Phi \sim U(0, 2\pi)$  и квадрат расстояния до центра  $R^2 \sim U(0, 1)$
- Другими словами, возьмём  $U_1, U_2 \sim U(0, 1)$  и положим  $R = \sqrt{U_1}$  и  $\Phi = 2\pi U_2$
- Координаты точки на диске:

$$G(u_1, u_2) = (\sqrt{u_1} \cos 2\pi u_2, \sqrt{u_1} \sin 2\pi u_2)$$

## Преобразование распределения: многомерный пример №2

- Якобиан  $J G$ :

$$\begin{pmatrix} \frac{1}{2\sqrt{u_1}} \cos 2\pi u_2 & \frac{1}{2\sqrt{u_1}} \sin 2\pi u_2 \\ -2\pi\sqrt{u_1} \sin 2\pi u_2 & 2\pi\sqrt{u_1} \cos 2\pi u_2 \end{pmatrix}$$

- Определитель:  $\det J G = \pi$
- Определитель обратного преобразования:  $\det J(G^{-1}) = \frac{1}{\pi}$
- Итог:

$$p_Y(u_1, u_2) = p_X(G^{-1}(u_1, u_2)) \frac{1}{\pi} = \frac{1}{\pi}$$

## Преобразование распределения: многомерный пример №2

- Итог:

$$p_Y(u_1, u_2) = \frac{1}{\pi}$$

- Мы получили равномерное распределение на диске!
- Множитель  $\frac{1}{\pi}$  ожидаем: при преобразовании из равномерного распределения на множестве  $A$  в равномерное на множестве  $B$  плотность вероятности должна умножиться на  $\frac{|A|}{|B|}$ , чтобы остаться нормированной

## Преобразование распределения: многомерный пример №3

- Возьмём  $U_1, U_2 \sim U(0, 1)$  и преобразуем их как

$$G(u_1, u_2) = (\sqrt{-2 \ln u_1} \cos 2\pi u_2, \sqrt{-2 \ln u_1} \sin 2\pi u_2)$$

- Тогда

$$x^2 + y^2 = -2 \ln u_1 \implies u_1 = \exp\left(-\frac{x^2 + y^2}{2}\right)$$

- Якобиан:

$$\mathbf{J} G = \begin{pmatrix} -\frac{1}{u_1 \sqrt{-2 \ln u_1}} \cos 2\pi u_2 & -\frac{1}{u_1 \sqrt{-2 \ln u_1}} \sin 2\pi u_2 \\ -2\pi \sqrt{-2 \ln u_1} \sin 2\pi u_2 & 2\pi \sqrt{-2 \ln u_1} \cos 2\pi u_2 \end{pmatrix}$$

- Определитель:

$$|\det \mathbf{J} G| = \left| -\frac{2\pi}{u_1} \right|$$

## Преобразование распределения: многомерный пример №3

- Определитель обратного преобразования:

$$|\det J G^{-1}| = \frac{u_1}{2\pi}$$

- Итог:

$$p(x, y) = \frac{1}{2\pi} \exp\left(-\frac{x^2 + y^2}{2}\right) = \left[\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)\right] \cdot \left[\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{y^2}{2}\right)\right]$$

- $\Rightarrow X$  и  $Y$  независимы, и имеют стандартное нормальное распределение
- Это стандартный алгоритм Бокса-Мюллера для генерации нормального распределения
- **N.B.:** Обычно используют вариацию – т.н. Marsaglia polar method – использующую только одну трансцендентную функцию  $\ln +$  rejection sampling в единичном круге (вероятность попасть –  $\frac{\pi}{4}$ )

# Генерация случайных семплов

- Есть очень много более продвинутых методов:
  - Ratio of uniforms
  - Ziggurat sampling
  - Adaptive rejection sampling
  - Metropolis-Hastings algorithm
  - Gibbs sampling
  - ..и другие

## Генерация случайных семплов: единичная сфера

- Для вычисления освещённости ламбертовых поверхностей нам нужно уметь генерировать случайный вектор на единичной полусфере вокруг вектора нормали  $N$
- Проще всего сгенерировать случайный вектор  $\omega$  на единичной сфере, и обратить его, если он смотрит не в ту сторону:  $\omega \mapsto -\omega$  если  $N \cdot \omega < 0$
- Как сгенерировать вектор на единичной сфере? Есть много способов:
  - Rejection sampling в единичном шаре + проекция
  - 3D normal distribution + проекция
  - Проекция с цилиндра
  - Явное обращение CDF

## Единичная сфера: rejection sampling

- Сгенерируем точку равномерно в  $[-1, 1]^3$
- Если  $|V|^2 = X^2 + Y^2 + Z^2 \leq 1$ , возвращаем  $\frac{V}{|V|}$
- Иначе, возвращаемся к первому пункту (это произойдёт с вероятностью  $\sim 0.48$ )

## Единичная сфера: normal + проекция

- Сгенерируем все три координаты как  $X, Y, Z \sim \mathcal{N}(0, 1)$
- Тогда плотность вероятности

$$p(x, y, z) = \frac{1}{(2\pi)^{3/2}} \exp\left(-\frac{x^2 + y^2 + z^2}{2}\right) = \frac{1}{(2\pi)^{3/2}} \exp\left(-\frac{r^2}{2}\right)$$

зависит только от расстояния, и равномерна по  
направлению

- Возвращаем  $\frac{v}{|v|}$

## Проекция с цилиндра

- Сгенерируем точку на цилиндре с равномерными углом  $\Phi \sim U(0, 2\pi)$  и высотой  $Z \sim U(-1, 1)$
- Спроектируем её на поверхность сферы параллельно плоскости XY:

$$x = \sqrt{1 - z^2} \cos \phi$$

$$y = \sqrt{1 - z^2} \sin \phi$$

- По теореме Архимеда о шаре и цилиндре получается равномерное распределение

## Явное обращение CDF в сферических координатах

- Сгенерируем две равномерных величины  $U_1, U_2 \sim U(0, 1)$  и положим

$$\Phi = \cos^{-1}(2U_1 - 1)$$

$$\Theta = 2\pi U_2$$

- Используем  $(\phi, \theta)$  как сферические координаты точки
- Вывод смотри в [этой статье](#)

# Единичная (полу)сфера

- Можете использовать любой понравившийся способ
- В любом случае плотность вероятности случайного вектора на единичной сфере равна  $\frac{1}{4\pi}$
- Плотность вероятности случайного вектора на единичной полусфере равна  $\frac{1}{2\pi}$
- Это будет важно для правильных коэффициентов Монте-Карло интегрирования

# Монте-Карло интегрирование + raytracing

- Итак, общий алгоритм вычисления изображения с помощью трассировки лучей и Монте-Карло интегрирования для решения уравнения рендеринга таков:
  - Генерируем лучи из камеры в центр пикселя в направлении  $d$
  - Вычисляем пересечение каждого луча со сценой
  - В точке пересечения вычисляем  $L_{out}(\omega)$  ( $\omega = -d$ ) как сумму излучения объекта в этой точке  $L_e(\omega)$  и интеграла от входящего излучения
  - Конкретный способ вычисления интеграла зависит от типа материала, но обычно сводится к Монте-Карло интегрированию **одним семплом**
  - Для вычисления значения  $L_{in}(\omega)$  в семплах рекурсивно пускаем дополнительные лучи
  - Усредняем цвет всех лучей, выпущенных из пикселя – это и есть цвет пикселя

## Монте-Карло интегрирование: ламбертова поверхность

- Ламбертова поверхность задаётся фиксированным цветом, и её BRDF константна  $f(p, \omega_{in}, \omega_{out}) = C$
- Добавим фиксированное значение излучения (emission) во всех направлениях:  $L_e(\omega_{out}) = E$
- Интеграл по входящему излучению выглядит как

$$\int_{\Omega(n)} C \cdot L_{in}(\omega) \cdot (\omega \cdot n) d\omega$$

- Апроксимация интеграла одним семплом  $\omega$  на единичной полусфере:

$$\frac{f(\omega)}{p(\omega)} = 2\pi C \cdot L_{in}(\omega) \cdot (\omega \cdot n)$$

## Ламбертова поверхность: алгоритм

- Генерируем случайный вектор на единичной полусфере  
 $\omega \in \Omega(n)$
- Вычисляем  $L_{in}(\omega)$  рекурсивным вызовом
- Возвращаем  $E + 2\pi C \cdot L_{in}(\omega) \cdot (\omega \cdot n)$

## Ламбертова поверхность: нормализация

- Нормализация BRDF требует, чтобы

$$\int_{\Omega(n)} C \cdot (\omega \cdot n) d\omega \leq 1$$

- Вычислим интеграл:

$$\int_{\Omega(n)} C \cdot (\omega \cdot n) d\omega = C \cdot \int_{\Omega(n)} (\omega \cdot n) d\omega = C \cdot \pi$$

- ⇒ Для ламбертовой поверхности, значение цвета (в каждом канале) не может быть больше чем  $\frac{1}{\pi}$ , что соответствует идеально белой поверхности
- Этим неудобно пользоваться, поэтому будем считать, что цвет задан как число  $C \in [0, 1]$ , и для применения в интегрировании его нужно разделить на  $\pi$

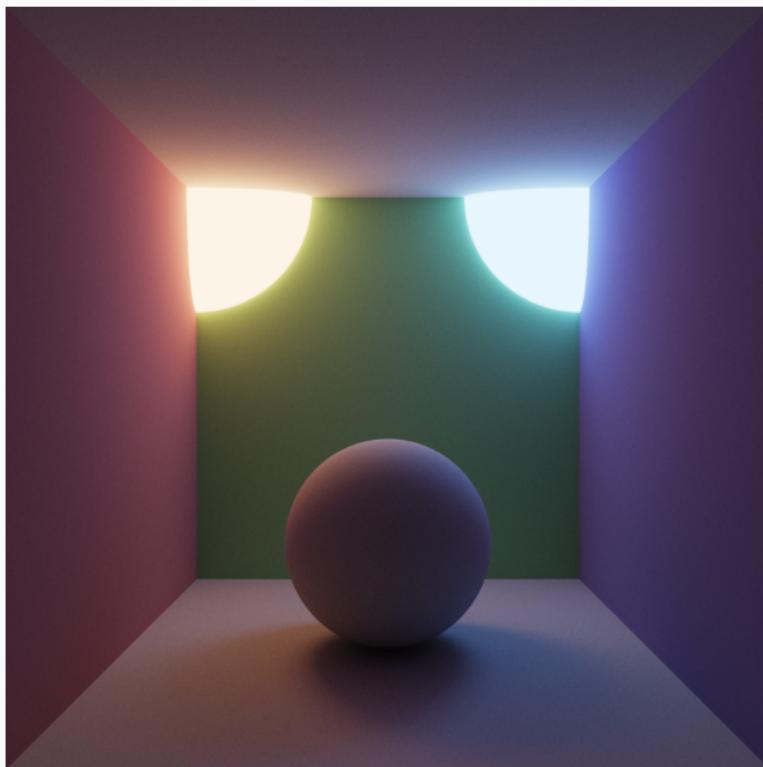
## Ламбертова поверхность: финальная формула

- С учётом нормализации цвета, получаем такую формулу

$$E + 2\pi \frac{C}{\pi} \cdot L_{in}(\omega) \cdot (\omega \cdot n) = E + 2C \cdot L_{in}(\omega) \cdot (\omega \cdot n)$$

- Эту формулу и надо использовать в коде!
- Мы умножили на  $2\pi$ , чтобы учесть плотность вероятности семпла, и разделили на  $\pi$  с учётом удобной нормализации цвета
- Одна из самых частых ошибок в raytracing'е – ошибиться с константами, – например, с количеством раз, когда сократились домножения и деления на  $\pi$

# Ламбертова поверхность



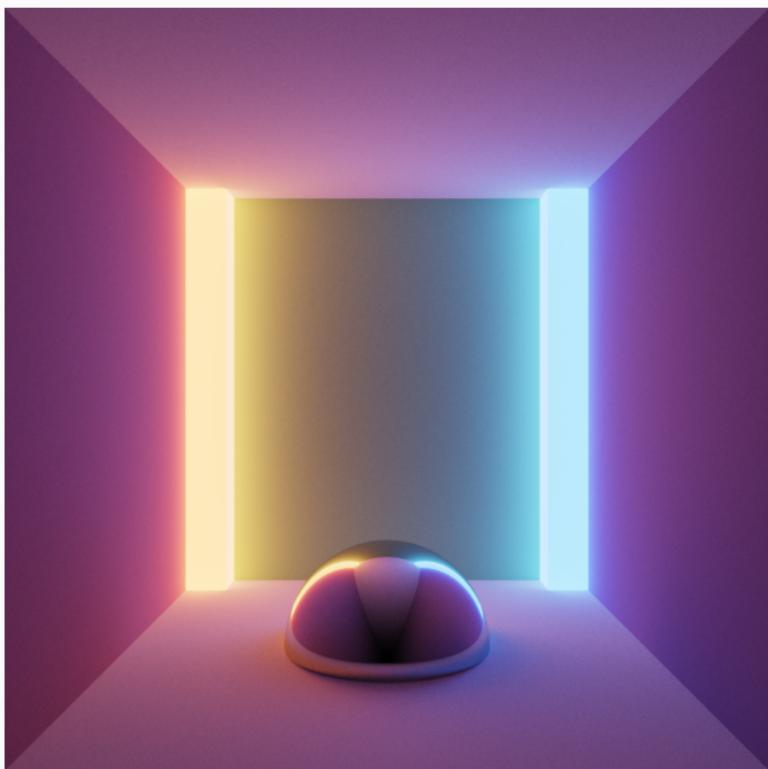
## Монте-Карло интегрирование: металлы

- Ничего не меняется с прошлой лекции: металл отражает весь свет в строго определённом направлении, окрашивая его в свой цвет:

$$L_{out}(\omega) = L_e(\omega) + C \cdot L_{in}(R_n(\omega))$$

- В этом случае, BRDF – делта-функция  $C \cdot \delta(\omega, R_n(\omega))$
- Нормировка:  $C \leq 1$ , никакого деления на  $\pi$  не нужно

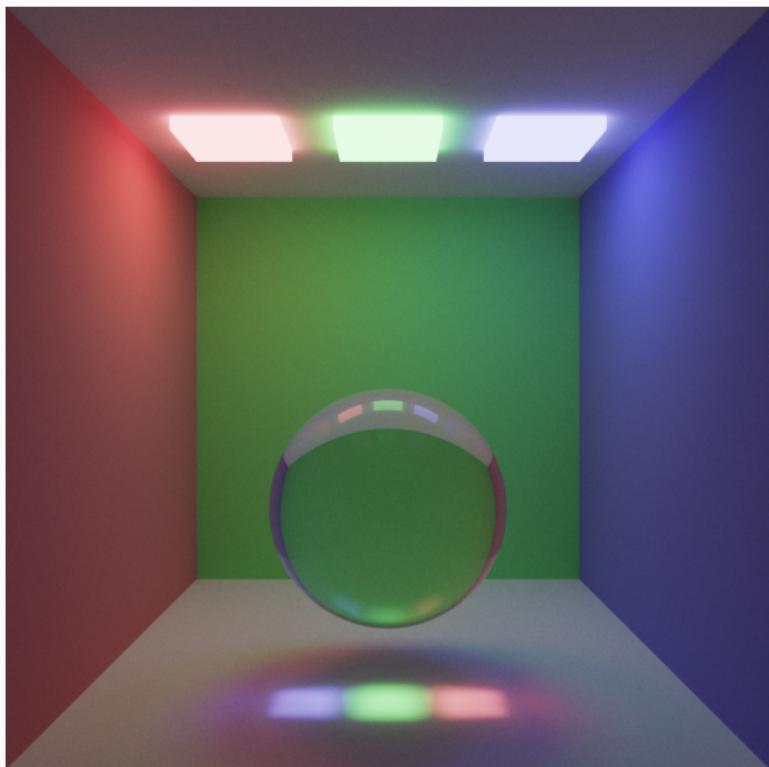
# Металл



## Монте-Карло интегрирование: диэлектрики

- Как в прошлой лекции, вычислим угол преломлённого луча  $\sin \theta_2$  и коэффициент отражения Френеля  $R$
- Если  $\sin \theta_2 > 1$  – полное внутреннее отражение, просто возвращаем отражённый свет
- Иначе, нам нужно вернуть отражённый луч с весом  $R$  и преломлённый луч с весом  $1 - R$
- Вместо того, чтобы вычислять два рекурсивных луча, бросим нечестную монетку: сгенерируем  $U \sim U(0, 1)$ 
  - Если  $U < R$ , возвращаем отражённый свет
  - Иначе возвращаем преломлённый свет
- Домножать возвращённый свет на  $R$  или  $1 - R$  не нужно: это автоматически учтётся за счёт вероятности этих случаев

# Диэлектрик



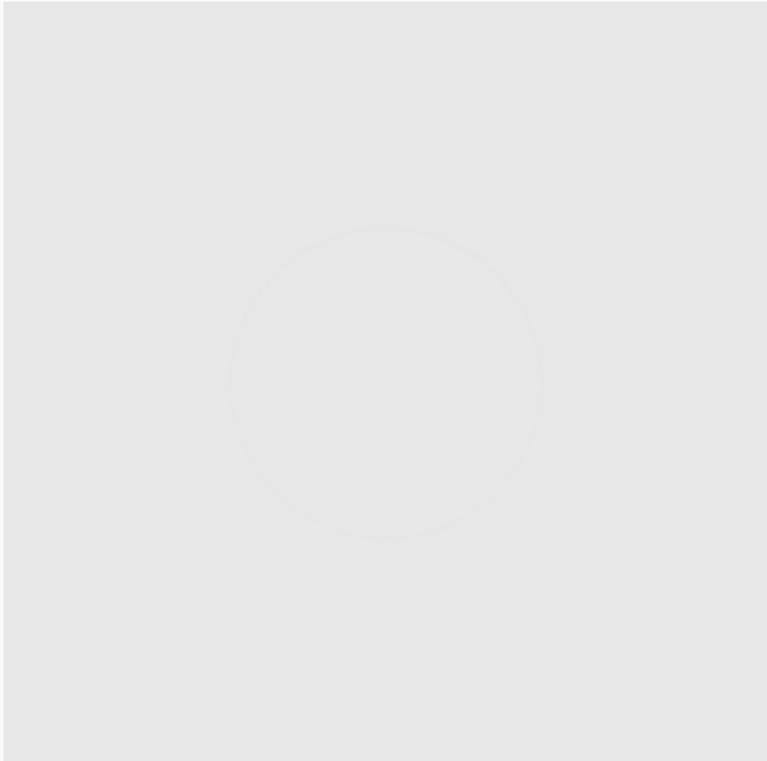
## White furnace test

- Пусть идеально белая ( $C = 1$ ) диффузная выпуклая поверхность освещается идеально белым фоном
- Тогда отражённый свет можно вычислить явно:

$$L_{out}(\omega) = \frac{1}{\pi} \cdot \int_{\Omega(n)} 1 \cdot (\omega \cdot n) d\omega = \frac{1}{\pi} \cdot \pi = 1$$

- Отражённый свет – идеально белый  $\implies$  мы не увидим поверхность – она сольётся с фоном
- Такая ситуация называется *white furnace test* и используется для тестирования алгоритмов рендеринга
- То же самое получится для металлов и диэлектриков

# White furnace test



*Белая сфера на белом фоне*

## Сглаживание (anti-aliasing)

- Из-за того, что мы посылаем лучи в центр пикселя, у нас остаётся один артефакт – алиасинг (лесенка из пикселей)
- Монте-Карло интегрирование позволяет практически бесплатно получить сглаживание (анти-алиасинг)
- Вместо центра пикселя возьмём случайную точку внутри пикселя:  $(x + u_1, y + u_2)$  где  $U_1, U_2 \sim U(0, 1)$

## Сглаживание (anti-aliasing)

