

# Фотореалистичный рендеринг *(aka raytracing)*

Лекция 2: Освещение и тени, отражение и преломление света, металлы и диэлектрики, коррекция цвета

---

2024

# Источники света

- Свет излучается некоторой *объёмной средой*: металл в лампе накаливания, газ в галогенной лампе, плазма на Солнце
- Чтобы честно учитывать такое излучение, нужны очень сложные уравнения *объёмного рендеринга*
- Обычно достаточно считать, что свет излучается только поверхностью объекта – источника света
- Всё ещё сложно учитывать, нужны интегральные уравнения (сл. лекция)

# Источники света

- В real-time графике обычно используют аппроксимации:
  - Бесконечно удалённый (направленный) источник света
  - Бесконечно маленький (точечный) источник света
- В Whitted-style raytracing'е удобно использовать эти же аппроксимации

## Направленный источник света

- Моделируют удалённые, мощные источники – солнце, луна
- Задаются вектором направления и интенсивностью (по трём цветовым каналам)

## Два замечания

- **N.B.:** Есть два соглашения: использовать вектор *на источник света*, или вектор *направления движения света*. Это противоположные векторы, и в формулах надо учитывать, какой вектор используется. Мы будем использовать вектор *на источник света*.
- **N.B.:** Интенсивность источника света, вообще говоря, ничем не ограничена, т.е. может быть сколь угодно большой. Это *количество света*, а не *цвет объекта*, который не может быть больше 1.

# Точечный источник света

- Моделируют относительно близкие источники – костёр, свеча, лампа
- Яркость обычно спадает при удалении от источника
- Задаются положением в пространстве и функцией зависимости интенсивности от расстояния
- Обычно берут такую функцию затухания (*attenuation*) с расстоянием  $R$ :

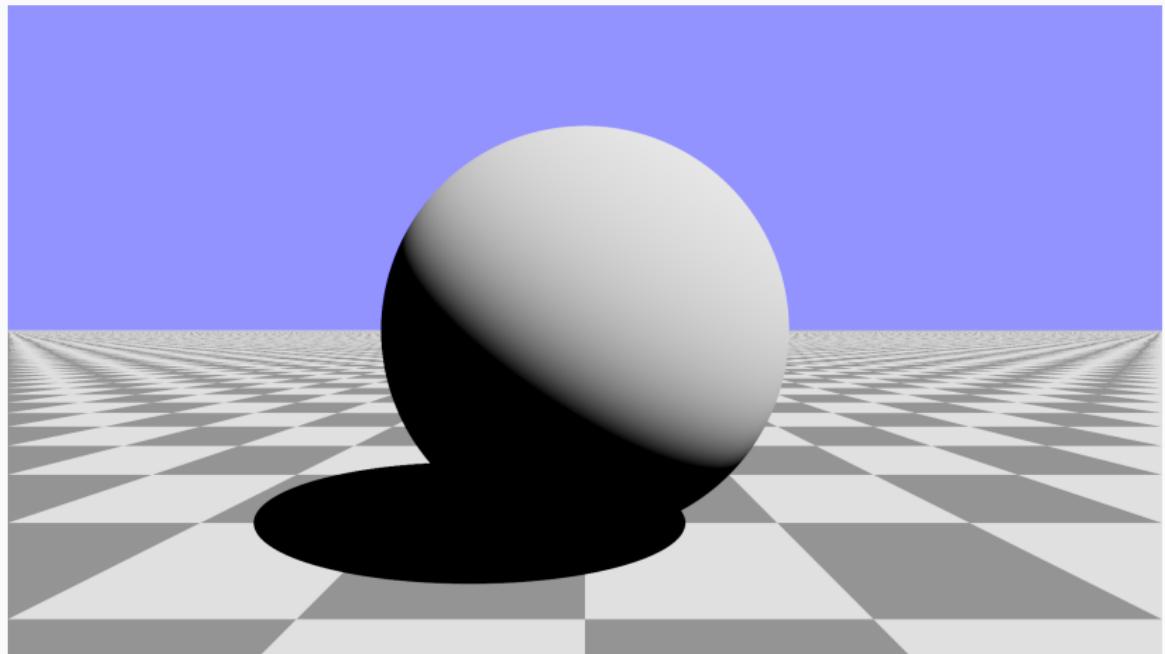
$$\frac{I}{C_0 + C_1R + C_2R^2}$$

- $I$  – интенсивность света без учёта затухания (по трём каналам)
- $C_0 \neq 0$  предотвращает сингулярность вблизи источника света

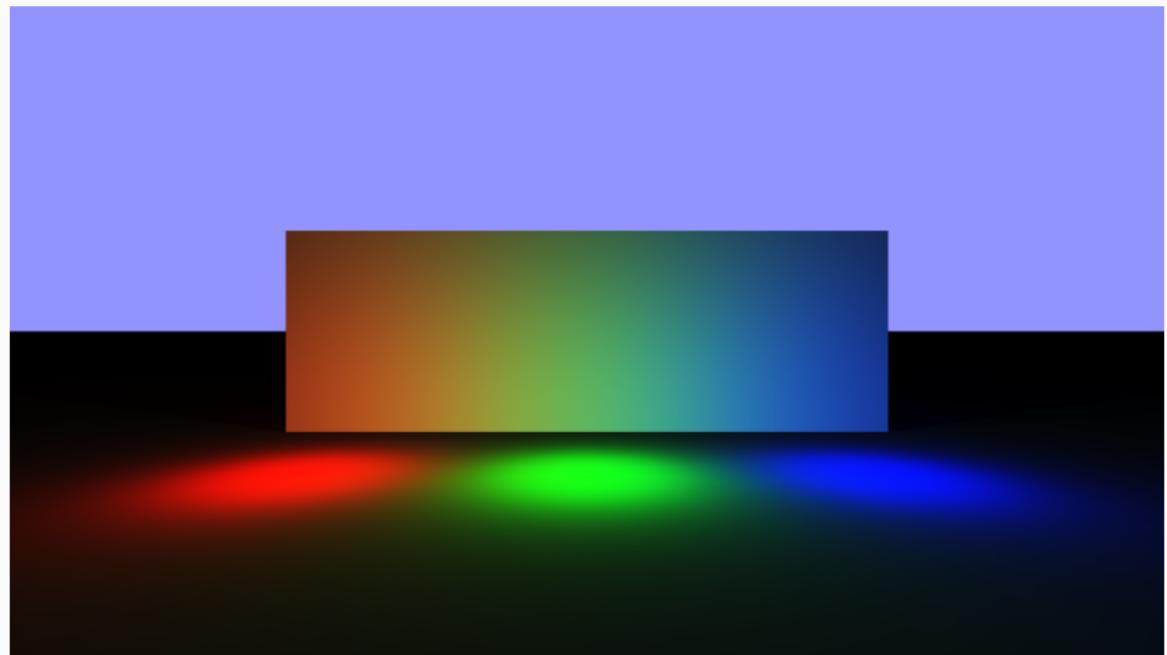
# Диффузные поверхности

- Как учесть источники света при вычислении цвета пикселя?  
Зависит от материала объекта (подробнее на следующей лекции)
- Один из простейших видов материалов – диффузная (она же – ламбертова) поверхность, отражающая падающий свет равномерно во всех направлениях
- Если свет приходит из направления  $L$  (это направление на свет) на поверхность с цветом  $\text{color}$  и нормалью  $N$ , то эта поверхность отражает  $\text{color} \cdot (L \cdot N)$  света в сторону камеры
- Если  $L \cdot N < 0$ , свет вообще не попадает на эту точку (светит с другой стороны поверхности)

## Диффузная поверхность, направленный источник



## Диффузная поверхность, точечные источники

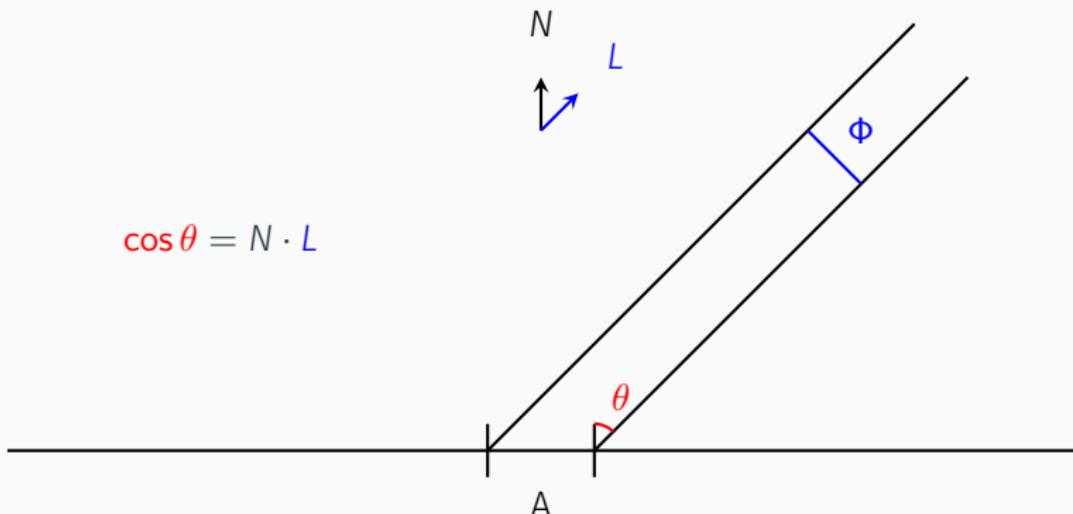


# Диффузные поверхности

- Как учесть источники света при вычислении цвета пикселя?  
Зависит от материала объекта (подробнее на следующей лекции)
- Один из простейших видов материалов – диффузная (она же – ламбертова) поверхность, отражающая падающий свет равномерно во всех направлениях
- Если свет приходит из направления  $L$  (это направление на свет) на поверхность с цветом  $\text{color}$  и нормалью  $N$ , то эта поверхность отражает  $\text{color} \cdot (L \cdot N)$  света в сторону камеры
- Если  $L \cdot N < 0$ , свет вообще не попадает на эту точку (светит с другой стороны поверхности)

## Косинус угла падения

- Откуда взялся  $L \cdot N$ ?
- На поверхность площадью  $A$  падает световой поток с площадью поперечного сечения  $\Phi$  под углом  $\theta$
- Тогда  $\Phi = A \cdot \cos \theta$
- $\Rightarrow$  На единицу площади приходится  $\cos \theta$  от общей плотности потока



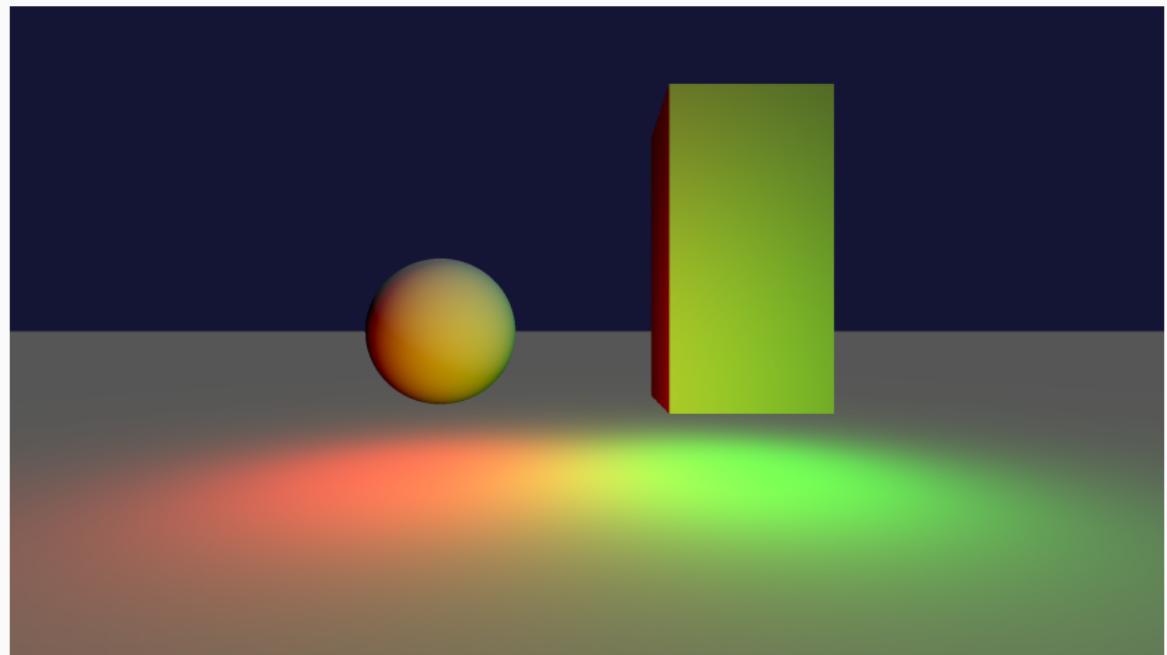
# Ambient освещение

- Часто в сцене присутствует рассеянный свет ‘отовсюду’: свет от неба, свет от звёзд, свет отразившийся от стен
- Правильный способ это учитывать – решать интегральное уравнение (следующая лекция)
- Дешёвый способ это учитывать – задать интенсивность ambient освещения и применять его ко всем точкам всех объектов (без косинусов, просто умножением на цвет объекта)

# Диффузная поверхность: алгоритм

- Строим луч из камеры в текущий пиксель
- Находим пересечение со сценой
- Вычисляем вклад ambient освещения
- Проходимся по всем (точечным и направленным) источникам света
  - Вычисляем направление на этот источник и его яркость в точке пересечения
  - Вычисляем освещённость точки этим источником
- Цвет пикселя – сумма вкладов всех источников света и ambient освещения

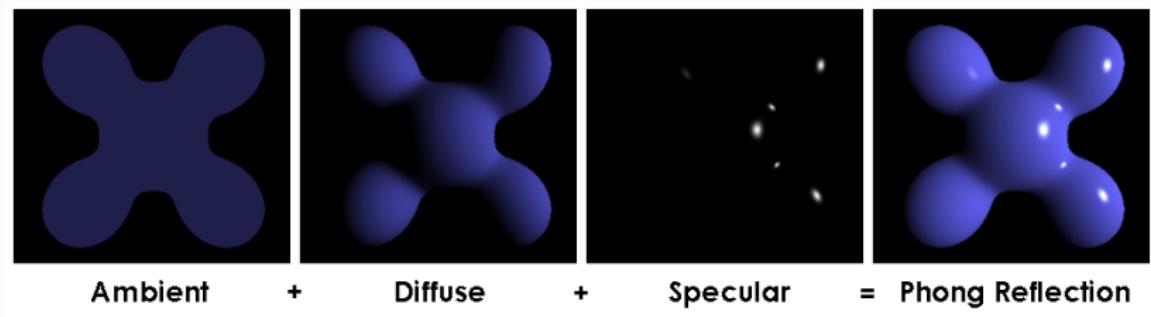
## Диффузная поверхность: много источников света



# Модель Фонга

- К диффузной модели можно добавить т.н. *specular* компоненту – эвристическую аппроксимацию света, вышедшего из источника света и отражённого шероховатой поверхностью в направлении камеры
- Ambient, diffuse и specular вместе дают модель Фонга – самую популярную простую модель освещения
- Мы не будем явно реализовывать specular составляющую, – вместо этого мы напрямую реализуем эффекты, которые она аппроксимирует

# Модель Фонга



# Модель Фонга

- К диффузной модели можно добавить т.н. *specular* компоненту – эвристическую аппроксимацию света, вышедшего из источника света и отражённого шероховатой поверхностью в направлении камеры
- Ambient, diffuse и specular вместе дают модель Фонга – самую популярную простую модель освещения
- Мы не будем явно реализовывать specular составляющую, – вместо этого мы напрямую реализуем эффекты, которые она аппроксимирует

# Нормаль к поверхности

- Раньше при вычислении пересечения луча с объектом нам было достаточно знать параметр  $t$ , описывающий точку пересечения
- Для рассчёта освещения нам нужна дополнительная информация: нормаль к поверхности в точке пересечения
- Вектор нормали **всегда** должен быть нормирован, многие формулы будут это неявно использовать!

## Нормаль к плоскости

- Для плоскости, описываемой уравнением  $N \cdot V = 0$ , вектор  $N$  и есть нормаль

## Нормаль к параллелепипеду

- Зависит от того, на какой грани было ближайшее пересечение:
  - Если это грань  $-X$  (т.е.  $P_X = -S_X$  в терминах прошлой лекции), то нормаль это вектор  $(-1, 0, 0)$
  - Если это грань  $+X$ , то нормаль это вектор  $(1, 0, 0)$
  - Если это грань  $-Y$ , то нормаль это вектор  $(0, -1, 0)$
  - И т.д.

## Нормаль к параллелепипеду

- Чтобы не делать тонну if'ов, можно схитрить: считая, что параллелепипед в центре координат, вычислить  $\frac{P}{S}$ , где  $P$  – точка пересечения, а  $S$  – вектор размеров параллелепипеда по трём осям
- Максимальная по модулю компонента вектора  $\frac{P}{S}$  будет равна единице; если занулить две остальные компоненты, то получится искомая нормаль

## Нормаль к сфере

- Если сфера имеет радиус 1, то вектор из начала координат (т.е. центра сферы) в точку пересечения  $P$  и есть нормаль
- Если сфера имеет произвольный радиус  $R$ , то нормаль это вектор  $P/R$

# Нормаль к сфере

- Сфера описывается уравнением

$$f(P) = X^2 + Y^2 + Z^2 - R^2 = 0$$

- В общем случае, нормаль к поверхности, заданной уравнением  $f(P) = 0$ , это нормированный вектор градиента

$$\frac{\nabla f(P)}{|\nabla f(P)|}$$

- В случае сферы

$$\nabla f(P) = (2X, 2Y, 2Z)$$

- N.B.** Множитель 2 можно игнорировать – он всё равно сократится при нормализации вектора

## Нормаль к эллипсоиду

- Эллипсоид описывается уравнением

$$f(P) = \frac{X^2}{R_X^2} + \frac{Y^2}{R_Y^2} + \frac{Z^2}{R_Z^2} - 1 = 0$$

- Вычислим градиент:

$$\nabla f(P) = \left( \frac{2X}{R_X^2}, \frac{2Y}{R_Y^2}, \frac{2Z}{R_Z^2} \right)$$

## Другой взгляд на эллипсоид

- Эллипсоид с радиусами  $(R_X, R_Y, R_Z)$  можно рассматривать как единичную сферу ( $R = 1$ ), к которой применили неизотропное масштабирование

$$\begin{pmatrix} R_X & 0 & 0 \\ 0 & R_Y & 0 \\ 0 & 0 & R_Z \end{pmatrix}$$

- Точке сферы  $(X, Y, Z)$  соответствует точка эллипсоида  $(X \cdot R_X, Y \cdot R_Y, Z \cdot R_Z)$
- Нормаль к сфере в этой точке  $(X, Y, Z)$  превращается в нормаль к эллипсоиду

$$\left( \frac{X \cdot R_X}{R_X^2}, \frac{Y \cdot R_Y}{R_Y^2}, \frac{Z \cdot R_Z}{R_Z^2} \right) = \left( \frac{X}{R_X}, \frac{Y}{R_Y}, \frac{Z}{R_Z} \right)$$

## Другой взгляд на эллипсоид

- Нормаль к сфере в этой точке  $(X, Y, Z)$  превращается в нормаль к эллипсоиду

$$\left( \frac{X \cdot R_X}{R_X^2}, \frac{Y \cdot R_Y}{R_Y^2}, \frac{Z \cdot R_Z}{R_Z^2} \right) = \left( \frac{X}{R_X}, \frac{Y}{R_Y}, \frac{Z}{R_Z} \right)$$

- ⇒ При применении неизотропного масштабирования, к нормали нужно применить *обратное* масштабирование

## Нормаль к объекту в общем положении

- Если объект был сдвинут, нормаль никак не меняется
- Если объект был повёрнут на кватернион  $Q$ , и мы вычислили нормаль в системе координат объекта (поворнув луч на  $\bar{Q}$ ), её нужно повернуть обратно в мировую систему координат тем же кватернионом  $Q$

# Нормаль при произвольном преобразовании

- Мы увидели две ситуации:
  - Если объект повёрнут, к нормали нужно применить *такое же* преобразование
  - Если объект отмасштабирован, к нормали нужно применить *обратное* преобразование
- Что происходит?

## Нормаль при произвольном преобразовании

- Нормаль  $N$  в точке поверхности объекта определяется тем, что она перпендикулярна касательной плоскости  $T$  в этой точке:

$$\forall V \in T \quad N \cdot V = 0$$

- Если к объекту применяется аффинное преобразование (линейное преобразование + сдвиг), к касательным векторам применяется его линейная часть  $A$ :

$$V \mapsto AV$$

- Преобразованная нормаль  $N'$  должна быть перпендикулярна преобразованным касательным векторам:

$$\forall V \in T \quad N' \cdot AV = 0$$

# Нормаль при произвольном преобразовании

- По свойствам сопряженного оператора (т.е. транспонированной матрицы)

$$N' \cdot AV = 0 \iff A^T N' \cdot V = 0$$

- Мы знаем, что исходная нормаль перпендикулярна касательным векторам:

$$A^T N' = N \implies N' = (A^T)^{-1} N$$

- Обращение и транспонирование матрицы коммутируют между собой, поэтому их композицию часто обозначают как

$$(A^T)^{-1} = (A^{-1})^T = A^{-T}$$

# Нормаль при произвольном преобразовании

- Итак, в общем случае, если к объекту применили аффинное преобразование с линейной частью  $A$ , то нормаль преобразуется посредством матрицы  $A^{-T}$
- **N.B.**: Нормаль после этого всё равно нужно отнормировать
- **N.B.**: Строго говоря, преобразование  $-A^{-T}$  тоже отвечает нашим требованиям; мы выбираем  $A^{-T}$  из условия сохранения ориентации пространства

# Нормаль при произвольном преобразовании

- Поворот – ортогональный оператор, т.е.  $A^T = A^{-1}$  и  $A^{-T} = A$
- $\Rightarrow$  Преобразование нормалей это такой же поворот
- Масштабирование – симметричный оператор (имеет ортогональный базис из собственных векторов), т.е.  $A^T = A$  и  $A^{-T} = A^{-1}$
- $\Rightarrow$  Преобразование нормалей это обратное масштабирование

## Нормаль: внутри или снаружи

- Описанные формулы для нормалей всегда дают внешнюю нормаль к объекту, т.е. смотрящую наружу объекта
- Если камера находится внутри объекта, нам нужна внутренняя нормаль, смотрящая в противоположную сторону!
- Проще всего это учесть по скалярному произведению направления луча  $D$  и нормали  $N$ : если  $D \cdot N < 0$ , то мы смотрим снаружи объекта, иначе – изнутри
- Если мы изнутри объекта, нужно перевернуть нормаль, и сделать пометку, что мы внутри (это будет важно для преломления света)
- **N.B.:** С таким подходом получится, что ‘внутренность’ плоскости это всё полупространство с одной стороны от этой плоскости – так и задумано

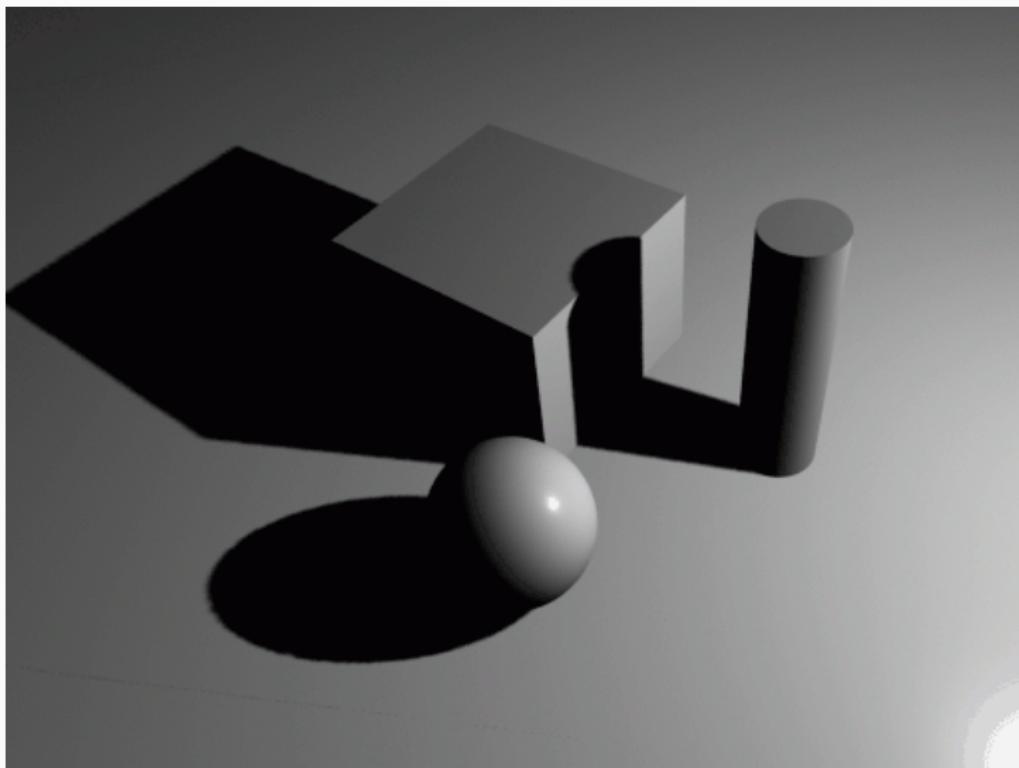
# Пересечение луча с геометрией

- К этому моменту функция пересечения луча с геометрией должна возвращать как минимум три вещи:
  - Параметр  $t$  точки пересечения
  - Нормаль  $N$  к поверхности
  - Пометка, смотрим мы изнутри, или снаружи
- Удобно завести под это отдельную структуру в коде, и возвращать `optional<intersection>`

## Тени: теория

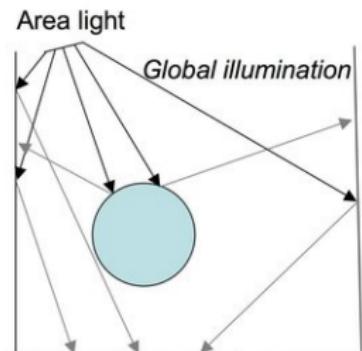
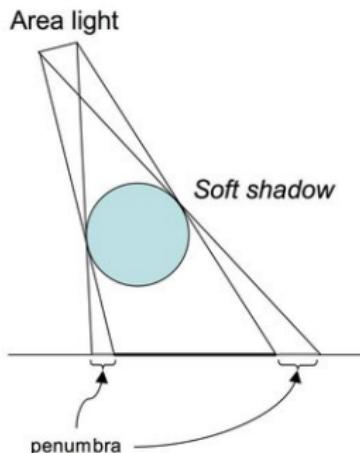
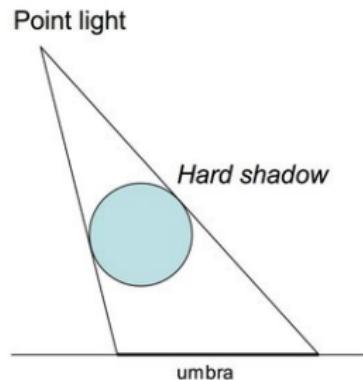
- Тень – область сцены, в которую не попадает (заблокирован чем-то) прямой свет из конкретного источника света
- Свойство точки по отношению к конкретному источнику света

# Тени

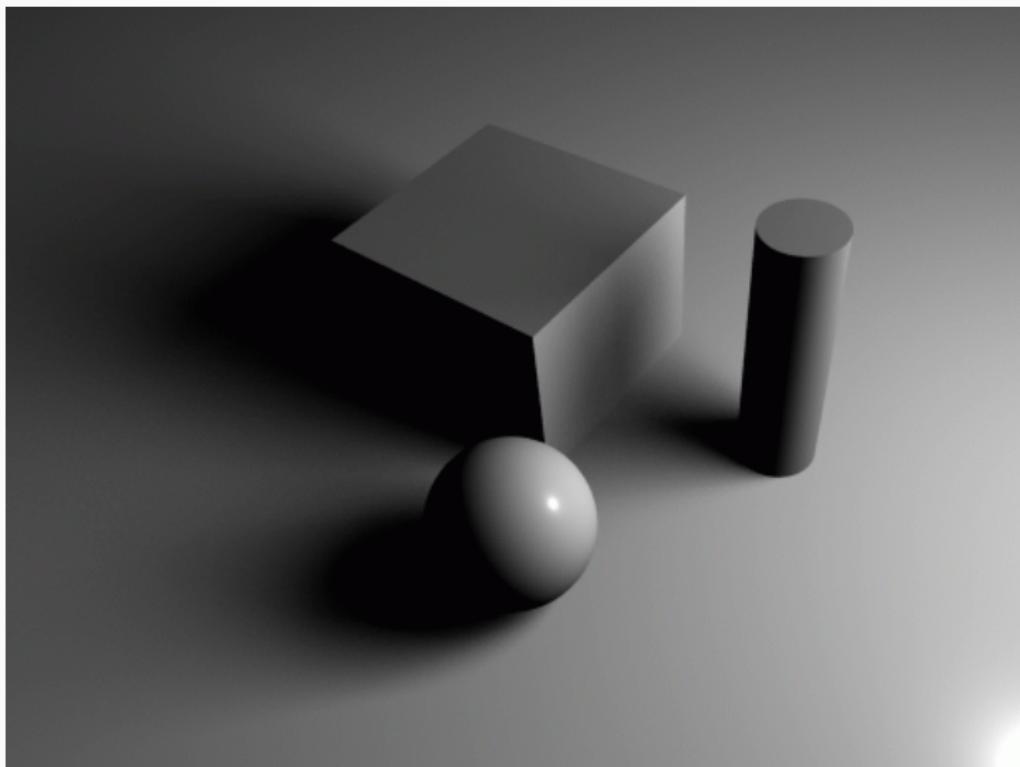


- Если источник света точечный (или бесконечно удалённый), тень – бинарное свойство: луч из точки сцены в источник света или пересекает что-то (точка в тени), или нет (точка не в тени)  $\Rightarrow$  жёсткие тени (*hard shadows*)
- Если источник света объёмный, точка сцены может находиться в тени относительно части источника света, и не находиться в тени относительно другой его части  $\Rightarrow$  мягкие тени (*soft shadows*)
  - Точки, полностью находящиеся в тени – *umbra*
  - Точки, частично находящиеся в тени – *penumbra* (полутень)

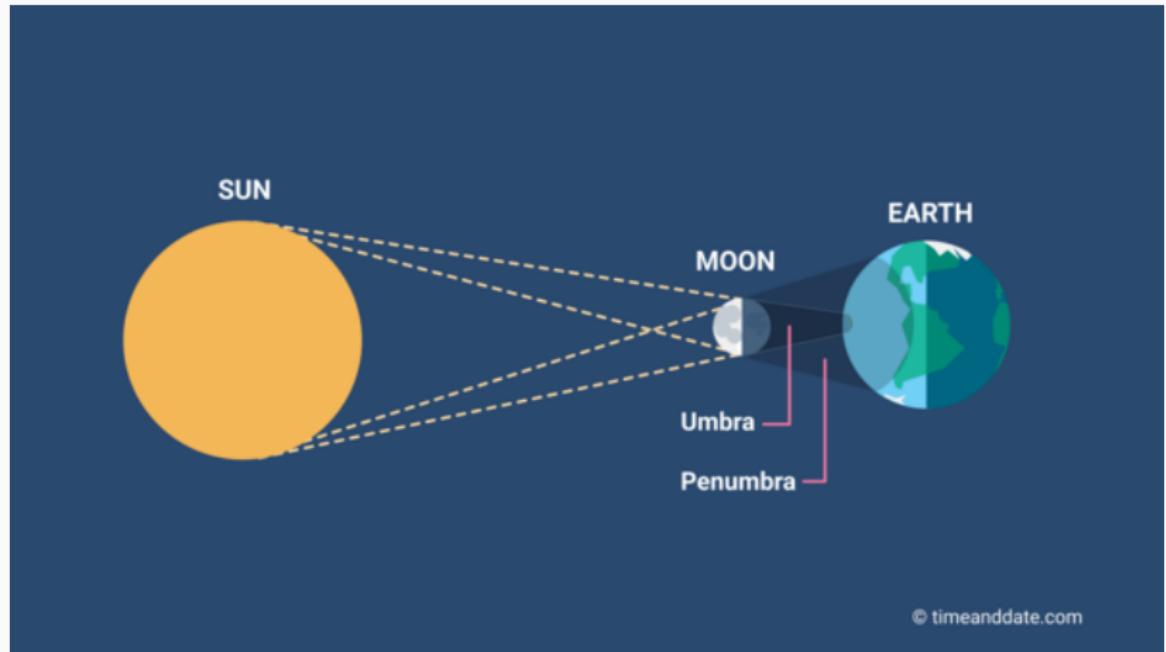
# Тени: мягкие vs жёсткие



## Мягкие тени



# Солнечное затмение



© timeanddate.com

# Тени: теория

- Реальные источники света – объёмные  $\Rightarrow$  реальные тени всегда мягкие
- Размер и форма penumbra зависит от размеров и формы источника света, а также от расстояния до него (чем дальше, тем меньше penumbra)
  - В пределе расстояния  $\rightarrow \infty$  мягкая тень вырождается в жёсткую

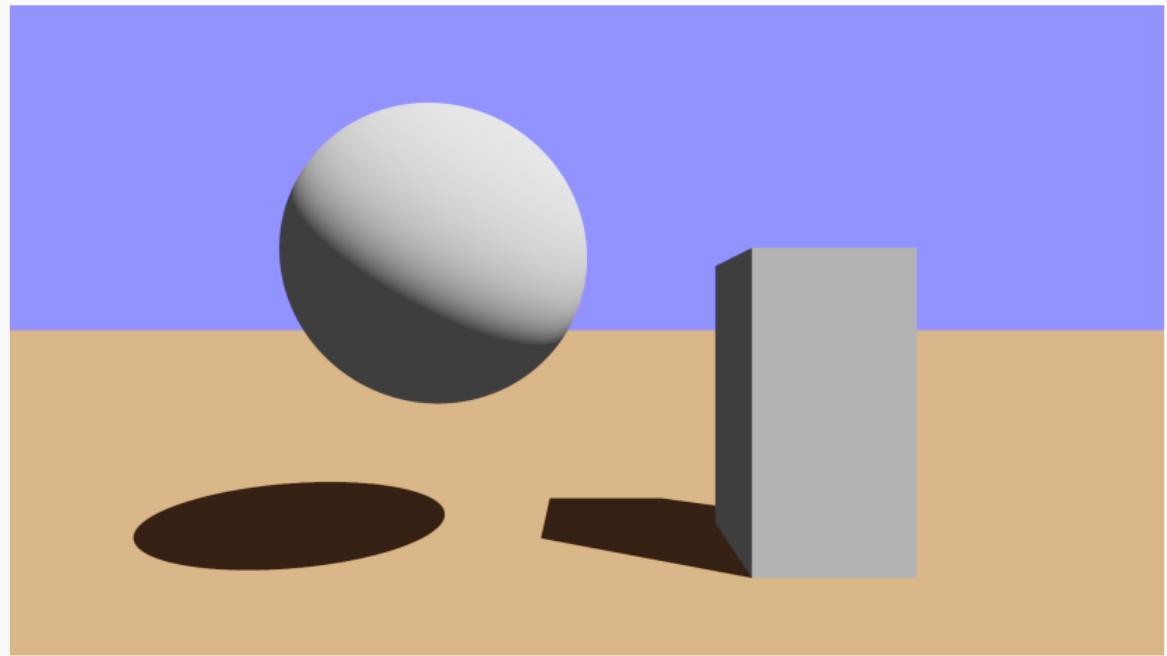
# Тени: raytracing

- Для получения честных мягких теней нужно решать интегральное уравнение (следующая лекция)
- Для получения жёсткой тени можно послать дополнительный луч из точки объекта в направлении источника света (*shadow ray*): если он что-то пересёк, то эта точка находится в тени этого источника света

## Жесткие тени: алгоритм

- При вычислении цвета пикселя, при учёте вклада конкретного источника света, строим луч из точки объекта в направлении источника света
- Пересекаем этот луч со сценой
  - Если пересечение есть, точка находится в тени, и вклад этого источника света равен нулю
  - В противном случае вычисляем вклад этого источника света, как и до этого

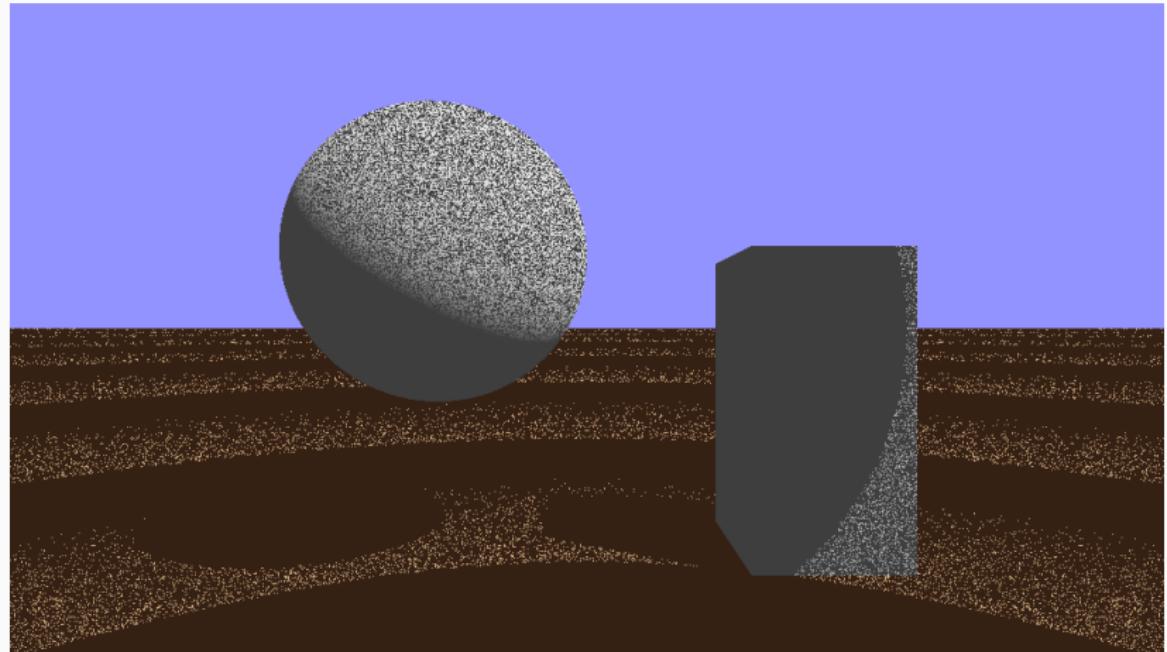
## Жесткие тени



## Жесткие тени: shadow acne

- Если в качестве начала shadow ray взять текущую освещаемую точку, возникнет артефакт *shadow acne*
- Это связано с тем, что пересечение луча со сценой иногда даёт нам ту же точку, в которой мы вычисляем освещение, а иногда нет – в сущности, мы видим результат округления floating-point вычислений
- В некоторых частных случаях это можно решить, запоминая, какой именно объект мы пересекли (работает только для непрозрачных выпуклых объектов)
- Проще всего решить проблему грязным хаком: явно подвинуть начало луча в сторону от поверхности (в направлении луча, или в направлении нормали к поверхности) на маленькое расстояние, скажем,  $10^{-4}$

## Shadow acne



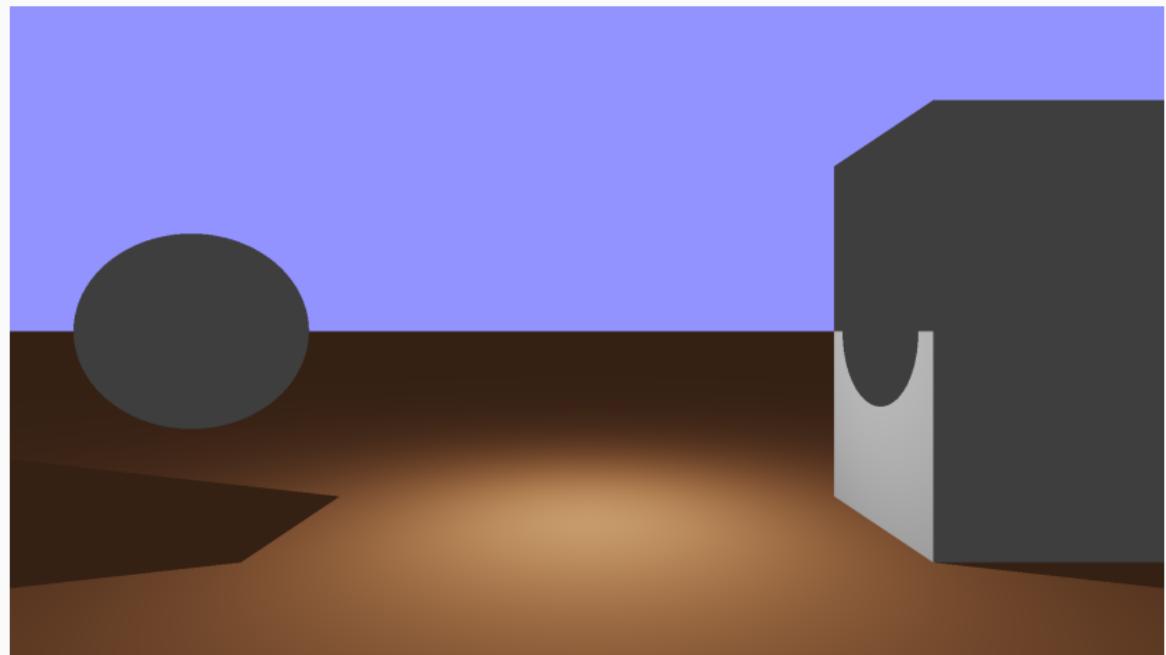
## Жесткие тени: shadow acne

- Если в качестве начала shadow ray взять текущую освещаемую точку, возникнет артефакт *shadow acne*
- Это связано с тем, что пересечение луча со сценой иногда даёт нам ту же точку, в которой мы вычисляем освещение, а иногда нет – в сущности, мы видим результат округления floating-point вычислений
- В некоторых частных случаях это можно решить, запоминая, какой именно объект мы пересекли (работает только для непрозрачных выпуклых объектов)
- Проще всего решить проблему грязным хаком: явно подвинуть начало луча в сторону от поверхности (в направлении луча, или в направлении нормали к поверхности) на маленькое расстояние, скажем,  $10^{-4}$

## Жесткие тени: точечный источник

- При вычислении теней от точечного источника нас интересует только пересечение луча с объектами ближе источника света, иначе получатся неправильные тени
- Если два способа это решить:
  - 1. После вычисления пересечения shadow ray со сценой сравнить параметр  $t$  пересечения с расстоянием до источника света: если источник ближе, считаем, что точка **не в тени**
  - 2. Передавать в функцию вычисления пересечения луча со сценой максимальное расстояние вдоль луча как дополнительный параметр (в обычной ситуации равный  $+\infty$ ): все пересечения с  $t > \text{max}$  нужно игнорировать
- Лучше сделать второй вариант: в дальнейшем он будет нам полезен для оптимизаций

## Неправильные тени от точечного источника



## Жесткие тени: точечный источник

- При вычислении теней от точечного источника нас интересует только пересечение луча с объектами ближе источника света, иначе получатся неправильные тени
- Если два способа это решить:
  - 1. После вычисления пересечения shadow ray со сценой сравнить параметр  $t_{\text{пересечения}}$  с расстоянием до источника света: если источник ближе, считаем, что точка **не в тени**
  - 2. Передавать в функцию вычисления пересечения луча со сценой максимальное расстояние вдоль луча как дополнительный параметр (в обычной ситуации равный  $+\infty$ ): все пересечения с  $t > \max$  нужно игнорировать
- Лучше сделать второй вариант: в дальнейшем он будет нам полезен для оптимизаций

## Более сложные материалы

- Диффузная модель хорошо описывает матовые, шероховатые поверхности
- Далеко не все поверхности в реальном мире – диффузные!
- В графике обычно рассматривают два основных вида нетривиальных поверхностей: *металлы* и *диэлектрики*
- Более сложные материалы часто строятся как комбинация диффузной, металлической, и диэлектрической моделей

# Металлы

- Особенность металлов в том, что они *отражают* приходящий свет
- То, сколько света отразится, определяется цветом металла
- Отражения от шероховатого металла – размытые (в пределе получится диффузная поверхность), и чтобы честно их учитывать, нужно, – вы не поверите! – решать интегральное уравнение
- Идеальные (неразмытые) отражения легко получить, послав дополнительный луч

## Направление отражённого луча

- Пусть свет пришёл из направления  $L$  в точку с нормалью  $N$
- Направление отражённого луча  $R$  можно вычислить по формуле

$$R = 2N \cdot (N \cdot L) - D$$

- У нас  $L = -D$ , где  $D$  – направление луча из камеры, и формула переписывается как

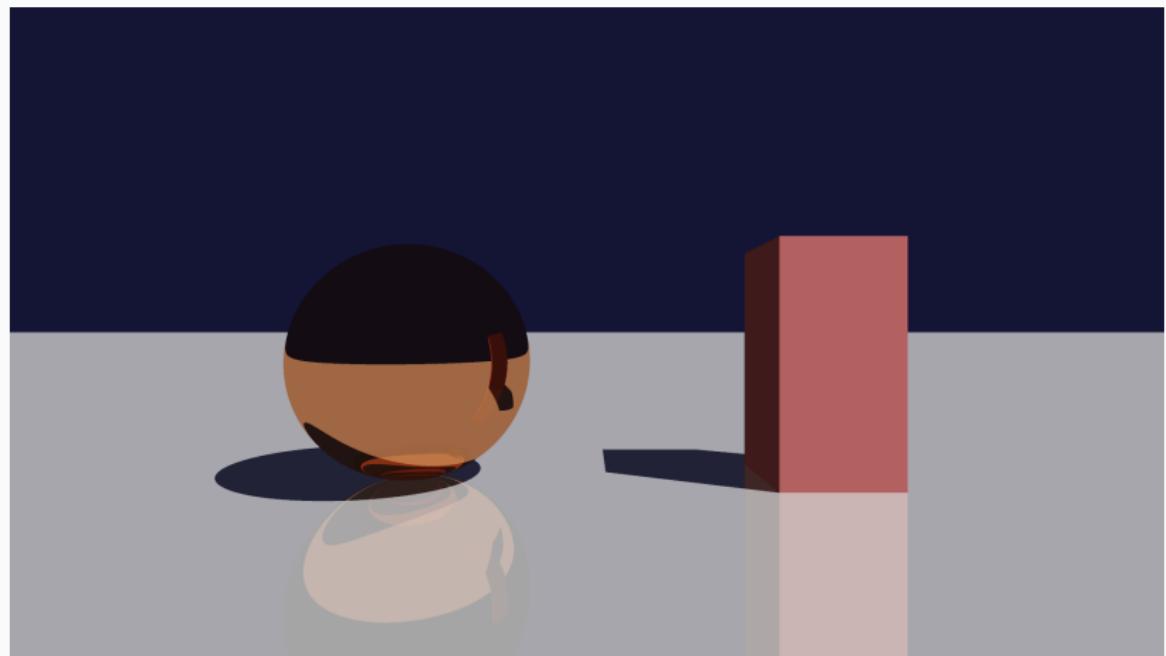
$$R = D - 2N \cdot (N \cdot D)$$

- **N.B.:** Как и с теневым лучом, начало отражённого луча нужно немного подвинуть в сторону от поверхности

## Металлическая поверхность: алгоритм

- Строим луч из камеры в текущий пиксель
- Находим пересечение со сценой
- Строим отражённый луч
- Рекурсивно вызываем функцию получения света по лучу в направлении отражённого луча
- Цвет пикселя – полученный рекурсивным вызовом свет, умноженный на цвет металлического объекта

# Металлическая поверхность



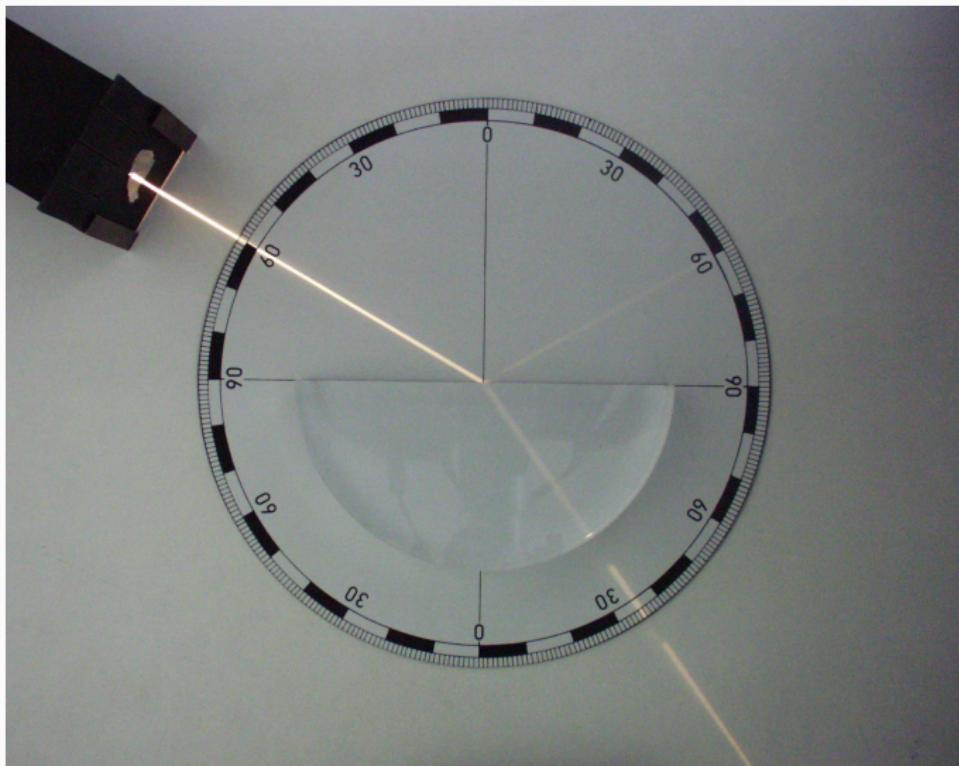
## Металлическая поверхность: рекурсия

- Мы впервые столкнулись с необходимостью *рекурсивно* вызвать функцию получения света по лучу
- В общем случае ничто не гарантирует, что эта рекурсия закончится за разумное число шагов
- Стандартное решение – ограничить максимальную глубину рекурсии некоторым числом, обычно от 4 до 16

# Диэлектрики

- Диэлектрик – вещество, не проводящее ток (в физике определение не совсем такое, но нам это не важно)
- Особенность диэлектриков в том, что они как отражают, так и преломляют свет
- За эти процессы отвечают закон Снеллиуса и уравнения Френеля
- **N.B.:** Непрозрачные объекты – часто тоже диэлектрики, но эффекты поглощения и рассеяния света внутри этих объектов делают их непрозрачными; обычно такие объекты моделируют как диффузные с отражающим покрытием, описываемые уравнениями Френеля

# Диэлектрики



# Диэлектрики

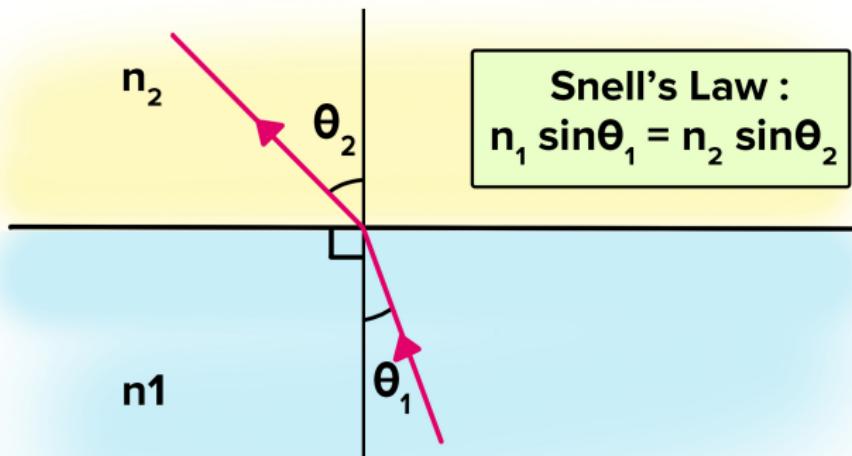
- Диэлектрик – вещество, не проводящее ток (в физике определение не совсем такое, но нам это не важно)
- Особенность диэлектриков в том, что они как отражают, так и преломляют свет
- За эти процессы отвечают закон Снеллиуса и уравнения Френеля
- **N.B.:** Непрозрачные объекты – часто тоже диэлектрики, но эффекты поглощения и рассеяния света внутри этих объектов делают их непрозрачными; обычно такие объекты моделируют как диффузные с отражающим покрытием, описываемые уравнениями Френеля

## Закон Снеллиуса (Snell's Law)

- Пусть свет идёт из среды с коэффициентом преломления  $\eta_1$  в среду с коэффициентом  $\eta_2$
- Тогда угол падения  $\theta_1$  (между падающим лучом и нормалью к поверхности) и угол преломления  $\theta_2$  связаны соотношением

$$\eta_1 \cdot \sin \theta_1 = \eta_2 \cdot \sin \theta_2$$

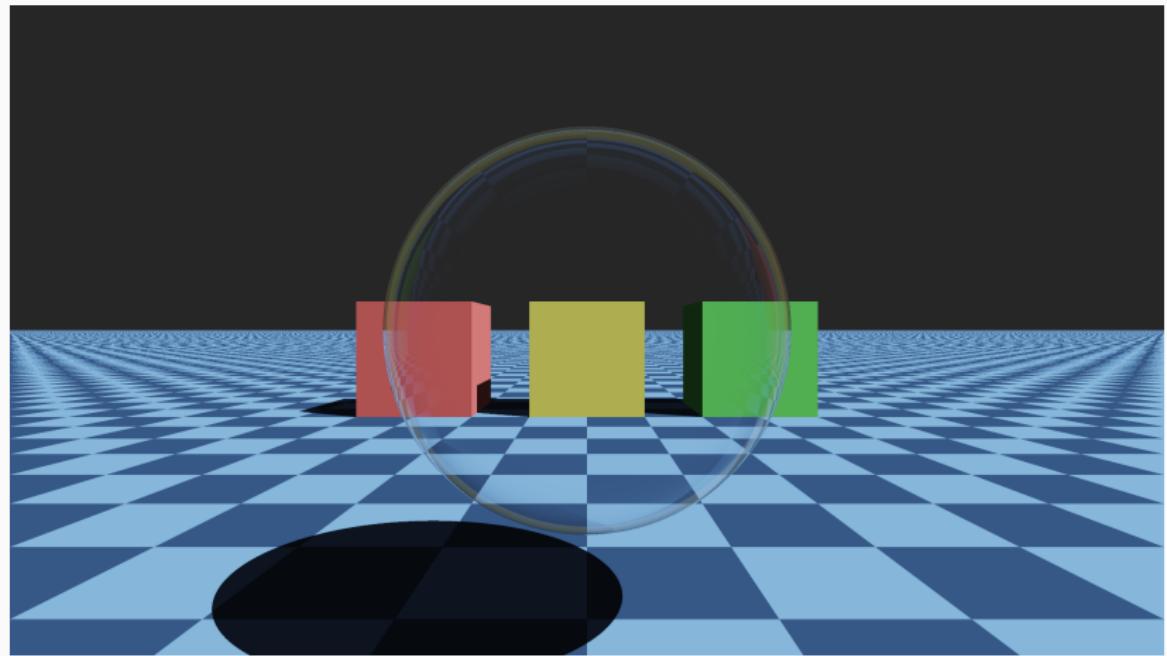
## Закон Снеллиуса (Snell's Law)



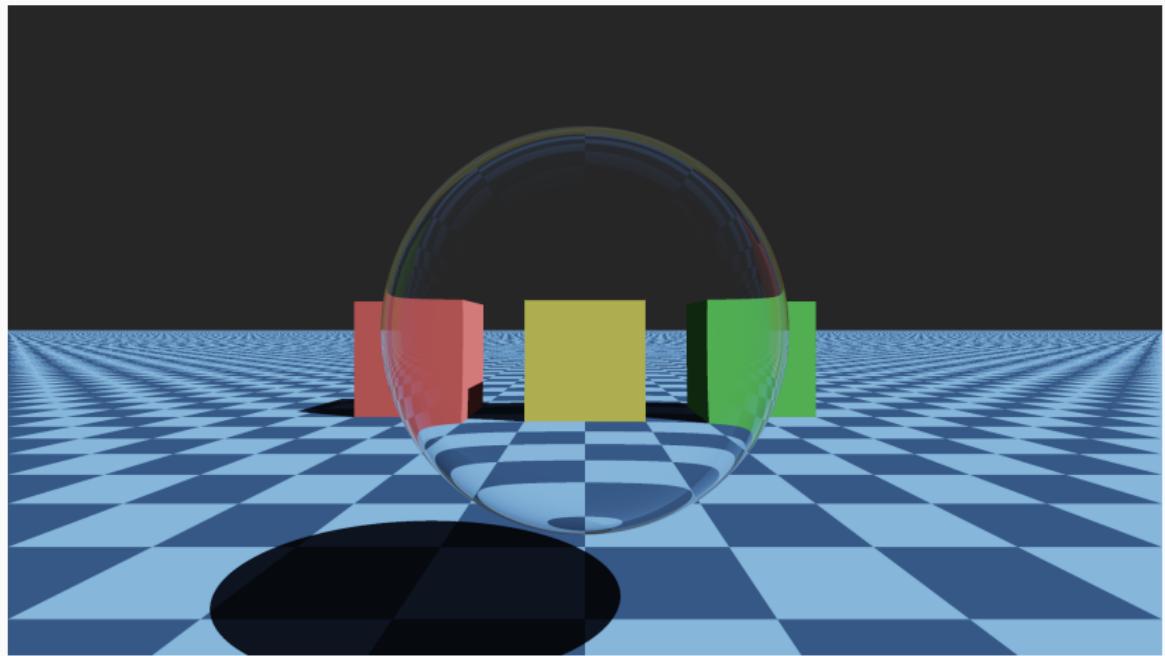
# Коэффициент преломления

- Коэффициент преломления (*refractive index* или *index of reflection – IOR*) связан со скоростью распространения света в среде
- Типичные значения:
  - Воздух: 1
  - Вода: 1.333
  - Стекло: 1.5
  - Алмаз: 2.4

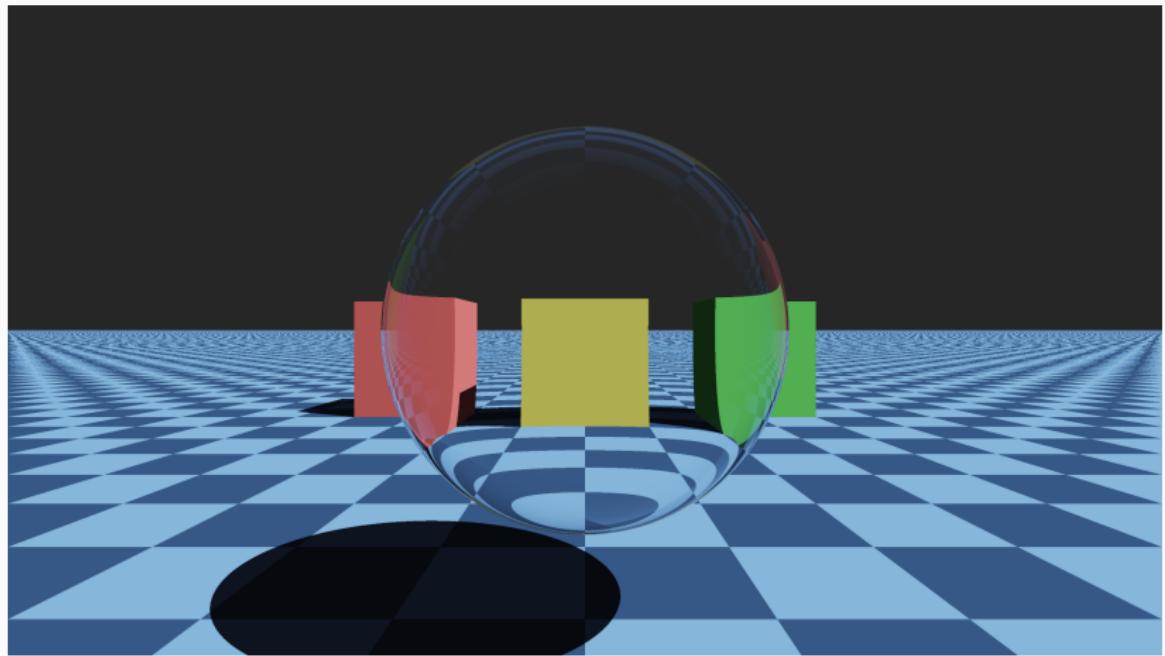
$\text{IOR} = 1$



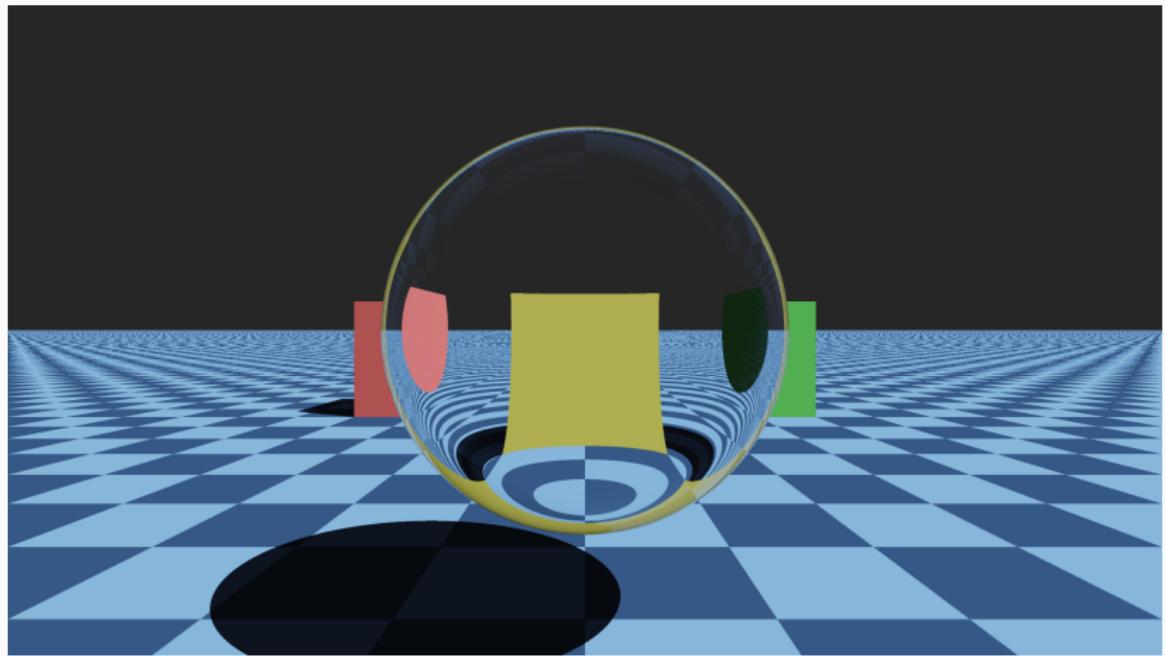
IOR = 1.01



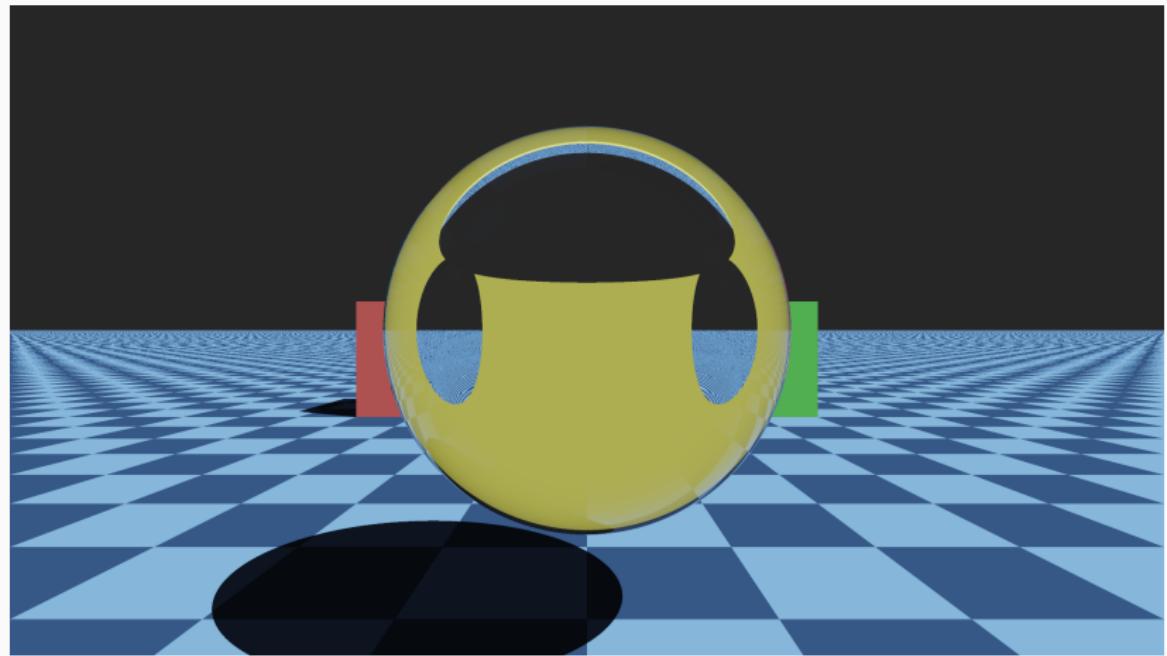
IOR = 1.02



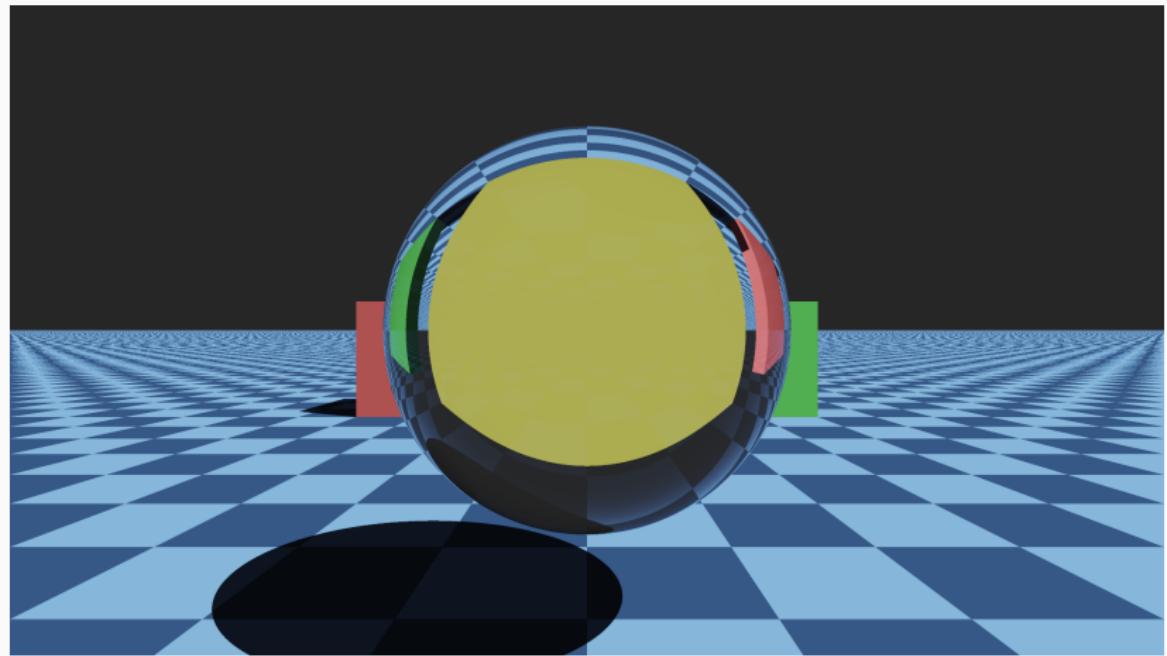
$\text{IOR} = 1.05$



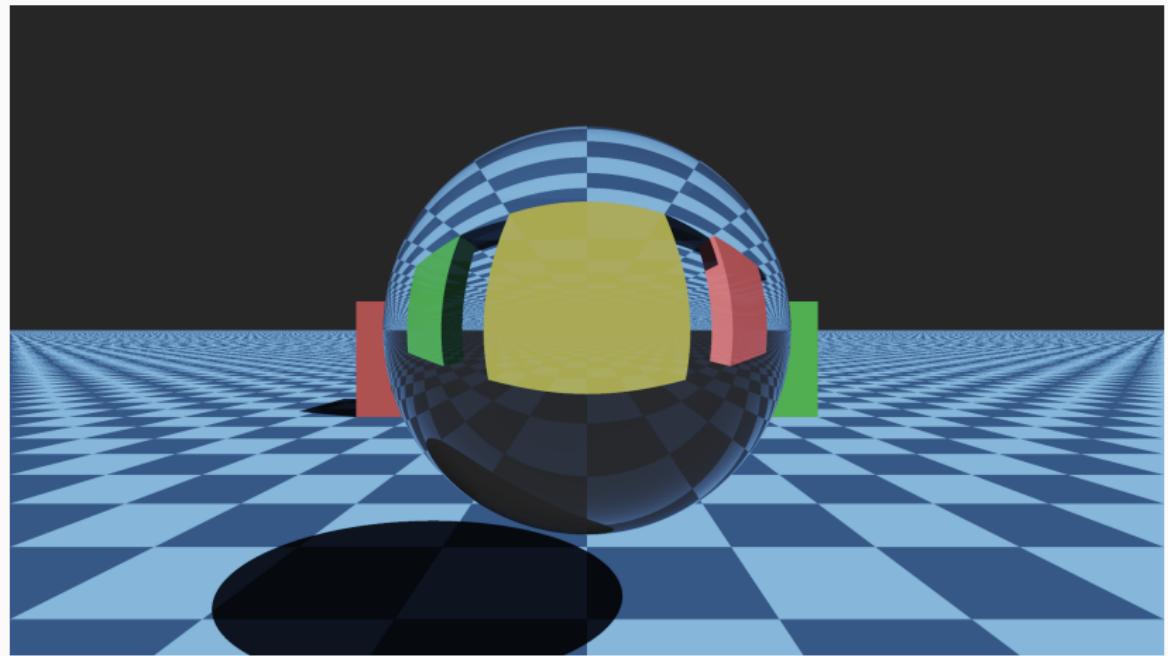
IOR = 1.1



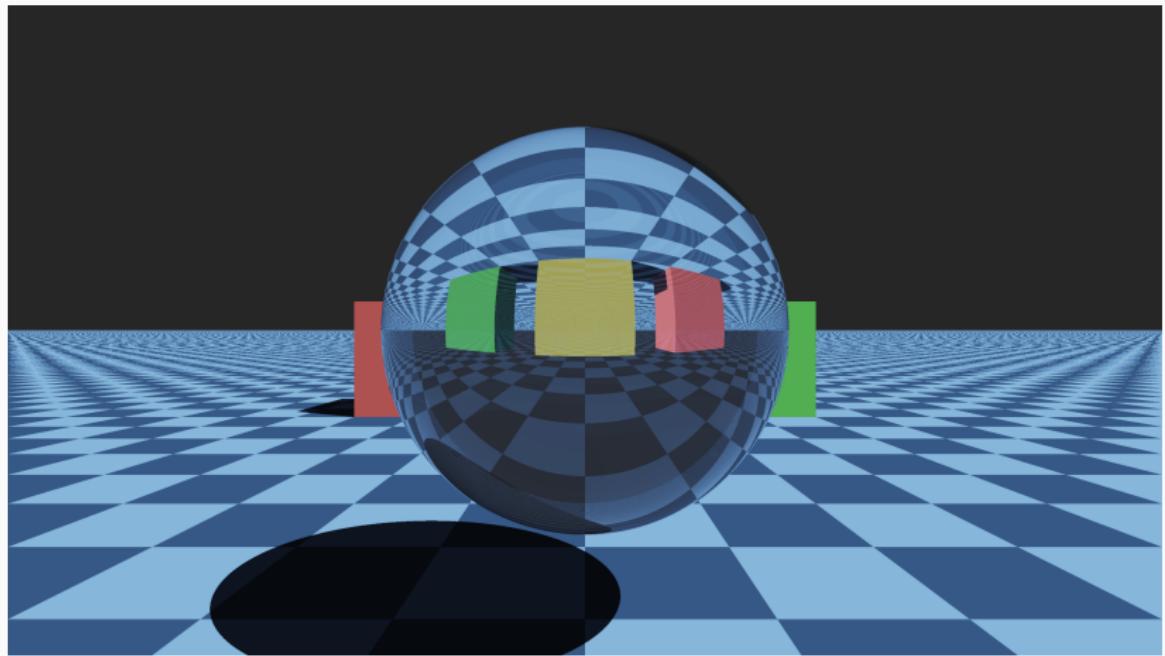
IOR = 1.3



IOR = 1.5



$\text{IOR} = 2$



# Направление преломлённого луча

- Из закона Снеллиуса можно вывести формулу для направления преломлённого луча
- Пусть свет падает из направления  $L$  на поверхность с нормалью  $N$
- Тогда направление преломлённого луча  $R$  можно вычислить как

$$\cos \theta_1 = N \cdot L$$

$$\sin \theta_2 = \frac{\eta_1}{\eta_2} \sqrt{1 - \cos^2 \theta_2} = \frac{\eta_1}{\eta_2} \sqrt{1 - (N \cdot L)^2}$$

$$\cos \theta_2 = \sqrt{1 - \sin^2 \theta_2}$$

$$R = \frac{\eta_1}{\eta_2}(-L) + \left( \frac{\eta_1}{\eta_2} \cos \theta_1 - \cos \theta_2 \right) \cdot N$$

# Направление преломлённого луча

- Нас интересует, откуда придет свет, который после преломления попадёт в направлении камеры
- $\Rightarrow$  Для луча из камеры в направлении  $D$  в формулах прошлого слайда нужно взять  $L = -D$
- $\eta_1$  положим равным 1 (воздух)
- $\eta_2$  – часть описания материала
- Если луч пришёл изнутри объекта,  $\eta_1$  и  $\eta_2$  нужно **поменять местами!**

## Полное внутреннее отражение

- Посмотрим внимательнее на формулу

$$\sin \theta_2 = \frac{\eta_1}{\eta_2} \sqrt{1 - \cos^2 \theta_2} = \frac{\eta_1}{\eta_2} \sqrt{1 - (N \cdot L)^2}$$

- Если  $\eta_1 > \eta_2$ , значение  $\sin \theta_2$  может оказаться больше 1 по модулю
- В этом случае преломления нет, и объект только отражает свет – происходит *полное внутреннее отражение*

## Полное внутреннее отражение



# Уравнения Френеля

- Описывают то, сколько света отразится, а сколько – пройдёт сквозь объект (и преломился по закону Снеллиуса)
- Сложные уравнения, учитывающие поляризацию света
- В графике почти всегда считается, что свет не поляризован
- Вместо настоящих уравнений Френеля используется *аппроксимация Шлика*

## Аппроксимация Шлика (Schlick's approximation)

- Пусть свет падает из направления  $L$  на диэлектрическую поверхность с нормалью  $N$
- Тогда отразится свет в количестве

$$R = R_0 + (1 - R_0) \cdot (1 - (N \cdot L))^5$$

- Соответственно, преломится свет в количестве  $1 - R$
- Здесь,  $R_0$  – коэффициент отражения, если смотреть прямо на поверхность; он равен

$$R_0 = \left( \frac{\eta_1 - \eta_2}{\eta_1 + \eta_2} \right)^2$$

## Диэлектрики: алгоритм

- Строим луч из камеры в текущий пиксель
- Находим пересечение со сценой
- Вычисляем  $\sin \theta_2$
- Если он  $\leq 1$ :
  - Вычисляем преломлённый и отражённый лучи
  - Рекурсивно получаем свет, приходящий из этих направлений
  - Цвет пикселя – среднее этих значений с коэффициентами согласно аппроксимации Шлика
- Если он  $> 1$ :
  - Полное внутреннее отражение, цвет пикселя это отражённый свет

## Диэлектрики: цвет

- Цвет полупрозрачного объекта определяется тем, как он окрашивает проходящий сквозь себя свет
- Правильно было бы учитывать то, как далеко свет шёл внутри полупрозрачного объекта, пока не вышел из него, и использовать формулы из volume rendering'a
- Пока будем использовать аппроксимацию: если мы смотрим на диэлектрик снаружи, то преломлённый свет (но **не отражённый**) нужно умножить на цвет объекта

## Гамма

- Пока мы не задумывались о том, что за значения мы записываем в пиксели выходной картинки
- Обычно, интенсивность света  $I$ , излучаемого монитором, нелинейно зависит от значения  $V$ , записанного в пикселе
- Это лучше соответствует восприятию света человеком и даёт больше точности тёмным цветам, которые человек различает лучше
- Почти всегда используется показательная функция:

$$I \sim V^\gamma \quad (1)$$

- $\gamma$  обычно равна 2.2 (некоторые компьютеры Macintosh использовали 1.8)

# Линейное значение пикселя vs линейная интенсивность излучения

Linear  
encoding

$$V_S = 0.00\ 0.10\ 0.20\ 0.30\ 0.40\ 0.50\ 0.60\ 0.70\ 0.80\ 0.90\ 1.0$$

Linear  
intensity

$$I = 0.00\ 0.10\ 0.20\ 0.30\ 0.40\ 0.50\ 0.60\ 0.70\ 0.80\ 0.90\ 1.0$$



## Гамма

- Пока мы не задумывались о том, что за значения мы записываем в пиксели выходной картинки
- Обычно, интенсивность света  $I$ , излучаемого монитором, нелинейно зависит от значения  $V$ , записанного в пикселе
- Это лучше соответствует восприятию света человеком и даёт больше точности тёмным цветам, которые человек различает лучше
- Почти всегда используется показательная функция:

$$I \sim V^\gamma \quad (1)$$

- $\gamma$  обычно равна 2.2 (некоторые компьютеры Macintosh использовали 1.8)

# Проблемы гаммы

- Искажается восприятие относительных яркостей, особенно при реалистичном рендеринге (e.g. объект в два раза ярче не будет выглядеть в два раза ярче)
- Неправильно выглядит освещение, наложение источников света, и т.д.

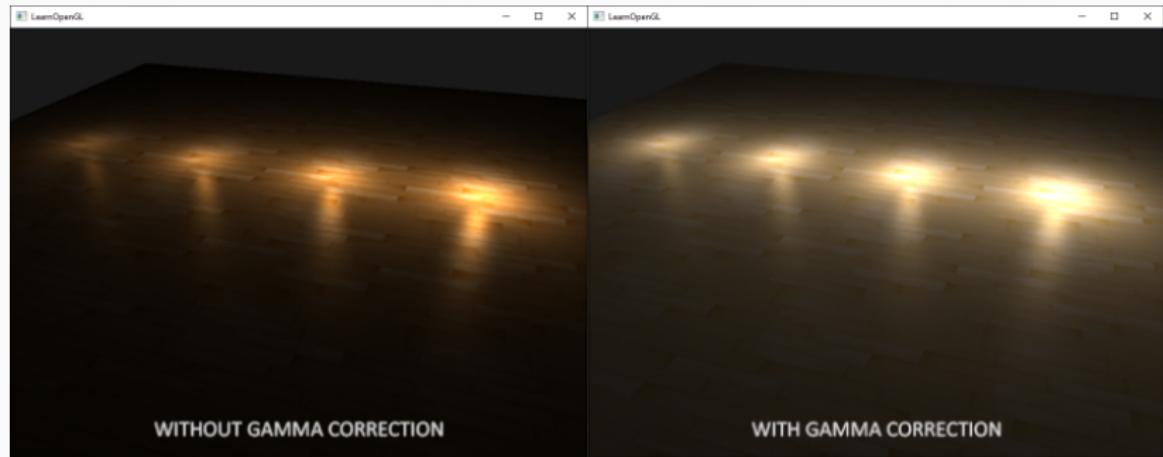
## Коррекция гаммы (gamma-correction)

- Коррекция гаммы – общий термин для применения любых нелинейных преобразований над интенсивностью пикселя
- В рендеринге под гамма-коррекцией обычно подразумевают применение обратного к  $V^{2.2}$  преобразования, чтобы получить линейную зависимость выходящего излучения от значения пикселя
- В нашем случае сводится к тому, что перед записью вычисленного цвета в пиксель, его нужно возвести в степень:  $V^{1/2.2}$  (до конвертации в диапазон 0..255)

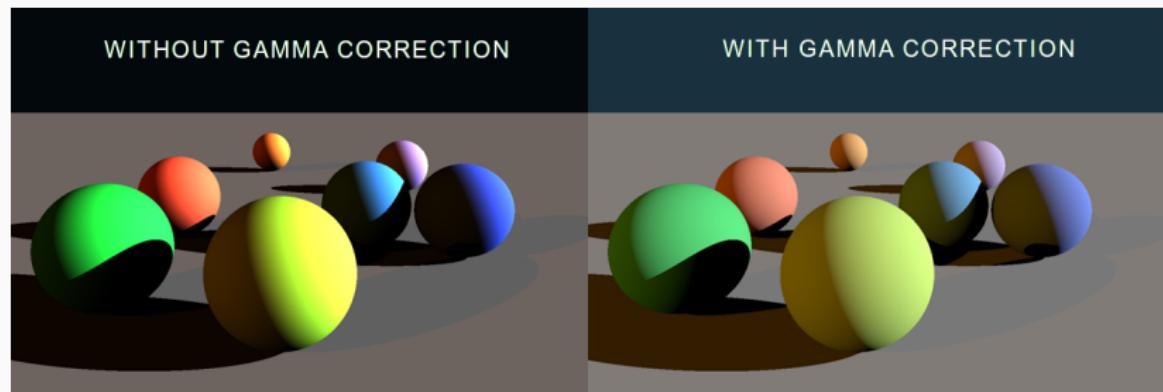
## Эффекты коррекции гаммы



# Эффекты коррекции гаммы



# Эффекты коррекции гаммы



# Эффекты коррекции гаммы

- Типичные эффекты гамма-коррекции:
  - Картинка становится ярче (чем темнее цвет, тем больше прирост яркости)
  - Картинка становится менее контрастной
  - Освещение выглядит правильнее
  - Переход от освещённой к неосвещённой области объекта выглядит более резким (как в реальности)
- Некоторые tone-mapping кривые уже включают в себя гамма-коррекцию!

- Проблема: после вычисления освещённости пикселя мы можем получить значения компонент цвета большие, чем 1, и при записи в пиксель они будут обрезаны до 1
- В реальном мире интенсивность света ничем не ограничена, и между солнечным днём и тёмной ночью может меняться на 15 порядков
- Объект может освещаться тусклым ambient освещением, а может сразу десятком ярких источников света

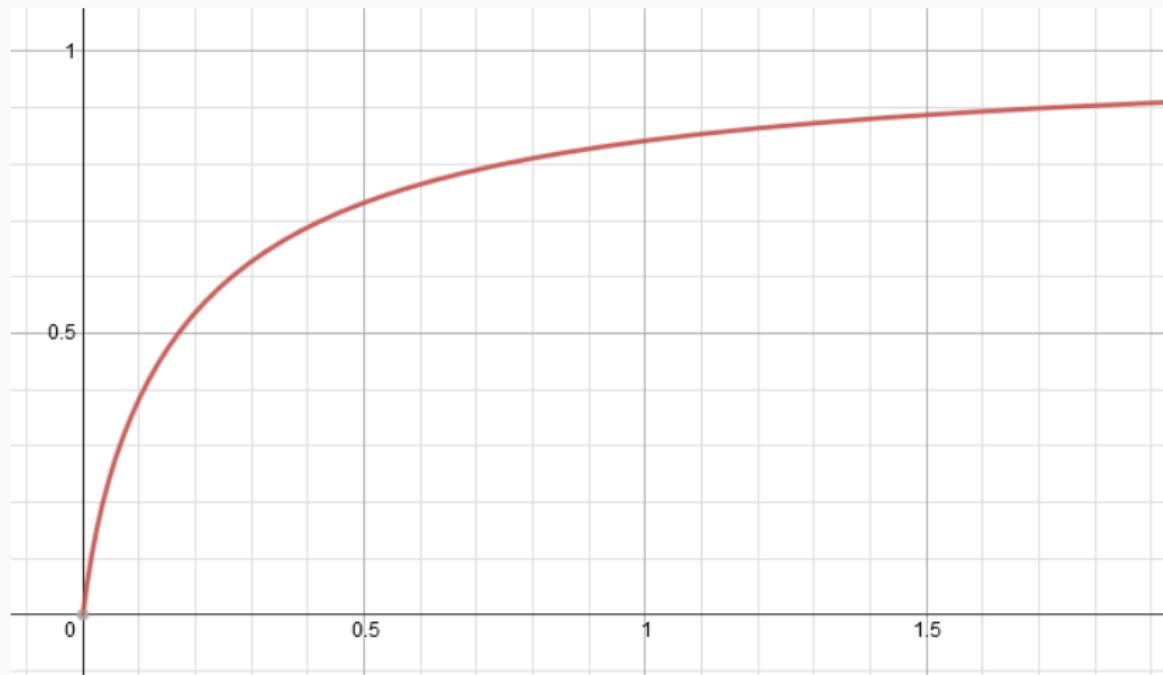
# HDR

- High Dynamic Range (HDR) – термин, означающий большой (не ограниченный  $[0, 1]$ ) диапазон интенсивностей света
- HDR текстура (например, environment map) – содержит значения, выходящие за диапазон  $[0, 1]$  (например, 32-bit floating-point)
- HDR рендеринг – позволяет отобразить HDR-освещение

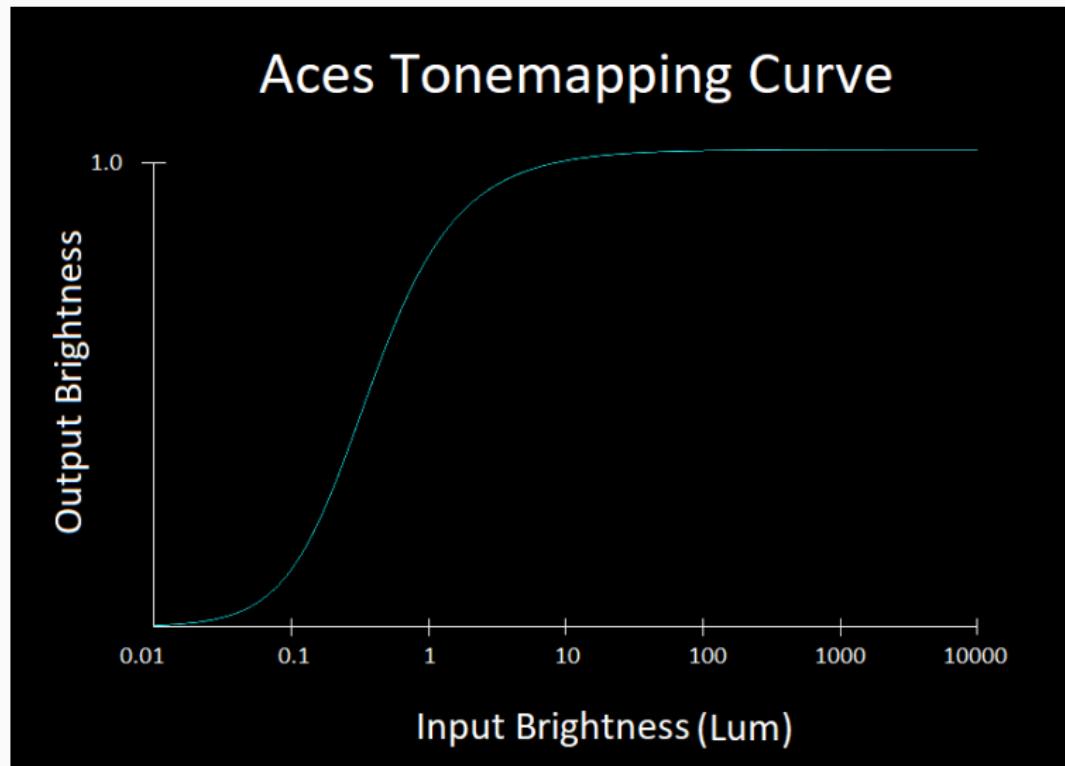
# Tone mapping

- Чтобы поддержать HDR рендеринг, нужно превратить диапазон освещённостей  $[0, \infty)$  в диапазон  $[0, 1]$
- В принципе, подойдёт любая монотонно возрастающая функция  $[0, \infty) \rightarrow [0, 1]$  – *tone-mapping curve*:
  - $x \mapsto \frac{x}{1+x}$  – Reinhard operator
  - $x \mapsto \arctan(x)$
  - $x \mapsto \frac{2}{1+e^{-x}} - 1$
- Иногда используют более сложные функции, взятые из кинематографа (filmic tone mapping)
- Часто всё равно ограничивают диапазон входных интенсивностей каким-то настраиваемым значением  $W$  (white), и делают нелинейное преобразование  $[0, W] \rightarrow [0, 1]$

# Reinhard operator



## Filmic tone mapping curve (ACES)



# Рациональная аппроксимация ACES

```
vec3 saturate(vec3 const & color)
{
    return clamp(color, vec3(0.0), vec3(1.0));
}

glm::vec3 aces_tonemap(glm::vec3 const & x)
{
    const float a = 2.51f;
    const float b = 0.03f;
    const float c = 2.43f;
    const float d = 0.59f;
    const float e = 0.14f;
    return saturate((x*(a*x+b))/(x*(c*x+d)+e));
}
```

- Мы будем использовать именно её