

# Компьютерная графика

Лекция 7: Распространение света, уравнение  
рендеринга, модели освещения

---

2025

## Почему важно освещение

- ≈90% информации человек воспринимает через зрение

## Почему важно освещение

- ≈90% информации человек воспринимает через зрение
- Зрение воспринимает то, как освещены окружающие объекты

## Почему важно освещение

- ≈90% информации человек воспринимает через зрение
- Зрение воспринимает то, как освещены окружающие объекты
- Человек с рождения учится воспринимать мир освещённым

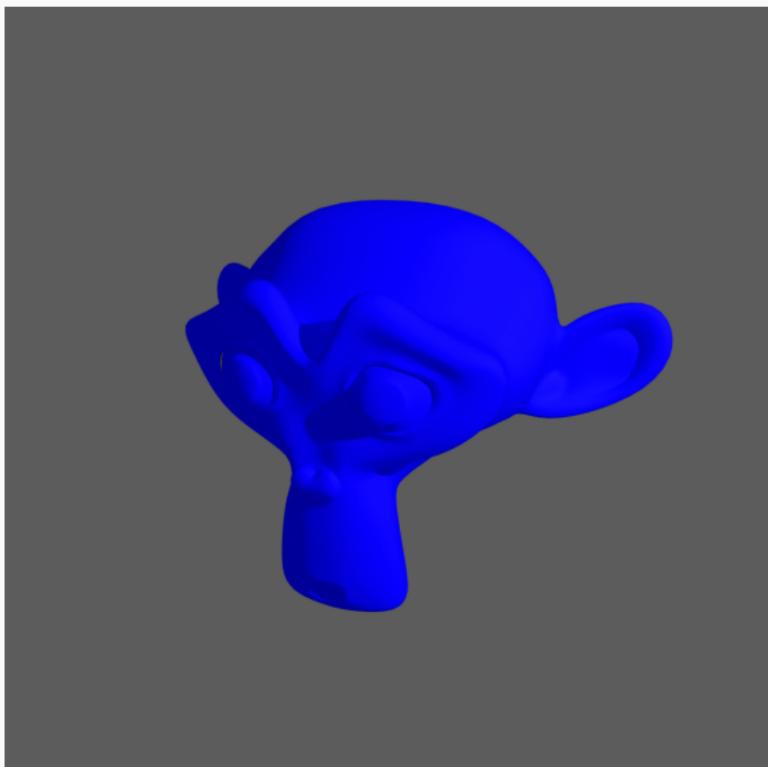
## Почему важно освещение

- Помогает понять форму объектов

## Почему важно освещение



## Почему важно освещение



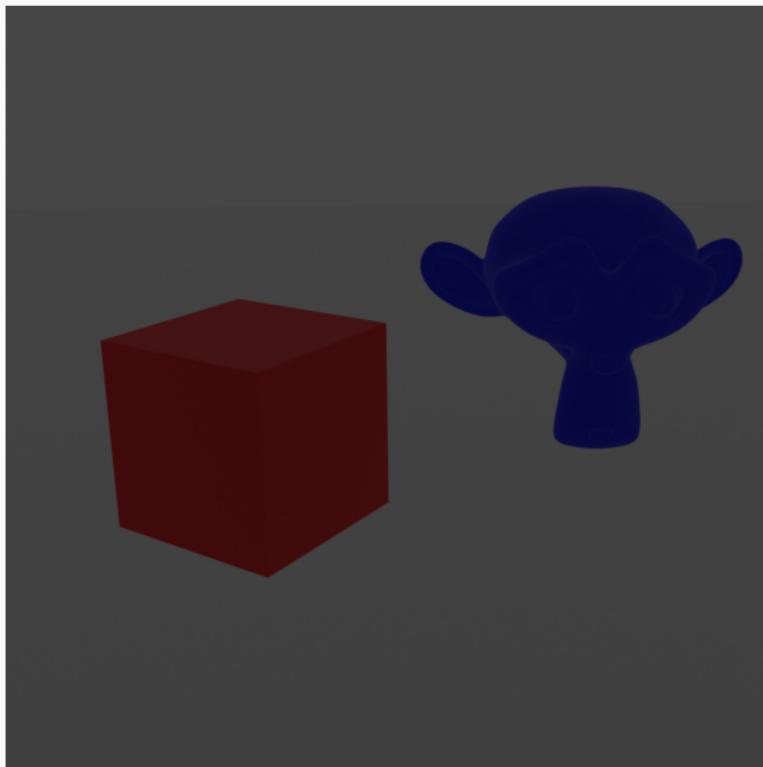
## Почему важно освещение

- Помогает понять форму объектов

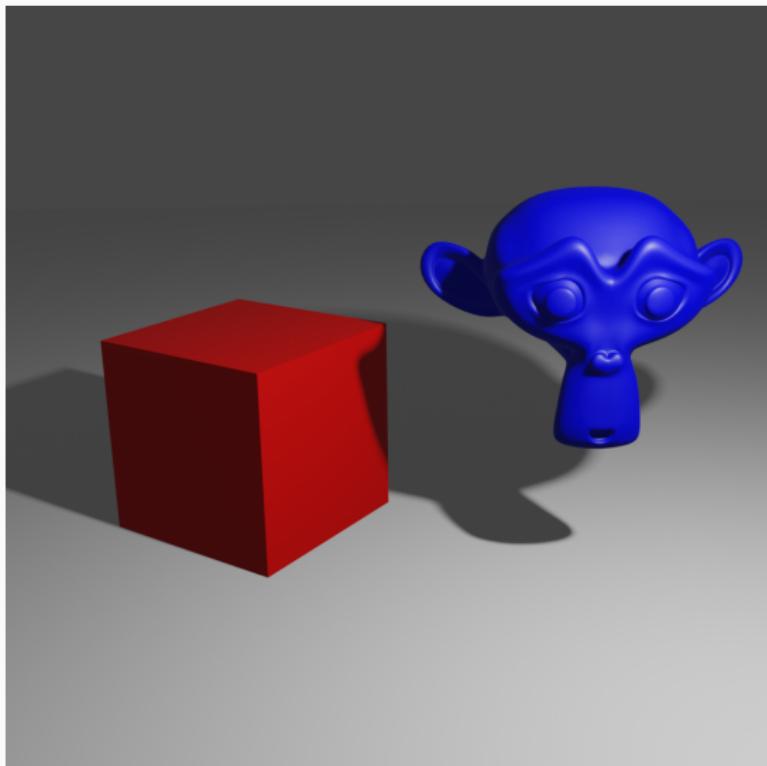
## Почему важно освещение

- Помогает понять форму объектов
- Помогает понять соотношение между объектами в пространстве

## Почему важно освещение



# Почему важно освещение



## Почему важно освещение

- Помогает понять форму объектов
- Помогает понять соотношение между объектами в пространстве

## Почему важно освещение

- Помогает понять форму объектов
- Помогает понять соотношение между объектами в пространстве
- Помогает понять материал объектов

## Почему важно освещение



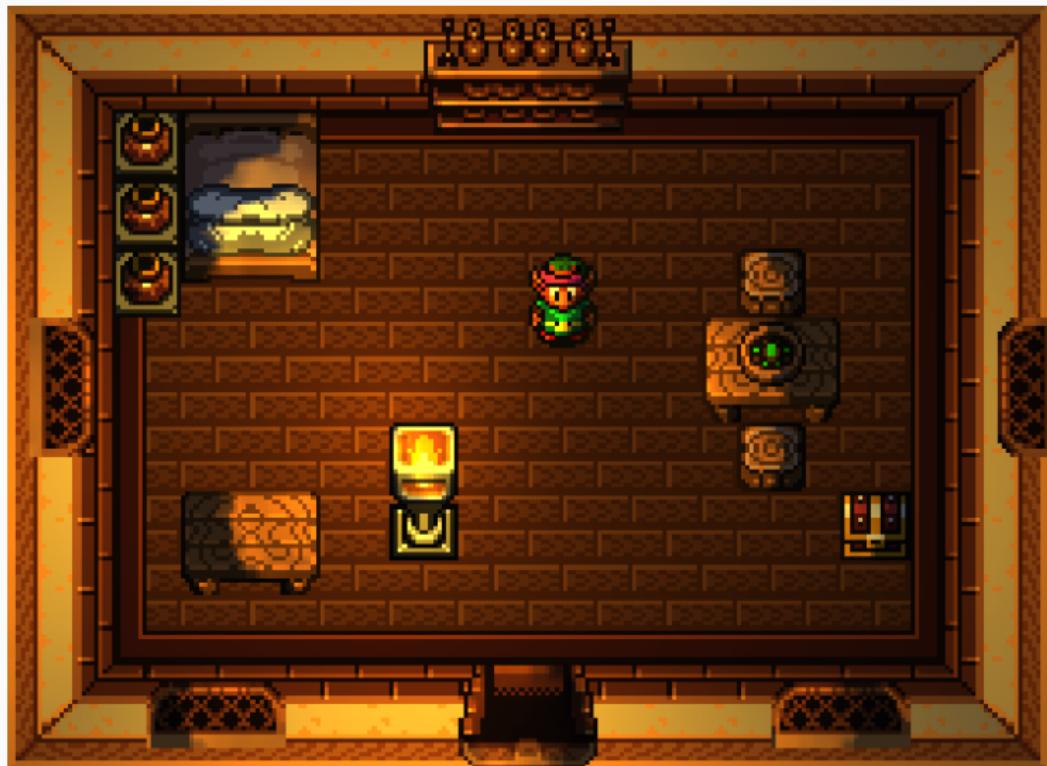
## Почему важно освещение

- Помогает понять форму объектов
- Помогает понять соотношение между объектами в пространстве
- Помогает понять материал объектов

## Почему важно освещение

- Помогает понять форму объектов
- Помогает понять соотношение между объектами в пространстве
- Помогает понять материал объектов
- Помогает создать атмосферу

## Почему важно освещение



## Почему важно освещение

- Помогает понять форму объектов
- Помогает понять соотношение между объектами в пространстве
- Помогает понять материал объектов
- Помогает создать атмосферу

## Почему важно освещение

- Помогает понять форму объектов
- Помогает понять соотношение между объектами в пространстве
- Помогает понять материал объектов
- Помогает создать атмосферу
- Помогает сделать объекты более привычными человеку

## Почему важно освещение



# Почему важно освещение



## Почему важно освещение

- Помогает понять форму объектов
- Помогает понять соотношение между объектами в пространстве
- Помогает понять материал объектов
- Помогает создать атмосферу
- Помогает сделать объекты более привычными человеку

# Почему важно освещение

- Помогает понять форму объектов
- Помогает понять соотношение между объектами в пространстве
- Помогает понять материал объектов
- Помогает создать атмосферу
- Помогает сделать объекты более привычными человеку
- ⇒ Какой бы графикой вы ни занимались, полезно добавить освещение

# Что такое свет

- Свет – электромагнитная волна

# Что такое свет

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

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

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

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

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

# 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)

# 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

- В графике обычно достаточно геометрической оптики – предела при  $\lambda \rightarrow 0$
- Свет распространяется прямыми лучами (исключение – граница раздела сред или неоднородные среды)

# 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 ( $c \rightarrow \infty$ )

# Свет

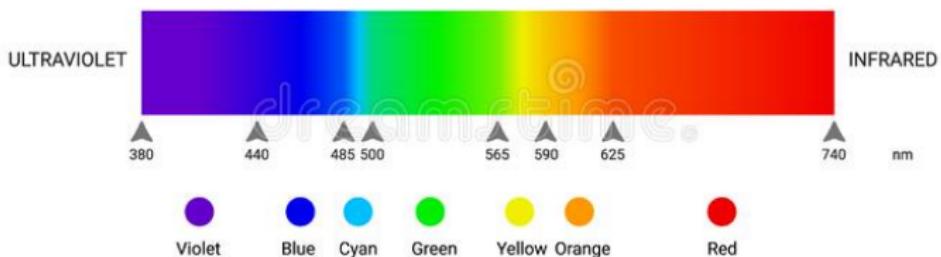
- В графике обычно достаточно геометрической оптики – предела при  $\lambda \rightarrow 0$
- Свет распространяется прямыми лучами (исключение – граница раздела сред или неоднородные среды)
- Свет распространяется бесконечно быстро ( $c \rightarrow \infty$ )
- Луч света разбивается в сумму монохроматических лучей, каждая имеет свою амплитуду (количество света)

# Свет

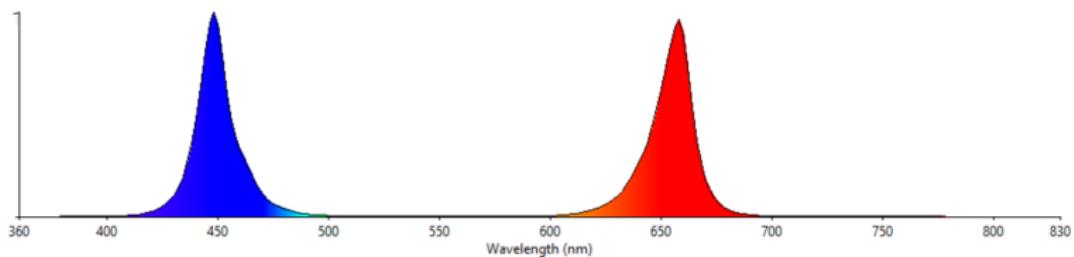
- В графике обычно достаточно геометрической оптики – предела при  $\lambda \rightarrow 0$
- Свет распространяется прямыми лучами (исключение – граница раздела сред или неоднородные среды)
- Свет распространяется бесконечно быстро ( $c \rightarrow \infty$ )
- Луч света разбивается в сумму монохроматических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

# Видимый спектр

## Visible spectrum



# Фиолетовый цвет



## Столкновение света с поверхностью

- Что происходит, когда луч света сталкивается с некой поверхностью?

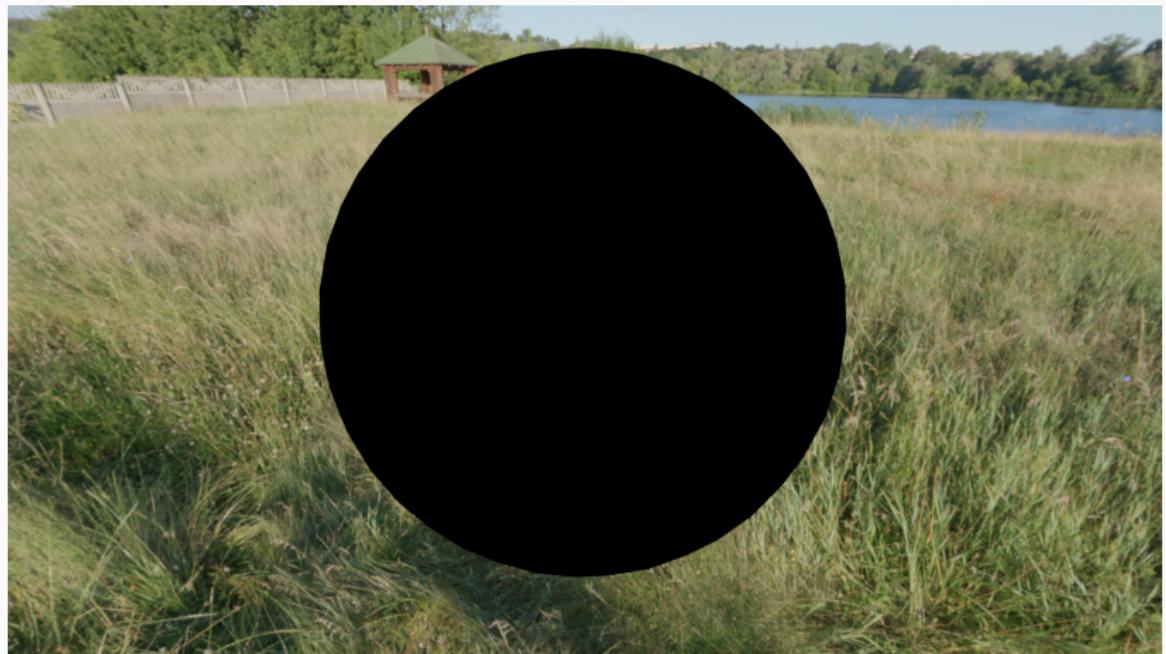
## Столкновение света с поверхностью

- Что происходит, когда луч света сталкивается с некой поверхностью?
- Зависит от её материала:

## Столкновение света с поверхностью

- Что происходит, когда луч света сталкивается с некой поверхностью?
- Зависит от её материала:
  - Абсолютно чёрное тело: поглощает весь свет

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

# 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

# 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

# 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

# Старое зеркало



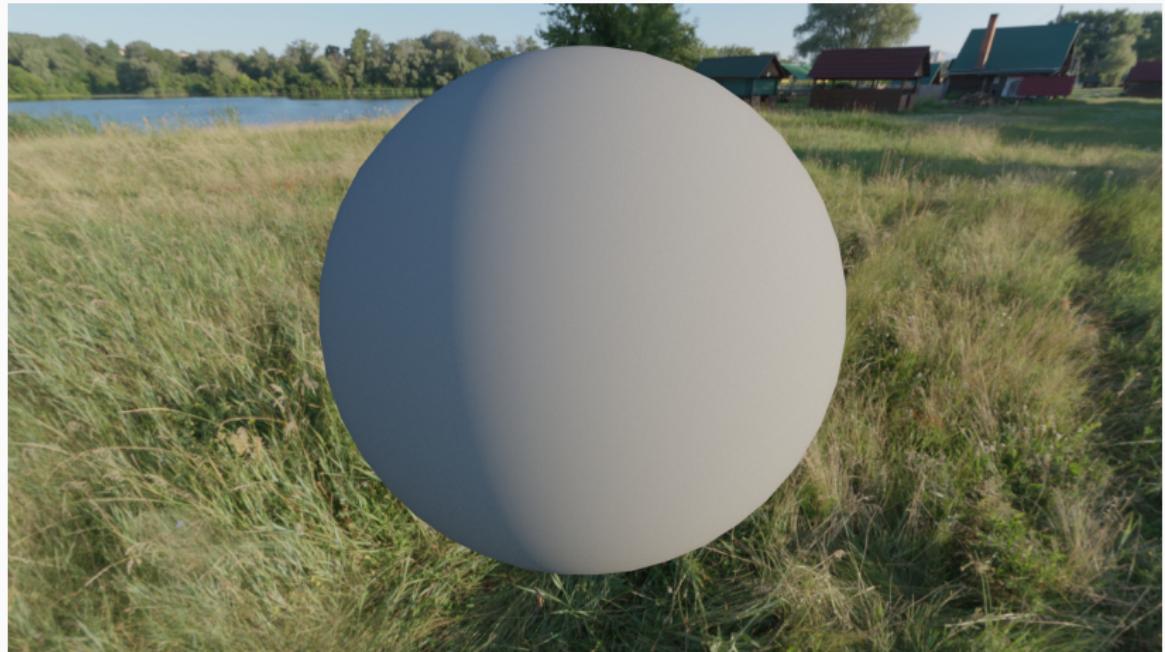
# Столкновение света с поверхностью

- 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

# 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

# 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

# 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
  - Вода, стекло: частично отражает, частично преломляет (закон Френеля)

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
  - Вода, стекло: частично отражает, частично преломляет (закон Френеля)

# 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
  - Вода, стекло: частично отражает, частично преломляет (закон Френеля)
  - Воск: частично отражает, частично преломляет, но свет заходит неглубоко (subsurface scattering)

Bock



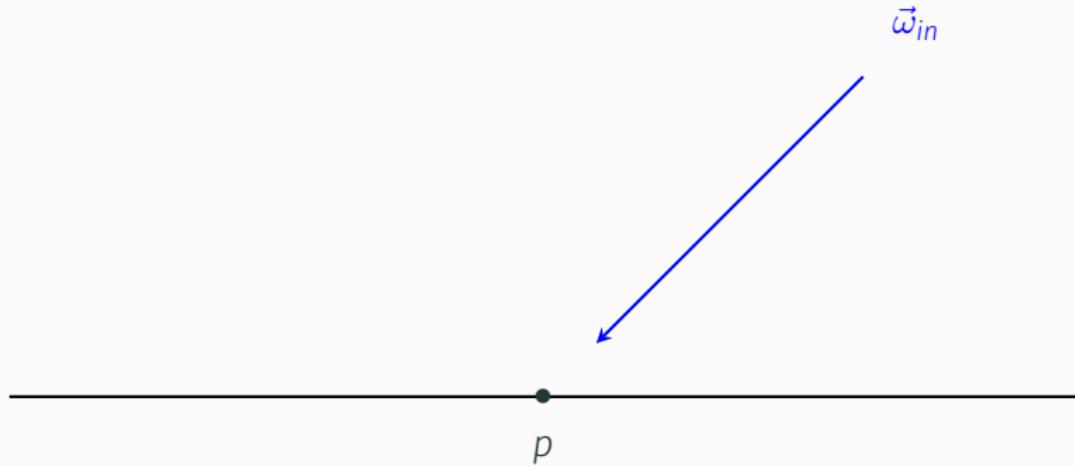
## Столкновение света с поверхностью

- Что происходит, когда луч света сталкивается с некой поверхностью?

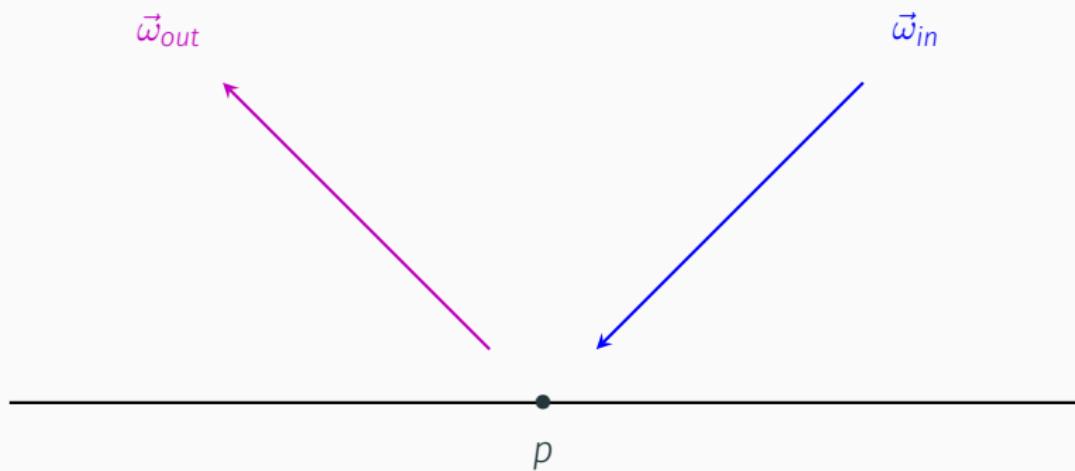
# Столкновение света с поверхностью

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

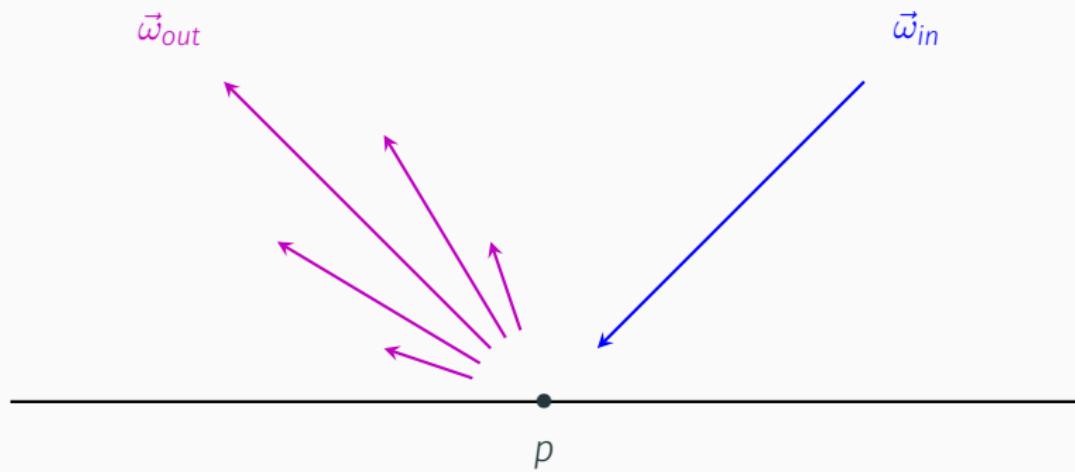
# Абсолютно чёрное тело



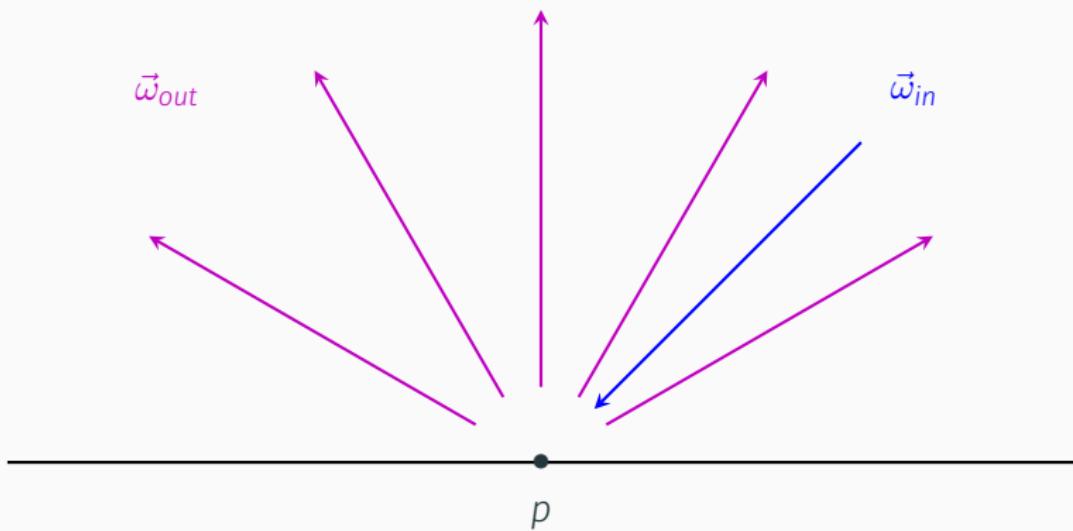
# Зеркало



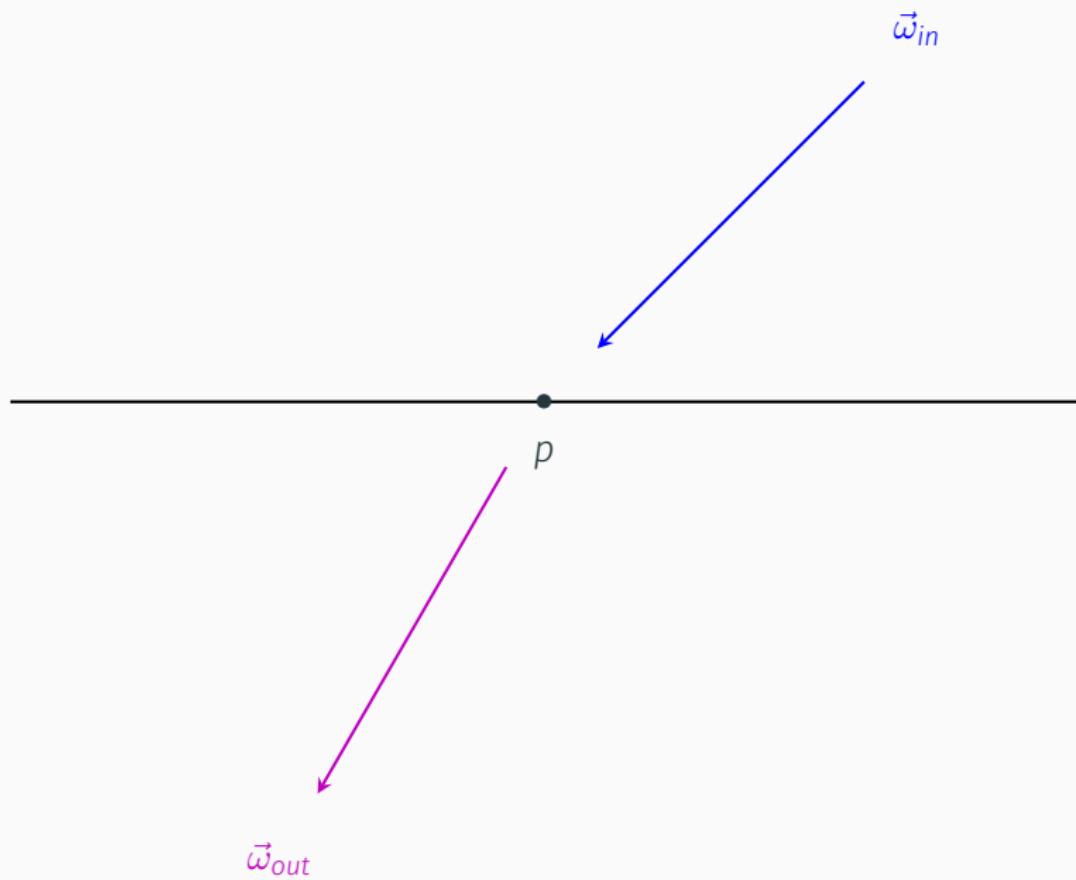
## Старое зеркало / лакированная поверхность



# Диффузная поверхность



# Стекло



## Уравнение рендеринга (Kajiya, 1986)

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

## Уравнение рендеринга (Kajiya, 1986)

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

- $I_{out}$  – количество света, выходящего из точки  $p$  в направлении  $\vec{\omega}_{out}$  с длиной волны  $\lambda$

## Уравнение рендеринга (Kajiya, 1986)

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

- $I_{out}$  – количество света, выходящего из точки  $p$  в направлении  $\vec{\omega}_{out}$  с длиной волны  $\lambda$
- $I_e$  – количество света, излучаемого поверхностью из точки  $p$  в направлении  $\vec{\omega}_{out}$  с длиной волны  $\lambda$

## Уравнение рендеринга (Kajiya, 1986)

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

- $I_{out}$  – количество света, выходящего из точки  $p$  в направлении  $\vec{\omega}_{out}$  с длиной волны  $\lambda$
- $I_e$  – количество света, излучаемого поверхностью из точки  $p$  в направлении  $\vec{\omega}_{out}$  с длиной волны  $\lambda$
- $I_{in}$  – количество света, приходящего в точку  $p$  из направления  $\vec{\omega}_{in}$  с длиной волны  $\lambda$

## Уравнение рендеринга (Kajiya, 1986)

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

- $I_{out}$  – количество света, выходящего из точки  $p$  в направлении  $\vec{\omega}_{out}$  с длиной волны  $\lambda$
- $I_e$  – количество света, излучаемого поверхностью из точки  $p$  в направлении  $\vec{\omega}_{out}$  с длиной волны  $\lambda$
- $I_{in}$  – количество 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  $\vec{\omega}_{out}$

## Уравнение рендеринга (Kajiya, 1986)

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

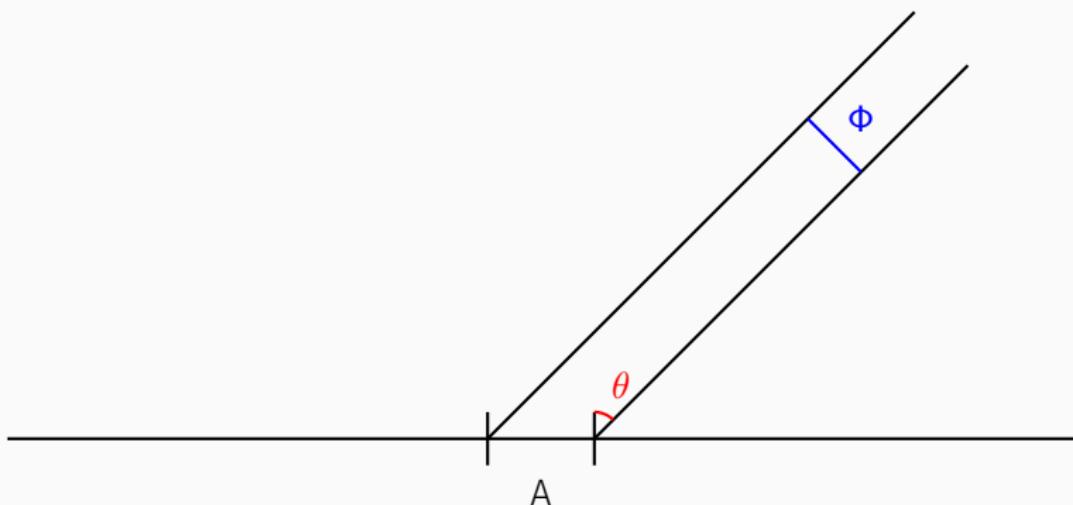
- $I_{out}$  – количество света, выходящего из точки  $p$  в направлении  $\vec{\omega}_{out}$  с длиной волны  $\lambda$
- $I_e$  – количество света, излучаемого поверхностью из точки  $p$  в направлении  $\vec{\omega}_{out}$  с длиной волны  $\lambda$
- $I_{in}$  – количество света, приходящего в точку  $p$  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  $p$

## Косинус угла падения

- Откуда взялся  $\vec{\omega}_{in} \cdot \vec{n}$ ?

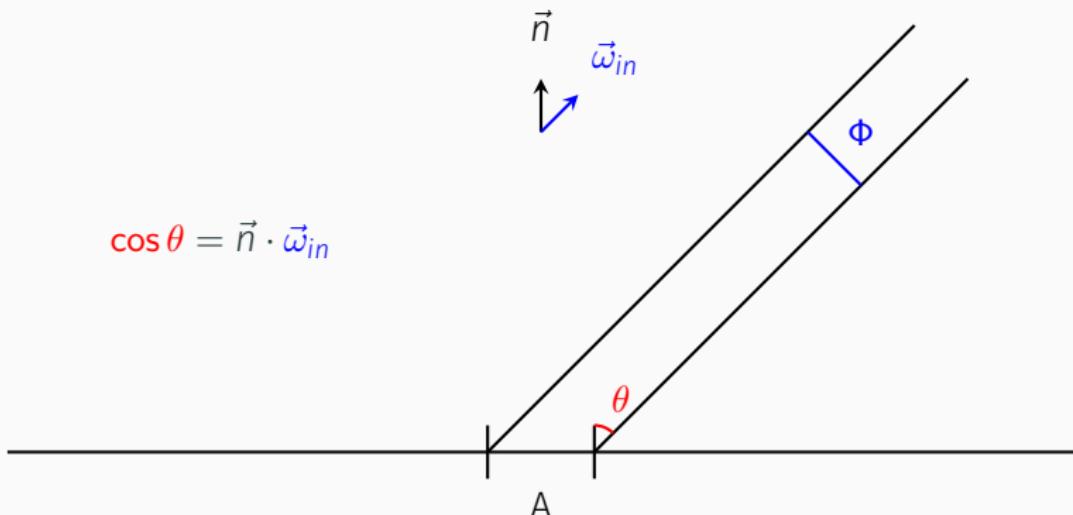
## Косинус угла падения

- Откуда взялся  $\vec{\omega}_{in} \cdot \vec{n}$ ?
- 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



## Косинус угла падения

- Откуда взялся  $\vec{\omega}_{in} \cdot \vec{n}$ ?
- 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



# Материал

- $f$  определяет материал объекта

# Материал

- $f$  определяет материал объекта
- Абсолютно чёрное тело:  $f(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) = 0$

# 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:  $R_{\vec{n}}(\vec{\omega}) = 2\vec{n} \cdot (\vec{n} \cdot \vec{\omega}) - \vec{\omega}$

# 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:  $f(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) = (R_{\vec{n}}(\vec{\omega}_{in}) \cdot \vec{\omega}_{out})^7$

# 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

# Цвет

- Зависимость  $f$  от  $\lambda$  обеспечивает цвет объектов
- Мы видим свет, отражённый объектом

# 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

## 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

## 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

# BSDF

- BSDF неотрицательна:  $f(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) \geq 0$

# 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

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

# 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

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

- Helmholtz reciprocity:

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

# 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

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

- Helmholtz reciprocity:

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

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

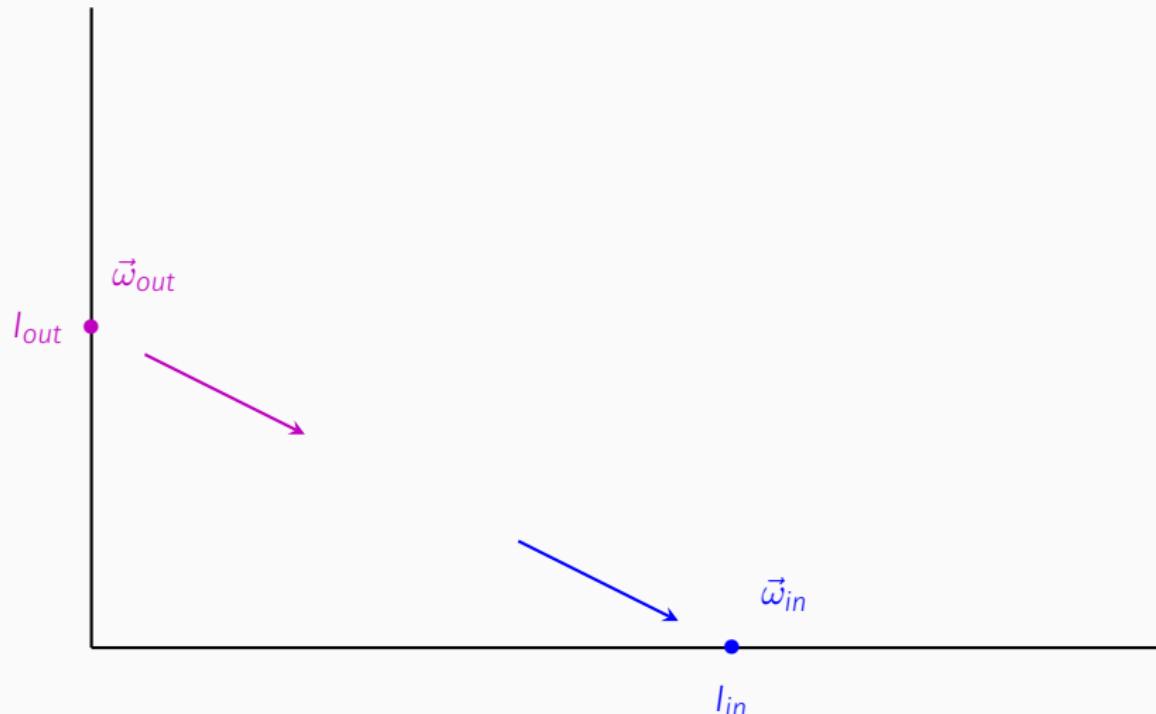
$$\mu_i \geq 0, \sum \mu_i \leq 1 \Rightarrow \sum \mu_i f_i$$

# Уравнение рендеринга

- Откуда взять  $I_{in}$ ?

# Уравнение рендеринга

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



# Уравнение рендеринга

- Интегральное уравнение для каждой точки каждой поверхности сцены

# Уравнение рендеринга

- Интегральное уравнение для каждой точки каждой поверхности сцены
- Включает зависимость от материала объектов в каждой точке

# 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

# 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**)

# 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**)
- Задача решения этого уравнения называется *Global Illumination (GI)*

# Уравнение рендеринга

- Как описывать геометрию?

# Уравнение рендеринга

- Как описывать геометрию?
- Как описывать материал (BRDF)?

# Уравнение рендеринга

- Как описывать геометрию?
- Как описывать материал (BRDF)?
- Как решать само уравнение?

# Уравнение рендеринга

- Как описывать геометрию?
- Как описывать материал (BRDF)?
- Как решать само уравнение?
- Как это выглядит в коде?

# Как описывать геометрию

- Зависит от используемого метода решения уравнения

## Как описывать геометрию

- Зависит от используемого метода решения уравнения
- Самый частый вариант – триангуляция поверхности

## Как описывать геометрию

- Зависит от используемого метода решения уравнения
- Самый частый вариант – триангуляция поверхности
- Параметрическая поверхность, сплайны, NURBS

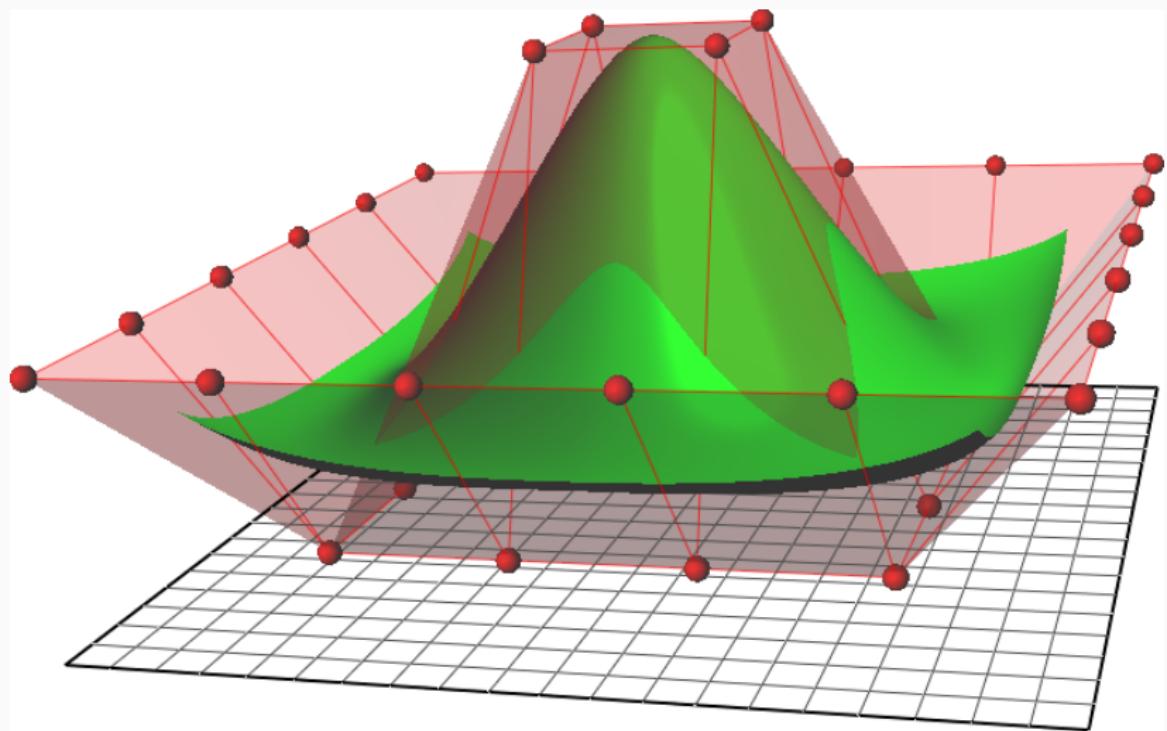
# Как описывать геометрию

- Зависит от используемого метода решения уравнения
- Самый частый вариант – триангуляция поверхности
- Параметрическая поверхность, сплайны, NURBS
- Неявная поверхность  $f(p) = 0$ , по  $\nabla f$  можно получить нормаль к поверхности

# Как описывать геометрию

- Зависит от используемого метода решения уравнения
- Самый частый вариант – триангуляция поверхности
- Параметрическая поверхность, сплайны, NURBS
- Неявная поверхность  $f(p) = 0$ , по  $\nabla f$  можно получить нормаль к поверхности
- Подвид неявных поверхностей – *signed distance function/field* (SDF) – функция  $f(p)$ , возвращающая расстояние до поверхности, удобна для некоторых алгоритмов (напр. raymarching)

# NURBS



## Как решать уравнение рендеринга: алгоритм radiosity

- Взять в качестве переменных среднюю освещённость в каждой вершине сцены, интерполировать освещённость в треугольниках, свести к системе линейных уравнений

## Как решать уравнение рендеринга: 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

## Как решать уравнение рендеринга: 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
- Обычно решается итеративными методами (Якоби, Гаусса-Зейделя)

# Как решать уравнение рендеринга: 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
- Обычно решается итеративными методами (Якоби, Гаусса-Зейделя)
- Каждая итерация учитывает ещё одно возможное отражение света (bounce) перед попаданием в камеру

# Как решать уравнение рендеринга: 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
- Обычно решается итеративными методами (Якоби, Гаусса-Зейделя)
- Каждая итерация учитывает ещё одно возможное отражение света (bounce) перед попаданием в камеру
- Плохо подходит для динамических и real-time сцен

# Как решать уравнение рендеринга: raytracing + monte-carlo integration

- Посыпать лучи из камеры, симулируя обратное распространение света (*raytracing*)
- В точке пересечения луча с объектом сцены аппроксимировать интеграл, пуская отражённые лучи в случайном направлении (*monte-carlo integration*)

# Как решать уравнение рендеринга: raytracing + monte-carlo integration

- Посыпать лучи из камеры, симулируя обратное распространение света (*raytracing*)
- В точке пересечения луча с объектом сцены аппроксимировать интеграл, пуская отражённые лучи в случайном направлении (*monte-carlo integration*)
- Основной алгоритм, использующийся при offline рендеринге

# Как решать уравнение рендеринга: raytracing + monte-carlo integration

- Посыпать лучи из камеры, симулируя обратное распространение света (*raytracing*)
- В точке пересечения луча с объектом сцены аппроксимировать интеграл, пуская отражённые лучи в случайном направлении (*monte-carlo integration*)
- Основной алгоритм, использующийся при offline рендеринге
- Требует как-то ограничить количество отражений (*bounces*)

# Как решать уравнение рендеринга: raytracing + monte-carlo integration

- Посыпать лучи из камеры, симулируя обратное распространение света (*raytracing*)
- В точке пересечения луча с объектом сцены аппроксимировать интеграл, пуская отражённые лучи в случайном направлении (*monte-carlo integration*)
- Основной алгоритм, использующийся при offline рендеринге
- Требует как-то ограничить количество отражений (*bounces*)
- Требует построить некую структуру данных (BVH – *bounding volume hierarchy*) для ускорения поиска пересечений лучей со сценой (RTX делают именно это)

# Как решать уравнение рендеринга: raytracing + monte-carlo integration

- Картинка получается очень шумной (зернистой) из-за случайности, очень много исследований направлено на борьбу с шумом (*importance sampling, resampling, reservoir sampling*)

# Как решать уравнение рендеринга: raytracing + monte-carlo integration

- Картинка получается очень шумной (зернистой) из-за случайности, очень много исследований направлено на борьбу с шумом (*importance sampling, resampling, reservoir sampling*)
- Хочется переиспользовать результаты вычислений между кадрами  $\implies$  нужно эти результаты сохранять в

# Как решать уравнение рендеринга: raytracing + monte-carlo integration

- Картинка получается очень шумной (зернистой) из-за случайности, очень много исследований направлено на борьбу с шумом (*importance sampling, resampling, reservoir sampling*)
- Хочется переиспользовать результаты вычислений между кадрами  $\implies$  нужно эти результаты сохранять в
  - Пикселях

# Как решать уравнение рендеринга: raytracing + monte-carlo integration

- Картинка получается очень шумной (зернистой) из-за случайности, очень много исследований направлено на борьбу с шумом (*importance sampling, resampling, reservoir sampling*)
- Хочется переиспользовать результаты вычислений между кадрами  $\Rightarrow$  нужно эти результаты сохранять в
  - Пикселях
  - Точках поверхностей (*surflets*)

# Как решать уравнение рендеринга: raytracing + monte-carlo integration

- Картинка получается очень шумной (зернистой) из-за случайности, очень много исследований направлено на борьбу с шумом (*importance sampling, resampling, reservoir sampling*)
- Хочется переиспользовать результаты вычислений между кадрами  $\Rightarrow$  нужно эти результаты сохранять в
  - Пикселях
  - Точких поверхностей (*surflets*)
  - Точких пространства (*voxels*)

# Как решать уравнение рендеринга: raytracing + monte-carlo integration

- Картинка получается очень шумной (зернистой) из-за случайности, очень много исследований направлено на борьбу с шумом (*importance sampling, resampling, reservoir sampling*)
- Хочется переиспользовать результаты вычислений между кадрами  $\Rightarrow$  нужно эти результаты сохранять в
  - Пикселях
  - Точких поверхностей (*surflets*)
  - Точких пространства (*voxels*)
- Использование для real-time – предмет активного исследования (см. *ReSTIR GI*)

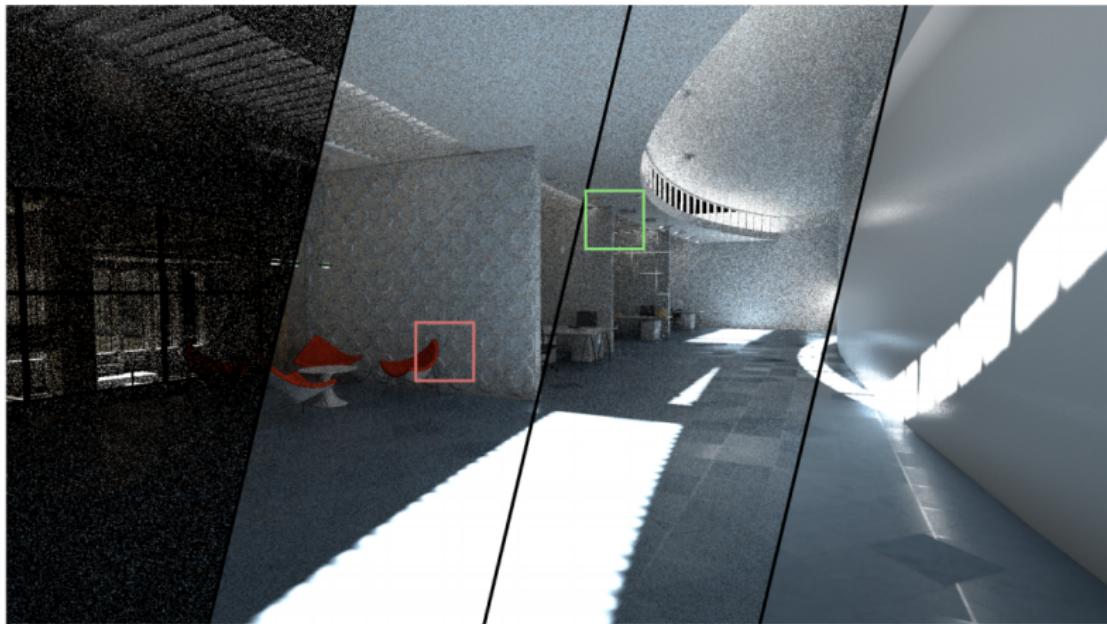
# ReSTIR GI

Path Traced  
(1spp) 8.0 ms  
0.265 MSE

ReSTIR GI  
(biased) 8.9 ms  
0.0175 MSE (15.1x)

ReSTIR GI  
(unbiased) 9.6 ms  
0.0224 MSE (11.8x)

Reference



## Как решать уравнение рендеринга: real-time подход

- Набор эвристических аппроксимаций

## Как решать уравнение рендеринга: real-time подход

- Набор эвристических аппроксимаций
- Прямое освещение:  $\vec{\omega}_{in}$  – направление на источник света,  
 $\vec{\omega}_{out}$  – направление на камеру

# Как решать уравнение рендеринга: real-time подход

- Набор эвристических аппроксимаций
- Прямое освещение:  $\vec{\omega}_{in}$  – направление на источник света,  $\vec{\omega}_{out}$  – направление на камеру
- Тени: учитывается, ‘виден’ ли объект источником света, или находится в тени

# Как решать уравнение рендеринга: real-time подход

- Набор эвристических аппроксимаций
- Прямое освещение:  $\vec{\omega}_{in}$  – направление на источник света,  $\vec{\omega}_{out}$  – направление на камеру
- Тени: учитывается, ‘виден’ ли объект источником света, или находится в тени
- Отражения (точные / размытые)

# Как решать уравнение рендеринга: real-time подход

- Набор эвристических аппроксимаций
- Прямое освещение:  $\vec{\omega}_{in}$  – направление на источник света,  $\vec{\omega}_{out}$  – направление на камеру
- Тени: учитывается, ‘виден’ ли объект источником света, или находится в тени
- Отражения (точные / размытые)
- Непрямое освещение (многократно отражённый свет) почти всегда игнорируется или эмулируется

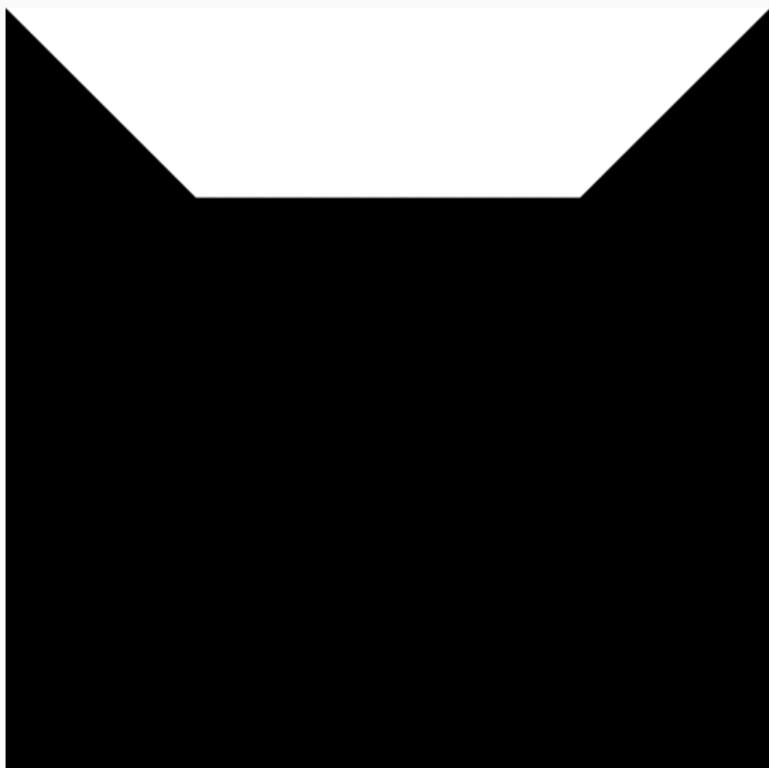
# Как решать уравнение рендеринга: real-time подход

- Набор эвристических аппроксимаций
- Прямое освещение:  $\vec{\omega}_{in}$  – направление на источник света,  $\vec{\omega}_{out}$  – направление на камеру
- Тени: учитывается, ‘виден’ ли объект источником света, или находится в тени
- Отражения (точные / размытые)
- Непрямое освещение (многократно отражённый свет) почти всегда игнорируется или эмулируется
  - Ambient освещение – свет, приходящий ‘отовсюду’ (с неба, отражённый от стен, и т.п.)

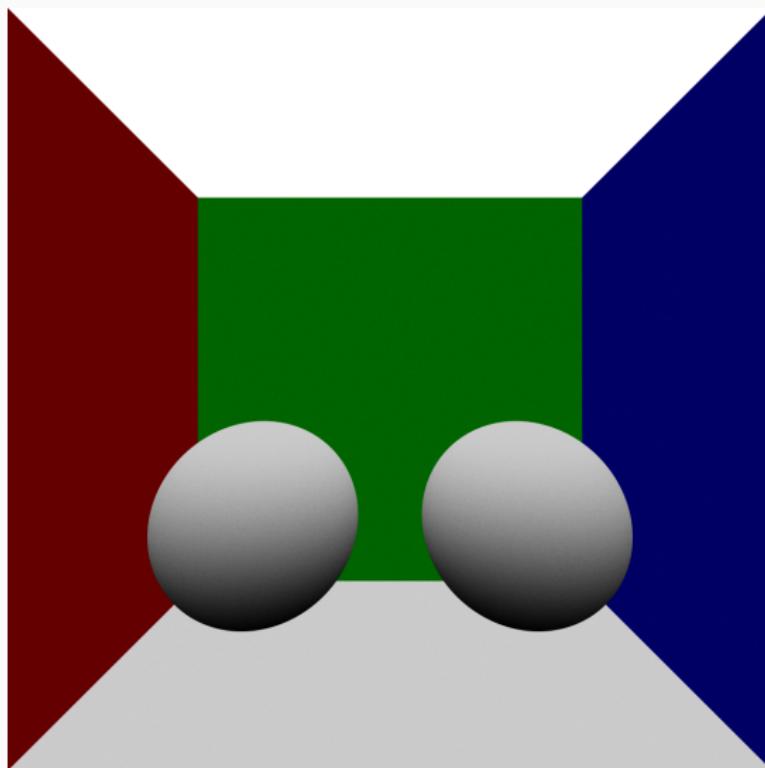
# Как решать уравнение рендеринга: real-time подход

- Набор эвристических аппроксимаций
- Прямое освещение:  $\vec{\omega}_{in}$  – направление на источник света,  $\vec{\omega}_{out}$  – направление на камеру
- Тени: учитывается, ‘виден’ ли объект источником света, или находится в тени
- Отражения (точные / размытые)
- Непрямое освещение (многократно отражённый свет) почти всегда игнорируется или эмулируется
  - Ambient освещение – свет, приходящий ‘отовсюду’ (с неба, отражённый от стен, и т.п.)
  - Ambient occlusion (AO) – затенение частей объекта, куда доходит меньше ambient света

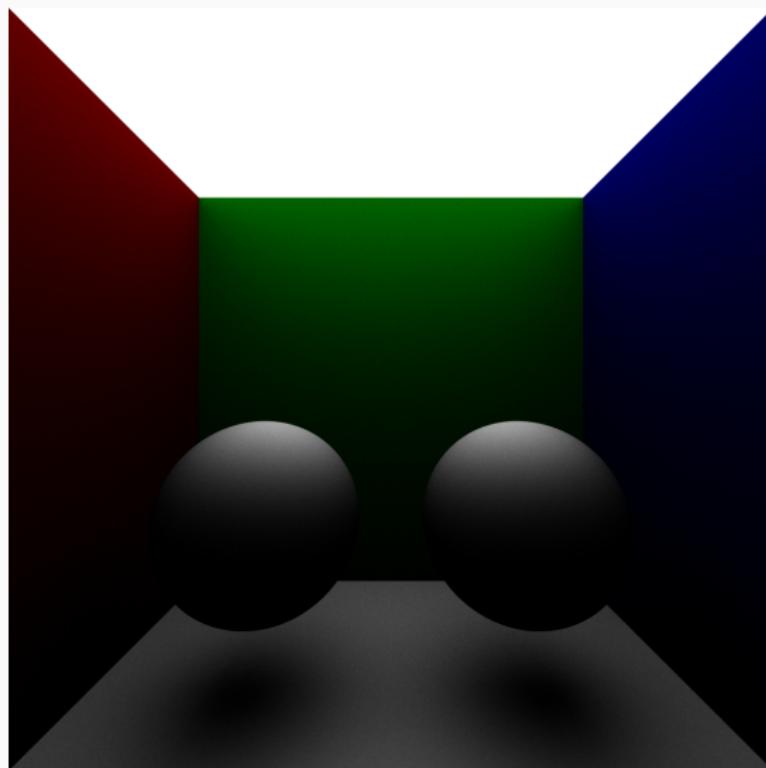
Только источник света



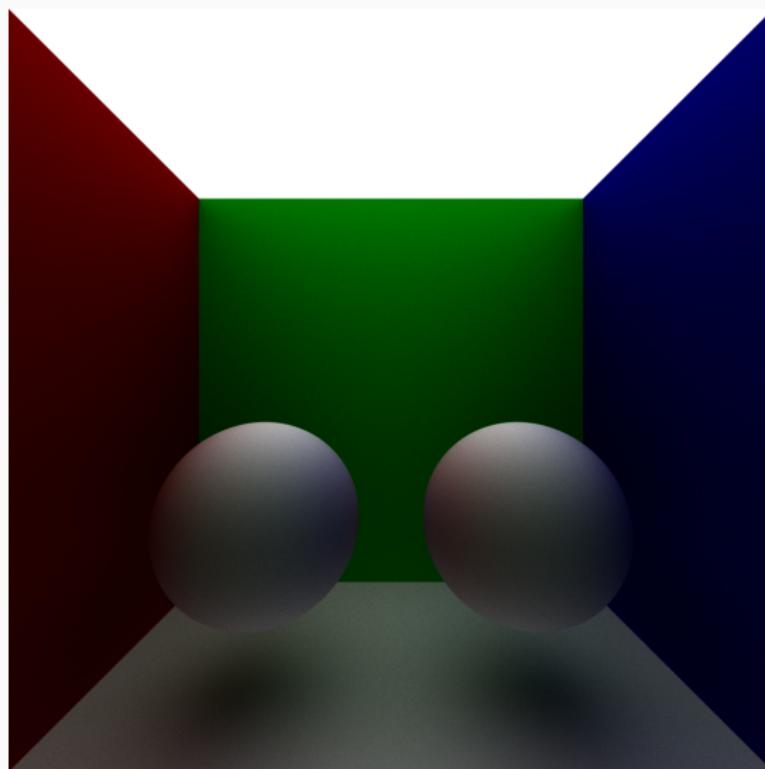
Только прямое освещение



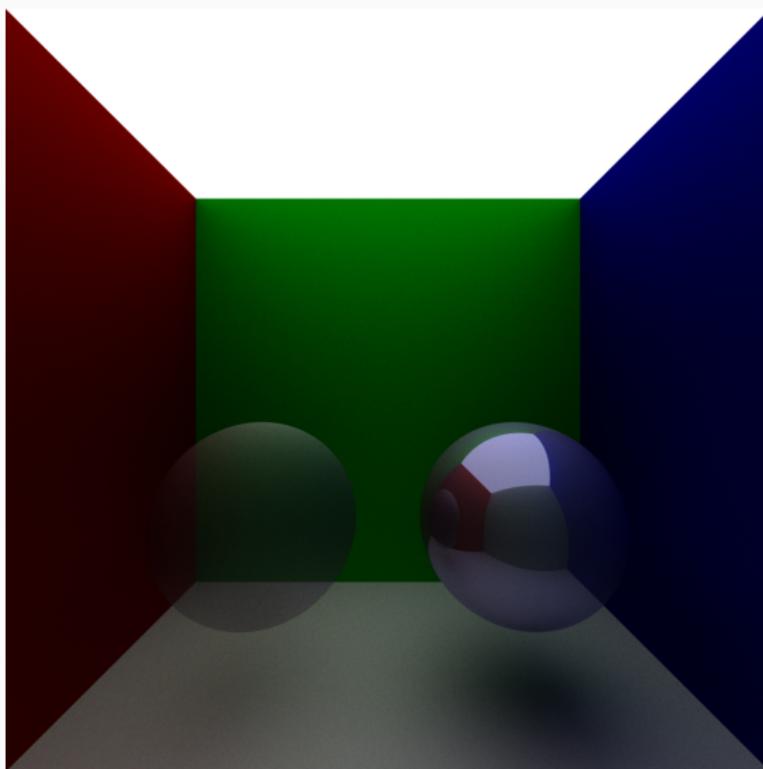
## Прямое освещение и тени



## Прямое и непрямое освещение, тени



## Прозрачность и отражение



## Откуда взять материал (BRDF)

- Эвристические формулы (Phong)

## Откуда взять материал (BRDF)

- Эвристические формулы (Phong)
- Вывод из предположений о структуре поверхности (*microfacets*)

## Откуда взять материал (BRDF)

- Эвристические формулы (Phong)
- Вывод из предположений о структуре поверхности (*microfacets*)
- Прямое измерение (MERL 100)

## Откуда взять материал (BRDF)

- Эвристические формулы (Phong)
- Вывод из предположений о структуре поверхности (*microfacets*)
- Прямое измерение (MERL 100)
- Симуляция

# Как описать материал (BRDF)

- Явная формула (точная, или аппроксимация) с набором параметров материала

# Как описать материал (BRDF)

- Явная формула (точная, или аппроксимация) с набором параметров материала
- Набор предподсчитанных значений, как-то записанных в текстуру

## Расчёт прямого освещения

- Почти всегда происходит во фрагментном шейдере

## Расчёт прямого освещения

- Почти всегда происходит во фрагментном шейдере
- Вычисления используют `vec3` для работы с цветом в `RGB`

# Расчёт прямого освещения

- Почти всегда происходит во фрагментном шейдере
- Вычисления используют `vec3` для работы с цветом в `RGB`
- Входные данные:

# Расчёт прямого освещения

- Почти всегда происходит во фрагментном шейдере
- Вычисления используют `vec3` для работы с цветом в `RGB`
- Входные данные:
  - Нормаль к поверхности

# Расчёт прямого освещения

- Почти всегда происходит во фрагментном шейдере
- Вычисления используют `vec3` для работы с цветом в `RGB`
- Входные данные:
  - Нормаль к поверхности
  - Параметры материала (цвет, гладкость, и т.п.)

# Расчёт прямого освещения

- Почти всегда происходит во фрагментном шейдере
- Вычисления используют `vec3` для работы с цветом в `RGB`
- Входные данные:
  - Нормаль к поверхности
  - Параметры материала (цвет, гладкость, и т.п.)
  - Источники света

# Расчёт прямого освещения

- Почти всегда происходит во фрагментном шейдере
- Вычисления используют `vec3` для работы с цветом в `RGB`
- Входные данные:
  - Нормаль к поверхности
  - Параметры материала (цвет, гладкость, и т.п.)
  - Источники света
- Полная освещённость = ambient освещение + вклад от каждого источника света

## Ambient освещение

- Свет, приходящий ‘отовсюду’

## Ambient освещение

- Свет, приходящий 'отовсюду'
  - На улице: небо

# Ambient освещение

- Свет, приходящий 'отовсюду'
  - На улице: небо
  - В помещении: диффузное отражение от стен, пола, потолка, других объектов

# Ambient освещение

- Свет, приходящий 'отовсюду'
  - На улице: небо
  - В помещении: диффузное отражение от стен, пола, потолка, других объектов
- Задаётся своим цветом

# Ambient освещение

- Свет, приходящий 'отовсюду'
  - На улице: небо
  - В помещении: диффузное отражение от стен, пола, потолка, других объектов
- Задаётся своим цветом
- Обычно добавляется чтобы точки, куда не попал другой свет, не выглядели совсем чёрными

## Источники света

- Два самых часто используемых типа: направленные и точечные

# Источники света

- Два самых часто используемых типа: направленные и точечные
- Направленные:

# Источники света

- Два самых часто используемых типа: направленные и точечные
- Направленные:
  - Задаются направлением и яркостью

# Источники света

- Два самых часто используемых типа: направленные и точечные
- Направленные:
  - Задаются направлением и яркостью
  - Моделируют удалённые источники – солнце, луна

# Источники света

- Два самых часто используемых типа: направленные и точечные
- Направленные:
  - Задаются направлением и яркостью
  - Моделируют удалённые источники – солнце, луна
- Точечные:

# Источники света

- Два самых часто используемых типа: направленные и точечные
- Направленные:
  - Задаются направлением и яркостью
  - Моделируют удалённые источники – солнце, луна
- Точечные:
  - Задаются расположением и функцией зависимости яркости от расстояния

# Источники света

- Два самых часто используемых типа: направленные и точечные
- Направленные:
  - Задаются направлением и яркостью
  - Моделируют удалённые источники – солнце, луна
- Точечные:
  - Задаются расположением и функцией зависимости яркости от расстояния
  - Моделируют (относительно) близкие источники – костёр, свеча, лампа

## Точечный источник света

- Яркость убывает с расстоянием

## Точечный источник света

- Яркость убывает с расстоянием
- Часто берут такую функцию затухания (*attenuation*):

$$\frac{1}{C_0 + C_1r + C_2r^2}$$

## Диффузная (ламбертова) BRDF

$$f(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) = \frac{C(\lambda)}{\pi}$$

- Отражает свет во всех направлениях одинаково
- $C(\lambda) \in [0, 1]$  – ‘альбедо’, определяет цвет
- $\pi$  – нормировочный множитель, часто входит в  $C(\lambda)$  (напр. в текстуру)

## Отражающая (зеркальная, specular) BRDF

$$f(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) = C(\lambda) (R_{\vec{n}}(\vec{\omega}_{in}) \cdot \vec{\omega}_{out})^{power}$$

- Отражает больше света в направлении отражённого луча:  
 $R_{\vec{n}}(\vec{\omega}) = 2\vec{n} \cdot (\vec{n} \cdot \vec{\omega}) - \vec{\omega}$
- Параметр *power* определяет ‘размытость’ отражения: чем он больше, тем оно менее размытое
  - Часто вычисляется на основе более удобного параметра *roughness*, например  $power = \frac{1}{roughness^2} - 1$
- Сложно правильно нормировать

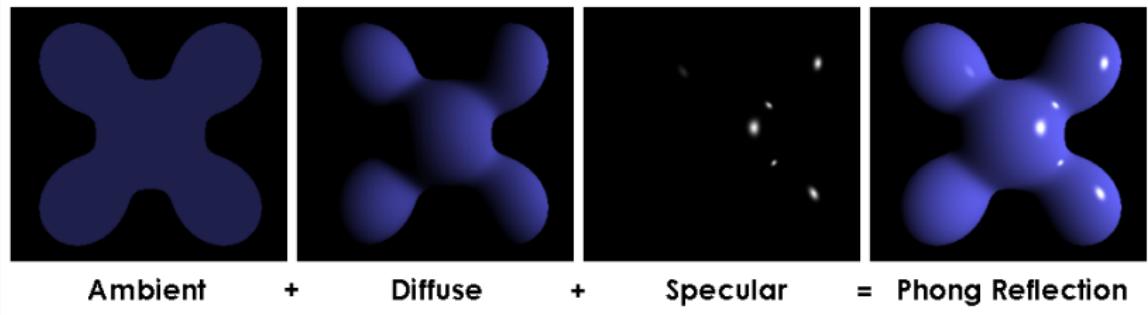
## Модель Фонга (Phong)

- Вклад источника света = diffuse + specular

## Модель Фонга (Phong)

- Вклад источника света = diffuse + specular
- Полная освещённость = ambient + diffuse + specular

# Модель Фонга (Phong)



# Модель Фонга + направленный источник света

```
uniform vec3 ambient_light_color;
uniform vec3 light_direction;
uniform vec3 light_color;
uniform vec3 view_direction;

in vec3 normal;
in vec3 ambient_albedo;
in vec3 diffuse_albedo;
in vec3 specular_albedo;
in float specular_power;

layout (location = 0) out vec4 out_color;

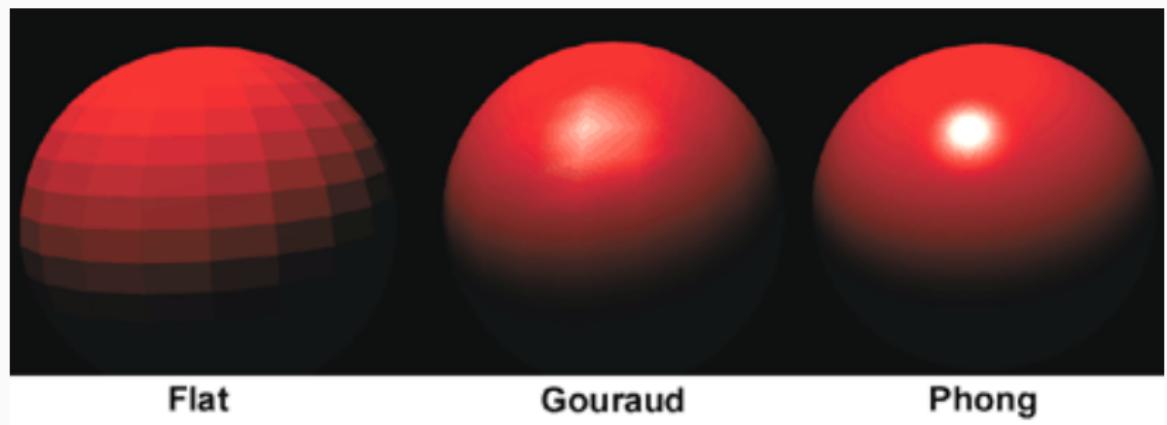
void main() {
    float cosine = dot(normal, light_direction);
    float light_factor = max(0.0, cosine);
    vec3 reflected_direction = 2.0 * normal * cosine - light_direction;

    vec3 ambient = ambient_light_color * ambient_albedo;
    vec3 diffuse = diffuse_albedo * light_color * light_factor;
    vec3 specular = specular_albedo * pow(max(0.0, dot(reflected_direction,
        view_direction)), specular_power);

    vec3 result_color = ambient + diffuse + specular;
    out_color = vec4(result_color, 1.0);
}
```

## Модель Гуро

- То же самое, что и модель Фонга, но в вершинном шейдере
- Результирующий цвет интерполируется между пикселями
- Выглядит плохо, в современности почти не используется



## Модель Блинна-Фонга (Blinn-Phong, modified Phong)

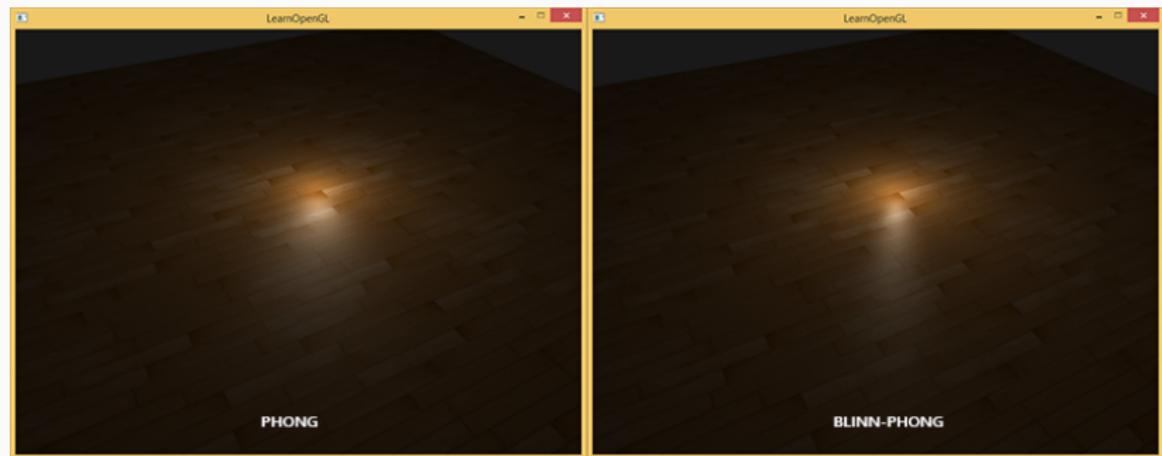
- Как модель Фонга, но по-другому вычисляется specular составляющая

## Модель Блинна-Фонга (Blinn-Phong, modified Phong)

- Как модель Фонга, но по-другому вычисляется specular составляющая
- $h = \frac{\vec{\omega}_{in} + \vec{\omega}_{out}}{\|\vec{\omega}_{in} + \vec{\omega}_{out}\|}$  – т.н. *halfway vector* (часто встречается в моделях освещения)

$$f_{specular}(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) = C(\lambda) (h \cdot n)^{power}$$

# Phong vs Blinn-Phong



# Microfacets

- Чтобы честно вывести формулу для BRDF (*physically-based BRDF*), нужно предположение о микроструктуре поверхности (*microsurface*)

## Microfacets

- Чтобы честно вывести формулу для BRDF (*physically-based BRDF*), нужно предположение о микроструктуре поверхности (*microsurface*)  $\Rightarrow$  микрофасеты (*microfacets*)

# Microfacets

- Чтобы честно вывести формулу для BRDF (*physically-based BRDF*), нужно предположение о микроструктуре поверхности (*microsurface*)  $\Rightarrow$  микрофасеты (*microfacets*)
- Предполагается, что поверхность состоит из маленьких площадок – микрофасетов

# Microfacets

- Чтобы честно вывести формулу для BRDF (*physically-based BRDF*), нужно предположение о микроструктуре поверхности (*microsurface*)  $\Rightarrow$  микрофасеты (*microfacets*)
- Предполагается, что поверхность состоит из маленьких площадок – микрофасетов
- Задаётся распределение ориентаций (нормалей) этих площадок

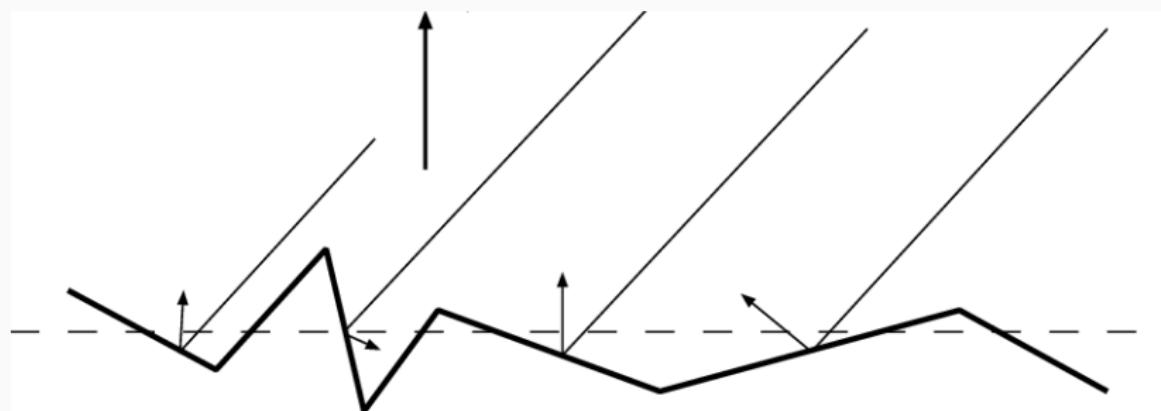
## Microfacets

- Чтобы честно вывести формулу для BRDF (*physically-based BRDF*), нужно предположение о микроструктуре поверхности (*microsurface*)  $\Rightarrow$  микрофасеты (*microfacets*)
- Предполагается, что поверхность состоит из маленьких площадок – микрофасетов
- Задаётся распределение ориентаций (нормалей) этих площадок
- Задаётся материал площадок

# Microfacets

- Чтобы честно вывести формулу для BRDF (*physically-based BRDF*), нужно предположение о микроструктуре поверхности (*microsurface*)  $\Rightarrow$  микрофасеты (*microfacets*)
- Предполагается, что поверхность состоит из маленьких площадок – микрофасетов
- Задаётся распределение ориентаций (нормалей) этих площадок
- Задаётся материал площадок
- Выводится формула для BRDF (обычно – приближённая)

# Microfacets



# Microfacets

- Oren-Nayar BRDF – микрофасетная модель, в которой фасеты имеют диффузную BRDF
- Torrance-Sparrow BRDF – микрофасетная модель, в которой фасеты – идеальные зеркала

## Модель Кука-Торренса (Cook-Torrance)

$$f(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) = (\vec{\omega}_{in} \cdot n) [(1 - s) C_d(\lambda) + s C_s(\lambda) R]$$

$$R = \frac{D \cdot G \cdot F}{4 \cdot (\vec{\omega}_{in} \cdot n) \cdot (\vec{\omega}_{out} \cdot n)}$$

## Модель Кука-Торренса (Cook-Torrance)

$$f(p, \vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) = (\vec{\omega}_{in} \cdot n) [(1 - s)C_d(\lambda) + sC_s(\lambda)R]$$

$$R = \frac{D \cdot G \cdot F}{4 \cdot (\vec{\omega}_{in} \cdot n) \cdot (\vec{\omega}_{out} \cdot n)}$$

- $C_d(\lambda), C_s(\lambda)$  – диффузное и отражающее альбено (цвет)
- $s$  – насколько сильно поверхность отражает свет (интерполизует между диффузной и specular BRDF)
- $D$  – плотность распределения (distribution) нормалей микрофасетов
- $G$  – коэффициент затенения геометрии (geometry) самой себя
- $F$  – коэффициент Френеля (Fresnel)
- Конкретный вид функций  $D, G, F$  может варьироваться (см. ссылки)

## Disney principled BRDF (2012)

- Основывается на микрофасетных моделях
- Не влезет на слайд :)
- Даёт огромное количество параметров для тонкой настройки, и потому удобна для художников
- Её вариации используют почти все крупные движки и программы (Unity, Unreal, Blender, ...)

# Blender principled BRDF



# Ссылки

Туториалы про освещение:

- [learnopengl.com/Lighting/Basic-Lighting](http://learnopengl.com/Lighting/Basic-Lighting)
- [opengl-tutorial.org/beginners-tutorials/tutorial-8-basic-shading](http://opengl-tutorial.org/beginners-tutorials/tutorial-8-basic-shading)
- [lighthouse3d.com/tutorials/glsl-tutorial/point-lights](http://lighthouse3d.com/tutorials/glsl-tutorial/point-lights)

Сложные BRDF:

- Нормировка Phong BRDF
- Описание Cook-Torrance BRDF и примеры функций D, G, F
- Стандартная BRDF в Unity
- Описание Disney BRDF

Raytracing:

- Книжка Raytracing in one weekend
- Видео про ReSTIR GI