

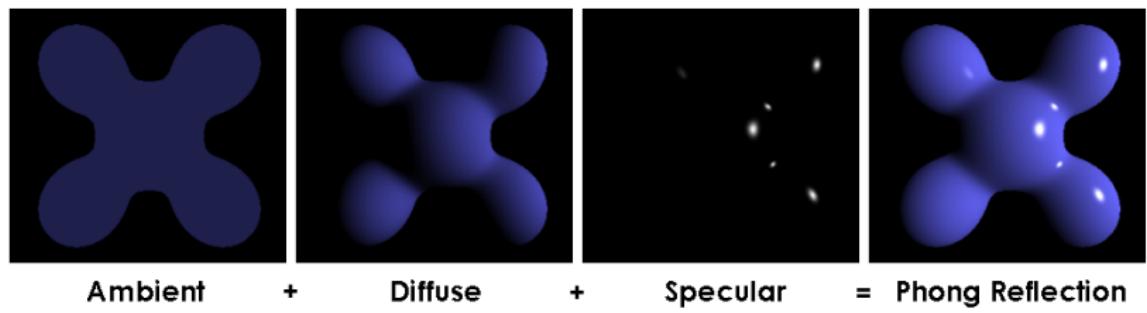
# Компьютерная графика

Лекция 7: Несколько источников света, normal mapping, material mapping, ambient occlusion, baking, отражения, environment mapping, HDR

2021

# Модель фонга

- ▶ Ambient + diffuse + specular



## Specular highlight

- ▶ Грубая аппроксимация отражений: прямой свет от источника света обычно значительно ярче, чем свет, отражённый другими объектами

## Specular highlight

- ▶ Грубая аппроксимация отражений: прямой свет от источника света обычно значительно ярче, чем свет, отражённый другими объектами

- ▶ `light_dir` - направление на источник света

- ▶ Вычисляем направление отражённого света

```
reflected = 2.0 * normal * dot(normal, light_dir)  
           - light_dir
```

## Specular highlight

- ▶ Грубая аппроксимация отражений: прямой свет от источника света обычно значительно ярче, чем свет, отражённый другими объектами

- ▶ `light_dir` - направление на источник света

- ▶ Вычисляем направление отражённого света

```
reflected = 2.0 * normal * dot(normal, light_dir)  
           - light_dir
```

- ▶ Если отражённый луч светит *примерно* в направлении камеры, добавим к цвету пикселя `specular`-компоненту

## Specular highlight

- ▶ Грубая аппроксимация отражений: прямой свет от источника света обычно значительно ярче, чем свет, отражённый другими объектами

- ▶ `light_dir` - направление на источник света
- ▶ Вычисляем направление отражённого света

```
reflected = 2.0 * normal * dot(normal, light_dir)  
           - light_dir
```

- ▶ Если отражённый луч светит *примерно* в направлении камеры, добавим к цвету пикселя *specular*-компоненту
- ▶ Обычно вычисляется как

```
pow(max(0.0, dot(reflected, camera_dir)), 16.0)
```

- ▶ `camera_dir` - направление на камеру
- ▶ 16 - настраиваемый параметр: чем меньше, тем ярче *specular highlight*

## Specular highlight

- ▶ Грубая аппроксимация отражений: прямой свет от источника света обычно значительно ярче, чем свет, отражённый другими объектами

- ▶ `light_dir` - направление на источник света
- ▶ Вычисляем направление отражённого света

```
reflected = 2.0 * normal * dot(normal, light_dir)  
           - light_dir
```

- ▶ Если отражённый луч светит *примерно* в направлении камеры, добавим к цвету пикселя *specular*-компоненту

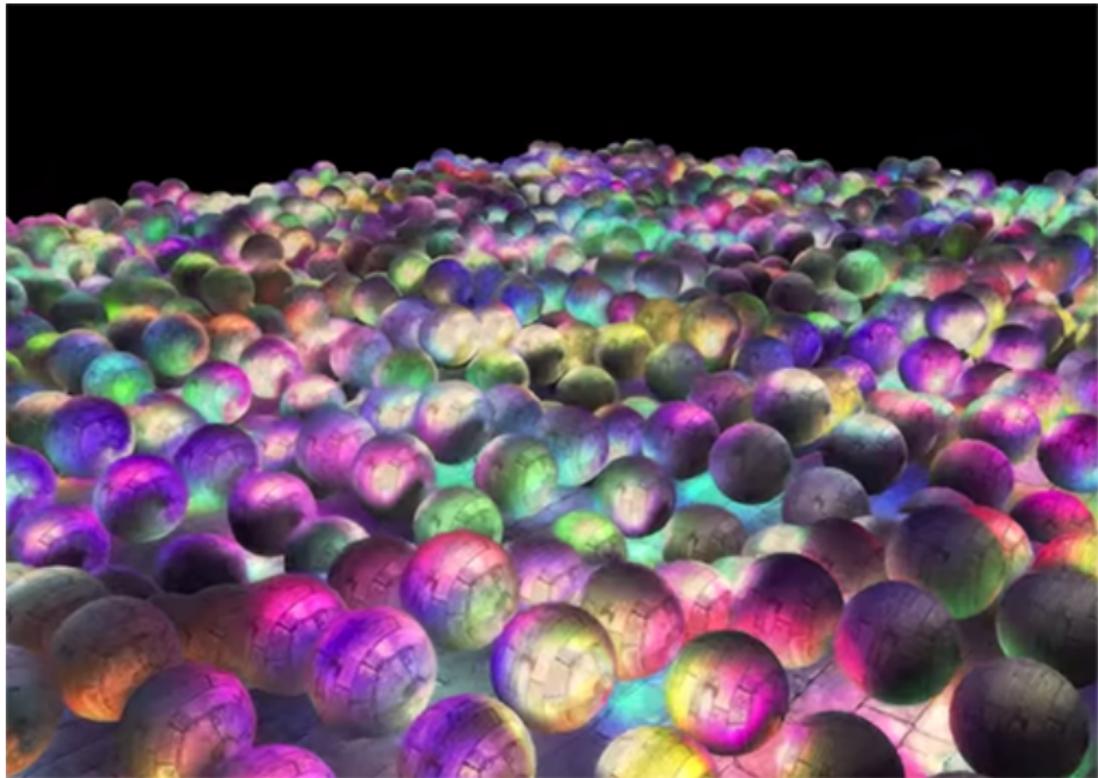
- ▶ Обычно вычисляется как

```
pow(max(0.0, dot(reflected, camera_dir)), 16.0)
```

- ▶ `camera_dir` - направление на камеру
- ▶ 16 - настраиваемый параметр: чем меньше, тем ярче specular highlight

- ▶ N.B. В таком виде получается ненормализованная BRDF

## Несколько источников света



## Несколько источников света

- ▶ Простейший способ - использовать по отдельному набору uniform-переменных для каждого источника света, и обрабатывать их все сразу в шейдере

## Несколько источников света

- ▶ Простейший способ - использовать по отдельному набору uniform-переменных для каждого источника света, и обрабатывать их все сразу в шейдере
- ▶ Можно использовать массивы uniform-переменных

## Несколько источников света

- ▶ Простейший способ - использовать по отдельному набору uniform-переменных для каждого источника света, и обрабатывать их все сразу в шейдере
- ▶ Можно использовать массивы uniform-переменных
  - ▶ Задаются как `vec3 light_position[N]` (`N` - некая константа)
  - ▶ Каждый элемент массива - отдельная uniform-переменная, для неё нужно отдельно вызвать `glGetUniformLocation`
  - ▶ Имя этой переменной - имя массива + индекс, например `light_position[0]`

## Несколько источников света

- ▶ Сколько источников света можно так обработать?

## Несколько источников света

- ▶ Сколько источников света можно так обработать?
- ▶ OpenGL гарантирует, что под uniform-переменные есть как минимум 1024 выделенных компоненты (vec3 занимает 3 компоненты, и т.п.)

## Несколько источников света

- ▶ Сколько источников света можно так обработать?
- ▶ OpenGL гарантирует, что под uniform-переменные есть как минимум 1024 выделенных компоненты (vec3 занимает 3 компоненты, и т.п.)
- ▶ В варианте 3 vec3 на источник (position, color, attenuation)
  - около 113 источников света

## Несколько источников света

- ▶ Сколько источников света можно так обработать?
- ▶ OpenGL гарантирует, что под uniform-переменные есть как минимум 1024 выделенных компоненты (vec3 занимает 3 компоненты, и т.п.)
- ▶ В варианте 3 vec3 на источник (position, color, attenuation)
  - около 113 источников света
- ▶ Чем больше источников, тем больше вычислений в шейдере, тем больше нагрузка на GPU

## Несколько источников света

- ▶ Такой подход выполняет много лишних операций:
  - ▶ Вычисляется полное освещение для объекта, который может быть скрыт другим объектом

## Несколько источников света

- ▶ Такой подход выполняет много лишних операций:
  - ▶ Вычисляется полное освещение для объекта, который может быть скрыт другим объектом
  - ▶ Вычисляется вклад в освещение для объекта, находящегося слишком далеко от источника света

## Несколько источников света

- ▶ Такой подход выполняет много лишних операций:
  - ▶ Вычисляется полное освещение для объекта, который может быть скрыт другим объектом
  - ▶ Вычисляется вклад в освещение для объекта, находящегося слишком далеко от источника света
- ▶ Можно частично решить, вычисляя для каждого объекта список источников, которыми он освещён ⇒ больше изменений uniform-переменных при рендеринге, меньше возможность для batching'а (объединения объектов в группы или в общие буферы)

## Несколько источников света

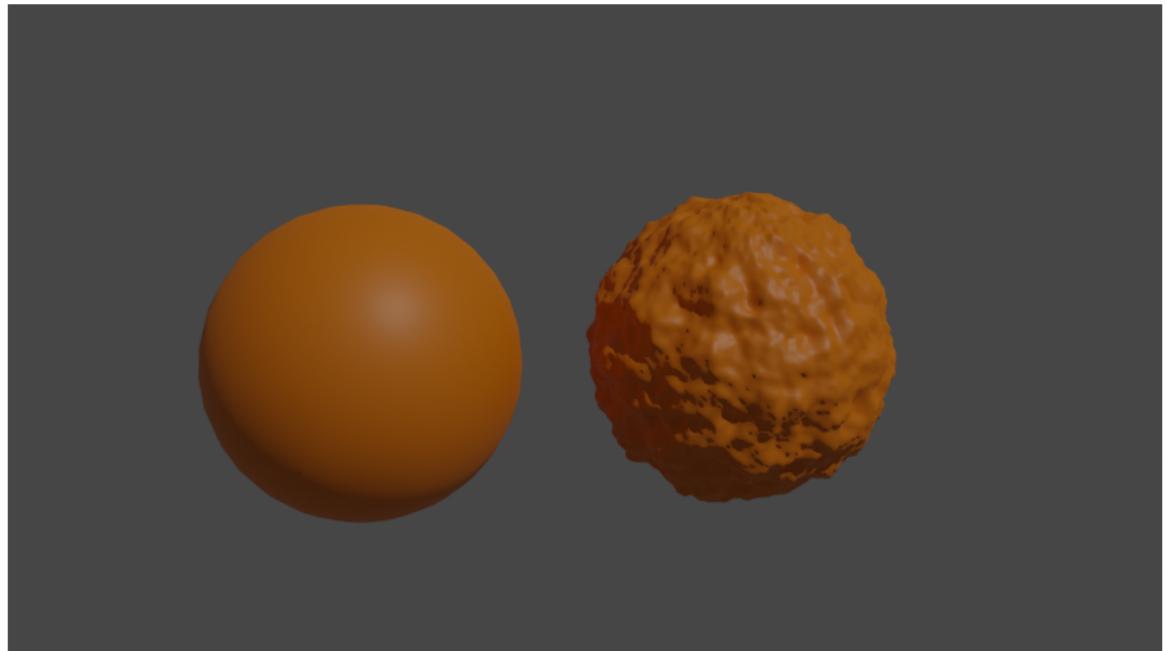
- ▶ Такой подход выполняет много лишних операций:
  - ▶ Вычисляется полное освещение для объекта, который может быть скрыт другим объектом
  - ▶ Вычисляется вклад в освещение для объекта, находящегося слишком далеко от источника света
- ▶ Можно частично решить, вычисляя для каждого объекта список источников, которыми он освещён ⇒ больше изменений uniform-переменных при рендеринге, меньше возможность для batching'а (объединения объектов в группы или в общие буферы)
- ▶ Более современные решения: deferred shading, tiled/clustered shading

## Несколько источников света

- ▶ [learnopengl.com/Lighting/Multiple-lights](http://learnopengl.com/Lighting/Multiple-lights)
- ▶ [tomdalling.com/blog/modern-opengl/08-even-more-lighting-directional-lights-spotlights-multiple-lights](http://tomdalling.com/blog/modern-opengl/08-even-more-lighting-directional-lights-spotlights-multiple-lights)
- ▶ [en.wikibooks.org/wiki/GLSL\\_Programming/GLUT/Multiple\\_Lights](http://en.wikibooks.org/wiki/GLSL_Programming/GLUT/Multiple_Lights)

## Normal mapping

- ▶ Хотим высокую детализацию поверхности объекта

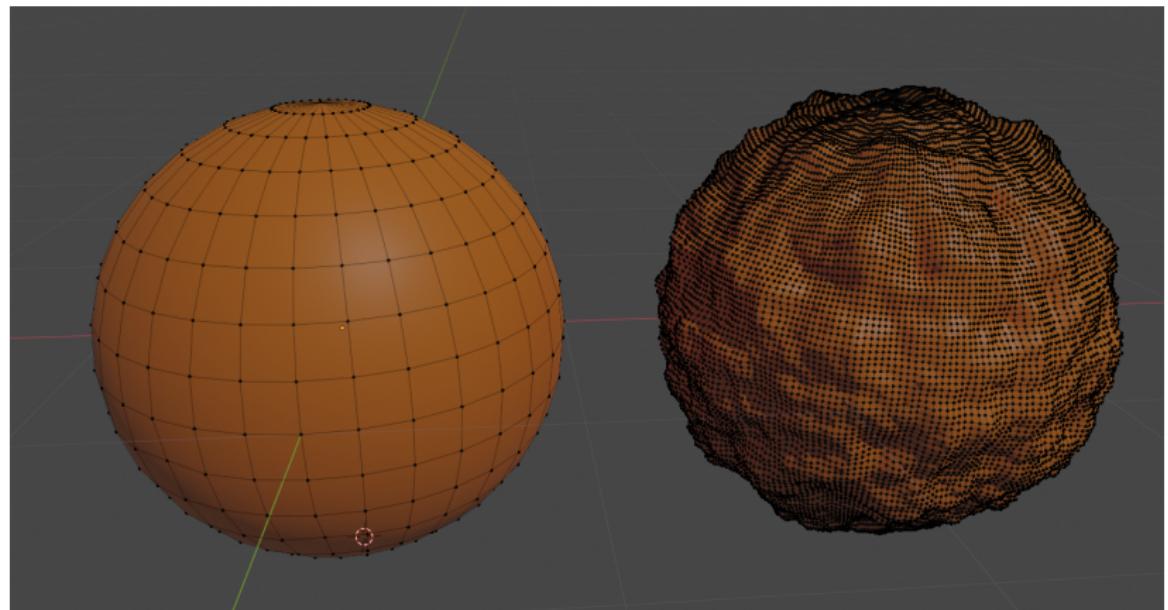


## Normal mapping

- Можно взять очень много вершин

## Normal mapping

- ▶ Можно взять очень много вершин
- ▶ Очень дорого, особенно когда объектов много



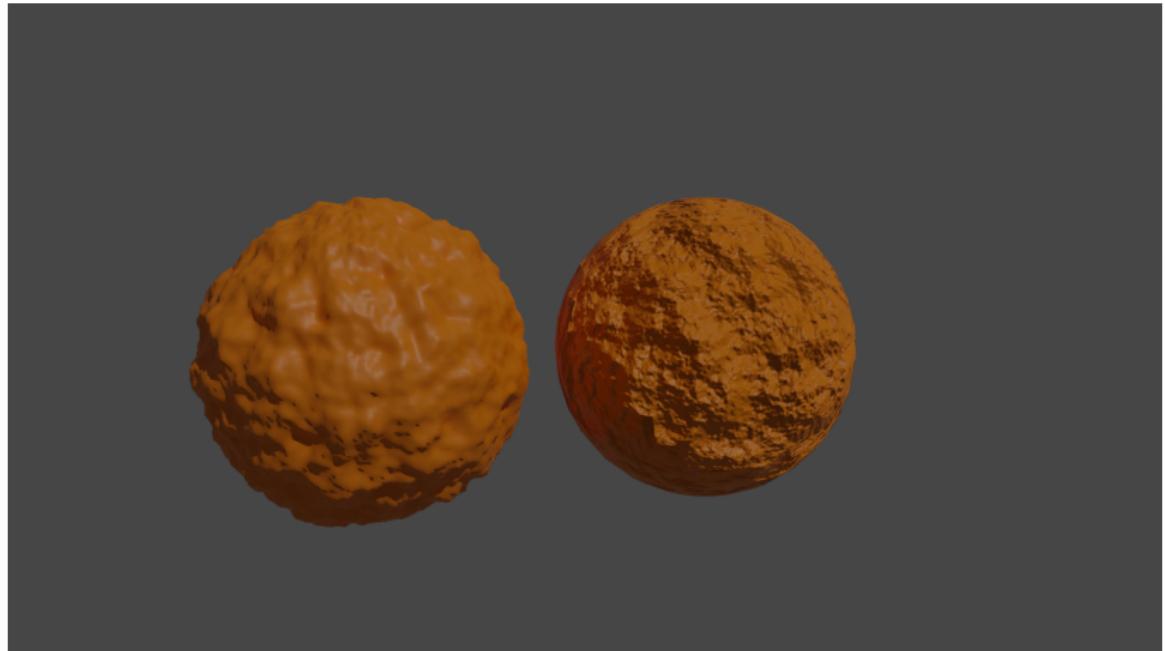
# Normal mapping

- ▶ Наблюдение: высокая детализация даёт 2 вещи
  - ▶ Детализацию координат вершин - слабо заметна
  - ▶ Детализацию нормалей вершин - сильно заметна (влияет на освещение)

# Normal mapping

- ▶ Наблюдение: высокая детализация даёт 2 вещи
  - ▶ Детализацию координат вершин - слабо заметна
  - ▶ Детализацию нормалей вершин - сильно заметна (влияет на освещение)
- ▶ Давайте откажемся от детализации координат!

## Normal mapping



## Normal mapping

- ▶ Normal mapping: используем текстуру, в которой закодированы нормали в точках объекта

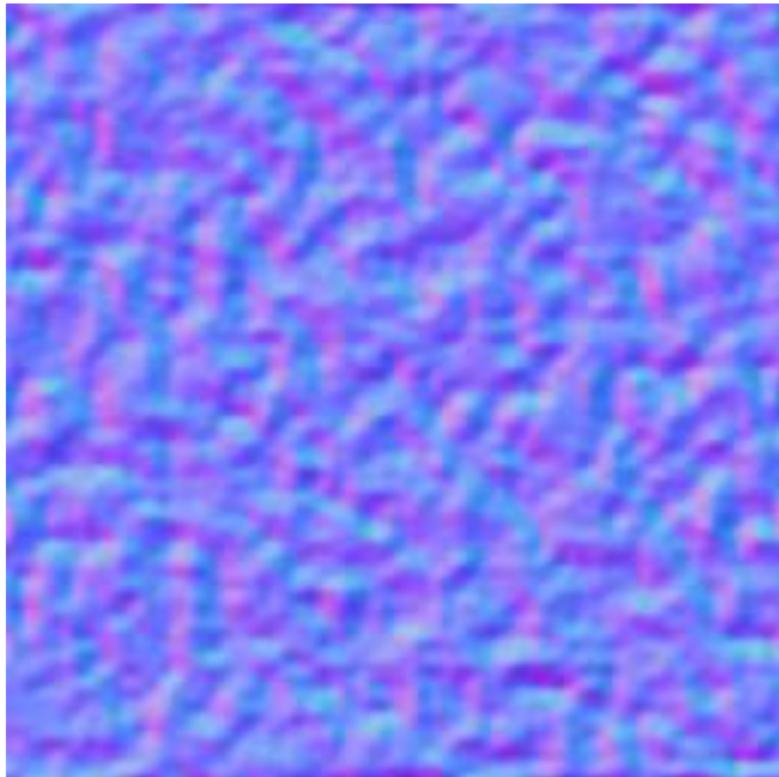
## Normal mapping

- ▶ Normal mapping: используем текстуру, в которой закодированы нормали в точках объекта
- ▶ Вместо нормалей, приходящих в вершинах, достаём нормали из текстуры во фрагментном шейдере

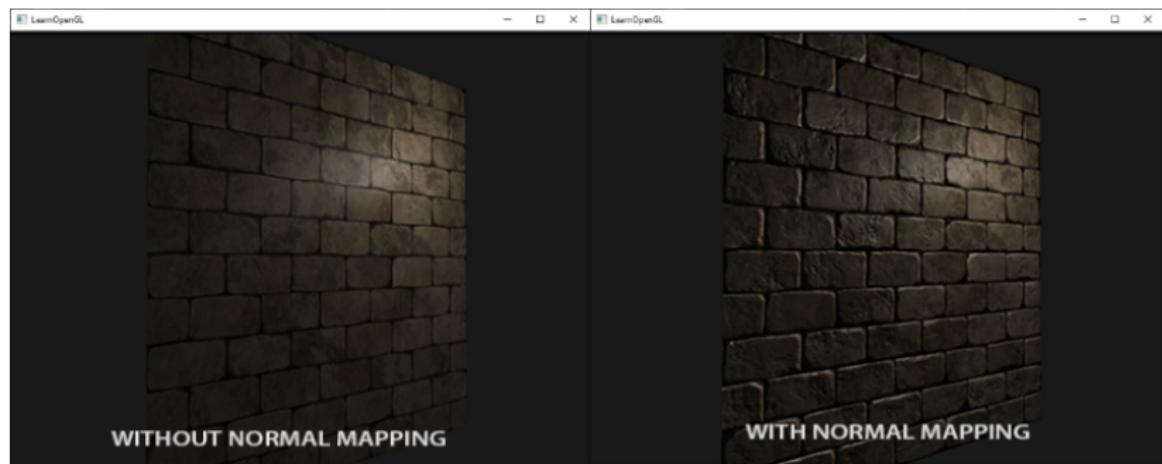
## Normal mapping

- ▶ Normal mapping: используем текстуру, в которой закодированы нормали в точках объекта
- ▶ Вместо нормалей, приходящих в вершинах, достаём нормали из текстуры во фрагментном шейдере
- ▶ При этом можно использовать слабо детализированную геометрию

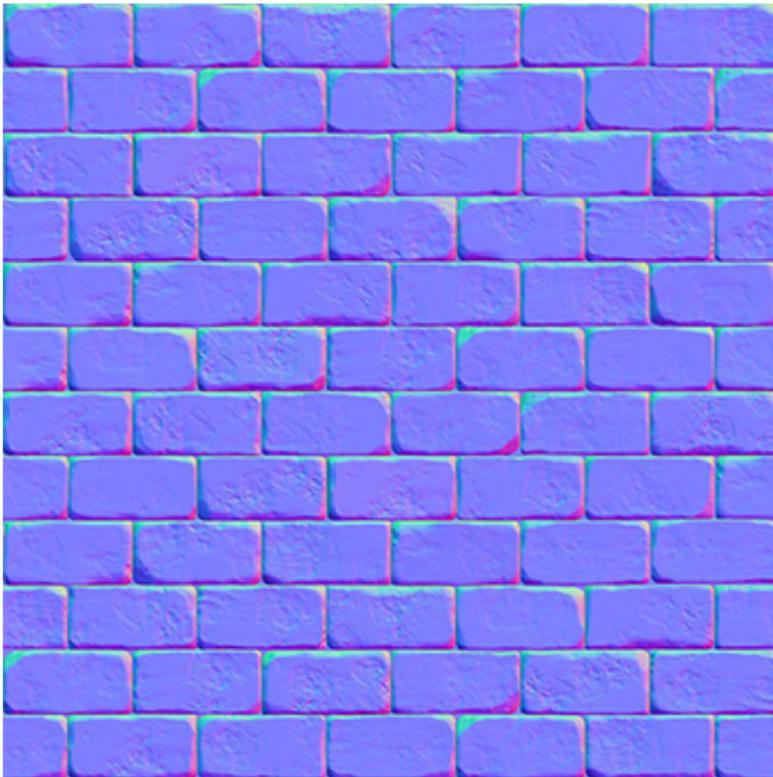
## Normal mapping



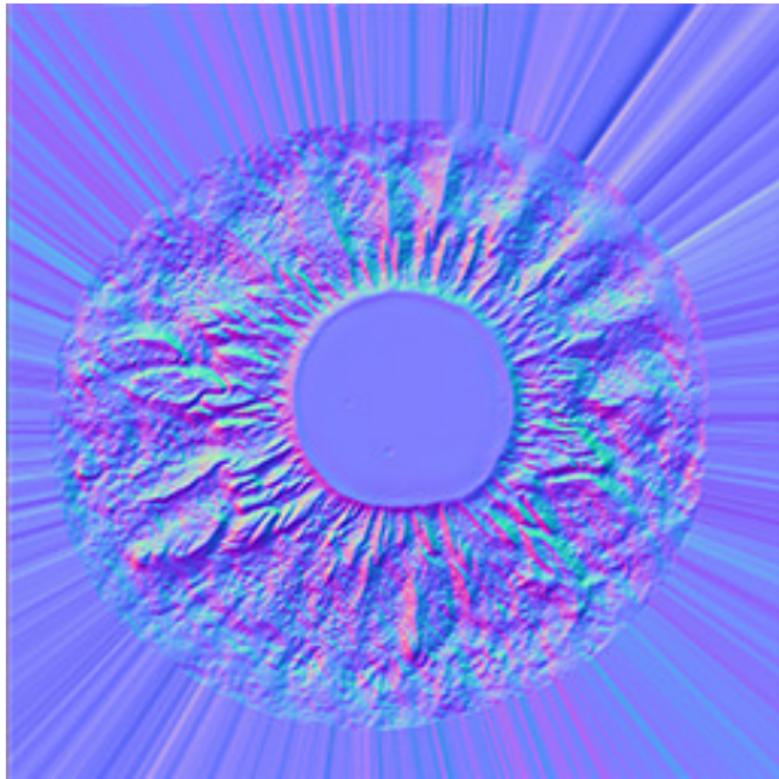
# Normal mapping



## Normal mapping



## Normal mapping



## Normal mapping: формат хранения

- ▶ Как хранить нормали в текстуре?

## Normal mapping: формат хранения

- ▶ Как хранить нормали в текстуре?
- ▶ Нормаль - 3х-компонентный нормированный вектор

## Normal mapping: формат хранения

- ▶ Как хранить нормали в текстуре?
- ▶ Нормаль - 3х-компонентный нормированный вектор
- ▶ Floating-point текстуры (GL\_RGB32F или GL\_RGB16F)
  - ▶ Занимают много места ( $3 \times 16\text{-bit} = 6$  байт на пиксель)
  - ▶ Излишняя точность
  - ▶ Значительная часть возможных значений (координаты больше 1 по модулю) вообще не используется

# Normal mapping: формат хранения

- ▶ Как хранить нормали в текстуре?
- ▶ Нормаль - 3х-компонентный нормированный вектор
- ▶ Floating-point текстуры (GL\_RGB32F или GL\_RGB16F)
  - ▶ Занимают много места ( $3 \times 16\text{-bit} = 6$  байт на пиксель)
  - ▶ Излишняя точность
  - ▶ Значительная часть возможных значений (координаты больше 1 по модулю) вообще не используется
- ▶ Signed normalized текстуры (GL\_RGB8\_SNORM)
  - ▶ Обычно достаточно точности
  - ▶ Компактные
  - ▶ Форматы изображений обычно не умеют работать с signed пикселями

## Normal mapping: формат хранения

- ▶ Как хранить нормали в текстуре?
- ▶ Нормаль - 3х-компонентный нормированный вектор
- ▶ Floating-point текстуры (GL\_RGB32F или GL\_RGB16F)
  - ▶ Занимают много места ( $3 \times 16\text{-bit} = 6$  байт на пиксель)
  - ▶ Излишняя точность
  - ▶ Значительная часть возможных значений (координаты больше 1 по модулю) вообще не используется
- ▶ Signed normalized текстуры (GL\_RGB8\_SNORM)
  - ▶ Обычно достаточно точности
  - ▶ Компактные
  - ▶ Форматы изображений обычно не умеют работать с signed пикселями
- ▶ Обычные unsigned normalized текстуры (GL\_RGB8, GL\_RGB10\_A2, GL\_RGB565, GL\_RGB5\_A1, GL\_R3\_G3\_B2)
  - ▶ Координаты хранятся в виде, отмасштабированном в диапазон [0, 1], т.е.  $0.5 * (x + 1)$

## Normal mapping: система координат

- ▶ В какой системе координат хранить нормали в текстуре?

## Normal mapping: система координат

- ▶ В какой системе координат хранить нормали в текстуре?
- ▶ В системе координат объекта
  - ▶ Нет лишних операций, просто читаем нормаль из текстуры

## Normal mapping: система координат

- ▶ В какой системе координат хранить нормали в текстуре?
- ▶ В системе координат объекта
  - ▶ Нет лишних операций, просто читаем нормаль из текстуры
- ▶ В локальной системе координат пикселя, учитывая геометрию объекта
  - ▶ Нужно восстанавливать локальную систему координат в шейдере (используя нормаль вершины)
  - ▶ Можно сэкономить и хранить только две координаты нормали
  - ▶ Обычно такая текстура с нормалами легче воспринимается на глаз

## Normal mapping: система координат

- ▶ В какой системе координат хранить нормали в текстуре?
- ▶ В системе координат объекта
  - ▶ Нет лишних операций, просто читаем нормаль из текстуры
- ▶ В локальной системе координат пикселя, учитывая геометрию объекта
  - ▶ Нужно восстанавливать локальную систему координат в шейдере (используя нормаль вершины)
  - ▶ Можно сэкономить и хранить только две координаты нормали
  - ▶ Обычно такая текстура с нормалями легче воспринимается на глаз
  - ▶ Такие текстуры часто выглядят синими, потому что направление Z считается нормалью вершины

# Normal mapping: туториалы

- ▶ [learnopengl.com/Advanced-Lighting/Normal-Mapping](http://learnopengl.com/Advanced-Lighting/Normal-Mapping)
- ▶ [opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping](http://opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping)
- ▶ [sudonull.com/post/14500-Learn-OpenGL-Lesson-55-Normal-Mapping](http://sudonull.com/post/14500-Learn-OpenGL-Lesson-55-Normal-Mapping)
- ▶ [habr.com/ru/post/415579](http://habr.com/ru/post/415579)

## Material mapping

- ▶ В обработке освещения часто встречаются настраиваемые параметры, помимо цвета (альбедо) объекта

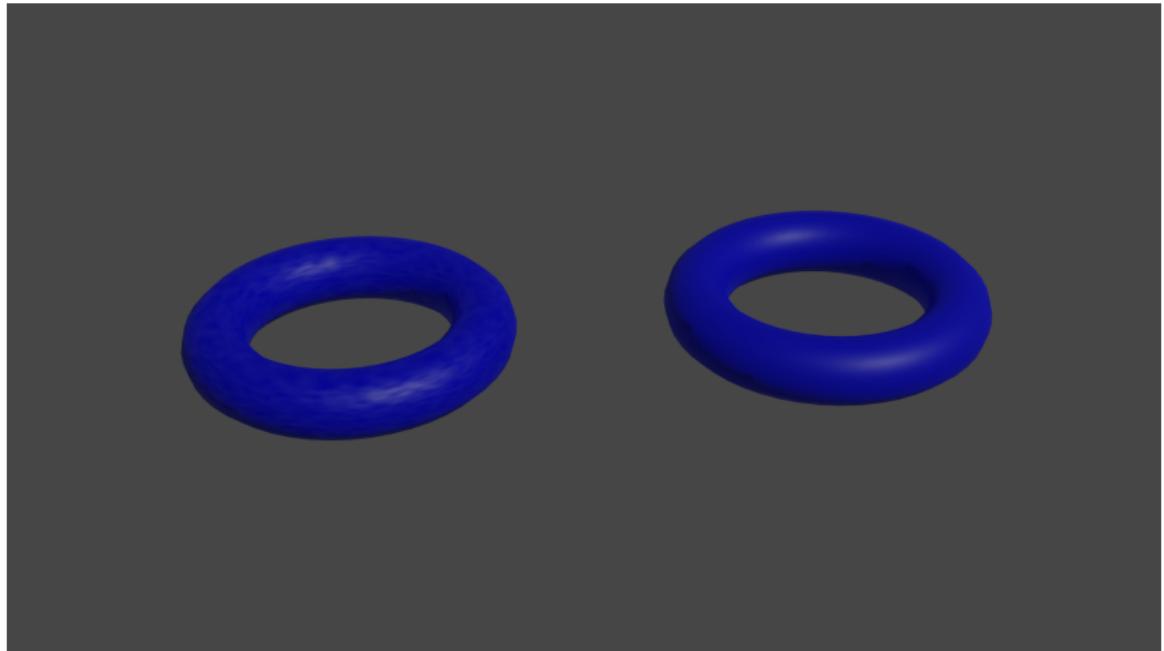
## Material mapping

- ▶ В обработке освещения часто встречаются настраиваемые параметры, помимо цвета (альбедо) объекта
- ▶ Например, показатель степени для specular highlight

## Material mapping

- ▶ В обработке освещения часто встречаются настраиваемые параметры, помимо цвета (альбедо) объекта
- ▶ Например, показатель степени для specular highlight
- ▶ Можно его тоже варьировать, читая из текстуры

## Material mapping



## Ambient occlusion

- ▶ Ambient освещение - светит "отовсюду"

## Ambient occlusion

- ▶ Ambient освещение - светит "отовсюду"
- ▶ Это свет, значит от него может быть тень

## Ambient occlusion

- ▶ Ambient освещение - светит "отовсюду"
- ▶ Это свет, значит от него может быть тень
- ▶ Какие-то части объекта получают больше ambient света, какие-то - меньше

## Ambient occlusion

- ▶ Ambient освещение - светит "отовсюду"
- ▶ Это свет, значит от него может быть тень
- ▶ Какие-то части объекта получают больше ambient света, какие-то - меньше
- ▶ Обычно затеняются углы, углубления, трещины, стыки объектов, и т.п.

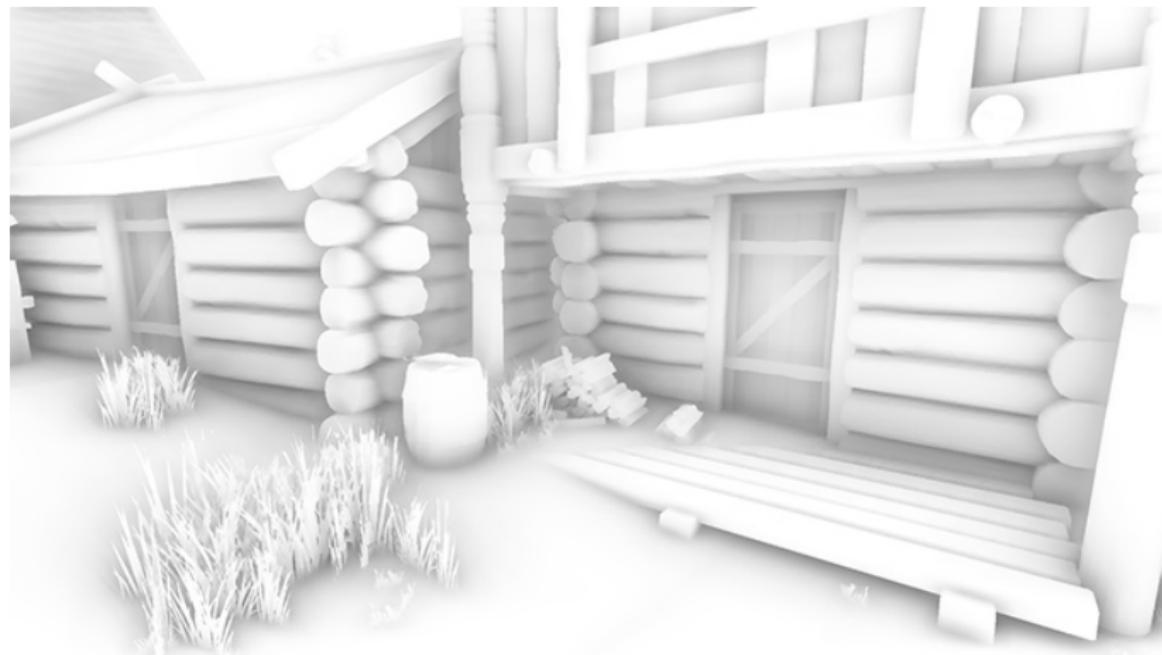
## Ambient occlusion

- ▶ Ambient освещение - светит "отовсюду"
- ▶ Это свет, значит от него может быть тень
- ▶ Какие-то части объекта получают больше ambient света, какие-то - меньше
- ▶ Обычно затеняются углы, углубления, трещины, стыки объектов, и т.п.
- ▶ Очень сильно увеличивает реализм изображения

## Ambient occlusion (Crysis, 2007)



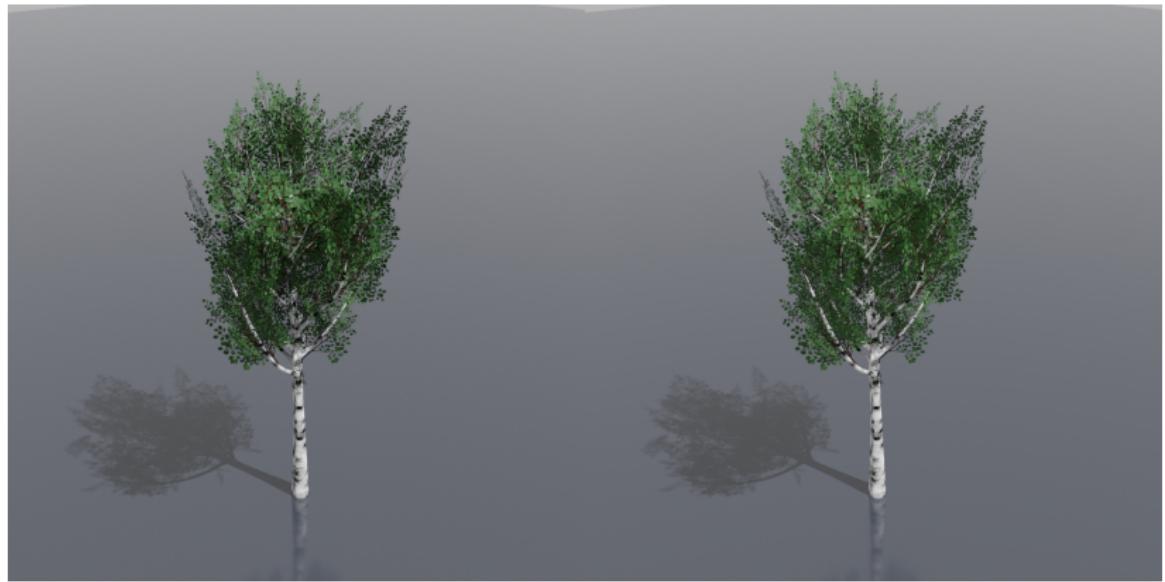
# Ambient occlusion



# Ambient occlusion



# Ambient occlusion



## Ambient occlusion

- ▶ В современности обычно используются real-time алгоритмы, использующие только геометрию сцены: SSAO, HBAO, HBAO+, HDAO, VXAO

## Ambient occlusion

- ▶ В современности обычно используются real-time алгоритмы, использующие только геометрию сцены: SSAO, HBAO, HBAO+, HDAO, VXAO
- ▶ Можно предподсчитать ambient occlusion для модели, записать её в текстуру и использовать в шейдере

## Ambient occlusion

- ▶ В современности обычно используются real-time алгоритмы, использующие только геометрию сцены: SSAO, HBAO, HBAO+, HDAO, VXAO
- ▶ Можно предподсчитать ambient occlusion для модели, записать её в текстуру и использовать в шейдере
- ▶ Работает быстрее, чем общие real-time алгоритмы

## Ambient occlusion

- ▶ В современности обычно используются real-time алгоритмы, использующие только геометрию сцены: SSAO, HBAO, HBAO+, HDAO, VXAO
- ▶ Можно предподсчитать ambient occlusion для модели, записать её в текстуру и использовать в шейдере
- ▶ Работает быстрее, чем общие real-time алгоритмы
- ▶ Не учитывает occlusion между разными объектами

## Ambient occlusion



Original model



With ambient occlusion



Extracted ambient occlusion map

# Baking

- ▶ Записывание предподсчитанных свойств объектов в текстуры в общем случае называется **baking** (запеканием)

# Baking

- ▶ Записывание предподсчитанных свойств объектов в текстуры в общем случае называется *baking* (запеканием)
- ▶ Все 3D-редакторы умеют запекать нормали, ambient occlusion, и многое другое

# Отражения

- ▶ Расчёт отражений в real-time - очень сложная задача

# Отражения

- ▶ Расчёт отражений в real-time - очень сложная задача
- ▶ Для плоских зеркал можно нарисовать сцену целиком, применив отражающее преобразование и нарисовав зеркало предварительно в stencil буфер

# Отражения

- ▶ Расчёт отражений в real-time - очень сложная задача
- ▶ Для плоских зеркал можно нарисовать сцену целиком, применив отражающее преобразование и нарисовав зеркало предварительно в stencil буфер
  - ▶ Рисовать заново всю сцену - дорого
  - ▶ Не работает для неплоских зеркал

# Отражения

- ▶ Расчёт отражений в real-time - очень сложная задача
- ▶ Для плоских зеркал можно нарисовать сцену целиком, применив отражающее преобразование и нарисовав зеркало предварительно в stencil буфер
  - ▶ Рисовать заново всю сцену - дорого
  - ▶ Не работает для неплоских зеркал
- ▶ Можно нарисовать сцену целиком, вычисляя проекцию объекта на зеркало в вершинном шейдере
  - ▶ Тоже дорого
  - ▶ Могут быть артефакты с растягиванием вершин

# Отражения

- ▶ Апроксимация: рисовать сцену в текстуру, эту текстуру накладывать на объект

# Отражения

- ▶ Апроксимация: рисовать сцену в текстуру, эту текстуру накладывать на объект
  - ▶ Мы пока не умеем рисовать в текстуры

# Отражения

- ▶ Апроксимация: рисовать сцену в текстуру, эту текстуру накладывать на объект
  - ▶ Мы пока не умеем рисовать в текстуры
- ▶ Можно использовать предподсчитанную текстуру!

## Environment mapping

- ▶ Предподсчитанная cubemap-текстура окружения

## Environment mapping

- ▶ Предподсчитанная сиветар-текстура окружения
- ▶ Не учитывает динамические объекты

## Environment mapping

- ▶ Предподсчитанная сиветар-текстура окружения
- ▶ Не учитывает динамические объекты
- ▶ Хорошо работает для нечётких/слабых зеркал (капот автомобиля, дверная ручка, гильза)

# Environment mapping



## Environment mapping: реализация

- ▶ Используя направление на камеру и нормаль поверхности, вычисляем направление отражённого луча

## Environment mapping: реализация

- ▶ Используя направление на камеру и нормаль поверхности, вычисляем направление отражённого луча
- ▶ Используем направление отражённого луча в качестве параметра для чтения из cube-map-текстуры

## Environment mapping: ссылки

- ▶ [learnopengl.com/Advanced-OpenGL/Cubemaps](http://learnopengl.com/Advanced-OpenGL/Cubemaps)

# HDR

- ▶ Проблема: после вычисления освещённости мы можем получить значения компонент цвета большие 1, и они будут сжаты до значений [0, 1]

# HDR

- ▶ Проблема: после вычисления освещённости мы можем получить значения компонент цвета большие 1, и они будут сжаты до значений [0, 1]
- ▶ В реальном мире интенсивность света может меняться на 15 порядков

# HDR

- ▶ Проблема: после вычисления освещённости мы можем получить значения компонент цвета большие 1, и они будут сжаты до значений [0, 1]
- ▶ В реальном мире интенсивность света может меняться на 15 порядков
- ▶ High Dynamic Range (HDR) - термин, означающий большой (не ограниченный [0, 1]) диапазон интенсивностей

# HDR

- ▶ Проблема: после вычисления освещённости мы можем получить значения компонент цвета большие 1, и они будут сжаты до значений [0, 1]
- ▶ В реальном мире интенсивность света может меняться на 15 порядков
- ▶ High Dynamic Range (HDR) - термин, означающий большой (не ограниченный [0, 1]) диапазон интенсивностей
- ▶ HDR текстура (например, environment map) - содержит значения, выходящие за диапазон [0, 1] (обычно 32-bit floating-point)

# HDR

- ▶ Проблема: после вычисления освещённости мы можем получить значения компонент цвета большие 1, и они будут сжаты до значений [0, 1]
- ▶ В реальном мире интенсивность света может меняться на 15 порядков
- ▶ High Dynamic Range (HDR) - термин, означающий большой (не ограниченный [0, 1]) диапазон интенсивностей
- ▶ HDR текстура (например, environment map) - содержит значения, выходящие за диапазон [0, 1] (обычно 32-bit floating-point)
- ▶ HDR рендеринг - позволяет отобразить HDR-освещение

## Tone mapping

- ▶ Нужно превратить диапазон освещённостей  $[0, \infty)$  в диапазон  $[0, 1]$

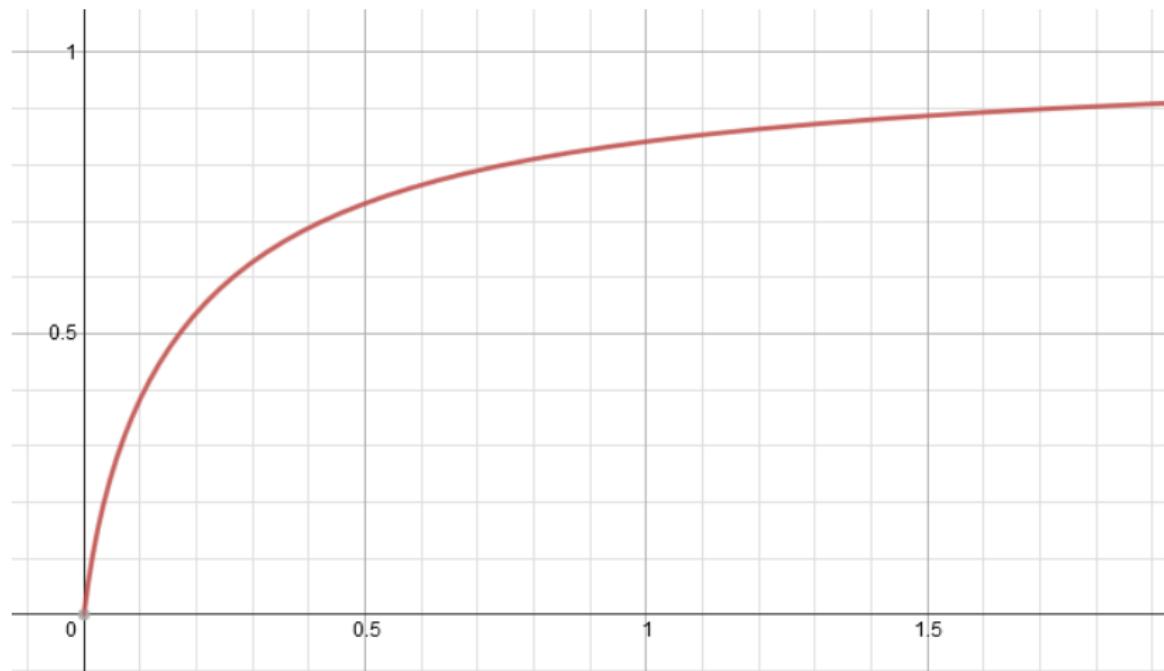
## Tone mapping

- ▶ Нужно превратить диапазон освещённостей  $[0, \infty)$  в диапазон  $[0, 1]$
- ▶ В принципе, подойдёт любая монотонно возрастающая функция  $[0, \infty) \rightarrow [0, 1]$ 
  - ▶  $x \mapsto \frac{x}{1+x}$  - Reinhard operator
  - ▶  $x \mapsto \arctan(x)$
  - ▶  $x \mapsto \frac{2}{1+e^{-x}} - 1$

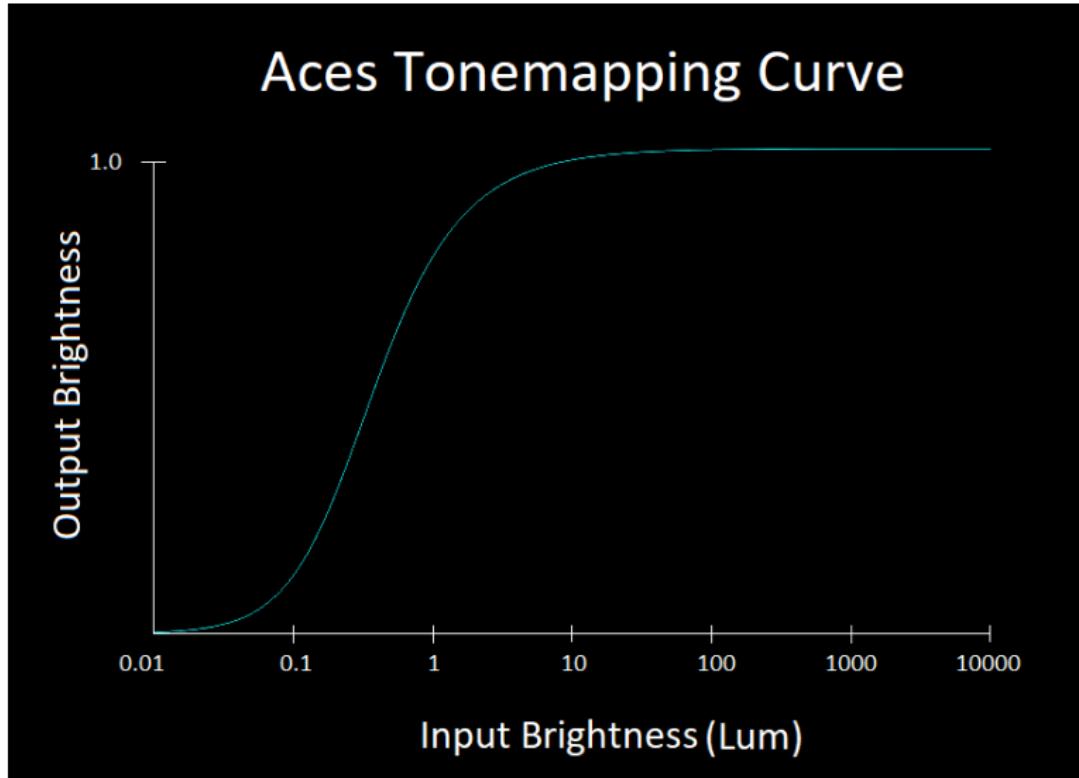
# Tone mapping

- ▶ Нужно превратить диапазон освещённостей  $[0, \infty)$  в диапазон  $[0, 1]$
- ▶ В принципе, подойдёт любая монотонно возрастающая функция  $[0, \infty) \rightarrow [0, 1]$ 
  - ▶  $x \mapsto \frac{x}{1+x}$  - 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)

## Reinhard operator



## Filmic tone mapping curve (ACES)



## HDR & tone mapping: ссылки

- ▶ [learnopengl.com/Advanced-Lighting/HDR](http://learnopengl.com/Advanced-Lighting/HDR)
- ▶ [skylum.com/blog/what-is-tone-mapping](http://skylum.com/blog/what-is-tone-mapping)
- ▶ [veneratech.com/what-is-hdr-tone-mapping](http://veneratech.com/what-is-hdr-tone-mapping)