

Компьютерная графика

Лекция 9: Вычисление проекции для shadow mapping, ESM, VSM, PSM, CSM, Ambient Occlusion

2023

Проекция для shadow map: точечный источник

- 6 перспективных проекций для 6 граней subemap-текстуры shadow map (одно рисование всей сцены на каждую из них)

Проекция для shadow map: точечный источник

- 6 перспективных проекций для 6 граней subemap-текстуры shadow map (одно рисование всей сцены на каждую из них)
- Каждая проекция смотрит в направлении некоторой оси: $\pm X, \pm Y, \pm Z$

Проекция для shadow map: точечный источник

- 6 перспективных проекций для 6 граней subemap-текстуры shadow map (одно рисование всей сцены на каждую из них)
- Каждая проекция смотрит в направлении некоторой оси: $\pm X$, $\pm Y$, $\pm Z$
- Соответствующие грани subemap-текстуры в OpenGL называются `GL_TEXTURE_CUBE_MAP_POSITIVE_X`, `GL_TEXTURE_CUBE_MAP_NEGATIVE_X` и т.д. (параметр `glTexImage2D`, `glFramebufferTexture2D`, etc)

Проекция для shadow map: точечный источник

- 6 перспективных проекций для 6 граней cubemap-текстуры shadow map (одно рисование всей сцены на каждую из них)
- Каждая проекция смотрит в направлении некоторой оси: $\pm X$, $\pm Y$, $\pm Z$
- Соответствующие грани cubemap-текстуры в OpenGL называются `GL_TEXTURE_CUBE_MAP_POSITIVE_X`, `GL_TEXTURE_CUBE_MAP_NEGATIVE_X` и т.д. (параметр `glTexImage2D`, `glFramebufferTexture2D`, etc)
- Каждую проекцию нужно будет покрутить вокруг оси взгляда, чтобы она совпала с тем, как эта грань cubemap текстуры определена в OpenGL (проще всего – подбором одного из четырёх вращений на 90°)

Проекция для shadow map: точечный источник

- 6 перспективных проекций для 6 граней cubemap-текстуры shadow map (одно рисование всей сцены на каждую из них)
- Каждая проекция смотрит в направлении некоторой оси: $\pm X, \pm Y, \pm Z$
- Соответствующие грани cubemap-текстуры в OpenGL называются `GL_TEXTURE_CUBE_MAP_POSITIVE_X`, `GL_TEXTURE_CUBE_MAP_NEGATIVE_X` и т.д. (параметр `glTexImage2D`, `glFramebufferTexture2D`, etc)
- Каждую проекцию нужно будет покрутить вокруг оси взгляда, чтобы она совпала с тем, как эта грань cubemap текстуры определена в OpenGL (проще всего – подбором одного из четырёх вращений на 90°)
- Угол видимой области – 90° по обеим осям (квадратная камера, `aspect ratio = 1`)

Проекция для shadow map: точечный источник

- `near` и `far` – максимальное (соответственно, минимальное), при котором не обрезается ничего лишнего

Проекция для shadow map: точечный источник

- `near` и `far` – максимальное (соответственно, минимальное), при котором не обрезается ничего лишнего
 - `near` – минимальное расстояние до вершин объектов сцены (часто просто хардкодится)
 - `far` – максимальное расстояние до вершин объектов сцены

Проекция для shadow map: точечный источник

- `near` и `far` – максимальное (соответственно, минимальное), при котором не обрезается ничего лишнего
 - `near` – минимальное расстояние до вершин объектов сцены (часто просто хардкодится)
 - `far` – максимальное расстояние до вершин объектов сцены
- Например, `far` можно посчитать как $\max_V |(V - C) \cdot D|$, где
 - C – координаты центра проекции (источника света)
 - D – единичный вектор направления взгляда проекции
 - V – вершины объектов сцены

Проекция для shadow map: точечный источник

- `near` и `far` – максимальное (соответственно, минимальное), при котором не обрезается ничего лишнего
 - `near` – минимальное расстояние до вершин объектов сцены (часто просто хардкодится)
 - `far` – максимальное расстояние до вершин объектов сцены
- Например, `far` можно посчитать как $\max_V |(V - C) \cdot D|$, где
 - C – координаты центра проекции (источника света)
 - D – единичный вектор направления взгляда проекции
 - V – вершины объектов сцены
- Можно заранее вычислить bounding box сцены и вычислять максимум только по его вершинам

Проекция для shadow map: направленный источник

- Одна ортографическая проекция: задаётся центром C и осями X, Y, Z

Проекция для shadow map: направленный источник

- Одна ортографическая проекция: задаётся центром C и осями X, Y, Z
- Z параллельный направлению на источник света

Проекция для shadow map: направленный источник

- Одна ортографическая проекция: задаётся центром C и осями X, Y, Z
- Z параллельный направлению на источник света
- X, Y перпендикулярны Z (и друг другу)
 - Можно выбрать любые, перпендикулярные Z
 - Можно попробовать выбрать их так, чтобы максимизировать эффективное разрешение shadow map (т.е. чтобы одному пикселю shadow map соответствовала как можно меньшая область пространства) – сводится к задаче OBB (oriented bounding box)

Проекция для shadow map: направленный источник

- Одна ортографическая проекция: задаётся центром C и осями X, Y, Z
- Z параллельный направлению на источник света
- X, Y перпендикулярны Z (и друг другу)
 - Можно выбрать любые, перпендикулярные Z
 - Можно попробовать выбрать их так, чтобы максимизировать эффективное разрешение shadow map (т.е. чтобы одному пикселю shadow map соответствовала как можно меньшая область пространства) – сводится к задаче OBB (oriented bounding box)
- Откуда взять C и длины векторов X, Y, Z ?

Проекция для shadow map: направленный источник

- Вычислим bounding box всей сцены

Проекция для shadow map: направленный источник

- Вычислим bounding box всей сцены
- В качестве C возьмём центр bounding box'а сцены

Проекция для shadow map: направленный источник

- Вычислим bounding box всей сцены
- В качестве C возьмём центр bounding box'a сцены
- Мы знаем направление вектора X , нужно вычислить его длину
 - Пусть \hat{X} – вектор направления (нормированный)
 - Мы хотим, чтобы для всех вершин сцены выполнялось
$$|(V - C) \cdot \hat{X}| \leq |X|$$

Проекция для shadow map: направленный источник

- Вычислим bounding box всей сцены
- В качестве C возьмём центр bounding box'a сцены
- Мы знаем направление вектора X , нужно вычислить его длину
 - Пусть \hat{X} – вектор направления (нормированный)
 - Мы хотим, чтобы для всех вершин сцены выполнялось $|(V - C) \cdot \hat{X}| \leq |X|$
 - Пройдёмся по всем восьми вершинам bounding box'a сцены и вычислим $\max_V |(V - C) \cdot \hat{X}|$ – это длина вектора X

Проекция для shadow map: направленный источник

- Вычислим bounding box всей сцены
- В качестве C возьмём центр bounding box'a сцены
- Мы знаем направление вектора X , нужно вычислить его длину
 - Пусть \hat{X} – вектор направления (нормированный)
 - Мы хотим, чтобы для всех вершин сцены выполнялось $|(V - C) \cdot \hat{X}| \leq |X|$
 - Пройдёмся по всем восьми вершинам bounding box'a сцены и вычислим $\max_V |(V - C) \cdot \hat{X}|$ – это длина вектора X
- Аналогично для Y, Z

Проекция для shadow map: направленный источник

- Описанный алгоритм слишком консервативен: в shadow map попадает вообще вся сцена

Проекция для shadow map: направленный источник

- Описанный алгоритм слишком консервативен: в shadow map попадает вообще вся сцена
- Можно взять bounding box области, видимой из основной камеры, и вычислить, что может бросать тень на эту область, и взять проекцию для shadow map, в которую попадают только эти объекты

Проекция для shadow map: направленный источник

- Описанный алгоритм слишком консервативен: в shadow map попадает вообще вся сцена
- Можно взять bounding box области, видимой из основной камеры, и вычислить, что может бросать тень на эту область, и взять проекцию для shadow map, в которую попадают только эти объекты
- Можно подобрать оси X, Y для проекции shadow map так, чтобы максимально плотно уместить видимую область

Проблемы shadow mapping

- В базовом shadow mapping хорошо видны пиксели shadow map (квадратная лесенка на границе тени)

Проблемы shadow mapping

- В базовом shadow mapping хорошо видны пиксели shadow map (квадратная лесенка на границе тени)
 - Чем больше сцена, тем хуже (пиксели становятся больше)

Проблемы shadow mapping

- В базовом shadow mapping хорошо видны пиксели shadow map (квадратная лесенка на границе тени)
 - Чем больше сцена, тем хуже (пиксели становятся больше)
 - Особенно плохо, если тень движется

Проблемы shadow mapping

- В базовом shadow mapping хорошо видны пиксели shadow map (квадратная лесенка на границе тени)
 - Чем больше сцена, тем хуже (пиксели становятся больше)
 - Особенно плохо, если тень движется
- Хочется сгладить тени, убрав пиксели и сделав аппроксимацию мягких теней

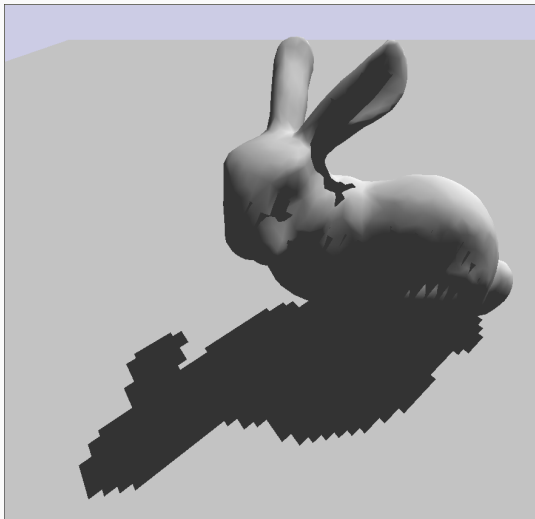
Проблемы shadow mapping

- В базовом shadow mapping хорошо видны пиксели shadow map (квадратная лесенка на границе тени)
 - Чем больше сцена, тем хуже (пиксели становятся больше)
 - Особенно плохо, если тень движется
- Хочется сгладить тени, убрав пиксели и сделав аппроксимацию мягких теней
- Саму shadow map сглаживать (размывать / использовать линейную фильтрацию / mipmaps) нет смысла: получатся интерполированные значения глубин, сравнение с ними даёт видимые артефакты и ничего не улучшает

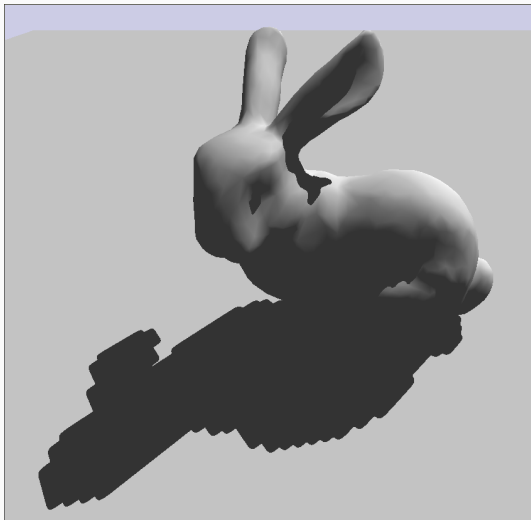
Проблемы shadow mapping

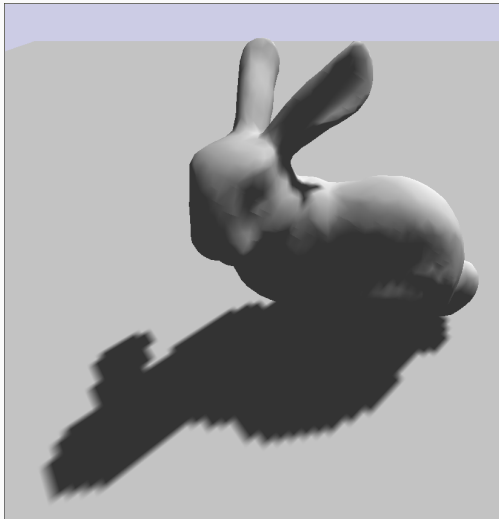
- В базовом shadow mapping хорошо видны пиксели shadow map (квадратная лесенка на границе тени)
 - Чем больше сцена, тем хуже (пиксели становятся больше)
 - Особенно плохо, если тень движется
- Хочется сгладить тени, убрав пиксели и сделав аппроксимацию мягких теней
- Саму shadow map сглаживать (размывать / использовать линейную фильтрацию / mipmaps) нет смысла: получатся интерполированные значения глубин, сравнение с ними даёт видимые артефакты и ничего не улучшает
- Встроенный PCF (shadow samplers) усредняет только по соседним 4 пикселям, иначе приходится делать размытие несепарабельным фильтром

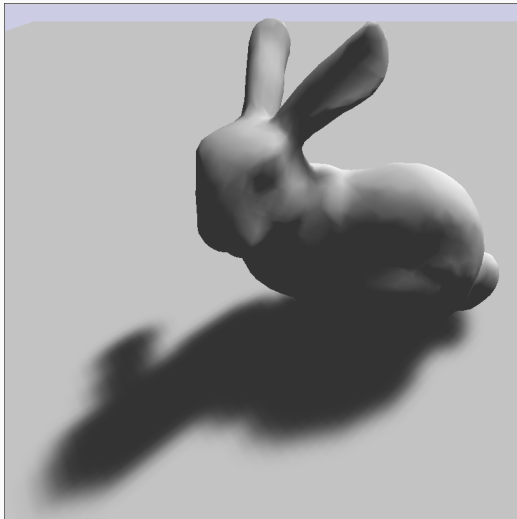
Пиксели shadow mapping



Пиксели shadow mapping (с линейной фильтрацией)







- Хочется, чтобы shadow map содержал значения, которые имеет смысл интерполировать (*фильтровать*)

Filtered shadow maps

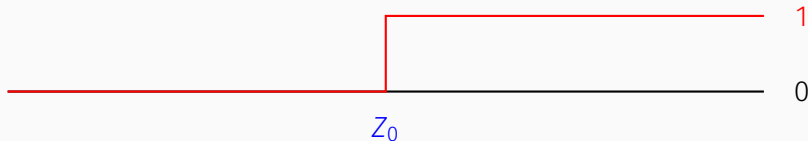
- Хочется, чтобы shadow map содержал значения, которые имеет смысл интерполировать (*фильтровать*)
- Тогда можно применить весь арсенал работы с текстурами: линейную фильтрацию, mipmaps, размытие, etc.

Filtered shadow maps

- Идея: рассматривать значение в одном пикселе shadow map как описание некоего распределения вероятности

Filtered shadow maps

- Идея: рассматривать значение в одном пикселе shadow map как описание некоего распределения вероятности
- График уровня освещённости в зависимости от расстояния до источника света (Z_0 – глубина объекта, бросающего тень)



- Распределения вероятности можно усреднять: если $F_1(z), F_2(z)$ – распределения вероятности, то $F(z) = (1 - t) \cdot F_1(z) + t \cdot F_2(z)$ – тоже распределение вероятности

- Распределения вероятности можно усреднять: если $F_1(z), F_2(z)$ – распределения вероятности, то $F(z) = (1 - t) \cdot F_1(z) + t \cdot F_2(z)$ – тоже распределение вероятности
- Есть много вариантов того, как представить распределение одним пикселем shadow map

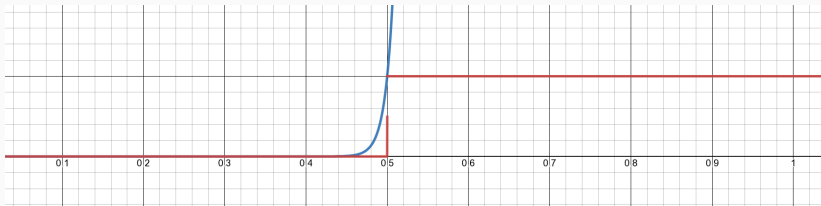
Exponential shadow maps (ESM)

- Идея: функцию распределения можно аппроксимировать экспонентой

$$F(z) = \min(1, \exp(C \cdot (z_0 - z))) = \min(1, \exp(-Cz) \exp(Cz_0))$$

(z – расстояние до освещаемого пикселя, z_0 – расстояние до источника тени)

Exponential shadow maps (ESM)



Exponential shadow maps (ESM)

- Идея: функцию распределения можно аппроксимировать экспонентой

$$F(z) = \min(1, \exp(C \cdot (z_0 - z))) = \min(1, \exp(-Cz) \exp(Cz_0))$$

(z – расстояние до освещаемого пикселя, z_0 – расстояние до источника тени)

Exponential shadow maps (ESM)

- Идея: функцию распределения можно аппроксимировать экспонентой

$$F(z) = \min(1, \exp(C \cdot (z_0 - z))) = \min(1, \exp(-Cz) \exp(Cz_0))$$

(z – расстояние до освещаемого пикселя, z_0 – расстояние до источника тени)

- Интерполяция таких функций сводится к интерполяции $\exp(Cz_0)$

Exponential shadow maps (ESM)

- Идея: функцию распределения можно аппроксимировать экспонентой

$$F(z) = \min(1, \exp(C \cdot (z_0 - z))) = \min(1, \exp(-Cz) \exp(Cz_0))$$

(z – расстояние до освещаемого пикселя, z_0 – расстояние до источника тени)

- Интерполяция таких функций сводится к интерполяции $\exp(Cz_0)$
- Будем записывать в shadow map значение $\exp(Cz_0)$

Exponential shadow maps (ESM)

- Идея: функцию распределения можно аппроксимировать экспонентой

$$F(z) = \min(1, \exp(C \cdot (z_0 - z))) = \min(1, \exp(-Cz) \exp(Cz_0))$$

(z – расстояние до освещаемого пикселя, z_0 – расстояние до источника тени)

- Интерполяция таких функций сводится к интерполяции $\exp(Cz_0)$
- Будем записывать в shadow map значение $\exp(Cz_0)$
- В шейдере будем умножать это значение на $\exp(-Cz)$

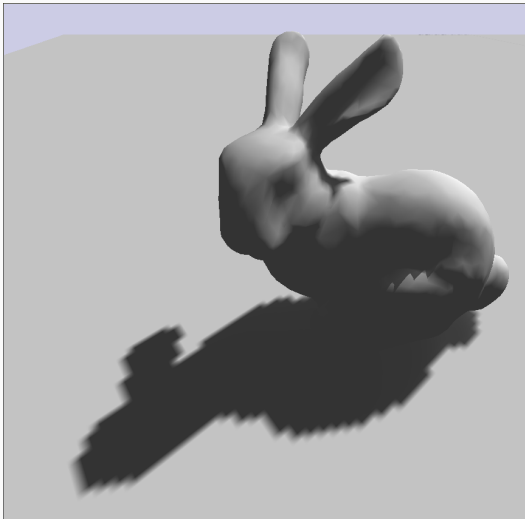
Exponential shadow maps (ESM)

- Идея: функцию распределения можно аппроксимировать экспонентой

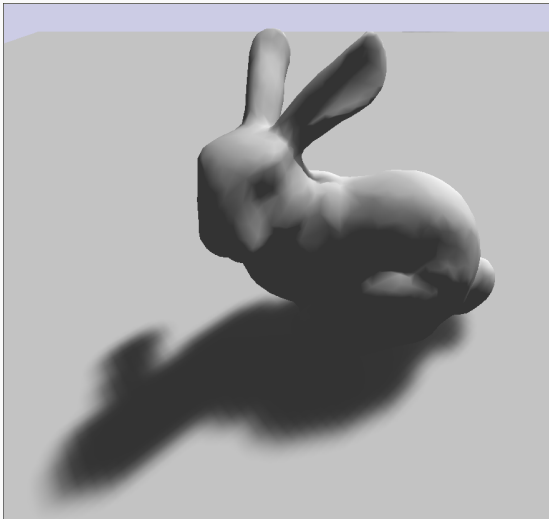
$$F(z) = \min(1, \exp(C \cdot (z_0 - z))) = \min(1, \exp(-Cz) \exp(Cz_0))$$

(z – расстояние до освещаемого пикселя, z_0 – расстояние до источника тени)

- Интерполяция таких функций сводится к интерполяции $\exp(Cz_0)$
- Будем записывать в shadow map значение $\exp(Cz_0)$
- В шейдере будем умножать это значение на $\exp(-Cz)$
- Если результат больше единицы ($z < z_0$), принимаем освещённость за единицу



ESM с линейной фильтрацией и 7x7 размытием по Гауссу



- + Shadow map можно фильтровать (размывать, mipmaps, etc)
- + Прост в реализации
- — Требуется floating-point буфер глубины или рендеринг в floating-point текстуру
- — Константу C приходится делать очень большой \Rightarrow могут быть проблемы с точностью

Variance shadow maps (VSM)

- Идея: запишем в shadow map мат. ожидание глубины и её квадрата

Variance shadow maps (VSM)

- Идея: запишем в shadow map мат. ожидание глубины и её квадрата
- Их можно усреднять:

$$\int z [(1-t)p_1(z) + tp_2(z)] dz = (1-t) \int zp_1(z)dz + t \int zp_2(z)dz \quad (1)$$

- (аналогично для z^2)

Variance shadow maps (VSM)

- Идея: запишем в shadow map мат. ожидание глубины и её квадрата
- Их можно усреднять:

$$\int z [(1-t)p_1(z) + tp_2(z)] dz = (1-t) \int zp_1(z)dz + t \int zp_2(z)dz \quad (1)$$

- (аналогично для z^2)
- Моделируем пиксель как дискретное распределение с единственным значением \Rightarrow мат.ожидание глубины совпадает с её значением (и аналогично для квадрата глубины)

Variance shadow maps (VSM)

- По мат.ожиданию глубины (μ) и её квадрата можно вычислить дисперсию

$$\sigma^2 = \mathbb{E}[z^2] - \mathbb{E}[z]^2 \quad (2)$$

Variance shadow maps (VSM)

- По мат.ожиданию глубины (μ) и её квадрата можно вычислить дисперсию

$$\sigma^2 = \mathbb{E}[z^2] - \mathbb{E}[z]^2 \quad (2)$$

- По мат.ожиданию глубины и дисперсии можно оценить освещённость через одностороннее неравенство Чебышёва (оно же – неравенство Кантелли):

$$P(z) = \frac{\sigma^2}{\sigma^2 + (z - \mu)^2} \quad (3)$$

Variance shadow maps (VSM): алгоритм

- В shadow map записываем глубину и её квадрат (нужна текстура с 2 компонентами)

Variance shadow maps (VSM): алгоритм

- В shadow map записываем глубину и её квадрат (нужна текстура с 2 компонентами)
- По желанию с результирующей текстурой shadow map производим фильтрацию (например, размытие)

Variance shadow maps (VSM): алгоритм

- В shadow map записываем глубину и её квадрат (нужна текстура с 2 компонентами)
- По желанию с результирующей текстурой shadow map производим фильтрацию (например, размытие)
- В шейдере читаем shadow map, вычисляем μ и σ^2

Variance shadow maps (VSM): алгоритм

- В shadow map записываем глубину и её квадрат (нужна текстура с 2 компонентами)
- По желанию с результирующей текстурой shadow map производим фильтрацию (например, размытие)
- В шейдере читаем shadow map, вычисляем μ и σ^2
- Если глубина нашего пикселя меньше μ , освещённость равна 1

Variance shadow maps (VSM): алгоритм

- В shadow map записываем глубину и её квадрат (нужна текстура с 2 компонентами)
- По желанию с результирующей текстурой shadow map производим фильтрацию (например, размытие)
- В шейдере читаем shadow map, вычисляем μ и σ^2
- Если глубина нашего пикселя меньше μ , освещённость равна 1
- Иначе, вычисляем освещённость через неравенство Чебышёва

Variance shadow maps (VSM): shadow bias

- VSM позволяет элегантно включить shadow bias

Variance shadow maps (VSM): shadow bias

- VSM позволяет элегантно включить shadow bias
- Моделируем пиксель как параллелограмм с линейно меняющейся глубиной

Variance shadow maps (VSM): shadow bias

- VSM позволяет элегантно включить shadow bias
- Моделируем пиксель как параллелограмм с линейно меняющейся глубиной
- Градиент глубины можно аппроксимировать с помощью $dFdx$ и $dFdy$ в шейдере

Variance shadow maps (VSM): shadow bias

- VSM позволяет элегантно включить shadow bias
- Моделируем пиксель как параллелограмм с линейно меняющейся глубиной
- Градиент глубины можно аппроксимировать с помощью $dFdx$ и $dFdy$ в шейдере
- Итоговый квадрат глубины вычисляется как

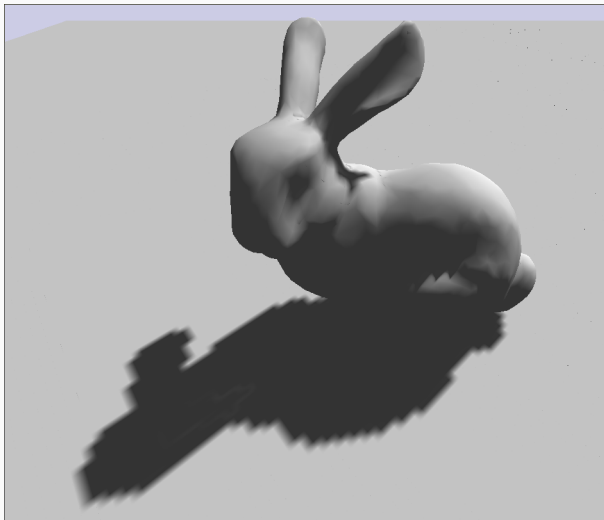
$$z^2 + \frac{1}{4} \left[\left(\frac{\partial z}{\partial x} \right)^2 + \left(\frac{\partial z}{\partial y} \right)^2 \right] \quad (4)$$

Variance shadow maps (VSM): shadow bias

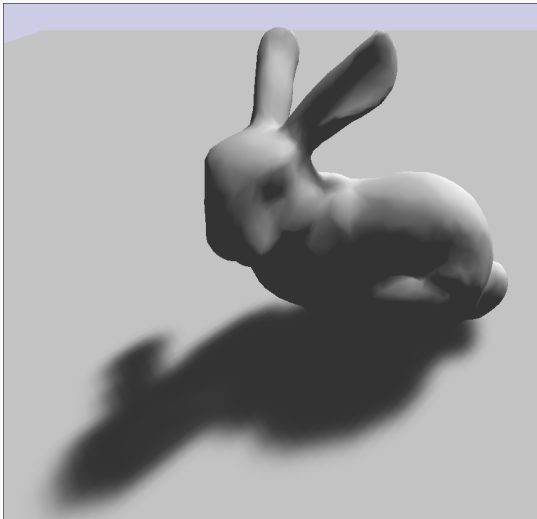
- VSM позволяет элегантно включить shadow bias
- Моделируем пиксель как параллелограмм с линейно меняющейся глубиной
- Градиент глубины можно аппроксимировать с помощью $dFdx$ и $dFdy$ в шейдере
- Итоговый квадрат глубины вычисляется как

$$z^2 + \frac{1}{4} \left[\left(\frac{\partial z}{\partial x} \right)^2 + \left(\frac{\partial z}{\partial y} \right)^2 \right] \quad (4)$$

- Убирает *почти* все артефакты shadow acne



VSM с линейной фильтрацией и 7x7 размытием по Гауссу



- + Shadow map можно фильтровать (размывать, mipmaps, etc)
- + Достаточно прост в реализации
- — Требуется floating-point буфер глубины или рендеринг в floating-point текстуру
- — Возникает артефакт light bleeding

Light bleeding



- Обычно возникает, когда σ^2 оказывается слишком большим (в близкие пиксели попали очень далёкие объекты)

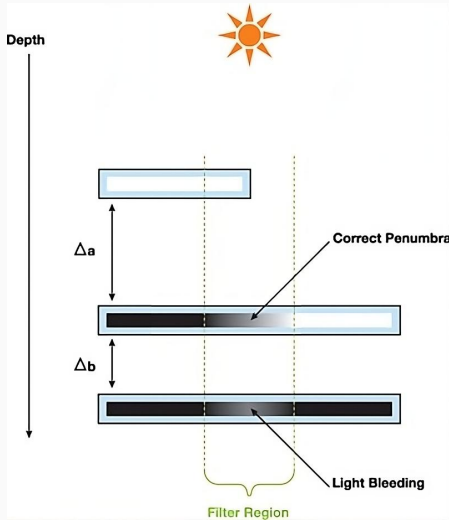
Light bleeding

- Обычно возникает, когда σ^2 оказывается слишком большим (в близкие пиксели попали очень далёкие объекты)
- Можно частично исправить, трактуя маленькие значения коэффициента освещённости как нулевые, и преобразовав остальные значения в диапазон $[0, 1]$

Light bleeding

- Обычно возникает, когда σ^2 оказывается слишком большим (в близкие пиксели попали очень далёкие объекты)
- Можно частично исправить, трактуя маленькие значения коэффициента освещённости как нулевые, и преобразовав остальные значения в диапазон $[0, 1]$
- Конкретные значения нужно подбирать в зависимости от сцены

Light bleeding



Convolution shadow maps (CSM)

- Записывает в shadow map коэффициенты преобразования Фурье от функции распределения

Convolution shadow maps (CSM)

- Записывает в shadow map коэффициенты преобразования Фурье от функции распределения
- + Можно фильтровать (размывать, mipmaps, etc)
- — Сложнее в реализации
- — Требуется многокомпонентных буферов (часто до 16 компонент на пиксель)
- — Возникают ringing-артефакты (см. Gibbs phenomenon)

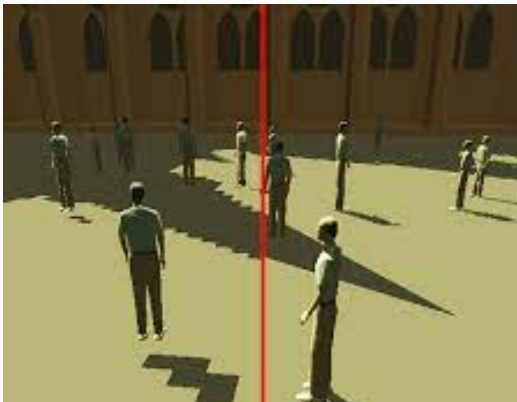
- ogldev.org/www/tutorial42/tutorial42.html
- learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping
- docs.microsoft.com/en-us/windows/win32/dxtecharts/common-techniques-to-improve-shadow-depth-maps
- Оригинальная статья про ESM
- Оригинальная статья про VSM
- Улучшения VSM, включая более реалистичные мягкие тени

- Если сцена очень большая, даже огромных (8k) shadow map текстур с хорошим сглаживанием может не хватить

- Если сцена очень большая, даже огромных (8k) shadow map текстур с хорошим сглаживанием может не хватить
- Классически есть два решения:
 - Использовать неравномерное соответствие пикселей shadow map и объектов в мире
 - Использовать более одной shadow map

Perspective shadow maps (PSM)

- При генерации shadow map помимо обычного преобразования вершин применяет ещё перспективное преобразование так, чтобы больше пикселей shadow map приходилось на близкие к камере объекты

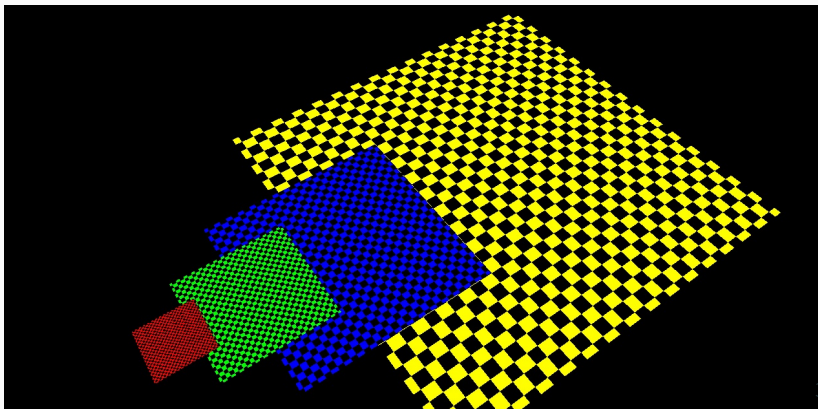


Perspective shadow maps (PSM)

- + Часто решает проблему точности shadow map
- — Сложен в реализации
- — Очень много худших случаев (например, камера смотрит на источник света), в которых алгоритм почти не работает

Cascaded shadow maps (CSM)

- Будем использовать несколько shadow maps (*cascades*): одну – для ближайших объектов, другую – для объектов на среднем расстоянии, третью – для далёких объектов, и т.п.



Cascaded shadow maps (CSM)

- + Прост в реализации
- + Легко комбинировать с VSM/ESM/etc
- — Большой расход памяти
- — Нужно аккуратно обработать переход между разными каскадами
- Самый часто использующийся сегодня алгоритм, поддерживается большинством движков

- Оригинальная статья про PSM
- PSM + улучшения
- Статья про CSM
- ogldev.org/www/tutorial49/tutorial49.html
- learnopengl.com/Guest-Articles/2021/CSM

- Ambient освещение светит 'отовсюду' – эвристическая аппроксимация многократных отражений в уравнении рендеринга

- Ambient освещение светит 'отовсюду' – эвристическая аппроксимация многократных отражений в уравнении рендеринга
- Это свет, значит от него может быть тень

- Ambient освещение светит 'отовсюду' – эвристическая аппроксимация многократных отражений в уравнении рендеринга
- Это свет, значит от него может быть тень
- Какие-то части сцены получают больше ambient света, какие-то – меньше

- Ambient освещение светит 'отовсюду' – эвристическая аппроксимация многократных отражений в уравнении рендеринга
- Это свет, значит от него может быть тень
- Какие-то части сцены получают больше ambient света, какие-то – меньше
- Обычно затеняются углы, углубления, трещины, стыки объектов, и т.п.

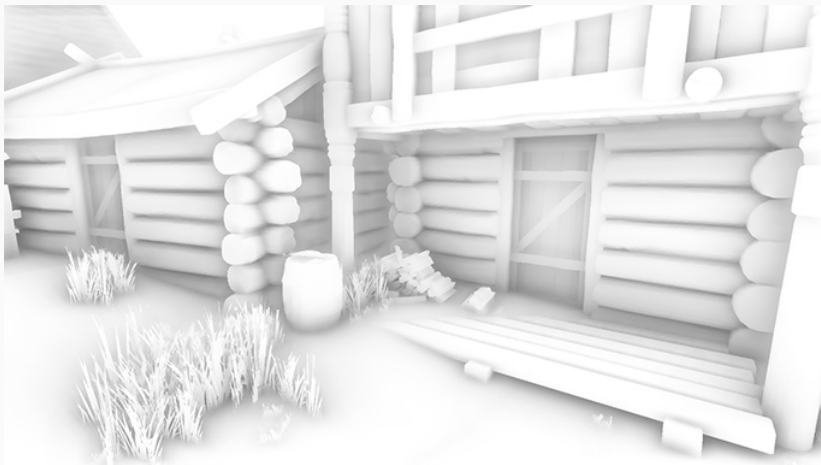
Ambient occlusion

- Ambient освещение светит ‘отовсюду’ – эвристическая аппроксимация многократных отражений в уравнении рендеринга
- Это свет, значит от него может быть тень
- Какие-то части сцены 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)



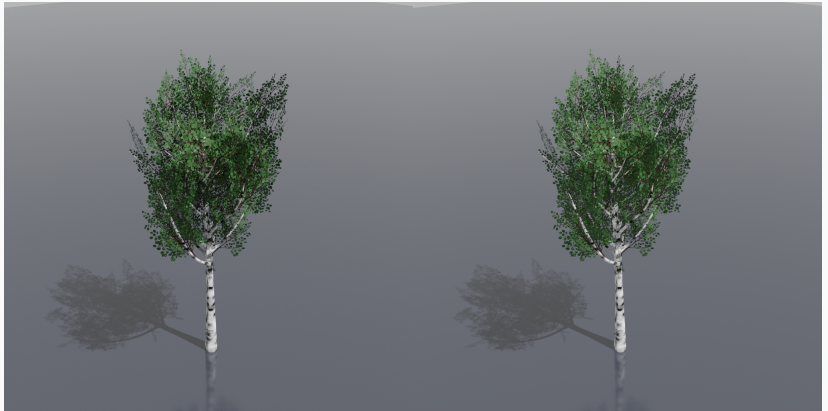
Ambient occlusion



Ambient occlusion



Ambient occlusion



- Baking: предподсчитать ambient occlusion для модели, записать её в одноканальную текстуру и использовать в шейдере как множитель при ambient освещении

- Baking: предподсчитать ambient occlusion для модели, записать её в одноканальную текстуру и использовать в шейдере как множитель при ambient освещении
 - + Быстро работает
 - — Не учитывает occlusion между разными объектами
 - — Не работает для анимированных объектов

Ambient occlusion



Original model



With ambient occlusion



Extracted ambient occlusion map

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

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

- Вычислять ambient occlusion на лету, используя какие-нибудь эвристики (алгоритмы SSAO, HBAO, HBAO+, HDAO, VXAO, ...)

- Вычислять ambient occlusion на лету, используя какие-нибудь эвристики (алгоритмы SSAO, HBAO, HBAO+, HDAO, VXAO, ...)
 - + Работает с любыми сценами
 - — Сложнее в реализации
 - — Медленнее работает
 - — Обычно требуют подгона параметров под сцену

- Screen-space ambient occlusion (SSAO) – один из первых real-time алгоритмов для генерации ambient occlusion

- Screen-space ambient occlusion (SSAO) – один из первых real-time алгоритмов для генерации ambient occlusion
- Термин *screen-space* означает, что алгоритм использует только те данные, которые попадают на экран в процессе рендеринга (т.е. не использует саму исходную геометрию, не использует рендеринг в текстуру с другого ракурса, и т.п.)

- Screen-space ambient occlusion (SSAO) – один из первых real-time алгоритмов для генерации ambient occlusion
- Термин *screen-space* означает, что алгоритм использует только те данные, которые попадают на экран в процессе рендеринга (т.е. не использует саму исходную геометрию, не использует рендеринг в текстуру с другого ракурса, и т.п.)
- Идея: вычислим пересечение полусферы некоторого радиуса в освещаемой точке с окружающей геометрией, объём этого пересечения – аппроксимация ambient occlusion (чем больше вокруг пикселя других объектов, тем слабее он освещён)

SSAO (Crytek, 2007)



- Вычислять настоящее пересечение – дорого, вместо этого аппроксимируем его монте-карло интегрированием: возьмём набор случайных точек в полусфере вокруг освещаемой точки и сравним их глубину со значением из Z-буфера

- Вычислять настоящее пересечение – дорого, вместо этого аппроксимируем его монте-карло интегрированием: возьмём набор случайных точек в полусфере вокруг освещаемой точки и сравним их глубину со значением из Z-буфера
- Обычно набор случайных точек фиксирован, передаётся в виде небольшой текстуры, где в цветах пикселей записаны координаты точек (относительно исходной точки)

- Вычислять настоящее пересечение – дорого, вместо этого аппроксимируем его монте-карло интегрированием: возьмём набор случайных точек в полусфере вокруг освещаемой точки и сравним их глубину со значением из Z-буфера
- Обычно набор случайных точек фиксирован, передаётся в виде небольшой текстуры, где в цветах пикселей записаны координаты точек (относительно исходной точки)
- Использование везде одного и того же набора точек приводит к *banding*-у

SSAO banding

- Banding – ситуация, когда хорошо видна дискретизация цветов

SSAO banding

- Banding – ситуация, когда хорошо видна дискретизация цветов
- В SSAO источник banding'a – монте-карло интегрирование: отношение $\frac{K}{N}$ (K – количество видимых точек, N – всех точек) может принимать небольшой набор (N+1) дискретных значений

SSAO banding

- Banding – ситуация, когда хорошо видна дискретизация цветов
- В SSAO источник banding'a – монте-карло интегрирование: отношение $\frac{K}{N}$ (K – количество видимых точек, N – всех точек) может принимать небольшой набор (N+1) дискретных значений
- Можно вращать набор случайных точек вокруг нормали освещаемой точки на угол, случайный для каждого освещаемого пикселя (угол тоже передаётся текстурой или вычисляется псевдо-случайным на основе координат пикселя)

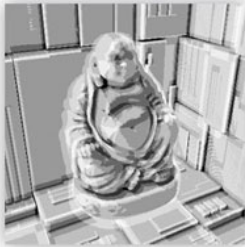
SSAO banding

- Banding – ситуация, когда хорошо видна дискретизация цветов
- В SSAO источник banding'a – монте-карло интегрирование: отношение $\frac{K}{N}$ (K – количество видимых точек, N – всех точек) может принимать небольшой набор (N+1) дискретных значений
- Можно вращать набор случайных точек вокруг нормали освещаемой точки на угол, случайный для каждого освещаемого пикселя (угол тоже передаётся текстурой или вычисляется псевдо-случайным на основе координат пикселя)
- Приводит к 'шуму' в изображении, который менее заметен и который можно размыть

SSAO banding

- Banding – ситуация, когда хорошо видна дискретизация цветов
- В SSAO источник banding'a – монте-карло интегрирование: отношение $\frac{K}{N}$ (K – количество видимых точек, N – всех точек) может принимать небольшой набор (N+1) дискретных значений
- Можно вращать набор случайных точек вокруг нормали освещаемой точки на угол, случайный для каждого освещаемого пикселя (угол тоже передаётся текстурой или вычисляется псевдо-случайным на основе координат пикселя)
- Приводит к 'шуму' в изображении, который менее заметен и который можно размыть
- Разменять banding на шум – *классический приём* real-time графики

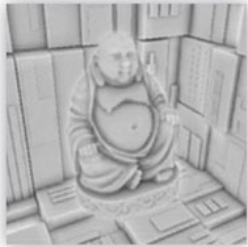
SSAO banding



low sample 'banding'



random rotation = noise



+ blur = acceptable

- Оригинальная статья про SSAO
- learnopengl.com/Advanced-Lighting/SSAO