Компьютерная графика

Лекция 10: Вычисление проекции для shadow mapping, PCF, ESM, VSM, PSM, CSM

2021

Проекция для shadow map: точечный источник

- 6 перспективных проекций для 6 граней cubemap-текстуры shadow map

Проекция для shadow map: точечный источник

- 6 перспективных проекций для 6 граней cubemap-текстуры shadow map
- 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 cubemap-текстуры shadow map
- Каждая проекция смотрит в направлении некоторой оси: $\pm X, \pm Y, \pm Z$
- Угол видимой области - 90° по обеим осям (квадратная камера)

Проекция для shadow map: точечный источник

- 6 перспективных проекций для 6 граней cubemap-текстуры shadow map
- Каждая проекция смотрит в направлении некоторой оси: $\pm X, \pm Y, \pm Z$
- Угол видимой области - 90° по обеим осям (квадратная камера)
- `near` и `far` - максимальное (соответственно, минимальное), при котором не обрезается ничего лишнего

Проекция для shadow map: точечный источник

- 6 перспективных проекций для 6 граней cubemap-текстуры shadow map
- Каждая проекция смотрит в направлении некоторой оси: $\pm X, \pm Y, \pm Z$
- Угол видимой области - 90° по обеим осям (квадратная камера)
- `near` и `far` - максимальное (соответственно, минимальное), при котором не обрезается ничего лишнего
  - `near` - минимальное расстояние до вершины объекта сцены
  - `far` - максимальное расстояние до вершины объекта сцены

Проекция для shadow map: точечный источник

- 6 перспективных проекций для 6 граней cubemap-текстуры shadow map
- Каждая проекция смотрит в направлении некоторой оси: $\pm X, \pm Y, \pm Z$
- Угол видимой области - 90° по обеим осям (квадратная камера)
- `near` и `far` - максимальное (соответственно, минимальное), при котором не обрезается ничего лишнего
  - `near` - минимальное расстояние до вершины объекта сцены
  - `far` - максимальное расстояние до вершины объекта сцены
  - `far` можно вычислить, зная bounding box'ы всех объектов сцены

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'а 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'а 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'а 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'а 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'а сцены и вычислим $\max\limits_{V} |(V - C) \cdot \hat{X}|$ - это длина вектора X
- Аналогично для Y, Z

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)
  - Тем хуже, чем больше сцена (пиксели становятся больше)
  - Особенно плохо, если тень движется

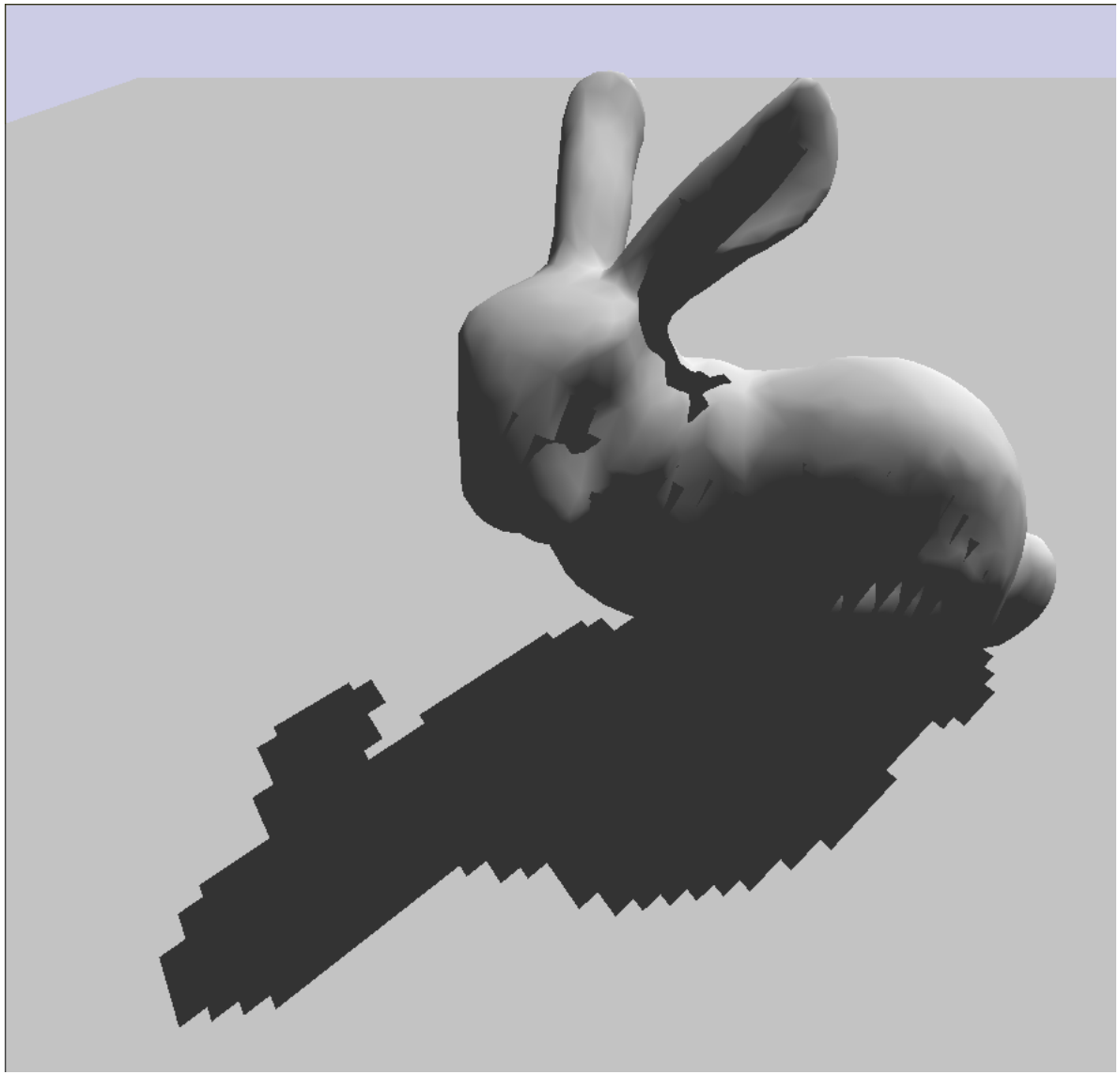
Проблемы shadow mapping

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

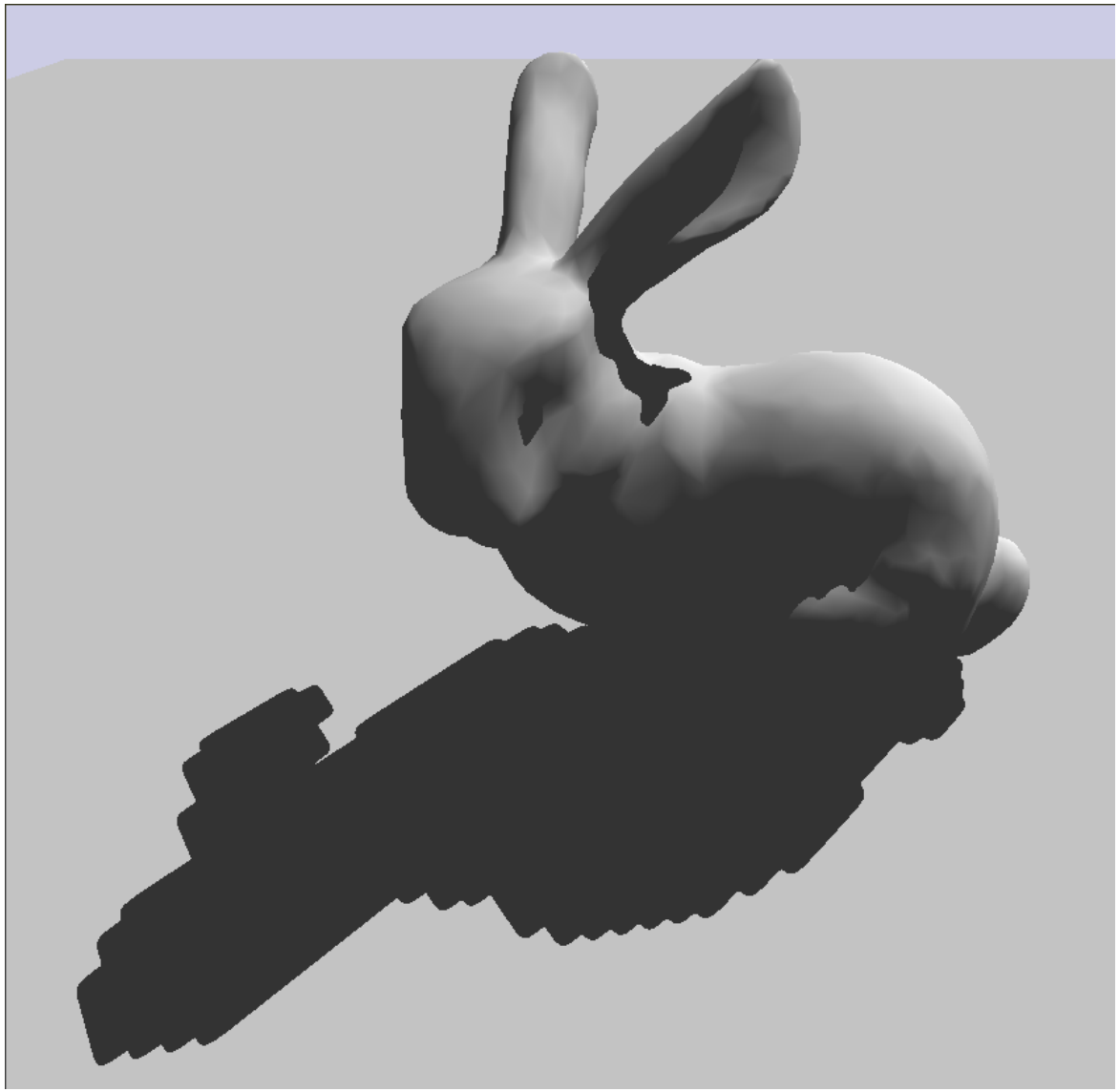
Проблемы shadow mapping

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

Пиксели shadow mapping



Пиксели shadow mapping (с линейной фильтрацией)



Percentage-closer filtering (PCF)

- Shadow mapping выдаёт 0 или 1 в зависимости от того, находится ли текущий пиксель в тени

Percentage-closer filtering (PCF)

- Shadow mapping выдаёт 0 или 1 в зависимости от того, находится ли текущий пиксель в тени
- PCF: прочитаем не один пиксель из shadow map, а несколько, и проинтерполируем результаты сравнения

Percentage-closer filtering (PCF)

- Shadow mapping выдаёт 0 или 1 в зависимости от того, находится ли текущий пиксель в тени
- PCF: прочитаем не один пиксель из shadow map, а несколько, и проинтерполируем результаты сравнения
- Можно взять много соседних пикселей и усреднить результат сравнения по Гауссу
- Можно взять четыре соседних пикселя и проинтерполировать между ними

Percentage-closer filtering (PCF): пример кода

```
float shadow_factor = 0.0;
const int N = 3;
vec2 shadow_map_size = vec2(textureSize(shadow_map, 0));
float total = float((2 * N + 1) * (2 * N + 1));
for (int dx = -N; dx <= N; ++dx) {
    for (int dy = -N; dy <= N; ++dy) {
        if (texture(shadow_map, shadow_pos.xy
            + vec2(dx, dy) / shadow_map_size).r > shadow_pos.z)
            shadow_factor += 1.0 / total;
    }
}
```

Shadow sampler

- В OpenGL есть ограниченная поддержка PCF в виде shadow sampler'ов

Shadow sampler

- В OpenGL есть ограниченная поддержка PCF в виде shadow sampler'ов
- Для текстуры нужно настроить параметры:
  - `GL_TEXTURE_COMPARE_MODE` в значение `GL_COMPARE_REF_TO_TEXTURE` (по умолчанию - `GL_NONE`)
  - `GL_TEXTURE_COMPARE_FUNC` в значение `GL_LESS` (теоретически могут быть `GL_ALWAYS` и т.п.)

Shadow sampler

- В OpenGL есть ограниченная поддержка PCF в виде shadow sampler'ов
- Для текстуры нужно настроить параметры:
  - `GL_TEXTURE_COMPARE_MODE` в значение `GL_COMPARE_REF_TO_TEXTURE` (по умолчанию - `GL_NONE`)
  - `GL_TEXTURE_COMPARE_FUNC` в значение `GL_LESS` (теоретически могут быть `GL_ALWAYS` и т.п.)
- Текстуру с `GL_COMPARE_REF_TO_TEXTURE` нельзя использовать как `sampler2D`, для неё есть отдельный тип sampler'а: `sampler2DShadow`

Shadow sampler

- В OpenGL есть ограниченная поддержка PCF в виде shadow sampler'ов
- Для текстуры нужно настроить параметры:
  - `GL_TEXTURE_COMPARE_MODE` в значение `GL_COMPARE_REF_TO_TEXTURE` (по умолчанию - `GL_NONE`)
  - `GL_TEXTURE_COMPARE_FUNC` в значение `GL_LESS` (теоретически могут быть `GL_ALWAYS` и т.п.)
- Текстуру с `GL_COMPARE_REF_TO_TEXTURE` нельзя использовать как `sampler2D`, для неё есть отдельный тип sampler'а: `sampler2DShadow`
- Функция `texture(shadow_map, texcoord)` принимает трёхмерный вектор текстурных координат:
  - Первые две - обычные X,Y текстурные координаты
  - Третья - используется для сравнения со значением в shadow map

Shadow sampler

- В OpenGL есть ограниченная поддержка PCF в виде shadow sampler'ов
- Для текстуры нужно настроить параметры:
  - `GL_TEXTURE_COMPARE_MODE` в значение `GL_COMPARE_REF_TO_TEXTURE` (по умолчанию - `GL_NONE`)
  - `GL_TEXTURE_COMPARE_FUNC` в значение `GL_LESS` (теоретически могут быть `GL_ALWAYS` и т.п.)
- Текстуру с `GL_COMPARE_REF_TO_TEXTURE` нельзя использовать как `sampler2D`, для неё есть отдельный тип sampler'а: `sampler2DShadow`
- Функция `texture(shadow_map, texcoord)` принимает трёхмерный вектор текстурных координат:
  - Первые две - обычные X,Y текстурные координаты
  - Третья - используется для сравнения со значением в shadow map
- Если сравнение успешно, возвращается 1, иначе 0 (функция сравнения настраивается `GL_TEXTURE_COMPARE_FUNC`)

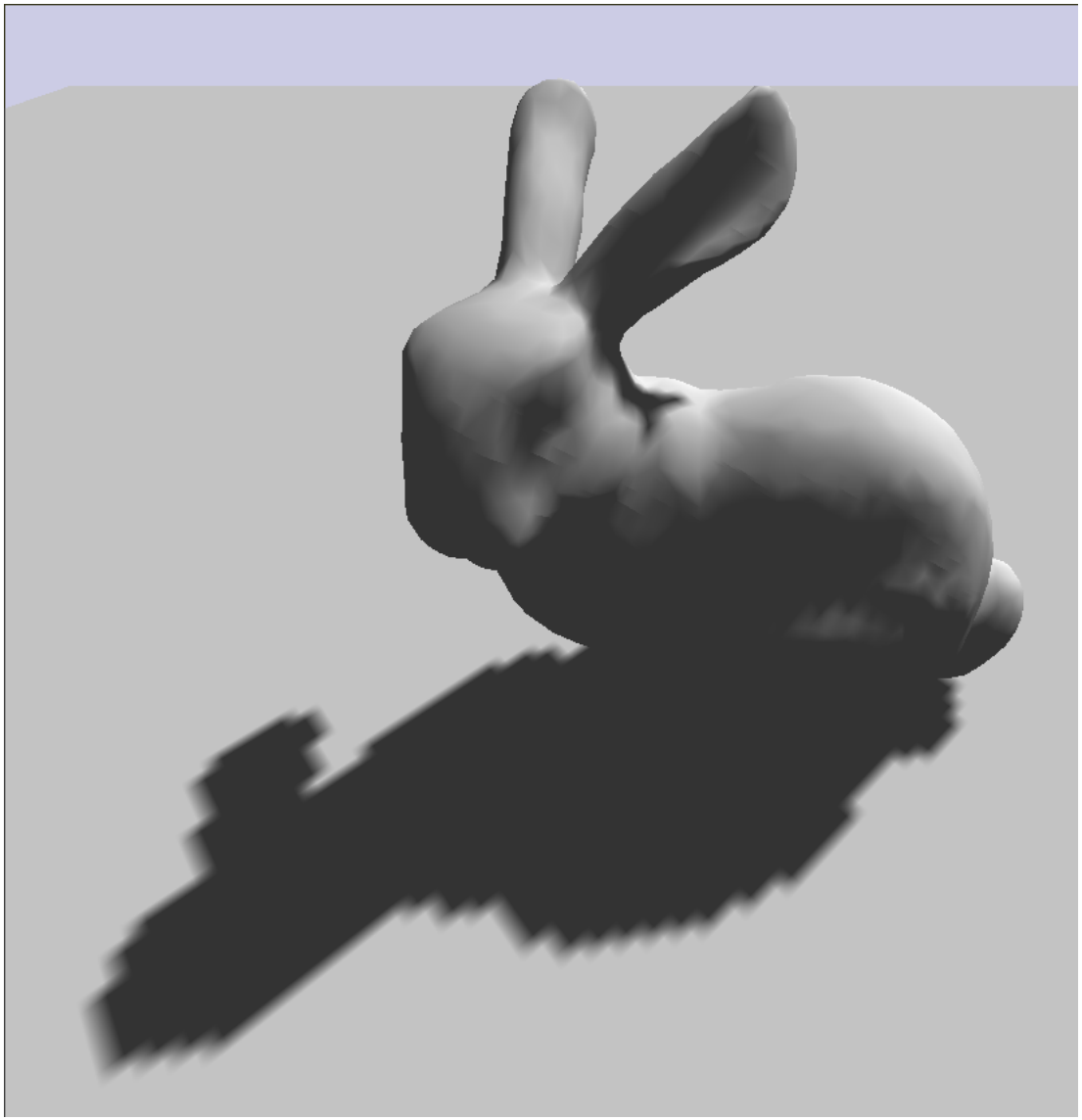
Shadow sampler

- В OpenGL есть ограниченная поддержка PCF в виде shadow sampler'ов
- Для текстуры нужно настроить параметры:
  - `GL_TEXTURE_COMPARE_MODE` в значение `GL_COMPARE_REF_TO_TEXTURE` (по умолчанию - `GL_NONE`)
  - `GL_TEXTURE_COMPARE_FUNC` в значение `GL_LESS` (теоретически могут быть `GL_ALWAYS` и т.п.)
- Текстуру с `GL_COMPARE_REF_TO_TEXTURE` нельзя использовать как `sampler2D`, для неё есть отдельный тип sampler'а: `sampler2DShadow`
- Функция `texture(shadow_map, texcoord)` принимает трёхмерный вектор текстурных координат:
  - Первые две - обычные X,Y текстурные координаты
  - Третья - используется для сравнения со значением в shadow map
- Если сравнение успешно, возвращается 1, иначе 0 (функция сравнения настраивается `GL_TEXTURE_COMPARE_FUNC`)
- Для такой текстуры можно включить линейную фильтрацию - интерполироваться будут не значения глубины, а результаты сравнения

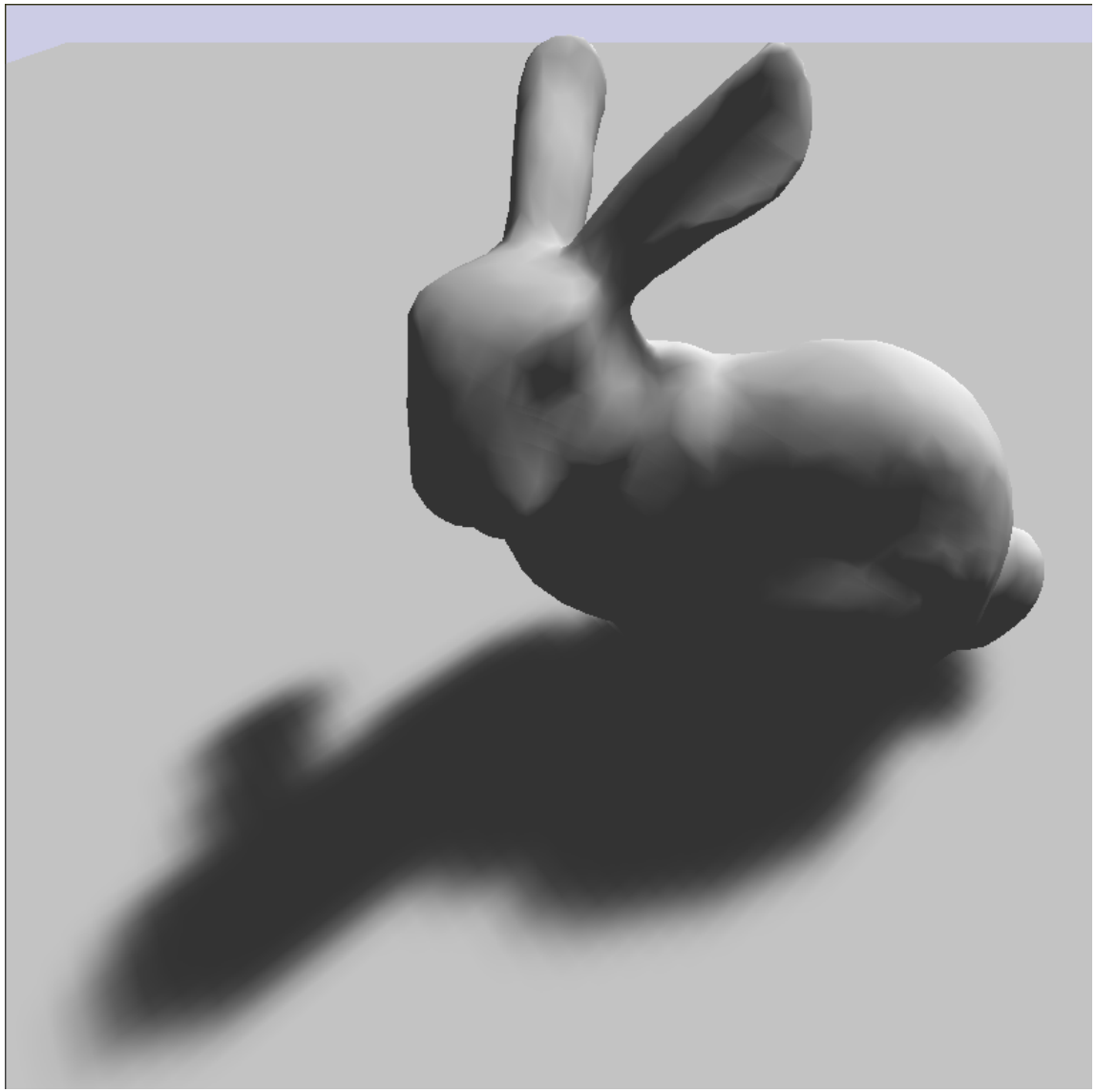
Shadow sampler: пример кода

```
float shadow_factor = texture(shadow_map,
    shadow_pos.xyz);
```

Shadow sampler с линейной фильтрацией



Shadow sampler с линейной фильтрацией и 7x7 размытием по Гауссу



Проблемы PCF

- + Есть аппаратная поддержка (shadow sampler)
- — Аппаратная поддержка даёт только интерполяцию между 4 соседними пикселями
- — Размытие всё равно приходится вычислять для каждого пикселя сцены уже при расчёте освещения - дорого и нельзя применить сепарабельные фильтры

Filtered shadow maps

- Хочется, чтобы shadow map содержал значения, которые имеет смысл интерполировать

Filtered shadow maps

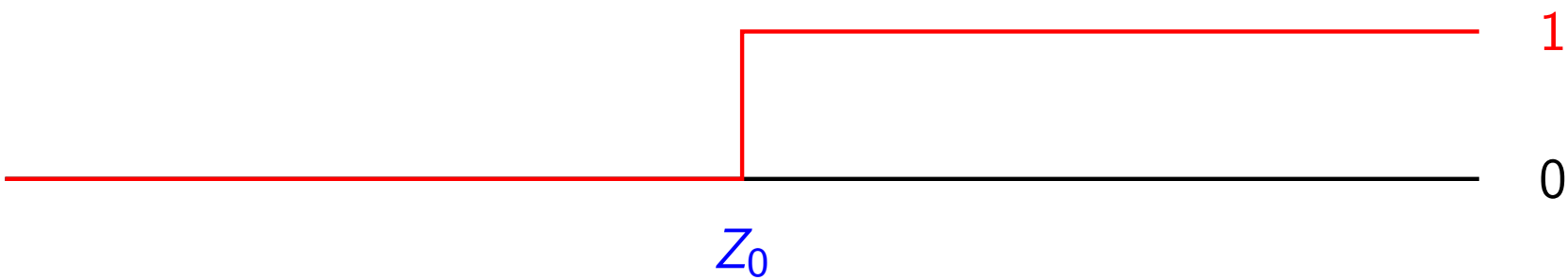
- Хочется, чтобы shadow map содержал значения, которые имеет смысл интерполировать
- Тогда можно применить весь арсенал работы с текстурами: линейную фильтрацию, mipmaps, размытие, etc.

Filtered shadow maps

- Идея: рассматривать значение в одном пикселе shadow map как описывающем некое распределение вероятности

Filtered shadow maps

- Идея: рассматривать значение в одном пикселе shadow map как описывающем некое распределение вероятности
- График уровня освещённости в зависимости от расстояния до источника света:



Filtered shadow maps

- 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

Filtered shadow maps

- 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 $F(z) = \exp(C \cdot (z_0 - z)) = \exp(-Cz)\exp(Cz_0)$ (z - расстояние до освещаемого пикселя, z_0 - расстояние до источника тени)

Exponential shadow maps (ESM)

- Идея: функцию распределения можно аппроксимировать экспонентой $F(z) = \exp(C \cdot (z_0 - z)) = \exp(-Cz)\exp(Cz_0)$ (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 $F(z) = \exp(C \cdot (z_0 - z)) = \exp(-Cz)\exp(Cz_0)$ (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)

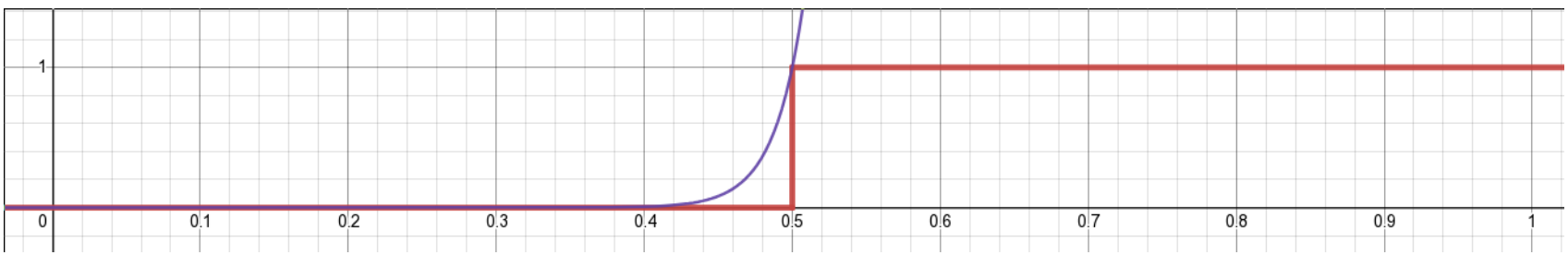
- Идея: функцию распределения можно аппроксимировать экспонентой $F(z) = \exp(C \cdot (z_0 - z)) = \exp(-Cz)\exp(Cz_0)$ (z - расстояние до освещаемого пикселя, z_0 - расстояние до источника тени)
- Интерполяция таких функций сводится к интерполяции $\exp(Cz_0)$
- Будем записывать в shadow map значение $\exp(Cz_0)$
- В шейдере будем умножать это значение на $\exp(-Cz)$

Exponential shadow maps (ESM)

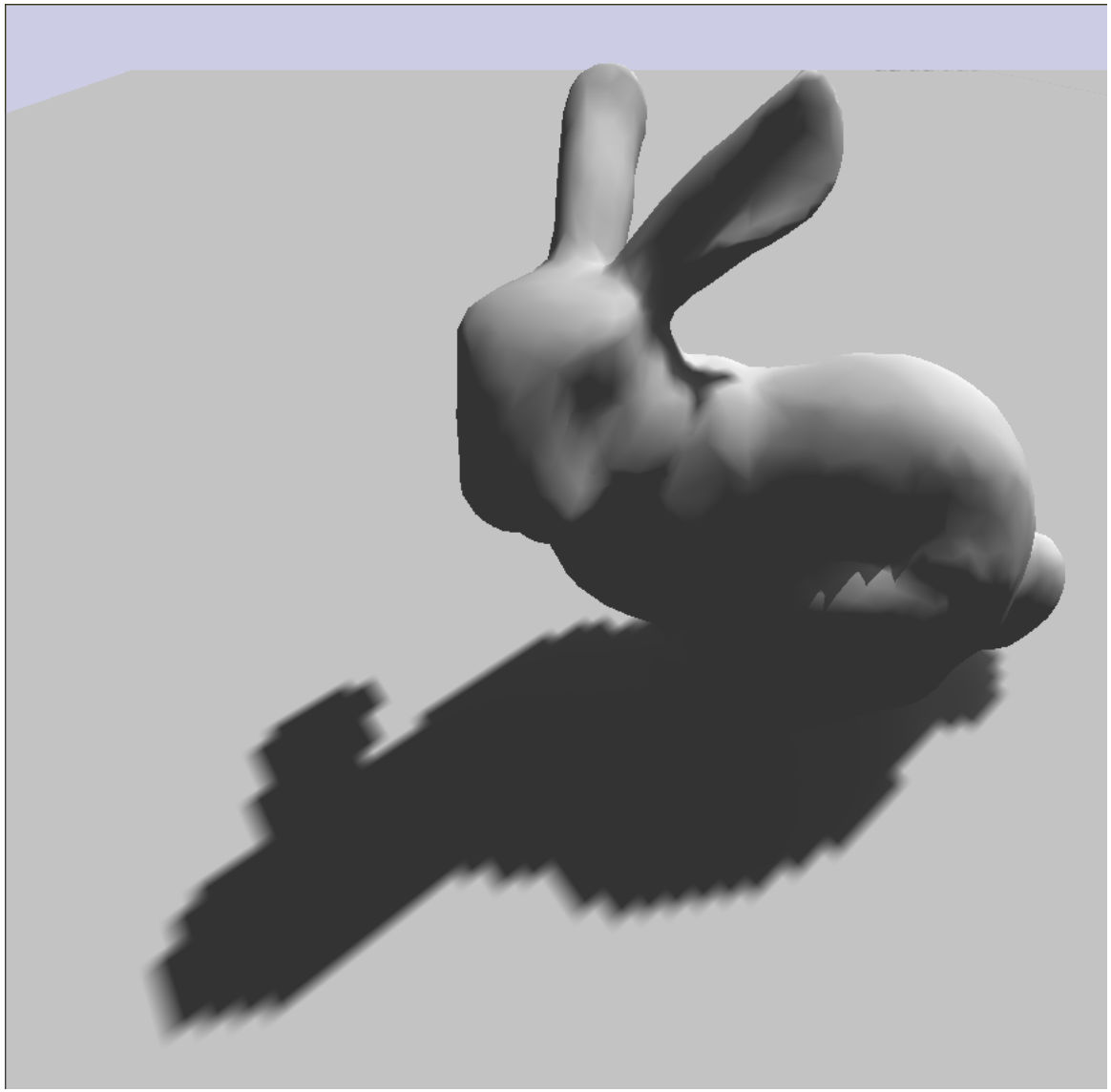
- Идея: функцию распределения можно аппроксимировать экспонентой $F(z) = \exp(C \cdot (z_0 - z)) = \exp(-Cz)\exp(Cz_0)$ (z - расстояние до освещаемого пикселя, z_0 - расстояние до источника тени)
- Интерполяция таких функций сводится к интерполяции $\exp(Cz_0)$
- Будем записывать в shadow map значение $\exp(Cz_0)$
- В шейдере будем умножать это значение на $\exp(-Cz)$
- Если результат больше единицы ($z < z_0$), принимаем освещённость за единицу

Exponential shadow maps (ESM)

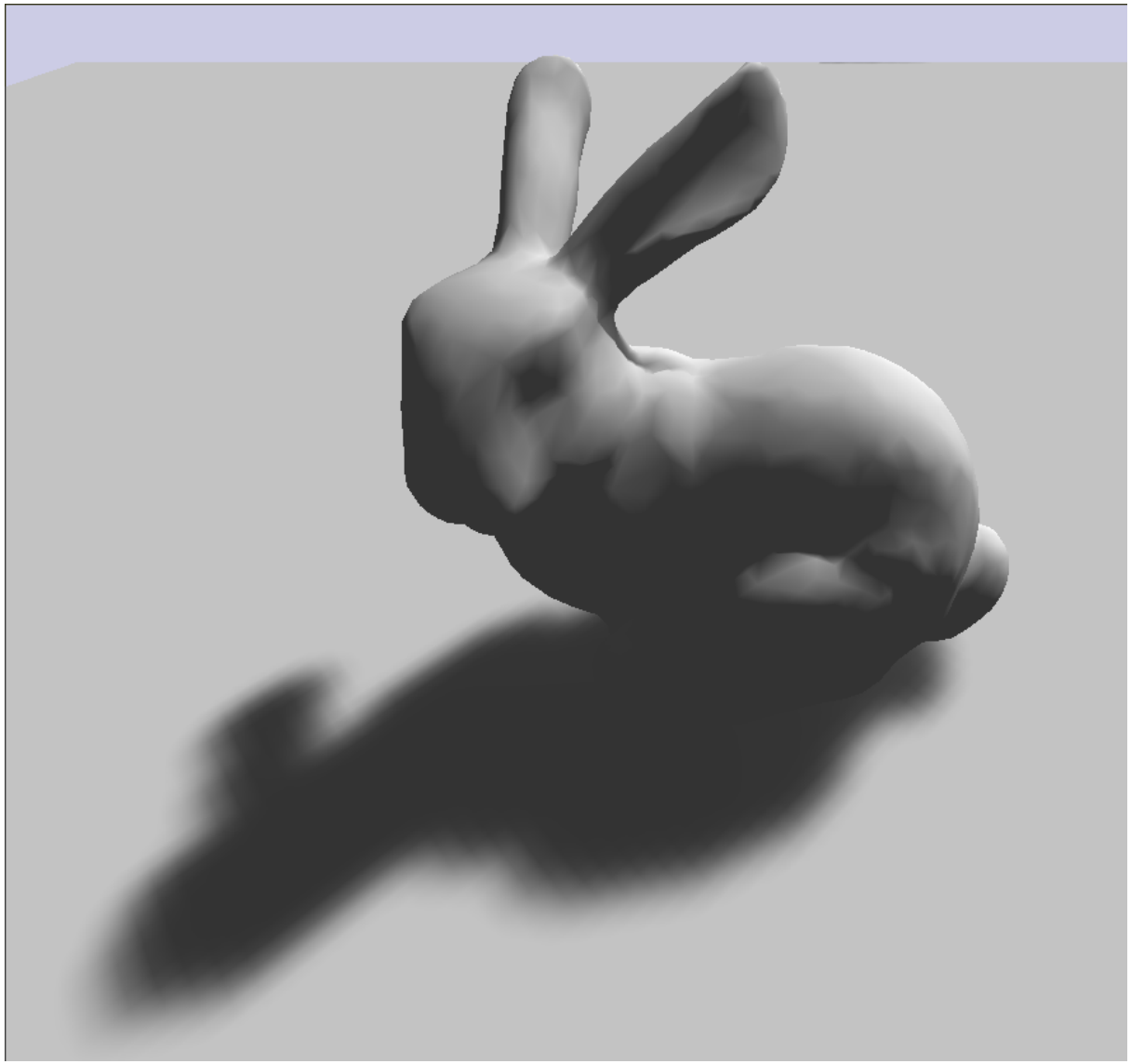
- Идея: функцию распределения можно аппроксимировать экспонентой $F(z) = \exp(C \cdot (z_0 - z)) = \exp(-Cz)\exp(Cz_0)$ (z - расстояние до освещаемого пикселя, z_0 - расстояние до источника тени)
- Интерполяция таких функций сводится к интерполяции $\exp(Cz_0)$
- Будем записывать в shadow map значение $\exp(Cz_0)$
- В шейдере будем умножать это значение на $\exp(-Cz)$
- Если результат больше единицы ($z < z_0$), принимаем освещённость за единицу



ESM с линейной фильтрацией



ESM с линейной фильтрацией и 7x7 размытием по Гауссу



ESM

- + Shadow map можно фильтровать (размывать, mipmaps, etc)
- + Прост в реализации
- — Требует floating-point буфер глубины или рендеринг в floating-point текстуру
- — При фильтрации возникают артефакты из-за того, что результирующая освещённость может быть больше 1

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:

$$\int z\left[(1-t)p_1(z)+tp_2(z)\right]dz = (1-t)\int zp_1(z)dz + t\int zp_2(z)dz \tag{1}$$

- (аналогично для z^2)

Variance shadow maps (VSM)

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

$$\int z\left[(1-t)p_1(z)+tp_2(z)\right]dz=(1-t)\int zp_1(z)dz+t\int zp_2(z)dz \tag{1}$$

- (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

$$\sigma^2 = \mathbb{E}[z^2] - \mathbb{E}[z]^2 \tag{2}$$

Variance shadow maps (VSM)

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

$$\sigma^2 = \mathbb{E}[z^2] - \mathbb{E}[z]^2 \tag{2}$$

- По мат.ожиданию глубины и дисперсии можно вычислить освещённость через неравенство Чебышёва:

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

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 (e.g. 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 (e.g. 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 (e.g. 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 (e.g. 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

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

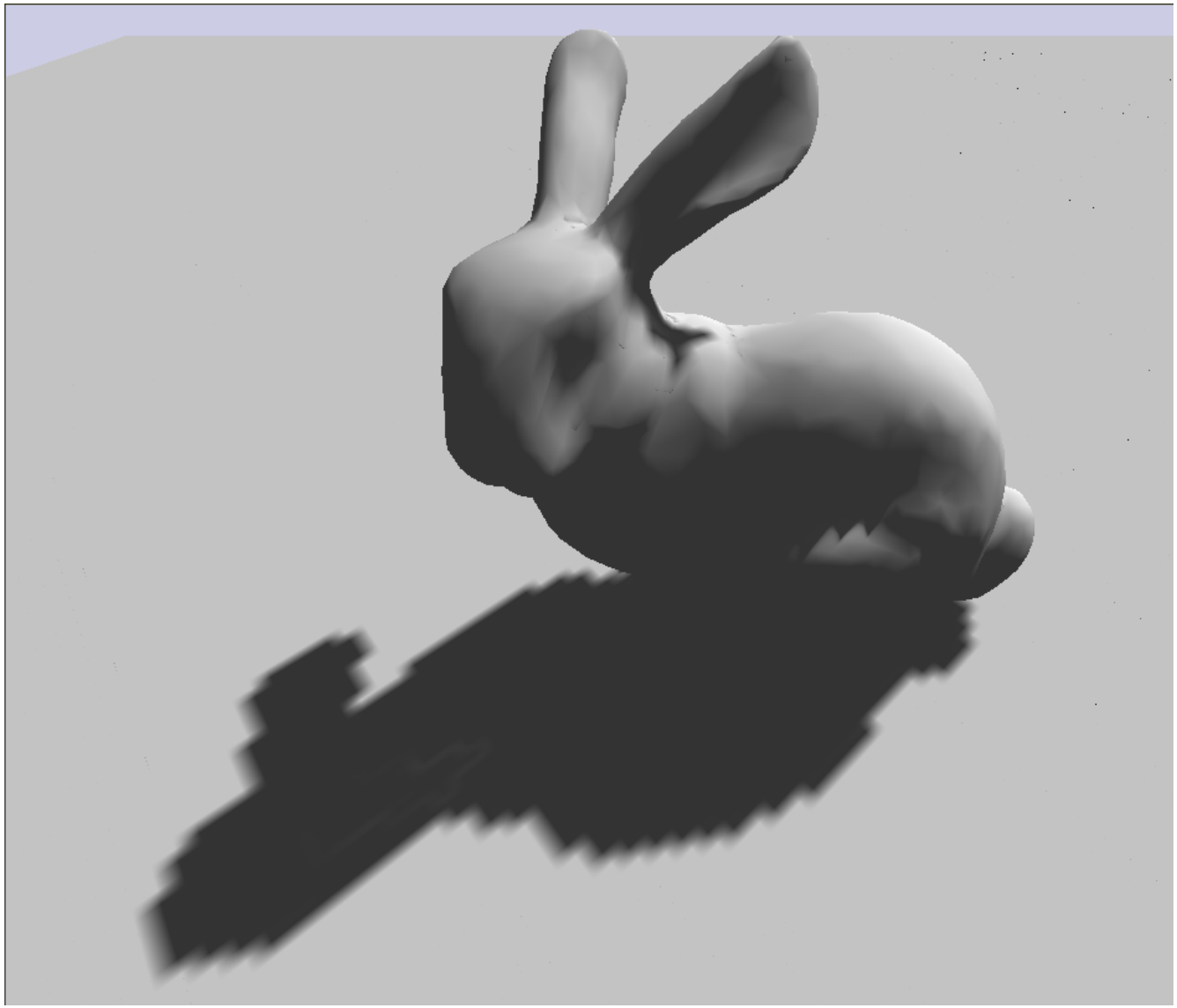
Variance shadow maps (VSM): shadow bias

- 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

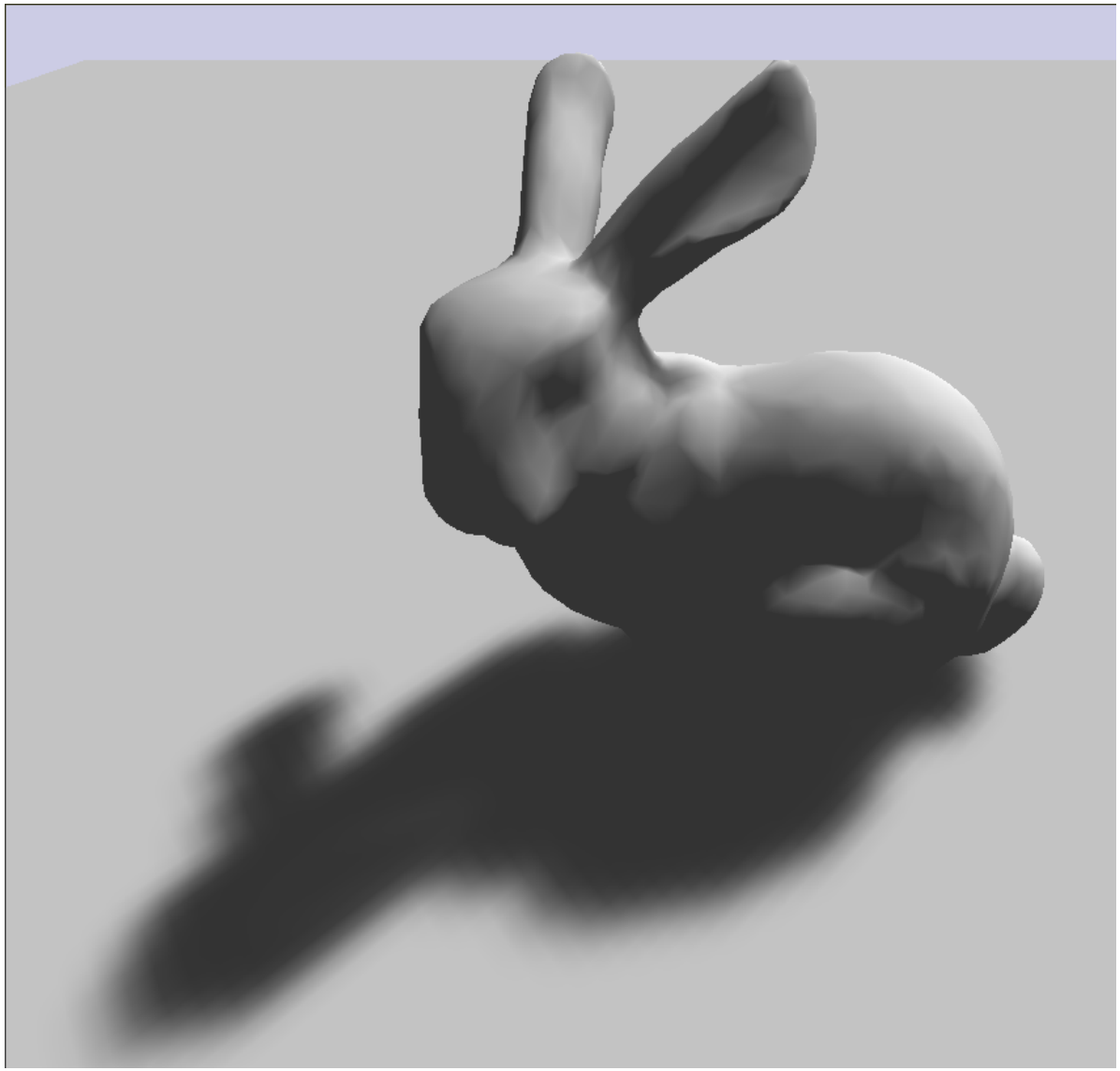
$$z^2 + \frac{1}{4}\left[\left(\frac{\partial z}{\partial x}\right)^2 + \left(\frac{\partial z}{\partial y}\right)^2\right] \tag{4}$$

- Убирает *почти* все артефакты shadow acne

VSM с линейной фильтрацией



VSM с линейной фильтрацией и 7x7 размытием по Гауссу



VSM

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

Light bleeding

Light bleeding

- Обычно возникает, когда σ^2 оказывается слишком большим

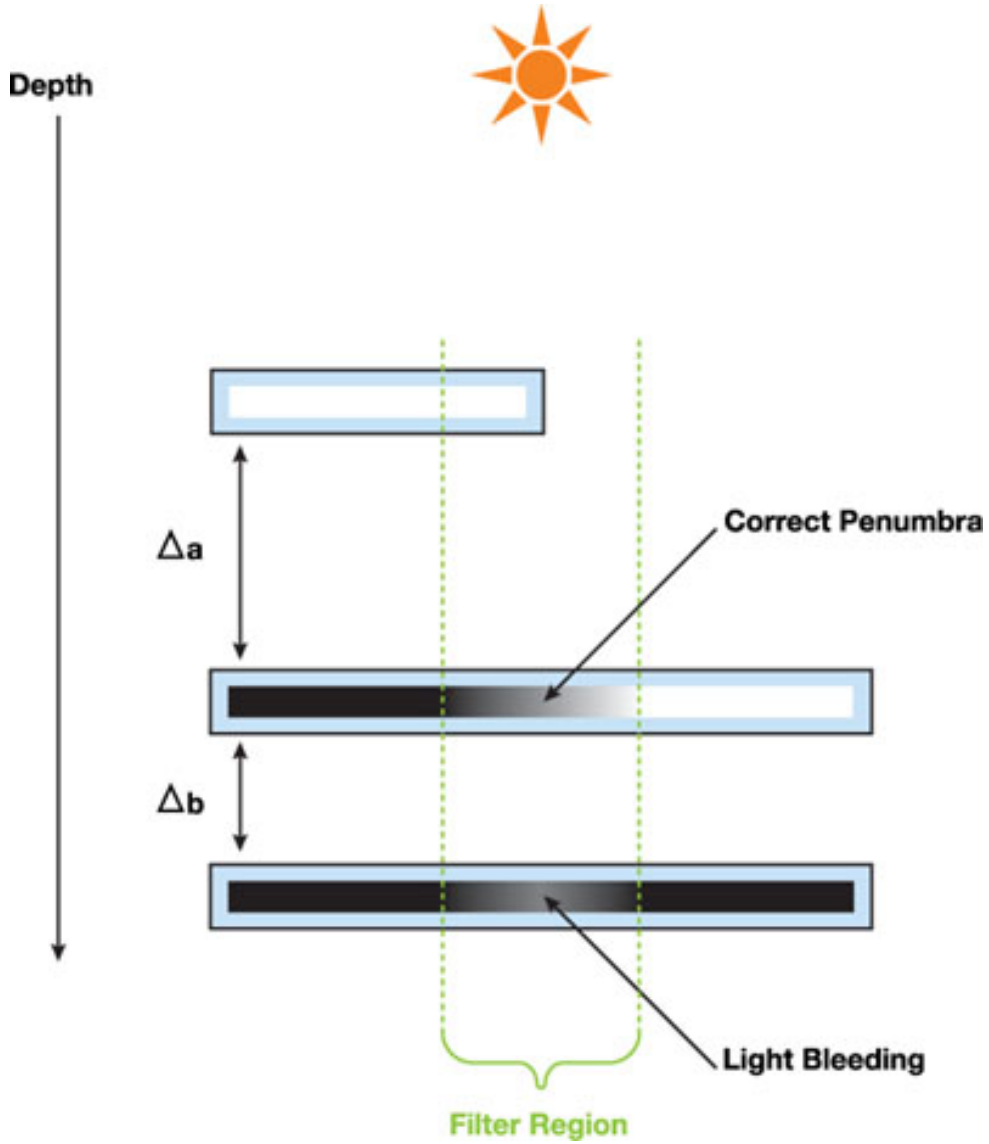
Light bleeding

- Обычно возникает, когда σ^2 оказывается слишком большим
- Можно 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
- Можно 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
- — Требует многокомпонентных буферов (часто до 16 компонент на пиксель)
- — Возникают ringing-артефакты (см. Gibbs phenomenon)

Ссылки

- 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

Тени на больших сценах

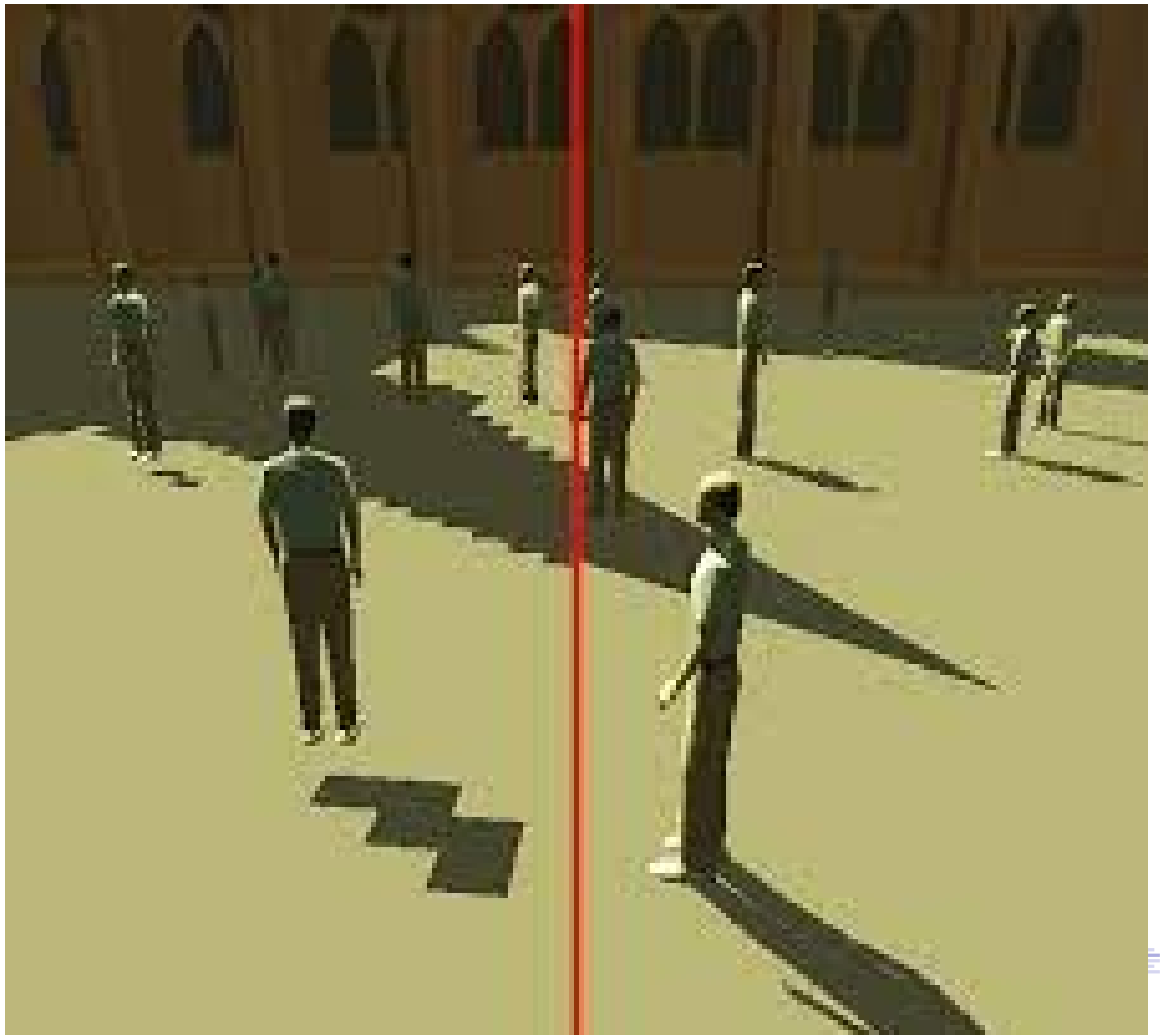
- Если сцена очень большая, даже огромных (8k) shadow map текстур с хорошим сглаживанием может не хватить

Тени на больших сценах

- 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

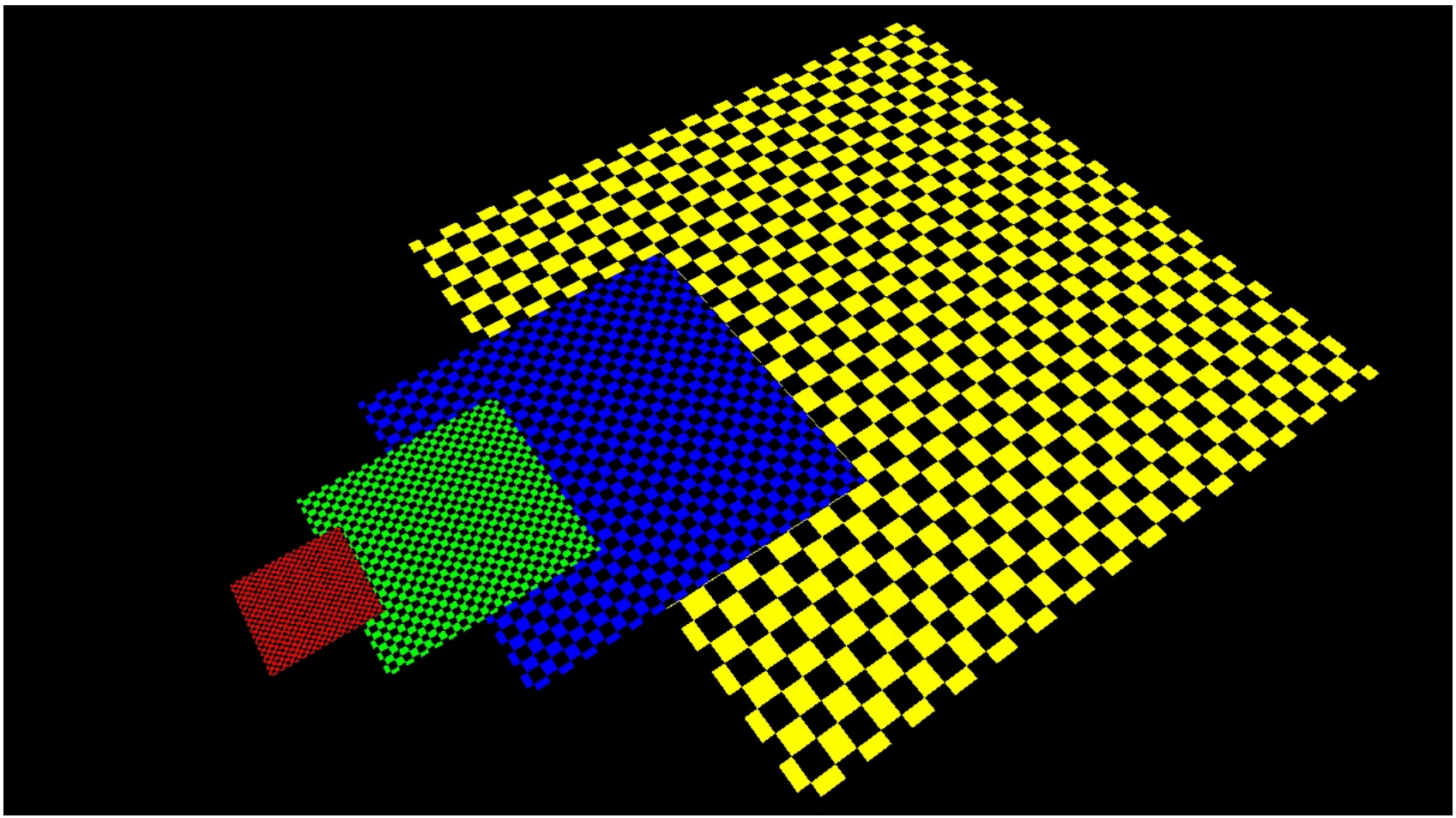


Perspective shadow maps (PSM)

- + 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
- + Можно комбинировать с VSM/ESM/etc
- — Большой расход памяти
- — Нужно аккуратно обработать переход между разными каскадами
- Самый часто использующийся сегодня алгоритм

Ссылки

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