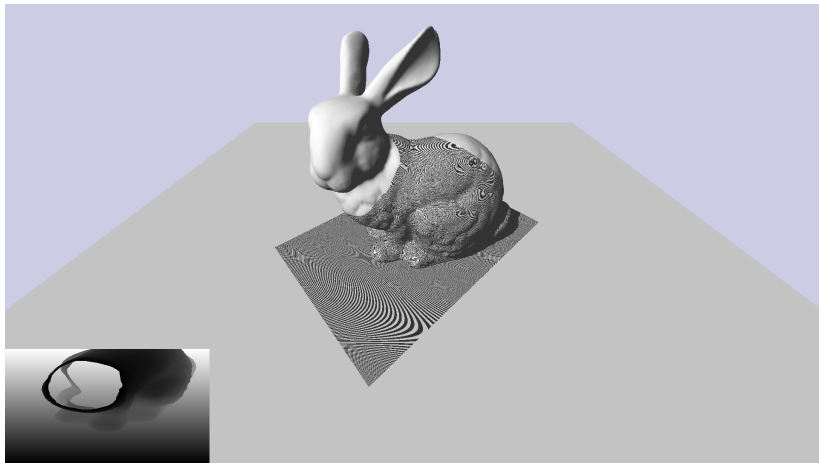


# Компьютерная графика

## Практика 9: Shadow mapping 2

2021

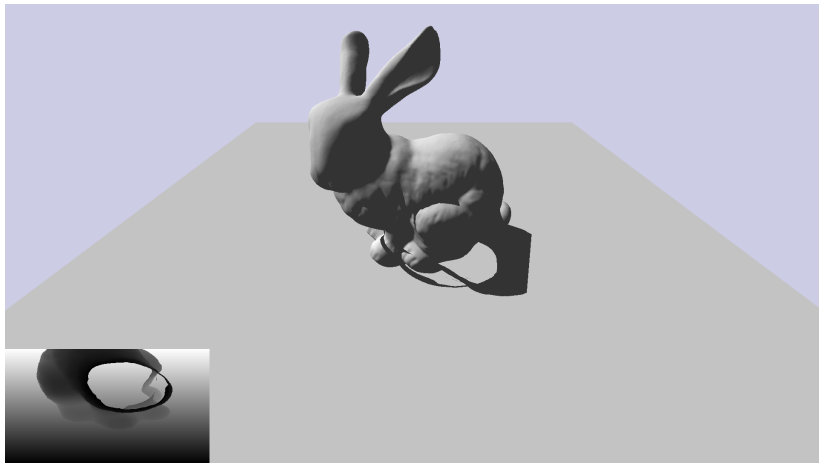


- ▶ В этой практике мы рисуем front-facing грани в shadow map, а не back-facing грани!

# Задание 1

Подбираем shadow bias

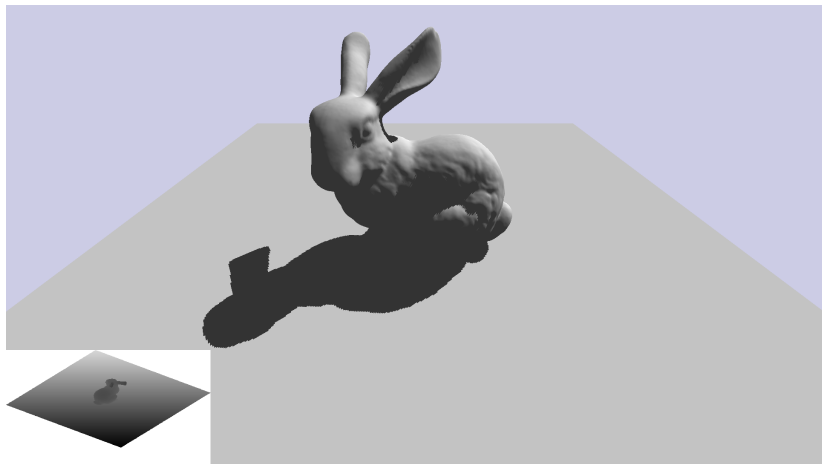
- ▶ Нужно прибавить в шейдере константу (bias) к значению, прочитанному из shadow map
- ▶ Слишком большое значение приведёт к заметному peter panning (тень будет съезжать в сторону от модели)
- ▶ Слишком маленького значения будет недостаточно, чтобы убрать shadow acne



## Задание 2

Вычисляем хорошую матрицу проекции для shadow map

- ▶ Нужно найти центр видимой области и оси  $X, Y, Z$
- ▶ Направления осей  $\hat{X}, \hat{Y}, \hat{Z}$  уже посчитаны, нужно только вычислить их длину
- ▶ После чтения сцены из файла нужно посчитать её bounding box (min/max по всем координатам вершин), её центр  $C$  – центр видимой области
- ▶ Пройдясь по всем 8 вершинам  $V$  bounding box'a сцены, можно посчитать модуль скалярного произведения  $|(V - C) \cdot \hat{X}|$ , максимум таких значений – длина вектора  $X$
- ▶ Аналогично для  $Y$  и  $Z$
- ▶ Используя  $X, Y, Z, C$  можно построить матрицу ортогографической проекции (см. слайды 4ой лекции)

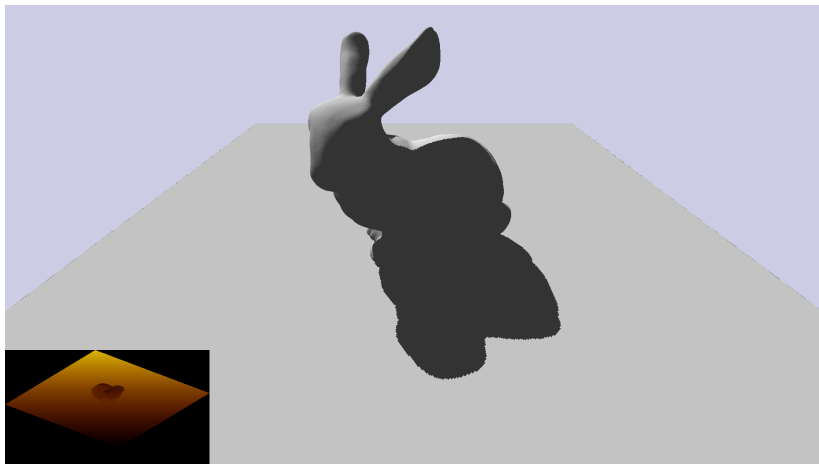


## Задание 3

### Variance shadow maps

- ▶ Включаем для shadow map линейную фильтрацию
- ▶ Устанавливаем текстуре shadow map internal format в `GL_RG32F` (format и type не принципиальны, можно `GL_RGBA` и `GL_FLOAT`)
- ▶ Добавляем shadow map к фреймбуферу как `GL_COLOR_ATTACHMENT0` вместо `GL_DEPTH_ATTACHMENT`
- ▶ Создаём renderbuffer (и выделяем ему память через `glRenderbufferStorage`), который будет использоваться как буфер глубины при рисовании shadow map, и добавляем его как `GL_DEPTH_ATTACHMENT` фреймбуфера
- ▶ Во фрагментном шейдере, рисуя shadow map, добавляем out-переменную и пишем в неё `vec4(z, z * z, 0.0, 0.0)` (z можно достать из `gl_FragCoord`)



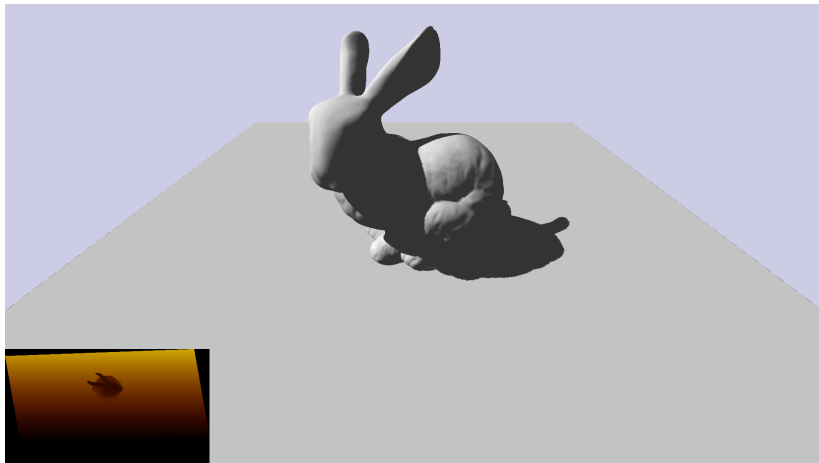


## Задание 4

### Variance shadow maps

- ▶ В основном фрагментном шейдере читаем данные из shadow map и используем неравенство Чебышёва для вычисления освещённости:

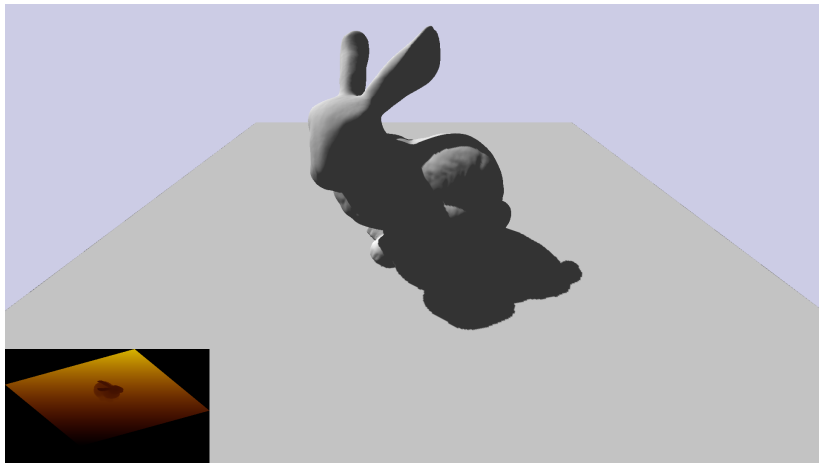
```
vec2 data = texture(shadow_map, shadow_pos.xy).rg;
float mu = data.r;
float sigma = data.g - mu * mu;
float z = shadow_pos.z;
float factor = (z < mu) ? 1.0
    : sigma / (sigma + (z - mu) * (z - mu));
```



## Задание 5

### Исправляем артефакты

- ▶ Добавляем наклон поверхности к среднему значению квадрата глубины при генерации shadow map:
  - ▶ Через  $dFdx(z)$  и  $dFdy(z)$  можно получить градиент глубины по  $X$  и  $Y$
  - ▶ К квадрату глубины добавляем  $\frac{1}{4} \left[ \left( \frac{\partial Z}{\partial X} \right)^2 + \left( \frac{\partial Z}{\partial Y} \right)^2 \right]$
- ▶ Добавляем shadow bias (вычитаем константу из значения  $z$ , использующегося для вычисления освещённости)
- ▶ Значение, получающееся из формулы неравенства Чебышёва, преобразуем: диапазон  $[0, \delta]$  переходит в  $0$ , а диапазон  $[\delta, 1]$  переходит в  $[0, 1]$  ( $\delta$  – некое фиксированное значение, например,  $0.125$ )



## Задание 6

### Размываем shadow map

- ▶ Вместо чтения одного пикселя из shadow map, читаем набор значений из соседних пикселей и усредняем по Гауссу
- ▶ Полученный двумерный вектор используем для вычисления освещённости через неравенство Чебышёва
- ▶ Размываем не результат вычисления освещённости, а сами данные из shadow map!
- ▶ **N.B.:** по-хорошему это размытие нужно делать отдельными проходами с отдельными шейдерами и отдельным размытием по X и Y (см. *separable Gaussian blur*)

