

# Компьютерная графика

## Практика 4: Индексы, перспективная проекция, буфер глубины

2021

# Напоминание про VBO и VAO

- ▶ VBO – хранит данные, ничего не знает о формате
- ▶ VAO – описывает формат и расположение атрибутов вершин
- ▶ Концептуально, расположение = id буфера + сдвиг
- ▶ VAO также хранит id текущего EBO (`GL_ELEMENT_ARRAY_BUFFER`) – оттуда берутся индексы
- ▶ Для рендеринга нужен только VAO
- ▶ Чтобы обновить данные, нужен только VBO
- ▶ Чтобы обновить индексы, нужен только EBO (можно использовать `target = GL_ARRAY_BUFFER`)

## Практика 4

- ▶ В этой практике нельзя менять код шейдеров!

## Практика 4

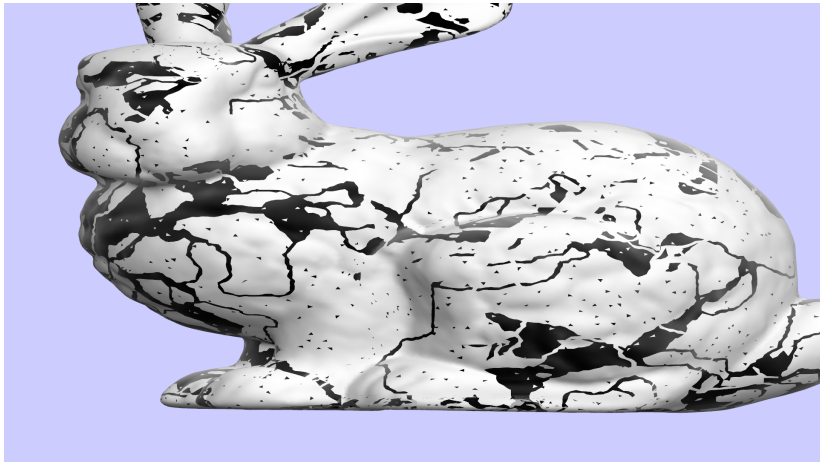


# Задание 1

Рисуем зайца

- ▶ Создаём VAO, VBO, EBO
- ▶ Загружаем данные (`bunny.vertices` и `bunny.indices`) в VBO и EBO
- ▶ Настраиваем атрибуты для VAO (нужно понять правильные настройки по вершинному шейдеру и по описанию структуры `obj_data::vertex` в `obj_parser.hpp`)
- ▶ Рисуем с помощью `glDrawElements`, `mode = GL_TRIANGLES`, обращаем внимание на тип индексов (`GL_UNSIGNED_INT`)
- ▶ N.B. заяц будет рисоваться странно из-за отключенного теста глубины, так и задумано

# Задание 1

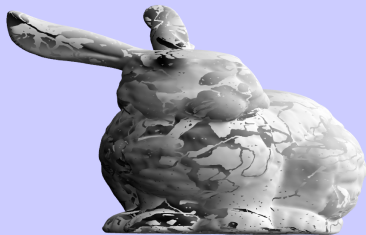


## Задание 2

### Вращаем зайца

- ▶ Меняем матрицу `transform` чтобы куб крутился в плоскости XZ
- ▶ В качестве угла нужно взять что-нибудь зависящее от времени, например `float angle = time;`
- ▶ Заяц будет обрезаться по  $z \in [-1, 1]$  (особенно хорошо видно, что иногда обрезается хвост)
- ▶ Отмасштабируем его по всем осям `float scale = 0.5f;`, тоже с помощью матрицы `transform`

## Задание 2





## Задание 3

Добавляем перспективу

- ▶ Выбираем значения `near`, `far`, `top`, `right`
  - ▶ `near` – маленький, но не слишком, в духе `0.001...0.1`
  - ▶ `far` – большой, но не слишком, в духе `10.0...1000.0`
  - ▶ Отношение `right/near` – тангенс половины угла обзора, например `right = near` это  $90^\circ$
  - ▶ Отношение `right/top` – aspect ratio экрана (`width/height`)
- ▶ В матрицу `view` записываем матрицу проекции с использованием выбранных значений (см. слайд с лекции)
- ▶ Заяц будет виден изнутри

## Задание 3



## Задание 4

Включаем тест глубины

- ▶ Сдвигаем зайца по оси  $Z$  на какое-то расстояние (например, на 3 единицы)
  - ▶ N.B: Камера смотрит в сторону  $-Z$ , т.е. сдвиг должен быть отрицательным
- ▶ Заяц будет рисоваться неправильно: задние грани перекрывают передние
- ▶ Включим тест глубины: `glEnable(GL_DEPTH_TEST)`
- ▶ Не забываем очищать буфер глубины в начале каждого кадра: `glClear(GL_DEPTH_BUFFER_BIT)`
  - ▶ Можно это делать одновременно с очисткой цветового буфера:  
`glClear(GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT)`

## Задание 4



## Задание 5

### Двигаем зайца

- ▶ Заведём переменные `bunny_x` и `bunny_y` с координатами центра зайца по  $X$  и  $Y$
- ▶ Изменим матрицу `transform`, добавив соответствующие сдвиги по  $X$  и  $Y$
- ▶ Перед рисованием каждого кадра обновим положение зайца:
  - ▶ Если нажата клавиша влево `SDLK_LEFT`, сдвинем куб влево: `cube_x -= speed * dt`
  - ▶ Аналогично `SDLK_RIGHT`, `SDLK_DOWN`, `SDLK_UP`
  - ▶ Состояние нажатости клавиш уже доступно в словаре `button_down`

## Задание 5



## Задание 6

Играем с face culling

- ▶ Включим back-face culling: `glEnable(GL_CULL_FACE)`
- ▶ Ничего не изменится: заяц сделан так, чтобы все треугольники были CCW
- ▶ Изменим режим: `glCullFace(GL_FRONT)`
- ▶ Должны быть видны задние грани куба и не видны передние
- ▶ Выглядеть будет странно – не пугайтесь, попробуйте увидеть и понять, что происходит :)

## Задание 6





## Задание 7\*

Три вращающихся зайца

- ▶ Рисуем три зайца в разных местах, вращающихся в плоскостях XY, XZ, YZ соответственно
- ▶ Три раза `glUniformMatrix + glDrawElements`
- ▶ Всё ещё можно двигать стрелочками, т.е. должны учитываться `bunny_x` и `bunny_y`

## Задание 7

