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`)

Задание 1

Рисуем куб

- Создаём VAO, VBO, EBO

Задание 1

Рисуем куб

- Создаём VAO, VBO, EBO
- Загружаем данные в VBO и EBO

Задание 1

Рисуем куб

- Создаём VAO, VBO, EBO
- Загружаем данные в VBO и EBO
- Настраиваем атрибуты для VAO

Задание 1

Рисуем куб

- Создаём VAO, VBO, EBO
- Загружаем данные в VBO и EBO
- Настраиваем атрибуты для VAO
- Рисуем с помощью `glDrawElements`, `mode = GL_TRIANGLES`, обращаем внимание на тип индексов (`GL_UNSIGNED_INT`)

Задание 2

Вращаем куб

- Меняем матрицу `transform` чтобы куб крутился в плоскости XZ

Задание 2

Вращаем куб

- Меняем матрицу `transform` чтобы куб крутился в плоскости XZ
- В качестве угла можно взять что-нибудь зависящее от времени, например `float angle = time;`

Задание 2

Вращаем куб

- Меняем матрицу `transform` чтобы куб крутился в плоскости XZ
- В качестве угла можно взять что-нибудь зависящее от времени, например `float angle = time;`
- Куб будет обрезаться по $z \in [-1, 1]$

Задание 2

Вращаем куб

- Меняем матрицу `transform` чтобы куб крутился в плоскости XZ
- В качестве угла можно взять что-нибудь зависящее от времени, например `float angle = time;`
- Куб будет обрезаться по $z \in [-1, 1]$
- Отмасштабируем его по всем осям `float scale = 0.5f`, тоже с помощью матрицы `transform`

Задание 3

Добавляем перспективу

- Выбираем значения `near`, `far`, `top`, `right`
  - `near` - маленький, но не слишком, в духе $0.001 \ldots 0.1$
  - `far` - большой, но не слишком, в духе $10.0 \ldots 1000.0$
  - Отношение `right/near` - тангенс угла обзора, например `right = near` это 45°
  - Отношение `right/top` - aspect ratio экрана (`width/height`)

Задание 3

Добавляем перспективу

- Выбираем значения `near`, `far`, `top`, `right`
  - `near` - маленький, но не слишком, в духе $0.001 \ldots 0.1$
  - `far` - большой, но не слишком, в духе $10.0 \ldots 1000.0$
  - Отношение `right/near` - тангенс угла обзора, например `right = near` это 45°
  - Отношение `right/top` - aspect ratio экрана (`width/height`)
- В матрицу `view` записываем матрицу проекции с использованием выбранных значений

Задание 3

Добавляем перспективу

- Выбираем значения `near`, `far`, `top`, `right`
  - `near` - маленький, но не слишком, в духе $0.001 \ldots 0.1$
  - `far` - большой, но не слишком, в духе $10.0 \ldots 1000.0$
  - Отношение `right/near` - тангенс угла обзора, например `right = near` это 45°
  - Отношение `right/top` - aspect ratio экрана (`width/height`)
- В матрицу `view` записываем матрицу проекции с использованием выбранных значений
- Куб будет виден изнутри

Задание 4

Включаем тест глубины

- Сдвигаем куб по оси Z на какое-то расстояние (например, на 5 единиц)
  - N.B: Камера смотрит в сторону -Z, т.е. сдвиг должен быть отрицательным

Задание 4

Включаем тест глубины

- Сдвигаем куб по оси Z на какое-то расстояние (например, на 5 единиц)
  - N.B: Камера смотрит в сторону -Z, т.е. сдвиг должен быть отрицательным
- Куб будет рисоваться неправильно: задние грани перекрывают передние

Задание 4

Включаем тест глубины

- Сдвигаем куб по оси Z на какое-то расстояние (например, на 5 единиц)
  - N.B: Камера смотрит в сторону -Z, т.е. сдвиг должен быть отрицательным
- Куб будет рисоваться неправильно: задние грани перекрывают передние
- Включим тест глубины: `glEnable(GL_DEPTH_TEST)`

Задание 4

Включаем тест глубины

- Сдвигаем куб по оси Z на какое-то расстояние (например, на 5 единиц)
  - N.B: Камера смотрит в сторону -Z, т.е. сдвиг должен быть отрицательным
- Куб 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)`

Задание 5

Двигаем куб

- Заведём переменные `cube_x` и `cube_y` с координатами центра куба по X и Y

Задание 5

Двигаем куб

- Заведём переменные `cube_x` и `cube_y` с координатами центра куба по X и Y
- Изменим матрицу `transform`, добавив соответствующие сдвиги по X и Y

Задание 5

Двигаем куб

- Заведём переменные `cube_x` и `cube_y` с координатами центра куба по X и Y
- Изменим матрицу `transform`, добавив соответствующие сдвиги по X и Y
- Перед рисованием каждого кадра обновим положение куба:
  - Если нажата `SDLK_LEFT`, сдвинем куб влево:
    `cube_x -= speed * dt`
  - Аналогично `SDLK_RIGHT`, `SDLK_DOWN`, `SDLK_UP`

Задание 6

Играем с face culling

- Включим back-face culling: `glEnable(GL_CULL_FACE)`

Задание 6

Играем с face culling

- Включим back-face culling: `glEnable(GL_CULL_FACE)`
- Ничего не изменится: куб сделан так, чтобы все треугольники были CCW

Задание 6

Играем с face culling

- Включим back-face culling: `glEnable(GL_CULL_FACE)`
- Ничего не изменится: куб сделан так, чтобы все треугольники были CCW
- Изменим режим: `glCullFace(GL_FRONT)`

Задание 6

Играем с face culling

- Включим back-face culling: `glEnable(GL_CULL_FACE)`
- Ничего не изменится: куб сделан так, чтобы все треугольники были CCW
- Изменим режим: `glCullFace(GL_FRONT)`
- Должны быть видны задние грани куба и не видны передние

Задание 7

Три вращающихся куба

- Рисуем три куба в разных местах, вращающихся в плоскостях XY, XZ, YZ соответственно

Задание 7

Три вращающихся куба

- Рисуем три куба в разных местах, вращающихся в плоскостях XY, XZ, YZ соответственно
- Три раза `glUniformMatrix` + `glDrawElements`

Задание 7

Три вращающихся куба

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