Компьютерная графика

Практика 12: Объёмный рендеринг

2021

Задание 1

Находим пересечение с кубом

- Во фрагментном шейдере заводим координаты крайних точек куба (`vec3 box_min(-1.0), box_max(1.0)`) - можно как uniform'ы, можно как const переменные

Задание 1

Находим пересечение с кубом

- Во фрагментном шейдере заводим координаты крайних точек куба (`vec3 box_min(-1.0), box_max(1.0)`) - можно как uniform'ы, можно как const переменные
- Вычисляем **нормированный** вектор направления из камеры в текущую точку поверхности куба - это вектор направления луча

Задание 1

Находим пересечение с кубом

- Во фрагментном шейдере заводим координаты крайних точек куба (`vec3 box_min(-1.0), box_max(1.0)`) - можно как uniform'ы, можно как const переменные
- Вычисляем **нормированный** вектор направления из камеры в текущую точку поверхности куба - это вектор направления луча
- Вычисляем пересечение этого луча с кубом: интервал $[t_{min}, t_{max}]$ для которых $p + t \cdot d$ пересекает куб

Задание 1

Находим пересечение с кубом

- Во фрагментном шейдере заводим координаты крайних точек куба (`vec3 box_min(-1.0), box_max(1.0)`) - можно как uniform'ы, можно как const переменные
- Вычисляем **нормированный** вектор направления из камеры в текущую точку поверхности куба - это вектор направления луча
- Вычисляем пересечение этого луча с кубом: интервал $[t_{min}, t_{max}]$ для которых $p + t \cdot d$ пересекает куб
- Делаем `tmin >= 0`, чтобы не включать часть пересечения сзади камеры

Задание 1

Находим пересечение с кубом

- Во фрагментном шейдере заводим координаты крайних точек куба (`vec3 box_min(-1.0), box_max(1.0)`) - можно как uniform'ы, можно как const переменные
- Вычисляем **нормированный** вектор направления из камеры в текущую точку поверхности куба - это вектор направления луча
- Вычисляем пересечение этого луча с кубом: интервал $[t_{min}, t_{max}]$ для которых $p + t \cdot d$ пересекает куб
- Делаем `tmin >= 0`, чтобы не включать часть пересечения сзади камеры
- В качестве цвета пикселя выводим `tmax - tmin` (это значение часто будет больше единицы, так что можно его нормировать, разделив на 3.5)

Задание 2

Загружаем 3D текстуру

- Создаём текстуру типа `GL_TEXTURE_3D`

Задание 2

Загружаем 3D текстуру

- Создаём текстуру типа `GL_TEXTURE_3D`
- Устанавливаем min фильтр в `GL_LINEAR`, mag в `GL_LINEAR_MIPMAP_NEAREST`

Задание 2

Загружаем 3D текстуру

- Создаём текстуру типа `GL_TEXTURE_3D`
- Устанавливаем min фильтр в `GL_LINEAR`, mag в `GL_LINEAR_MIPMAP_NEAREST`
- Устанавливаем wrap_r, wrap_s, wrap_t в `GL_CLAMP_TO_EDGE`

Задание 2

Загружаем 3D текстуру

- Создаём текстуру типа `GL_TEXTURE_3D`
- Устанавливаем min фильтр в `GL_LINEAR`, mag в `GL_LINEAR_MIPMAP_NEAREST`
- Устанавливаем wrap_r, wrap_s, wrap_t в `GL_CLAMP_TO_EDGE`
- Считываем данные из файла `house256` (256x256x256, 4 байта на пиксель) и загружаем в текстуру (`glTexImage3D`)
  - Если GPU не потянет, можно взять файл поменьше - `house128` или `house64` (128x128x128 и 64x64x64 соответственно)

Задание 2

Загружаем 3D текстуру

- Создаём текстуру типа `GL_TEXTURE_3D`
- Устанавливаем min фильтр в `GL_LINEAR`, mag в `GL_LINEAR_MIPMAP_NEAREST`
- Устанавливаем wrap_r, wrap_s, wrap_t в `GL_CLAMP_TO_EDGE`
- Считываем данные из файла `house256` (256x256x256, 4 байта на пиксель) и загружаем в текстуру (`glTexImage3D`)
  - Если GPU не потянет, можно взять файл поменьше - `house128` или `house64` (128x128x128 и 64x64x64 соответственно)
- Добавляем текстуру в шейдер (`sampler3D`), выводим в качестве цвета значение из текстуры в точке `position`
  - Нужно перевести пространственные координаты в текстурные: `(position - box_min) / box_size`
  - В качестве альфа-канала возьмём 1, иначе ничего не увидим

Задание 3

Вычисляем среднее значение цвета вдоль луча

- В вершинном шейдере делаем цикл (например, в 64 шага); шаг цикла соответствует 1/64 части отрезка $[t_{min}, t_{max}]$, т.е. $[t_{min} + \frac{\Delta t \cdot i}{64}, t_{min} + \frac{\Delta t \cdot (i+1)}{64}]$, где $\Delta t = t_{max} - t_{min}$
  - Каждому значению $t \in [t_{min}, t_{max}]$ соответствует точка $p + t \cdot d$
  - Вместо 64 можно взять любое другое число; чем больше, тем красивее и медленнее

Задание 3

Вычисляем среднее значение цвета вдоль луча

- В вершинном шейдере делаем цикл (например, в 64 шага); шаг цикла соответствует 1/64 части отрезка $[t_{min}, t_{max}]$, т.е. $[t_{min} + \frac{\Delta t \cdot i}{64}, t_{min} + \frac{\Delta t \cdot (i+1)}{64}]$, где $\Delta t = t_{max} - t_{min}$
  - Каждому значению $t \in [t_{min}, t_{max}]$ соответствует точка $p + t \cdot d$
  - Вместо 64 можно взять любое другое число; чем больше, тем красивее и медленнее
- Вычисляем цвет из текстуры в текущей точке (лучше взять середину отрезка, т.е. точку соответствующую $\frac{i+0.5}{64}$), и переводим в premultiplied формат (RGB-часть нужно домножить на альфа-канал)
  - Лучше читать 0-ой mipmap-уровень (`textureLod(sampler, texcoord, 0.0)`)

Задание 3

Вычисляем среднее значение цвета вдоль луча

- В вершинном шейдере делаем цикл (например, в 64 шага); шаг цикла соответствует 1/64 части отрезка $[t_{min}, t_{max}]$, т.е. $[t_{min} + \frac{\Delta t \cdot i}{64}, t_{min} + \frac{\Delta t \cdot (i+1)}{64}]$, где $\Delta t = t_{max} - t_{min}$
  - Каждому значению $t \in [t_{min}, t_{max}]$ соответствует точка $p + t \cdot d$
  - Вместо 64 можно взять любое другое число; чем больше, тем красивее и медленнее
- Вычисляем цвет из текстуры в текущей точке (лучше взять середину отрезка, т.е. точку соответствующую $\frac{i+0.5}{64}$), и переводим в premultiplied формат (RGB-часть нужно домножить на альфа-канал)
  - Лучше читать 0-ой mipmap-уровень (`textureLod(sampler, texcoord, 0.0)`)
- Суммируем усреднённое значение цвета по всем точкам (цвет в точке нужно домножить на длину одного отрезка, т.е. на $\frac{\Delta t}{64}$)

Задание 3

Вычисляем среднее значение цвета вдоль луча

- В вершинном шейдере делаем цикл (например, в 64 шага); шаг цикла соответствует 1/64 части отрезка $[t_{min}, t_{max}]$, т.е. $[t_{min} + \frac{\Delta t \cdot i}{64}, t_{min} + \frac{\Delta t \cdot (i+1)}{64}]$, где $\Delta t = t_{max} - t_{min}$
  - Каждому значению $t \in [t_{min}, t_{max}]$ соответствует точка $p + t \cdot d$
  - Вместо 64 можно взять любое другое число; чем больше, тем красивее и медленнее
- Вычисляем цвет из текстуры в текущей точке (лучше взять середину отрезка, т.е. точку соответствующую $\frac{i+0.5}{64}$), и переводим в premultiplied формат (RGB-часть нужно домножить на альфа-канал)
  - Лучше читать 0-ой mipmap-уровень (`textureLod(sampler, texcoord, 0.0)`)
- Суммируем усреднённое значение цвета по всем точкам (цвет в точке нужно домножить на длину одного отрезка, т.е. на $\frac{\Delta t}{64}$)
- Усреднённый цвет выводим в качестве цвета

Задание 4

Вычисляем настоящую интенсивность цвета используя front-to-back проход

- Заводим коэффициент интенсивности `C = 16.0` (чем больше, тем менее прозрачной будет сцена) - он используется для перевода значений альфа-канала в плотность среды

Задание 4

Вычисляем настоящую интенсивность цвета используя front-to-back проход

- Заводим коэффициент интенсивности `C = 16.0` (чем больше, тем менее прозрачной будет сцена) - он используется для перевода значений альфа-канала в плотность среды
- Вычисляем реальную плотность вдоль `i`-ого отрезка как $1 - \exp(-C\frac{\Delta t}{64}A)$, где A - значение альфа-канала, и перезаписываем в альфа-канал текущего (прочитанного из текстуры) цвета

Задание 4

Вычисляем настоящую интенсивность цвета используя front-to-back проход

- Заводим коэффициент интенсивности `C = 16.0` (чем больше, тем менее прозрачной будет сцена) - он используется для перевода значений альфа-канала в плотность среды
- Вычисляем реальную плотность вдоль `i`-ого отрезка как $1 - \exp(-C\frac{\Delta t}{64}A)$, где A - значение альфа-канала, и перезаписываем в альфа-канал текущего (прочитанного из текстуры) цвета
- Переводим цвет в premultiplied формат (домножаем RGB-часть на альфа-канал, который мы перезаписали в предыдущем пункте)

Задание 4

Вычисляем настоящую интенсивность цвета используя front-to-back проход

- Заводим коэффициент интенсивности `C = 16.0` (чем больше, тем менее прозрачной будет сцена) - он используется для перевода значений альфа-канала в плотность среды
- Вычисляем реальную плотность вдоль `i`-ого отрезка как $1 - \exp(-C\frac{\Delta t}{64}A)$, где A - значение альфа-канала, и перезаписываем в альфа-канал текущего (прочитанного из текстуры) цвета
- Переводим цвет в premultiplied формат (домножаем RGB-часть на альфа-канал, который мы перезаписали в предыдущем пункте)
- Прибавляем к результату текущий цвет, умноженный на `1 - result.a` (чтобы учесть прозрачность уже посчитанного цвета)

Задание 4

Вычисляем настоящую интенсивность цвета используя front-to-back проход

- Заводим коэффициент интенсивности `C = 16.0` (чем больше, тем менее прозрачной будет сцена) - он используется для перевода значений альфа-канала в плотность среды
- Вычисляем реальную плотность вдоль `i`-ого отрезка как $1 - \exp(-C\frac{\Delta t}{64}A)$, где A - значение альфа-канала, и перезаписываем в альфа-канал текущего (прочитанного из текстуры) цвета
- Переводим цвет в premultiplied формат (домножаем RGB-часть на альфа-канал, который мы перезаписали в предыдущем пункте)
- Прибавляем к результату текущий цвет, умноженный на `1 - result.a` (чтобы учесть прозрачность уже посчитанного цвета)
- После суммирования применяем к RGB-части результирующего пикселя гамма-коррекцию

Задание 5

Добавляем тени

- Для каждого цвета, прочитанного из текстуры (т.е. на каждой итерации цикла) вычисляем его освещённость (число в диапазоне $[0, 1]$)

Задание 5

Добавляем тени

- Для каждого цвета, прочитанного из текстуры (т.е. на каждой итерации цикла) вычисляем его освещённость (число в диапазоне $[0, 1]$)
- Чтобы вычислить освещённость, пускаем луч из текущей точки в направлении источника света и пересекаем с кубом (делаем `tmin >= 0`, иначе тени будут неверными)

Задание 5

Добавляем тени

- Для каждого цвета, прочитанного из текстуры (т.е. на каждой итерации цикла) вычисляем его освещённость (число в диапазоне $[0, 1]$)
- Чтобы вычислить освещённость, пускаем луч из текущей точки в направлении источника света и пересекаем с кубом (делаем `tmin >= 0`, иначе тени будут неверными)
- Вдоль этого луча запускаем внутренний цикл, аналогичный внешнему, но нас интересует только альфа-канал результата; освещённость - это `1 - A`
  - Число итераций лучше взять не очень большим, например, 8 или 16
  - Можно упростить вычисления, просто просуммировав $C \cdot A \cdot \frac{\Delta t}{16}$, и посчитав освещённость как `exp(-sum)`
  - Так как итераций мало, лучше читать какой-нибудь низкодетализированный mipmap текстуры (например, 4ый)

Задание 5

Добавляем тени

- Для каждого цвета, прочитанного из текстуры (т.е. на каждой итерации цикла) вычисляем его освещённость (число в диапазоне $[0, 1]$)
- Чтобы вычислить освещённость, пускаем луч из текущей точки в направлении источника света и пересекаем с кубом (делаем `tmin >= 0`, иначе тени будут неверными)
- Вдоль этого луча запускаем внутренний цикл, аналогичный внешнему, но нас интересует только альфа-канал результата; освещённость - это `1 - A`
  - Число итераций лучше взять не очень большим, например, 8 или 16
  - Можно упростить вычисления, просто просуммировав $C \cdot A \cdot \frac{\Delta t}{16}$, и посчитав освещённость как `exp(-sum)`
  - Так как итераций мало, лучше читать какой-нибудь низкодетализированный mipmap текстуры (например, 4ый)
- Домножаем RGB-часть цвета текущей точки (во внешнем цикле) на значение освещённости