Компьютерная графика

Практика 6: Освещение

2021

Задание 1

Добавляем ambient освещение

- Добавляем uniform-переменную `vec3 ambient` во фрагментный шейдер

Задание 1

Добавляем ambient освещение

- Добавляем uniform-переменную `vec3 ambient` во фрагментный шейдер
- Используем её для вычисления отражённого ambient освещения по формуле `color = ambient * albedo` (альбедо читается из текстуры)

Задание 1

Добавляем ambient освещение

- Добавляем uniform-переменную `vec3 ambient` во фрагментный шейдер
- Используем её для вычисления отражённого ambient освещения по формуле `color = ambient * albedo` (альбедо читается из текстуры)
- Устанавливаем значение uniform-переменной (`glGetUniformLocation`, `glUniform3f`)

Задание 2

Добавляем точечный источник света

- Добавляем uniform-переменные, описывающие источник света
  - `vec3 light_position` - координаты источника
  - `vec3 light_color` - цвет источника
  - `vec3 light_attenuation` - параметры затухания источника с расстоянием

Задание 2

Добавляем точечный источник света

- Добавляем uniform-переменные, описывающие источник света
  - `vec3 light_position` - координаты источника
  - `vec3 light_color` - цвет источника
  - `vec3 light_attenuation` - параметры затухания источника с расстоянием
- Для вычисления освещённости нам нужны расстояние от рисуемого пикселя до источника света, и направление на него $\Rightarrow$ нужно передать во фрагментный шейдер интерполированную позицию пикселя (**не** `gl_Position`, а что-то в духе `model * in_position`)

Задание 2

Добавляем точечный источник света

- Добавляем uniform-переменные, описывающие источник света
  - `vec3 light_position` - координаты источника
  - `vec3 light_color` - цвет источника
  - `vec3 light_attenuation` - параметры затухания источника с расстоянием
- Для вычисления освещённости нам нужны расстояние от рисуемого пикселя до источника света, и направление на него $\Rightarrow$ нужно передать во фрагментный шейдер интерполированную позицию пикселя (**не** `gl_Position`, а что-то в духе `model * in_position`)
- Вычисляем освещённость:
  `albedo * light_color * cosine * intensity`

Задание 2

Добавляем точечный источник света

- Добавляем uniform-переменные, описывающие источник света
  - `vec3 light_position` - координаты источника
  - `vec3 light_color` - цвет источника
  - `vec3 light_attenuation` - параметры затухания источника с расстоянием
- Для вычисления освещённости нам нужны расстояние от рисуемого пикселя до источника света, и направление на него $\Rightarrow$ нужно передать во фрагментный шейдер интерполированную позицию пикселя (**не** `gl_Position`, а что-то в духе `model * in_position`)
- Вычисляем освещённость:
  `albedo * light_color * cosine * intensity`
- Устанавливаем какие-нибудь значения для uniform-переменных (в качестве `attenuation` можно взять, например, `vec3(1.0, 0.0, 0.1)`)
  - Если источник плохо видно, можно сделать его поярче - компоненты `light_color` могут быть и больше 1

Задание 3

Три источника света

- Заменяем uniform-переменные, описывающие источник, на массивы uniform-переменных: `vec3 light_position[3]` и т.п.

Задание 3

Три источника света

- Заменяем uniform-переменные, описывающие источник, на массивы uniform-переменных: `vec3 light_position[3]` и т.п.
- Во фрагментном шейдере суммируем отраженный свет каждого источника

Задание 3

Три источника света

- Заменяем uniform-переменные, описывающие источник, на массивы uniform-переменных: `vec3 light_position[3]` и т.п.
- Во фрагментном шейдере суммируем отраженный свет каждого источника
- Для `glGetUniformLocation` в качестве имени переменной указываем `light_position[0]`, `light_position[1]`, и т.д.

Задание 3

Три источника света

- Заменяем uniform-переменные, описывающие источник, на массивы uniform-переменных: `vec3 light_position[3]` и т.п.
- Во фрагментном шейдере суммируем отраженный свет каждого источника
- Для `glGetUniformLocation` в качестве имени переменной указываем `light_position[0]`, `light_position[1]`, и т.д.
- В сумме будет 9 uniform-переменных (3 массива по 3), нужно задать им всем значения

Задание 3

Три источника света

- Заменяем uniform-переменные, описывающие источник, на массивы uniform-переменных: `vec3 light_position[3]` и т.п.
- Во фрагментном шейдере суммируем отраженный свет каждого источника
- Для `glGetUniformLocation` в качестве имени переменной указываем `light_position[0]`, `light_position[1]`, и т.д.
- В сумме будет 9 uniform-переменных (3 массива по 3), нужно задать им всем значения
- Лучше, если источники света будут двигаться (например, крутиться вокруг общего центра)

Задание 4

Добавляем стены

- Меняя матрицу `model` и вызывая рисование ещё раз (`glm::rotate`, `glm::translate`, `glUniformMatrix`, `glDrawArrays`) добавляем три стены: слева, справа, и сзади

Задание 4

Добавляем стены

- Меняя матрицу `model` и вызывая рисование ещё раз (`glm::rotate`, `glm::translate`, `glUniformMatrix`, `glDrawArrays`) добавляем три стены: слева, справа, и сзади
  - N.B. Исходная геометрия - плоскость $[-10, 10] \times [-10, 10]$ параллельная XY, с Z = 0
  - Задняя стена - плоскость, параллельная XY, с Z = -10
  - Левая стена - плоскость, параллельная YZ, с X = -10
  - Правая стена - плоскость, параллельная YZ, с X = 10

Задание 4

Добавляем стены

- Меняя матрицу `model` и вызывая рисование ещё раз (`glm::rotate`, `glm::translate`, `glUniformMatrix`, `glDrawArrays`) добавляем три стены: слева, справа, и сзади
  - N.B. Исходная геометрия - плоскость $[-10, 10] \times [-10, 10]$ параллельная XY, с Z = 0
  - Задняя стена - плоскость, параллельная XY, с Z = -10
  - Левая стена - плоскость, параллельная YZ, с X = -10
  - Правая стена - плоскость, параллельная YZ, с X = 10
  - Не забываем очищать матрицу перед записью в неё нового преобразования: `model = glm::mat4(1.f)`

Задание 4

Добавляем стены

- Меняя матрицу `model` и вызывая рисование ещё раз (`glm::rotate`, `glm::translate`, `glUniformMatrix`, `glDrawArrays`) добавляем три стены: слева, справа, и сзади
  - N.B. Исходная геометрия - плоскость $[-10, 10] \times [-10, 10]$ параллельная XY, с Z = 0
  - Задняя стена - плоскость, параллельная XY, с Z = -10
  - Левая стена - плоскость, параллельная YZ, с X = -10
  - Правая стена - плоскость, параллельная YZ, с X = 10
  - Не забываем очищать матрицу перед записью в неё нового преобразования: `model = glm::mat4(1.f)`
- Если стена не освещается источниками света, значит она обращена в другую сторону - нужно повернуть её на 180 градусов
  - N.B. функции GLM принимают радианы

Задание 5

Добавляем normal mapping

- Нормаль в вершинном шейдере нам больше не нужна

Задание 5

Добавляем normal mapping

- Нормаль в вершинном шейдере нам больше не нужна
- Добавляем во фрагментный шейдер uniform-текстуру `sampler2D normal_map`

Задание 5

Добавляем normal mapping

- Нормаль в вершинном шейдере нам больше не нужна
- Добавляем во фрагментный шейдер uniform-текстуру `sampler2D normal_map`
- Читаем нормали из текстуры и переводим из диапазона $[0, 1]$ в диапазон $[-1, 1]$

Задание 5

Добавляем normal mapping

- Нормаль в вершинном шейдере нам больше не нужна
- Добавляем во фрагментный шейдер uniform-текстуру `sampler2D normal_map`
- Читаем нормали из текстуры и переводим из диапазона $[0, 1]$ в диапазон $[-1, 1]$
- Нормали заданы в исходной системе координат объекта, но мы сдвигаем/поворачиваем объект $\Rightarrow$ к нормалям надо применить матрицу `model` (её нужно объявить во фрагментном шейдере)

Задание 5

Добавляем normal mapping

- Нормаль в вершинном шейдере нам больше не нужна
- Добавляем во фрагментный шейдер uniform-текстуру `sampler2D normal_map`
- Читаем нормали из текстуры и переводим из диапазона $[0, 1]$ в диапазон $[-1, 1]$
- Нормали заданы в исходной системе координат объекта, но мы сдвигаем/поворачиваем объект $\Rightarrow$ к нормалям надо применить матрицу `model` (её нужно объявить во фрагментном шейдере)
- В качестве значения uniform-переменной снаружи записываем 0 (чтобы использовать текстуру из нулевого texture unit'а)

Задание 5

Добавляем normal mapping

- Нормаль в вершинном шейдере нам больше не нужна
- Добавляем во фрагментный шейдер uniform-текстуру `sampler2D normal_map`
- Читаем нормали из текстуры и переводим из диапазона $[0, 1]$ в диапазон $[-1, 1]$
- Нормали заданы в исходной системе координат объекта, но мы сдвигаем/поворачиваем объект $\Rightarrow$ к нормалям надо применить матрицу `model` (её нужно объявить во фрагментном шейдере)
- В качестве значения uniform-переменной снаружи записываем 0 (чтобы использовать текстуру из нулевого texture unit'а)
- Создаём текстуру типа `GL_TEXTURE_2D`, загружаем данные из `brick_normal`, настраиваем mip-mapping (`glGenTextures`, `glBindTexture`, `glTexImage2D`, `glTexParameteri`, опционально `glGenerateMipmap`)

Задание 5

Добавляем normal mapping

- Нормаль в вершинном шейдере нам больше не нужна
- Добавляем во фрагментный шейдер uniform-текстуру `sampler2D normal_map`
- Читаем нормали из текстуры и переводим из диапазона $[0, 1]$ в диапазон $[-1, 1]$
- Нормали заданы в исходной системе координат объекта, но мы сдвигаем/поворачиваем объект $\Rightarrow$ к нормалям надо применить матрицу `model` (её нужно объявить во фрагментном шейдере)
- В качестве значения uniform-переменной снаружи записываем 0 (чтобы использовать текстуру из нулевого texture unit'а)
- Создаём текстуру типа `GL_TEXTURE_2D`, загружаем данные из `brick_normal`, настраиваем mip-mapping (`glGenTextures`, `glBindTexture`, `glTexImage2D`, `glTexParameteri`, опционально `glGenerateMipmap`)
- При рендеринге делаем активным нулевой texture unit и делаем для него текущей нашу текстуру (`glActiveTexture`, `glBindTexture`)

Задание 6

Добавляем ambient occlusion mapping

- Ещё одна текстура: `sampler2D ao_map`

Задание 6

Добавляем ambient occlusion mapping

- Ещё одна текстура: `sampler2D ao_map`
- Читаем из неё значение ambient occlusion, умножаем на него ambient освещение в данной точке
  - Эффект можно сделать заметнее, возведя значение из `ao_map` в какую-нибудь степень (скажем, 4)

Задание 6

Добавляем ambient occlusion mapping

- Ещё одна текстура: `sampler2D ao_map`
- Читаем из неё значение ambient occlusion, умножаем на него ambient освещение в данной точке
  - Эффект можно сделать заметнее, возведя значение из `ao_map` в какую-нибудь степень (скажем, 4)
- Данные для текстуры - `brick_ao`, texture unit = 1

Задание 7

Добавляем specular light и material mapping

- Ещё одна текстура: `sampler2D roughness_map`

Задание 7

Добавляем specular light и material mapping

- Ещё одна текстура: `sampler2D roughness_map`
- Читаем из неё значение roughness, яркость specular компоненты = (1 - roughness)

Задание 7

Добавляем specular light и material mapping

- Ещё одна текстура: `sampler2D roughness_map`
- Читаем из неё значение roughness, яркость specular компоненты = (1 - roughness)
- Для вычисления specular нужно направление на камеру $\Rightarrow$ нужно передать позицию камеры как uniform в шейдер

Задание 7

Добавляем specular light и material mapping

- Ещё одна текстура: `sampler2D roughness_map`
- Читаем из неё значение roughness, яркость specular компоненты = (1 - roughness)
- Для вычисления specular нужно направление на камеру $\Rightarrow$ нужно передать позицию камеры как uniform в шейдер
- Вычисляем интенсивность specular как

```
pow(max(0.0, dot(reflected_direction,
    camera_direction)), 4.0) * (1.0 - roughness)
```

  - 4 - настраиваемый параметр: чем больше, тем более точечным получается отражение

Задание 7

Добавляем specular light и material mapping

- Ещё одна текстура: `sampler2D roughness_map`
- Читаем из неё значение roughness, яркость specular компоненты = (1 - roughness)
- Для вычисления specular нужно направление на камеру $\Rightarrow$ нужно передать позицию камеры как uniform в шейдер
- Вычисляем интенсивность specular как

```
pow(max(0.0, dot(reflected_direction,
    camera_direction)), 4.0) * (1.0 - roughness)
```

  - 4 - настраиваемый параметр: чем больше, тем более точечным получается отражение
- Добавляем specular-компоненту к результирующему освещению

Задание 7

Добавляем specular light и material mapping

- Ещё одна текстура: `sampler2D roughness_map`
- Читаем из неё значение roughness, яркость specular компоненты = (1 - roughness)
- Для вычисления specular нужно направление на камеру $\Rightarrow$ нужно передать позицию камеры как uniform в шейдер
- Вычисляем интенсивность specular как

```
pow(max(0.0, dot(reflected_direction,
    camera_direction)), 4.0) * (1.0 - roughness)
```

  - 4 - настраиваемый параметр: чем больше, тем более точечным получается отражение
- Добавляем specular-компоненту к результирующему освещению
- Данные для текстуры - `brick_roughness`, texture unit = 2

Задание 8

Добавляем tone mapping

- Применяем к результирующему цвету пикселя простейший Reinhard operator: `color = color / (vec3(1.0) + color)`