Московский авиационный институт
(национальный исследовательский университет)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №6 по курсу «Компьютерная графика»

Студент: М. А. Инютин
Преподаватель: А. В. Морозов
Группа: М8О-307Б-19
Дата:
Оценка:
Подпись:

Москва, 2022

Создание шейдерных анимационных эффектов в OpenGL 2.1

Задача: Для поверхности, созданной в предыдущей лабораторной работе, обеспечить выполнение шейдерного эффекта.

Вариант задания: Шаровой сектор.

Анимация: Изменение цвета источника рассеянного света по синусоидальному закону.

1 Описание

Шейдер (Shader) — программа для процессора графической карты (GPU), управляющая поведением шейдерной стадии графического конвейера и занимающаяся обработкой соответствующих входных данных.

В работе используются вершинный и фрагментный шейдеры. Первый трансформирует вершины из локального пространства в пространство камеры. Второй отвечает за закрасу геометрического объекта, интерполирует параметры и расчитывает цвет отдельно взятого пикселя.

Шейдеры можно писать на GLSL (Graphics Library Shader Language) — высокоуровневом языке используемом для шейдеров OpenGL. Синтаксис языка очень похож на синтаксис C.

Для создания анимационного эффекта при каждой отрисовке шейдеру передаётся время, на основании которого программа пересчитывает цвет источника по синусоидальному закону.

2 Исходный код

Почти полностью совпадает с кодом предыдущей лабораторной работы.

Фрагментный шейдер *Phong.frag* реализует затенение Фонга, интерполирующее нормаль к точке. Для создания анимации интесивность источника пересчитывается на основании переданного значения времени.

```
#version 150 core

in vec3 position;
in vec3 normal;
out vec4 color;

uniform bool useSingleColor;
uniform vec3 singleColor3f;
uniform vec3 ka3f;
uniform vec3 kd3f;
uniform vec3 ks3f;
uniform vec3 light3f;
uniform vec3 ia3f;
uniform vec3 il3f;
uniform float p;
uniform bool animate;
uniform float time;

uniform vec3 camera = vec3(0, 0, -1e9f);
uniform float k = 0.5;

vec3 animate_vec(vec3 v) {
    vec3 res;
    for (int i = 0; i < 3; ++i) {
        res[i] = 0.5f * (1.0f + sin(asin(v[i] * 2.0f - 1.0f) + time));
    }
    return res;
}

void main(void) {
    if (useSingleColor) {
        if (animate) {
            color = vec4(animate_vec(singleColor3f), 1.0f);
        } else {
            color = vec4(singleColor3f, 1.0f);
        }
        return;
    }
    vec3 il3f_res = il3f;
    if (animate) {
        il3f_res = animate_vec(il3f);
    }
```

```
43      vec3 res = singleColor3f;
44      vec3 l = light3f - position;
45      float d = length(l);
46      l = normalize(l);
47      vec3 s = normalize(camera - position);
48      vec3 r = normalize(2 * normal * dot(normal, l) - l);
49      float diffusal = dot(l, normal);
50      float specular = dot(r, s);
51      if (diffusal < 1e-3) {
52          diffusal = 0;
53          specular = 0;
54      }
55      if (specular < 1e-3) {
56          specular = 0;
57      }
58      for (int i = 0; i < 3; ++i) {
59          res[i] = res[i] * (ia3f[i] * ka3f[i] + il3f_res[i] * (kd3f[i] * diffusal + ks3f
                [i] * pow(specular, p)) / (d + k));
60      }
61      color = vec4(res, 1.0f);
62  }
```

3 Демонстрация работы программы

Источник света меняет свою интесивность по синусоидальному закону.