

Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной
математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Компьютерная графика»

Студент: Д. А. Ваньков
Преподаватель: Г. С. Филиппов
Группа: М8О-307Б
Дата:
Оценка:
Подпись:

Москва, 2019

Основы построения фотореалистичных изображений.

Задача: Создать графическое приложение с использованием OpenGL. Используя результаты Л.Р.№3, изобразить заданное тело (то же, что и в л.р. №3) с использованием средств OpenGL 2.1. Использовать буфер вершин. Точность аппроксимации тела задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель освещения на GLSL. Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме

Вариант №18:

Прямой усеченный круговой цилиндр

1 Описание

Для реализации я использовал библиотеки GLFW для работы с оконными приложениями, GLM для математических операций, GLEW для упрощения работы с функциями библиотек и кросс-платформенности, а также язык C++ и OpenGL/.

Библиотека позволяет отрисовывать следующие геометрические единицы: точку, прямую, треугольник. Поэтому для реализации отрисуем цилиндр как n -ти угольник в основании, составленный из n треугольников с общей вершиной в центре масс и еще $2 * n$ треугольников(боковых граней) для соединения 2-ух оснований. Значение n зависит от точности аппроксимации, задаваемой с консоли.

Координаты вершин треугольников из которых составим фигуру вычислим и подадим библиотеке в нормализованных координатах в диапазоне $[-1, 1]$.

Вращение реализуем при помощи кватернионов для более гладкого вращения и решения проблемы шарнирного замка. Для расчета новых координат при вращении будем преобразовывать кватернионы в эквивалентные матрицы вращения и делать матричное произведение для получения новых координат.

Для каждой точки фигуры в программе вычисляется нормаль поверхности в этой точке. Так же в программе задано положение(в координатах) источника света. С помощью этих координат и нормали в точке реализуется основное освещение в конкретной точке, чилу которого можно регулировать с помощью значения вводимого с клавиатуры, также как и значения силы фонового и бликового освещения реализованного в программе.

2 Исходный код

Вначале подключаются необходимые библиотеки. В файле «multyplyes.h» реализованы кватернионы вращения и все необходимые функции для работы с ними. В файле «Camera.h» реализованы функции для работы и оптимального расположения камеры в отрисовываемом пространстве. Остальные включения - подключения библиотек: GLFW, GLM, GLEW.

```
1 | #include <iostream>
2 | #define GLEW_STATIC
3 | #include <GL/glew.h>
4 | #include <GLFW/glfw3.h>
5 | #include <glm/glm.hpp>
6 | #include <glm/gtc/matrix_transform.hpp>
7 | #include <glm/gtc/type_ptr.hpp>
8 | #include "multyplyes.h"
9 | #include "Camera.h"
```

Затем задаются глобальные переменные через которые реализуется мгновенное вращение, масштаб фигуры и сигнал к возврату в исходное положение. Помимо этого здесь инициализируется камера, положение источника света и создается массив для регистрации нажатий клавиатуры. Мелкость разбиения для более точного отображения фигуры хранится в переменной APPROX.

```
1 | const GLuint WIDTH = 800, HEIGHT = 600;
2 | Camera camera(glm::vec3(0.0f, 0.0f, 3.0f));
3 | GLfloat lastX = WIDTH / 2.0;
4 | GLfloat lastY = HEIGHT / 2.0;
5 | bool keys[1024];
6 | bool ret = true;
7 | GLfloat APPROX = 0.01f;
8 | GLfloat chill_height = 0.8f;
9 | glm::vec3 lightPos(1.2f, 1.0f, 2.0f);
10 | GLfloat deltaTime = 0.0f; // Time between current frame and last frame
11 | GLfloat lastFrame = 0.0f; // Time of last frame
12 | GLfloat x_rotation = 0.0f, y_rotation = 0.0f, z_rotation = 0.0f;
13 | GLfloat scale = 1.0f;
```

Функции реализующие в себе обработку пользовательских действий, таких как обработка изменений экрана, нажатия клавиш и движения мыши, имеют следующие объявления:

```
1 | void key_callback(GLFWwindow* window, int key, int scancode, int action, int mode);
2 | void mouse_callback(GLFWwindow* window, double xpos, double ypos);
```

```

3 | void scroll_callback(GLFWwindow* window, double xoffset, double yoffset);
4 | void do_movement();
5 | void new_func_size_callback(GLFWwindow* window, int width, int height);

```

Для генерации вектора с вершинами для отрисовки цилиндра используется следующая функция, способная сгенерировать цилиндр в зависимости от параметра аппроксимации.

```

1 | GLfloat* get_figure();

```

В функции *main* происходит инициализация библиотек, последовательный вызов всех этих функций и проводятся основные математические операции.

Для отображения окна использую функции, встроенные в OpenGL.

```

1 | glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
2 |   glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
3 |   glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
4 |   glfwWindowHint(GLFW_RESIZABLE, GL_TRUE);

```

Затем нужно инициализировать GLEW и остальные функции для отображения фигуры, света и подгрузить шейдеры.

```

1 |   glewInit();
2 |   // Define the viewport dimensions
3 |   glViewport(0, 0, WIDTH, HEIGHT);
4 |   // OpenGL options
5 |   glEnable(GL_DEPTH_TEST);
6 |   // Build and compile our shader program
7 |   Shader lightingShader("shaders/lighting.vs", "shaders/lighting.frag");
8 |   Shader lampShader("shaders/lamp.vs", "shaders/lamp.frag");
9 |   GLfloat* vertices = get_figure();
10 |   // First, set the container's VAO (and VBO)
11 |   GLuint VBO, containerVAO;
12 |   glGenVertexArrays(1, &containerVAO);
13 |   glGenBuffers(1, &VBO);
14 |   glBindBuffer(GL_ARRAY_BUFFER, VBO);
15 |   glBufferData(GL_ARRAY_BUFFER, sizeof(GLfloat) * (1 + (unsigned)(2/APPROX)) * 9 * 4
16 |               * 2, vertices, GL_STATIC_DRAW);
17 |   glBindVertexArray(containerVAO);
18 |   // Position attribute
19 |   glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(GLfloat), (GLvoid*)0);
20 |   glEnableVertexAttribArray(0);
21 |   // Normal attribute
22 |   glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(GLfloat), (GLvoid*)(3 *
23 |               sizeof(GLfloat)));
24 |   glEnableVertexAttribArray(1);

```

```
23 || glBindVertexArray(0);
```

Остается только описать функцию обработки клавиш.

```
1 | void key_callback(GLFWwindow* window, int key, int scancode, int action, int mode)
2 | {
3 |     if (key == GLFW_KEY_ESCAPE && action == GLFW_PRESS)
4 |         glfwSetWindowShouldClose(window, GL_TRUE);
5 |     if (key >= 0 && key < 1024)
6 |     {
7 |         if (action == GLFW_PRESS)
8 |             keys[key] = true;
9 |         else if (action == GLFW_RELEASE)
10 |             keys[key] = false;
11 |     }
12 | }
```

3 Консоль

В консоли необходимо скомпилировать исходный код и запустить. Согласно заданию в окне необходимо будет ввести параметры освещения и точность аппроксимации.

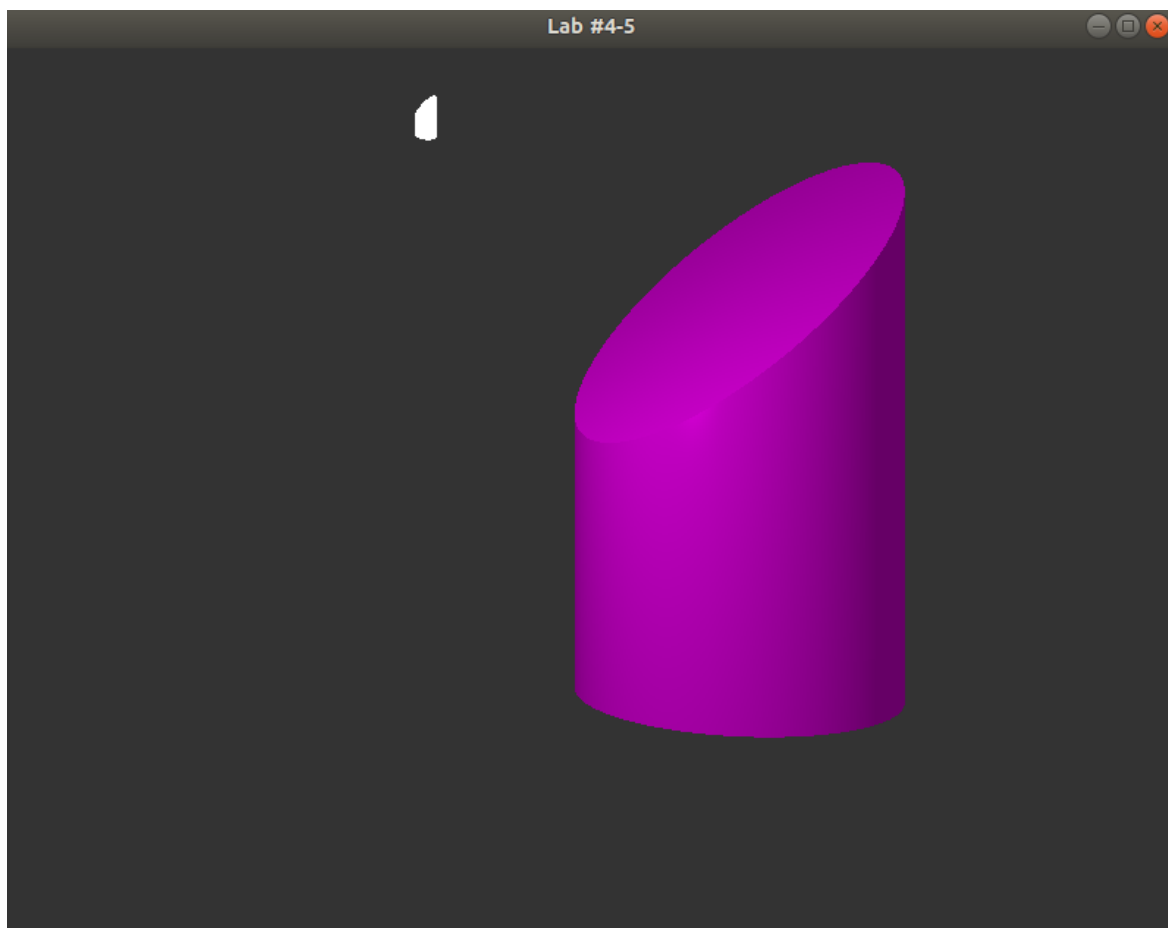
```
(base) chappybunny@chappybunny:~/CG/lab3$ g++ main.cpp -o start -lGL -lGLEW -lglfw
(base) chappybunny@chappybunny:~/CG/lab3$ ./start
Enter params of light:
>>Strenght of ambient light [0.0,1.0] (default 0.5): 0.5
>>Strenght of diffusion light [0.0,1.0] (default 0.5): 0.5
>>Strenght of specular light [0.0,1.0] (default 0.5): 0.5
4)Enter approximation parametr less then 1.0 (default ~0.02): 0.02

Success
Start
SUCCESSFUL::SHADER::PROGRAM::LINKING_SUCCESS
SUCCESSFUL::SHADER::PROGRAM::LINKING_SUCCESS
```

После откроется изображение фигуры в окне.

Это окно можно изменять по размерам и перемещать по экрану без всяких побочных эффектов, фигура подстраивается под изменение размеров экрана и масштабируется соответствующим образом.

С помощью нажатий клавиатуры можно вращать и масштабировать фигуру произвольным образом:



4 Выводы

Выполнив данную лабораторную работу по курсу «Компьютерная графика», я познакомился с OpenGL и ее функциями на языке C++. Как оказалось, данная библиотека очень удобна для отображения фигур.

В процессе написания необходимо было прочесть достаточное количество информации о библиотеке и про то, как в ней реализуются и отображаются объемные фигуры.