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: Используя результаты Л.Р.№2, аппроксимировать заданное тело выпуклым многогранником. Точность аппроксимации задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель закраски для случая одного источника света. 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++.

Библиотека позволяет отрисовывать следующие геометрические еденицы: 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 кординатах 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 приобразовывать кватернионы в эквивалентные матрицы вращения и делать матричное произведение для получения новых кординат.

Для каждой точки фигуры в программе вычисляется нормаль поверхности в этой точке. Так же в программе задано положение(в координатах) источнка света. С помощью этих кординат 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.

```
#include <iostream>
#define GLEW_STATIC
#include <GL/glew.h>
#include <GLFW/glfw3.h>
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>
#include "multyplyes.h"
#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.

```
const GLuint WIDTH = 800, HEIGHT = 600;
Camera camera(glm::vec3(0.0f, 0.0f, 3.0f));
GLfloat lastX = WIDTH / 2.0;
GLfloat lastY = HEIGHT / 2.0;
bool keys[1024];
bool ret = true;
GLfloat APPROX = 0.01f;
GLfloat chill_height = 0.8f;
glm::vec3 lightPos(1.2f, 1.0f, 2.0f);
GLfloat deltaTime = 0.0f; // Time between current frame and last frame
GLfloat lastFrame = 0.0f;  // Time of last frame
GLfloat x_rotation = 0.0f, y_rotation = 0.0f, z_rotation = 0.0f;
GLfloat scale = 1.0f;
```

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

```
void key_callback(GLFWwindow* window, int key, int scancode, int action, int mode);
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 heigh);
```

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

```
1 GLfloat* get_figure();
```

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

3 Консоль

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

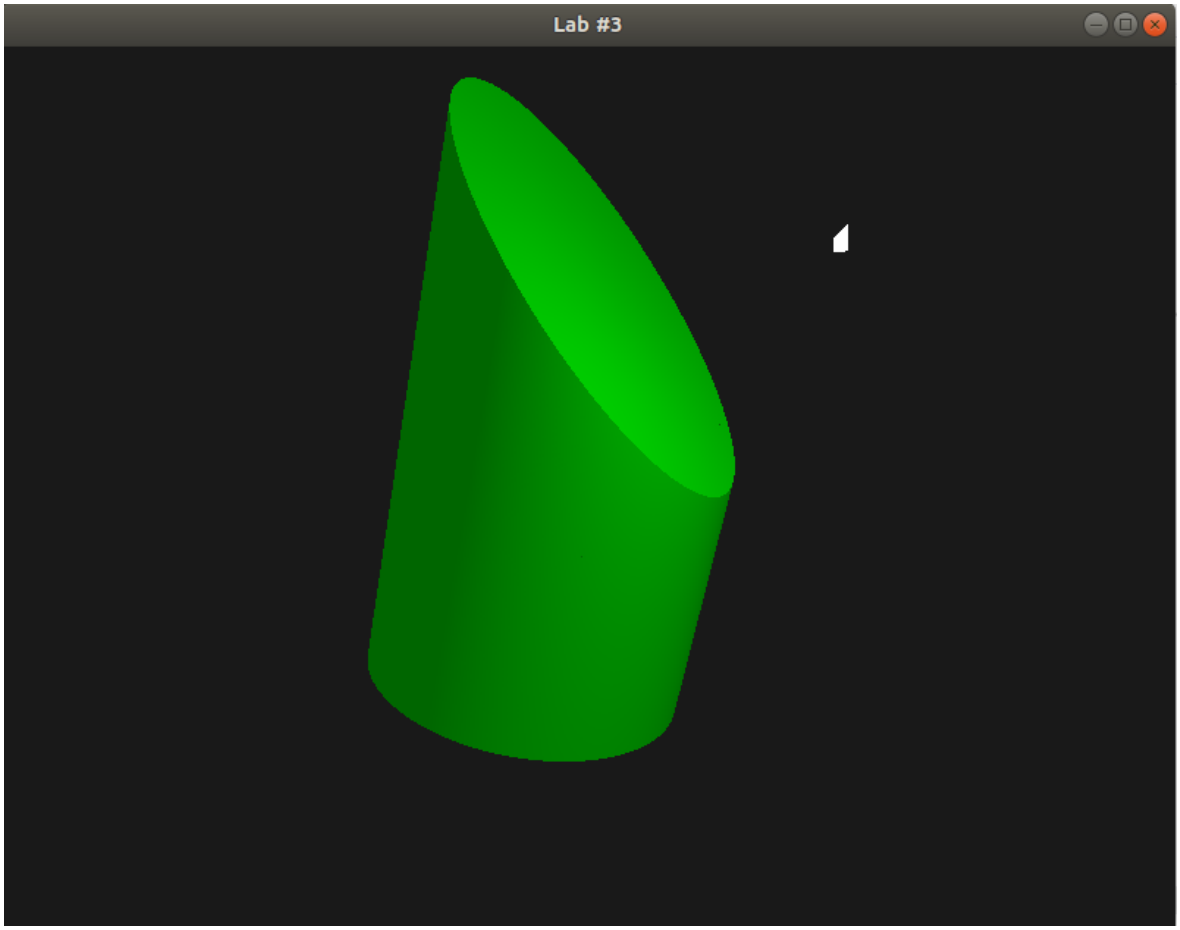
```
(base) chappybunny@chappybunny:~/CG/lab3$ g++ main.cpp -o start -lGL -lGLEW
-lglfw
(base) chappybunny@chappybunny:~/CG/lab3$ ./start
Enter params of light:
>>Strenght of ambient light [0.0,1.0] (default 0.5): 0.5
>>Strenght of diffusion light [0.0,1.0] (default 0.5): 0.5
>>Strenght of specular light [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 третью лабораторную работу по курсу «Компьютерная графика», я получил представление о том, как реализуется освещение в графических приложениях и попрактиковался в этом сам на языке C++. Также я научился создавать объемные фигуры с криволинеными поверхностями.

В процессе написания необходимо было прочитать про то как реализуются и отображаются объемные фигуры, то есть как математически расчитать и отрисовать данную фигуру.