

Московский авиационный институт  
(национальный исследовательский университет)

Факультет информационных технологий и прикладной  
математики

Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу «Компьютерная графика»

Студент: Я. С. Поскряков  
Преподаватель: Г. С. Филиппов  
Группа: М8О-306Б  
Дата:  
Оценка:  
Подпись:

Москва, 2019

# Каркасная визуализация выпуклого многогранника.

## Удаление невидимых линий

**Задача:** Разработать формат представления многогранника и процедуру его каркасной отрисовки в ортографической и изометрической проекциях. Обеспечить удаление невидимых линий и возможность пространственных поворотов и масштабирования многогранника. Обеспечить автоматическое центрирование и изменение размеров изображения при изменении размеров окна.

**Вариант №14:**

8-ми гранная прямая правильная пирамида.

## 1 Описание

Для реализации данной лабораторной работы я использовал библиотеки GLFW для работы с оконными приложениями, GLM для математических операций, GLEW для упрощения работы с функциями библиотек и кросс-платформенности, для языка C++.

Библиотеки позволяют отрисовывать следующие геометрические единицы: точку, прямую, треугольник. Поэтому для реализации отрисуем усеченную пирамиду как 4-ти угольник в нижнем и верхнем основаниях, составленные из 4 треугольников с общей вершиной в центре масс и еще 4 треугольников(боковых граней) с общей вершиной в вершине пирамиды.

Координаты вершин треугольников из которых составим фигуру вычислим и подадим библиотеке в нормированных 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.

Проблему удаления невидимых граней решает библиотечная реализация.

## 2 Исходный код

Вначале подключаются необходимые библиотеки. В файле «multiplyes.h» реализованы кватернионы вращения и все необходимые функции для работы с ними. Остальные включения - подключения библиоте: 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 "multplyes.h"

```

Затем задаются глобальные переменные через которые реализуется мгновенное вращение и сигнал к возврату в исходное положение.

```

1 | bool ret = false;
2 | GLfloat x_rotation = 0.0f, y_rotation = 0.0f, z_rotation = 0.0f;
3 | GLfloat scale = 1.0f;

```

В этой функции вызываются методы для инициализации используемых библиотек:

```

1 | GLFWwindow* InitOpenGL(GLuint width, GLuint height);

```

А здесь освобождается вся паамять выделенная для программы и обнуление внутренних состояний библиотек:

```

1 | void DestroyOpenGL(GLuint VAO, GLuint VBO, GLuint EBO);

```

Основной «игровой» цикл в котором происходит расчет поворотов при помощи матриц вращения и кватернионов, отрисовка фигуры и обработка функций обратного вызова вызывается этой функцией:

```

1 | void GameLoop(GLFWwindow* window, Shader& shader, GLuint VAO, GLuint VBO, GLuint EBO);

```

Методы для работы с клавиатурой представлены в 2-ух функциях: одна регистрирует нажатие, другая производит необходимые действия с глобальными переменными:

```

1 | void key_callback(GLFWwindow* window, int key, int scancode, int action, int mode);
2 | void do_movement();

```

Функция для обработки изменений размеров и положения окна:

```

1 | void new_func_size_callback(GLFWwindow* window, int width, int height);

```

Для генерации вектора с вершинами для отрисовки пирамиды используется следующая функция, способная сгенерировать усеченную пирамиду со сколь угодно большим количеством углов в основании. По заданию значение углов передаем равное 8:

```
1 || GLfloat* vertex_vector(GLuint num_of_angles);
```

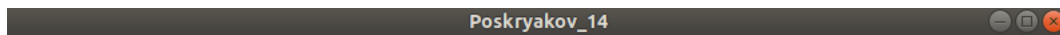
В функции *main* происходит последовательный вызов всех этих функций.

### 3 Консоль

В консоли необходимо скомпилировать исходный код и запустить.

```
(base) yar@yarmachine:~/CG/already/LAB2$ g++ main.cpp -o test -lGL -lGLEW -lglfw
(base) yar@yarmachine:~/CG/already/LAB2$ ./test
SUCCESSFUL::SHADER::PROGRAM::LINKING_SUCCESS
```

После откроется изображение графика в окне следующего типа:



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 вторую лабораторную работу по курсу «Компьютерная графика», я получил представление о том, как реализуется отображение объемных фигур на экране и научился создавать такие фигуры самостоятельно на языке C++.

В процессе решения задачи я столкнулся с такими трудностями, как отображение объемной фигуры на 2D мониторе. А также пришлось много читать о реализации вращения через кватернионы и решении проблемы шарнирного замка.