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: М. А. Бронников
Преподаватель: Г. С. Филиппов
Группа: М8О-307Б

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 №12:
5-ти гранная прямая правильная пирамида

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 пирамиду как 5-ти угольник в основании, составленный из 5 треугольников с общей вершиной в центре масс и еще 5 треугольников(боковых граней) с общей вершиной в вершине пирамиды.

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.

Проблему удаления невидимых граней решает библиотечная реализация.

2 Исходный код

Вначале подключаются необходимые библиотеки. В файле «multyplyes.h» реализованы кватернионы вращения и все необходимые функции для работы с ними. Остальные включения - подключения библиоте: GLFW, GLM, GLEW.

```
1 #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"
```

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

```
bool ret = false;
GLfloat x_rotation = 0.0f, y_rotation = 0.0f, z_rotation = 0.0f;
GLfloat scale = 1.0f;
```

В этой функции вызываются методы для инициализации используемых библиотек:

```
GLFWwindow* InitOpenGl(GLuint width, GLuint height);
```

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

```
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:

```
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:

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

Функция для обработки изменений размеров и положения окна:

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

Для генерации вектора с вершинами для отрисовки пирамиды используется следующая функция, способная сгенерировать пирамиду со сколь угодно большим количеством углов в основании. По заданию значение углов передаем равное 5:

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

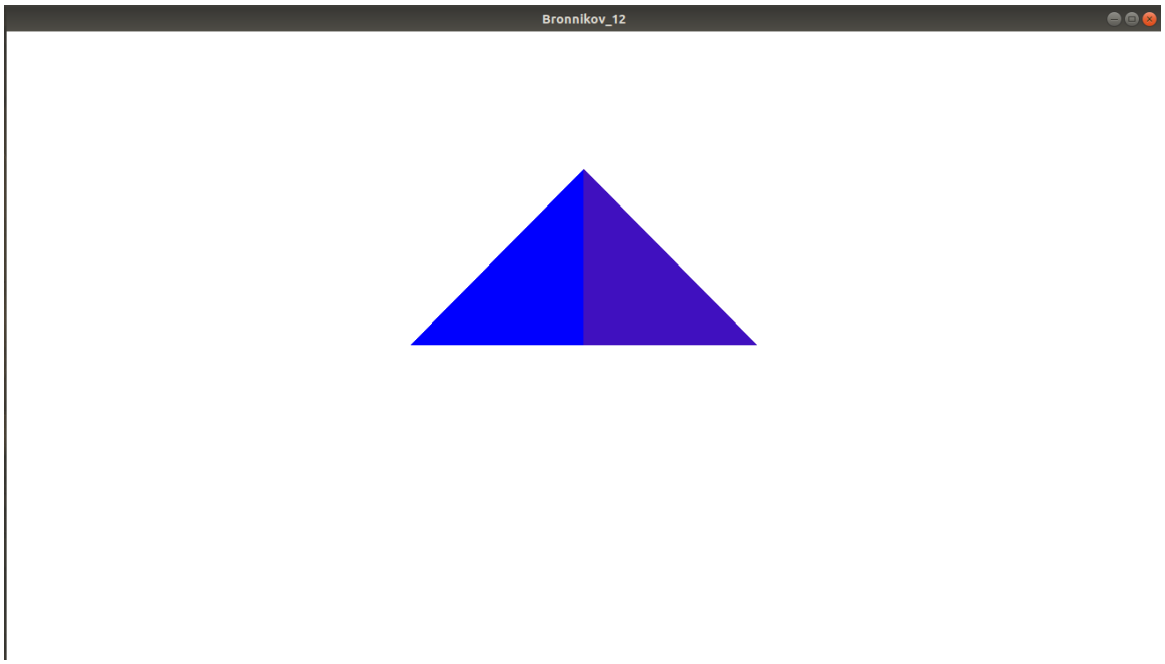
В функции *main* происходит последовательный вызов всех этих функций.

3 Консоль

В консоли необходимо скомпилировать исходный код и запустить. В окне необходимо будет ввести параметр a, от которого зависит вид графика.

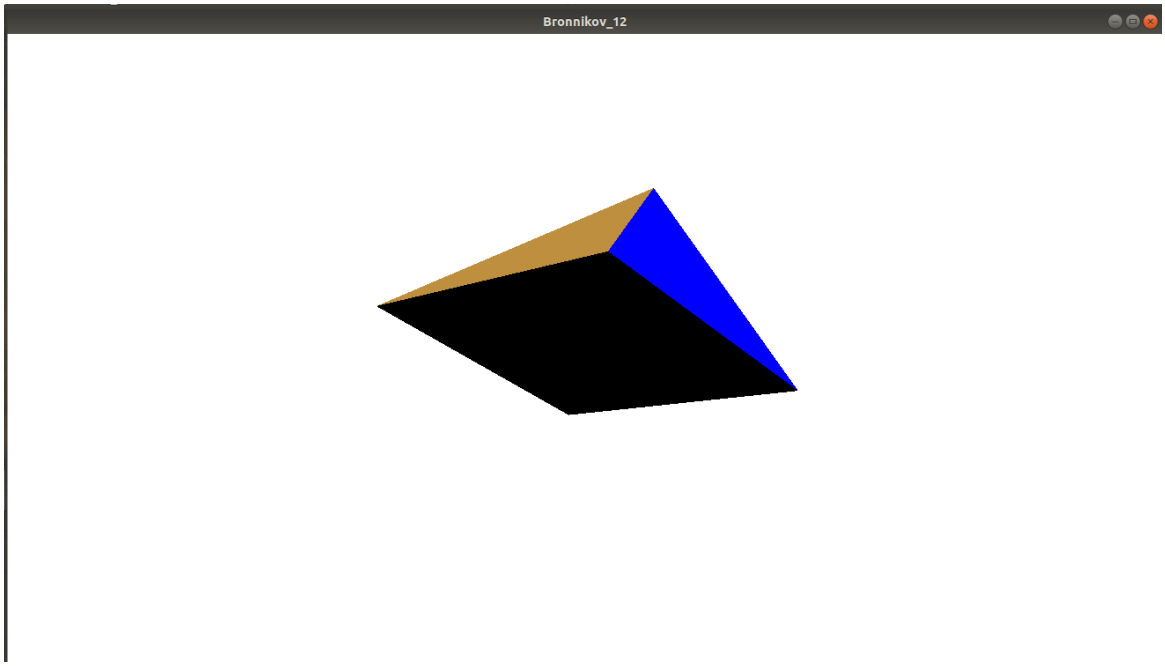
```
(base) max@max-X550CC:~/KG/lab2$ g++ main.cpp -o hello -lGL -lGLEW -lglfw
(base) max@max-X550CC:~/KG/lab2$ ls
hello  main.cpp  multyplyes.h  Shader.h  shaders
(base) max@max-X550CC:~/KG/lab2$ ./hello
SUCCESSFUL::SHADER::PROGRAM::LINKING_SUCCESS
(base) max@max-X550CC:~/KG/lab1$ exit
```

После откроется изображение графика в окне следующего типа:



Это окно можно изменять по размерам и перемещать по экрану без всяких побочных эффектов, график 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 создавть такие фигуры самостоятельно на языке C++.

В процессе решения задачи я столкнулся с такими трудностями, как отображение объемной фигуры на 2D мониторе. Немало злопот мне доставил поиск информации о функциях обратного вызова для реализации корректного масштабирования при изменении размеров окна, а также для обработки нажатий клавиш. Пришлось поэксперементировать с реализацией вращения через кватернионы и решением проблемы шарнирного замка.

Данная лабораторная придала мне уверенности в том, что я способен реализовывать не только консольные приложения, но и те, в которых есть какая-либо графика. Однако мне необходимо для этого еще очень много практиковаться в решении более сложных задач.