**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине** «Компьютерная 3D-графика»

Тема: Знакомство с графической библиотекой OpenGl.

Реализация примитивов.

Вариант № 44

|  |  |  |
| --- | --- | --- |
| Студенты гр. 3303 |  | Левшин П.И. |
|  |  | Переверзев Е.А. |
| Преподаватель |  | Герасимова Т.В. |

Санкт-Петербург

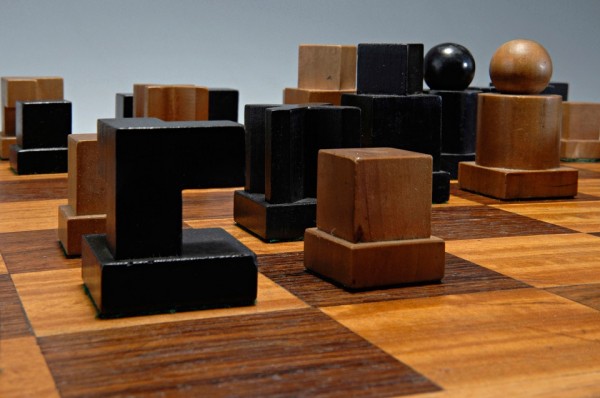
2017

**Цель работы.**

Ознакомление с графической библиотекой OpenGL.

**Содержательная постановка задачи.**

Реализовать примитивы (прямоугольники, окружности, квадраты), в соответствии с заданием, выданным преподавателем. Использование при реализации шейдеров обязательно.



**Среда разработки и язык программирования.**

Для дальнейшего выполнения лабораторных работ, была выбрана среда разработки VisualStudio 2013 , язык программирования c#, библиотека Tao Framework.

**Основные теоретические положения.**

**Tao Framework** — это библиотека, предоставляющая разработчикам .NET и Mono доступ к возможностям библиотеки OpenGL.

**Шейдер** - представляет собой часть шейдерной программы, которая заменяет собой часть графического конвейера видеокарты.

**Шейдерная программа** - это небольшая программа, состоящая из шейдеров (вершинного и фрагментного, возможны и другие) и выполняющаяся на графическом процессоре видеокарты.

**Вершинный шейдер** - заменяет часть графического конвейера, выполняющего преобразования, связанные с данными вершин. Такие как умножение вершин и нормалей на матрицу проекции и моделирования, установка цветов вершин, установка материалов освещения.

**Фрагментный шейдер** - заменяет часть графического ковейера (ГК), обрабатывая каждый полученный на предыдущих стадиях ГК фрагмент (не пиксель). Обработка может включать такие стадии, как получение данных из текстуры, просчет освещения, просчет смешивания.

**Разработка программы.**

В поставленном преподавателем задании можно увидеть следующие объекты:

* Прямоугольный параллелепипед
* Тройной угол
* Крест
* Цилиндр
* Сфера

Построение объектов производилось кинематическим способом:

1. Для параллелепипеда, тройного угла и креста
   1. Задание вершин прямоугольника
   2. Задание текстурных координат
   3. Формирование фигур за счет поворота копирования и поворота плоскости.
2. Для цилиндра и сферы
   1. Задание вершин сегмента фигуры
   2. Задание текстурных координат
   3. Формирование фигур за счет поворота сегмента вокруг оси.

Отображение фигур выполняется с помощью шейдеров:

Вершинный шейдер (Параллелепипед, Тройной угол, Крест)

#version 130

in vec3 vertexPosition;

in vec2 vertexUV;

out vec2 uv;

uniform mat4 projection\_matrix;

uniform mat4 view\_matrix;

uniform mat4 model\_matrix;

void main(void)

{

uv = vertexUV;

gl\_Position = projection\_matrix \* view\_matrix \* model\_matrix \* vec4(vertexPosition, 1);

}

Фрагментный шейдер (Параллелепипед, Тройной угол, Крест)

#version 130

uniform sampler2D texture;

in vec2 uv;

out vec4 fragment;

void main(void)

{

fragment = texture2D(texture, uv);

}

Вершинный шейдер (Цилиндр, Сфера)

#version 130

in vec3 vertexPosition;

in vec2 vertexUV;

out vec2 uv;

uniform mat4 segment\_matrix;

uniform mat4 rotate\_matrix;

uniform mat4 projection\_matrix;

uniform mat4 view\_matrix;

uniform mat4 model\_matrix;

void main(void)

{

uv = vertexUV;

if (vertexPosition.x == 0.5){

gl\_Position = projection\_matrix \* view\_matrix \* model\_matrix \* rotate\_matrix \* segment\_matrix \* vec4(1,0,0,1);

vec4 tempVec = ((rotate\_matrix \* segment\_matrix \* vec4(0.5,0,0,1)) + vec4(0.5, 0, 0.5, 0));

uv = vec2(tempVec.x, tempVec.z);

}

else {

vec4 tempVec = ((rotate\_matrix \* (vec4(vertexUV.x,0,vertexUV.y,1) - vec4(0.5, 0, 0.5, 0))) + vec4(0.5, 0, 0.5, 0));

uv = vec2(tempVec.x, tempVec.z);

gl\_Position = projection\_matrix \* view\_matrix \* model\_matrix \* rotate\_matrix \* vec4(vertexPosition, 1);

}

}

Фрагментный шейдер (Цилиндр, Сфера)

#version 130

in vec3 vertexPosition;

in vec2 vertexUV;

out vec2 uv;

uniform vec2 ind;

uniform mat4 projection\_matrix;

uniform mat4 view\_matrix;

uniform mat4 model\_matrix;

void main(void)

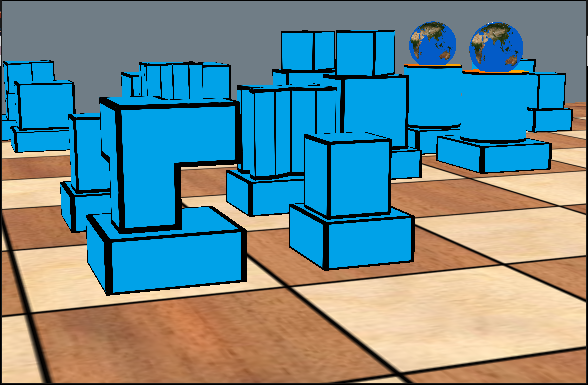
{

gl\_Position = projection\_matrix \* view\_matrix \* model\_matrix \* vec4(vertexPosition, 1);

uv = vec2(smoothstep(0, ind.y, ind.x + vertexUV.x),vertexUV.y);

}

**Результат работы программы.**



**Вывод**

В результате выполнения данной лабораторной работы были изучены основы работы с графической библиотекой OpenGL, а также реализованы объекты:

* Прямоугольный параллелепипед
* Тройной угол
* Крест
* Цилиндр
* Сфера