МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Череповецкий государственный университет»

**Лабораторная работа № 1**

**«Построение примитивов с помощью библиотеки OpenGL»**

**Выполнил:**

студент гр. 1ИВТпб-01-31оп

Климов А.Г.

**Проверил:** преподаватель

Юдина О.В., доцент  
Отметка о зачете:

Череповец

2017 год

**Цель работы**

Изучить основные процедуры и функции библиотеки OpenGL, приобрести практические навыки построения объемных геометрических фигур.

**Задание**

Построить с помощью библиотеки OpenGL куб в перспективной проекции, вращающийся вокруг своего центра. Направление вращения выбирает пользователь.

Внизу экрана должна выводиться текстовая строка вида «ФИО студента».

**Текст программы**

#include <GL/glut.h>

#include <iostream>

// Функция для вывода на экран

void display();

// Функция для управления

void specialKeys(int key, int x, int y);

// Переменные поворота

double rotate\_y = 0;

double rotate\_x = 0;

// Строка с фамилией и инициалами

std::string str;

void drawText(const char \*text, int length, int x, int y)

{

glMatrixMode(GL\_PROJECTION); // Задаём матрицу для 2-д рисования (режим матрицы проекций)

double \*matrix = new double[16];

glGetDoublev(GL\_PROJECTION\_MATRIX, matrix);

glLoadIdentity(); // Загрузка единичной матрицы

glOrtho(0, 400, 0, 400, -5, 5); // Границы объема отсечения

glMatrixMode(GL\_MODELVIEW); // Говорит о том, что работа будет теперь просмотром, а не проектом

glLoadIdentity(); // Загрузка единичной матрицы

glPushMatrix(); // Положить в стек текущую матрицу

glLoadIdentity(); // Загрузка единичной матрицы

glRasterPos2i(x, y); // Установить текущее растровое положение

for (int i = 0; i<length; i++)

{

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15, (int)text[i]); // Вывод одного символа в текущей позиции

}

glPopMatrix(); // Возвращаемся к старой системе координат

glMatrixMode(GL\_PROJECTION); // Задаём матрицу для 2-д рисования (режим матрицы проекций)

glLoadMatrixd(matrix); // Заменяет текущую матрицу новой

glMatrixMode(GL\_MODELVIEW); // Возврат в модельно-видовой режим

}

void display() {

// Очищаем экран и z-буферизацию

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

// Сброс трансформаций

glLoadIdentity();

// Вращение при изменении пользователем значений rotate\_x и rotate\_y

glRotatef(rotate\_x, 1.0, 0.0, 0.0);

glRotatef(rotate\_y, 0.0, 1.0, 0.0);

// Красная сторона - ПЕРЕДНЯЯ

glBegin(GL\_POLYGON);

glColor3f(1.0, 0.0, 0.0);

glVertex3f(0.6, -0.4, -0.4);

glVertex3f(0.6, 0.4, -0.4);

glVertex3f(-0.6, 0.4, -0.4);

glVertex3f(-0.6, -0.4, -0.4);

glEnd();

// Белая сторона - ЗАДНЯЯ

glBegin(GL\_POLYGON);

glColor3f(1.0, 1.0, 1.0);

glVertex3f(0.6, -0.4, 0.4);

glVertex3f(0.6, 0.4, 0.4);

glVertex3f(-0.6, 0.4, 0.4);

glVertex3f(-0.6, -0.4, 0.4);

glEnd();

// Фиолетовая сторона - ПРАВАЯ

glBegin(GL\_POLYGON);

glColor3f(1.0, 0.0, 1.0);

glVertex3f(0.6, -0.4, -0.4);

glVertex3f(0.6, 0.4, -0.4);

glVertex3f(0.6, 0.4, 0.4);

glVertex3f(0.6, -0.4, 0.4);

glEnd();

// Зеленая сторона - ЛЕВАЯ

glBegin(GL\_POLYGON);

glColor3f(0.0, 1.0, 0.0);

glVertex3f(-0.6, -0.4, 0.4);

glVertex3f(-0.6, 0.4, 0.4);

glVertex3f(-0.6, 0.4, -0.4);

glVertex3f(-0.6, -0.4, -0.4);

glEnd();

// Синяя сторона - ВЕРХНЯЯ

glBegin(GL\_POLYGON);

glColor3f(0.0, 0.0, 1.0);

glVertex3f(0.6, 0.4, 0.4);

glVertex3f(0.6, 0.4, -0.4);

glVertex3f(-0.6, 0.4, -0.4);

glVertex3f(-0.6, 0.4, 0.4);

glEnd();

// Оранжевая сторона - НИЖНЯЯ

glBegin(GL\_POLYGON);

glColor3f(1.0, 0.2, 0.0);

glVertex3f(0.6, -0.4, -0.4);

glVertex3f(0.6, -0.4, 0.4);

glVertex3f(-0.6, -0.4, 0.4);

glVertex3f(-0.6, -0.4, -0.4);

glEnd();

// Вывод текста

str = "Klimov A.G.";

drawText(str.data(), str.size(), 10, 10);

// Функции, которые дают эффект двойной буферизации

glFlush();

glutSwapBuffers();

}

void specialKeys(int key, int x, int y) {

if (key == GLUT\_KEY\_RIGHT)

rotate\_y += 5;

else if (key == GLUT\_KEY\_LEFT)

rotate\_y -= 5;

else if (key == GLUT\_KEY\_UP)

rotate\_x += 5;

else if (key == GLUT\_KEY\_DOWN)

rotate\_x -= 5;

// Запрос обновления экрана

glutPostRedisplay();

}

int main(int argc, char\* argv[]) {

glutInit(&argc, argv); // Инициализация GLUT и обработка пользовательских параметров

// Запрашиваем окно с поддержкой двойной буферизации, z-буферизации и цветовой схемы True Color

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

// Задаём позицию окна

glutInitWindowPosition(500, 200);

// Задаём размер окна

glutInitWindowSize(400, 400);

// Создаем окно

glutCreateWindow("Лабораторная работа 1");

// Активируем тест глубины Z-буферизации

glEnable(GL\_DEPTH\_TEST);

// Функции обратного вызова

glutDisplayFunc(display);

glutSpecialFunc(specialKeys);

// Передаем контроль над событиями в GLUT

glutMainLoop();

return 0;

}

**Контрольные вопросы**

*1. Каковы особенности работы с OpenGL под операционной системой Windows?*

OpenGL представляет собой универсальную графическую библиотеку, которая может быть реализована в любой оконной среде.

Для работы с OpenGL в Windows используется понятие контекста воспроизведения (rendering context), который связывает OpenGL с оконной системой Windows. Если обычный контекст устройства (device context) содержит информацию, относящуюся к графическим компонентам GDI, то контекст воспроизведения содержит информацию, относящуюся к OpenGL.

Таким образом, чтобы начать работать с командами OpenGL, приложение должно создать, как минимум, один контекст воспроизведения и сделать его текущим.

Перед созданием контекста воспроизведения необходимо установить формат пикселей. Для установки формата пикселей используется функция Windows GDIint ChoosePixelFormat(HDC, const PIXELFORMATDESCRIPTOR), выбирающая наиболее подходящий формат исходя из информации, переданной в полях структуры PIXELFORMATDESCRIPTOR.

После того как найден подходящий формат пикселей, следует установить его в контексте устройства при помощи функции BOOL SetPixelFormat(HDChDC, inl pixelFormat, const PIXELFORMATDESCRIPTOR).

Для работы с контекстом воспроизведения в Windows существуют функции HGLRC wglCreateContext(HDC hDC) и BOOL wglMakeCurrent(HDC hDC,HGLRC hGLRC).

Первая из них создаёт новый контекст воспроизведения OpenGL, который подходит для рисования на устройстве, задаваемом контекстом hDC. Вторая функция устанавливает текущий контекст воспроизведения.

По окончании работы с OpenGL созданный контекст воспроизведения необходимо удалить. Для этого существует функция BOOLwglDeleteContext(HGLRC hGLRC).

Текущий контекст воспроизведения можно узнать при помощи функции HGLRC wglGetCurrentContext().

*2. Что такое видовая матрица?*

Видовая матрица отвечает за систему координат создаваемой трехмерной модели. Обычно видовая матрица используется для таких преобразований как перенос, масштабирование и поворот.

Видовая матрица задается командой glMatrixMode(GL\_MODELVIEW). Для стандартных операций используются следующие команды:

1. glRotate – задается матрица вращения;

2. glTranslate – задается матрица переноса;

3. glScale – матрица масштабирования.

При этом данные команды формируют матрицу на основе параметров и умножают ее на текущую видовую матрицу. Результат умножения записывается в качестве видовой матрицы.

Область просмотра задается командой glViewport, которая задает преобразование из координат OPENGL в координаты окна.

Ориентация задается командой glLookAt, в которой задаются три вектора: вектор, описывающий точку наблюдения; вектор, описывающий центр сцены и вектор, задающий положительное направление оси Y.

Видовая матрица отвечает за расположение оси координат.

*3. Как задать матрицу проецирования?*

Матрица проецирования задается командой glMatrixMode(GL\_PROJECTION). Затем необходимо задать матрицу проецирования. Для этого существуют следующие функции:

1. glLoadMatrix - загружает матрицу из массива.

2. glLoadIdentity – загружает единичную матрицу.

3. glMultMatrix – умножает на матрицу, заданную как массив.

Для работы с основными видами проецирования используются вспомогательные команды:

1. Задание ортографической проекции выполняется через команду glOrtho, которая задает параллелограмм видимости в пространстве сцены. При этом считается, что глаз наблюдателя расположен в точке (0,0,0).

2. Для перспективного проецирования используется команда glPerspective, которая задает усечённый конус видимости в пространстве.

При этом данные команды формируют матрицу на основе параметров и умножают ее на текущую матрицу проецирования. Результат умножения записывается в качестве матрицы проецирования.

Матрица проецирования отвечает за координаты самой фигуры.

*4. Список изображений.*

Если нужно несколько раз обращаться к одной и той же группе команд, эти команды можно объединить в так называемый список изображений и вызывать его при необходимости. Для того чтобы создать новый список изображений надо поместить все команды, которые должны в него войти между командными скобками:

GlNewList, glEndList

Для различения списков используются целые положительные числа, задаваемые при создании списка значением параметра list, а параметр mode определяет режим обработки команд, входящих в список:

GL\_COMPILE (команды записываются в список без выполнения);

GL\_COMPILE\_AND\_EXECUTE (команды сначала выполняются, а затем записываются в список).

После того, как список создан, его можно вызвать командой glCallList, указав в параметре list идентификатор нужного списка. Чтобы вызвать сразу несколько списков, можно воспользоваться командой glCallLists, вызывающей n списков с идентификаторами из массива. Для удаления списков используется команда glDeleteLists. Номер списка вычисляется относительно базового значения. По умолчанию базовое значение равно нулю. Установка нового базового значения осуществляется посредством вызова glListBase.

*5. Как выводится текст в OpenGL?*

Для вывода текста в OpenGL требуется создать список изображения для каждой буквы. Данную операцию можно выполнить с помощью функции wglUseFontBitmaps, которая создает набор списков изображений для каждой буквы. Например: glCallLists(24,GL\_UNSIGNED\_BYTE,"Hello Win32 OpenGL World");