МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Компьютерная графика»

Тема: Примитивы OpenGL

Студент гр. 8383 _____ Ларин А.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург

2021

Задание

Разработать программу, реализующую представление определенного набора примитивов из имеющихся в библиотеке OpenGL (GL_POINT, GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUADS, GL_QUAD_STRIP, GL_POLYGON).

Разработанная на базе шаблона программа должна быть пополнена возможностями остановки интерактивно различных атрибутов примитивов рисования через вызов соответствующих элементов интерфейса пользователя

Общие сведения

В данной лабораторной работе должны быть рассмотрены следующие примитивы:

GL_POINTS – каждая вершина рассматривается как отдельная точка, параметры которой не зависят от параметров остальных заданных точек. При этом вершина n определяет точку n. Рисуется N точек (n – номер текущей вершины, N – общее число вершин).

Основой графики OpenGL являются вершины. Для их определения используется команда glVertex.

void glVertex[2 3 4][s i f d](type coord)

Вызов команды определяется четырьмя координатами x, y, z и w. При этом вызов glVertex2* устанавливает координаты x и y, координата z полагается равной 0, а w – 1. Вызов glVertex3* устанавливает координаты x, y, z, а w равно 1.

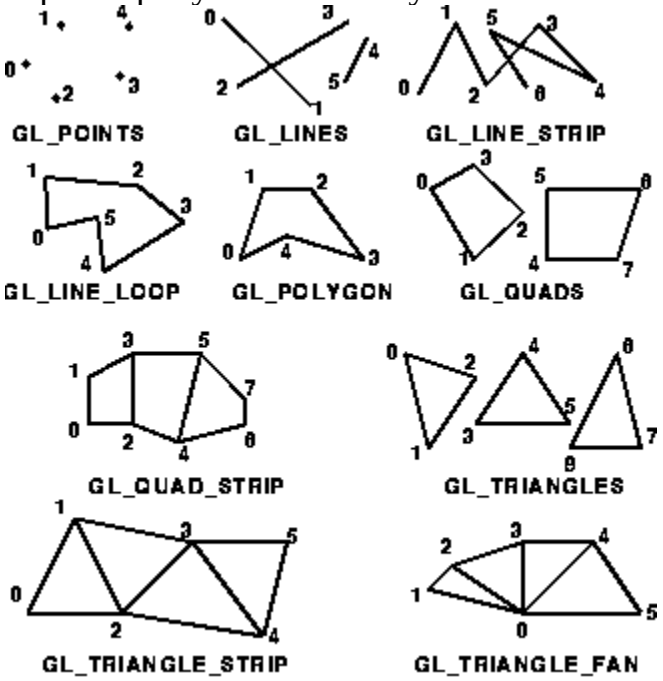
GL_LINES – каждая пара вершин рассматривается как независимый отрезок. Первые две вершины определяют первый отрезок, следующие две – второй отрезок и т.д., вершины (2n-1) и 2n определяют отрезок n. Всего рисуется N/2 линий. Если число вершин нечетно, то последняя просто игнорируется.

GL_LINE_STRIP – в этом режиме рисуется последовательность из одного или нескольких связанных отрезков. Первая вершина задает начало первого отрезка, а вторая – конец первого, который является также началом второго. В общем случае, вершина n (n > 1) определяет начало отрезка n и конец отрезка (n - 1). Всего рисуется (N - 1) отрезок.

GL_LINE_LOOP – осуществляется рисование замкнутой кривой линии. Первая вершина задает начало первого отрезка, а вторая – конец первого, который является также началом второго. В общем случае, вершина n (n > 1) определяет начало отрезка n и конец отрезка (n - 1). Первая вершина является концом последнего отрезка. Всего рисуется N отрезков.

GL_TRIANGLES – каждая тройка вершин рассматривается как независимый треугольник. Вершины (3n-2), (3n-1), 3n (в таком порядке) определяют треугольник n. Если число вершин не кратно 3, то оставшиеся (одна или две) вершины игнорируются. Всего рисуется N/3 треугольника.

GL_TRIANGLE_STRIP - в этом режиме рисуется группа связанных треугольников, имеющих общую грань. Первые три вершины определяют первый треугольник, вторая, третья и четвертая – второй и т.д. для нечетного n вершины n, (n+1) и (n+2) определяют треугольник n. Для четного n треугольник определяют вершины (n+1), n и (n+2). Всего рисуется (N-2) треугольника.
GL_TRIANGLE_FAN - в этом режиме рисуется группа связанных треугольников, имеющих общие грани и одну общую вершину. Первые три вершины определяют первый треугольник, первая, третья и четвертая – второй и т.д. Всего рисуется (N-2) треугольника.
GL_QUADS – каждая группа из четырех вершин рассматривается как независимый четырехугольник. Вершины (4n-3), (4n-2), (4n-1) и 4n определяют четырехугольник n. Если число вершин не кратно 4, то оставшиеся (одна, две или три) вершины игнорируются. Всего рисуется N/4 четырехугольника.
GL_QUAD_STRIP – рисуется группа четырехугольников, имеющих общую грань. Первая группа из четырех вершин задает первый четырехугольник. Третья, четвертая, пятая и шестая задают второй четырехугольник.
GL_POLYGON – задет многоугольник. При этом число вершин равно числу вершин рисуемого многоугольника.



GL_FRONT - для лицевых граней, GL_BACK - для обратных граней, GL_FRONT_AND_BACK - для всех граней. Параметр mode может быть равен:
GL_POINT при таком режиме будут отображаться только вершины многоугольников.
GL_LINE при таком режиме многоугольник будет представляться набором отрезков.
GL_FILL при таком режиме многоугольники будут закрашиваться текущим цветом с учетом освещения, и этот режим установлен по умолчанию.
Также можно указывать, какой тип граней отображать на экране. Для этого сначала на-до установить соответствующий режим вызовом команды glEnable

(GL_CULL_FACE), а за- тем выбрать тип отображаемых граней с помощью команды void glCullFace (GLenum mode).
Вызов с параметром GL_FRONT приводит к удалению из изображения всех лицевых граней, а с параметром GL_BACK – обратных (установка по умолчанию).
Кроме рассмотренных стандартных примитивов в библиотеках GLU и GLUT описаны более сложные фигуры, такие как сфера, цилиндр, диск (в GLU) и сфера, куб, конус, тор, тетраэдр, додекаэдр, икосаэдр, октаэдр и чайник (в GLUT). Автоматическое наложение текстуры предусмотрено только для фигур из библиотеки GLU.
Например, чтобы нарисовать сферу или цилиндр, надо сначала создать объект специального типа GLUquadricObj с помощью команды GLUquadricObj* gluNewQuadric(void);а затем вызвать соответствующую команду:

```
void gluSphere (GLUquadricObj * qobj, GLdouble radius,
                GLint slices, GLint stacks);

void gluCylinder (GLUquadricObj * qobj,
                  GLdouble baseRadius,
                  GLdouble topRadius,
                  GLdouble height, GLint slices,
                  GLint stacks);
```

где параметр slices задает число разбиений вокруг оси z, a stacks – вдоль оси z.

Цель, требования и рекомендации к выполнению задания

Цель выполнения задания: ознакомление с основными примитивами OpenGL.
Требования и рекомендации к выполнению задания:
- проанализировать полученное задание, выделить информационные объекты и действия;
- разработать программу с использованием требуемых примитивов и атрибутов.

Задания

Разработать программу, реализующую представление определенного набора примитивов (4) из имеющихся в OpenGL (GL_POINT, GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUADS, GL_QUAD_STRIP, GL_POLYGON).
Разработанная на базе шаблона программа должна быть пополнена возможностями остановки интерактивно различных атрибутов примитивов рисования через вызов соответствующих элементов интерфейса пользователя

Пример выполнения задания

GL_LINES – каждая пара вершин рассматривается как независимый отрезок. Первые две вершины определяют первый отрезок, следующие две – второй отрезок и т. д., вершины (2n-1) и 2n определяют отрезок n. Всего рисуется N/2 линий. Если число вершин нечетно, то последняя просто игнорируется.
Листинг программы:

```
glLineWidth(12.0f);        // определение толщины линии
   glColor3f(0.0f, 0.1f, 0.9f);
   glBegin(GL_LINES);      // сплошная линия
```

```
        glVertex2f(-0.75f, 0.45f);
        glVertex2f(0.75f, 0.45f);
    glEnd();

//в одной строке три отрезка с разным стилем рисования
     glColor3f(0.5f, 0.7f, 1.0f);
     glEnable(GL_LINE_STIPPLE);
     glLineStipple(1,0x0101);    // точечный пунктир
    glBegin(GL_LINES);
        glVertex2f(-0.75f, -0.25f);
        glVertex2f(-0.21f, -0.25f);
     glEnd();

     glColor3f(0.0f, 0.1f, 0.9f);
     glLineStipple(1,0x00FF);    // штриховой пунктир
    glBegin(GL_LINES);
        glVertex2f(-0.21f, -0.25f);
        glVertex2f(0.21f, -0.25f);
     glEnd();

     glColor3f(0.0f, 0.5f, 0.7f);
     glLineStipple(1,0x1C47);    // штрих точка штрих
    glBegin(GL_LINES);
        glVertex2f(0.21f, -0.25f);
        glVertex2f(0.75f, -0.25f);
     glEnd();

glDisable(GL_LINE_STIPPLE);
```

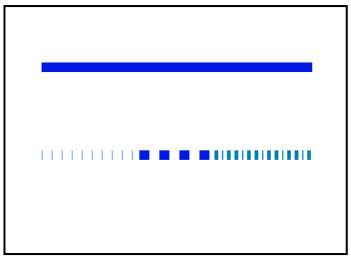


Рис. - пример работы программы

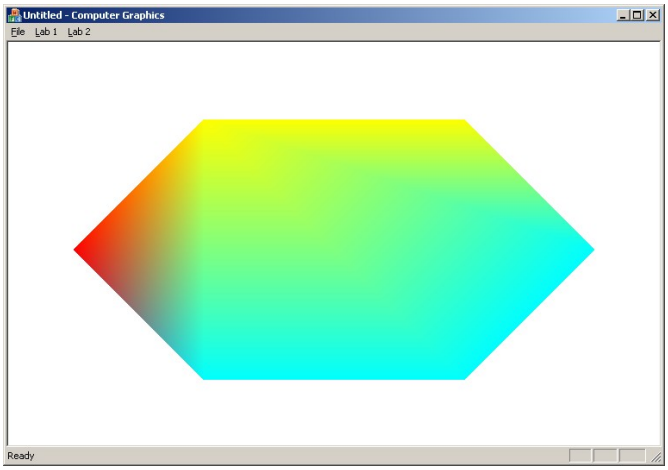


Рис. Вывод многоугольника

GL_POLYGON – задает многоугольник. При этом число вершин равно числу вершин рисуемого многоугольника. Код программы

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
glBegin(GL_POLYGON);
    glColor4d(1.0, 1.0, 0.0, 0.7);
    glVertex2d(-0.4,  0.0);
    glVertex2d( 0.4,  0.0);
    glColor4d(0.0, 1.0, 1.0, 0.7);
    glVertex2d( 0.8, -0.4);
    glVertex2d( 0.4, -0.8);
    glVertex2d(-0.4, -0.8);
    glColor4d(1.0, 0.0, 0.0, 0.7);
    glVertex2d( -0.8, -0.4);
glEnd();
glDisable(GL_POLYGON_STIPPLE);
```

Выполнение

Задание вполнено на языке C++ с использованием фреймворка Qt5.
Qt предоставляет возможность использование OpenGL посредством класса QGLWidget. От него отнаследован класс CustomGL.

```
class CustomGL : public QGLWidget
```

, в котором переопределены методы:

- `void initializeGL();`
  Задает контекст рендеринга. Заключается в установке фона белым
- `void resizeGL(int nWidth, int nHeight);`
  `Для корректного масштабирования. Размер окна задается через glViewport.`
- `void paintGL();`
  Вызов этой функции происходит при рисовании. Она в свою очередь вызывает функцию `scene` где прописана вся логика рисования.

В данном классе так же прописаны шесть точек и цвета для них.
Так же определена переменная, хранящая тип примитива и слот для задания данной переменной из GUI.
Функция `scene` проверяет требуемый тип геометрии, которую требуется отрисовать и вызывает `drawGeometry` с нужным параметром.
`drawGeometry` задает размер точек и линий и последовательно задает шесть предварительно заданных точек(`glVertex2f`) и их цвета(`glColor3f`).
В результате на экране отображается нужный примитив.

В коде главного окна инициализируются элементы интерфейса, помещаются в layout'ы и связываются сигналы со слотами.

Выводы.

Были изучены основы работы с OpenGL, отрисованы первые примитивы и осуществлено управление отрисовкой при помощи GUI.