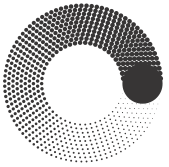
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

***Факультет информационных технологий***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.04.02 «Информационные системы и технологии»,**

**ЛАБОРАТОРНАЯ РАБОТА № 7**

**Дисциплина: Разработка мобильных приложений для Android**

**Тема: Работа с OpenGL ES**

**Выполнил(а): студент(ка) группы 224-371** Лейн Ф.Е.

(Фамилия И.О.)

**Проверил(а):** Якубовский К.И.

(Фамилия И.О. )

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**ЦЕЛЬ РАБОТЫ**

Разработать с использование OpenGL ES 2.0 (или выше) приложение, в котором отображается вращающийся куб.

**ХОД РАБОТЫ**

Для рендеринга куба, был создан класс Cube, хранящий данные вершин, вертиксов и цветов, что показано в листинге 1. А также, обеспечивающий отрисовку этого куба в методе draw(GL10 gl). Полный исходных код этого класса находится в приложении А.

Листинг 1 – данные для отрисовки куба

**private static final float**[] ***VERTICES*** = {  
 -1.0f, -1.0f, -1.0f,  
 1.0f, -1.0f, -1.0f,  
 1.0f, 1.0f, -1.0f,  
 -1.0f, 1.0f, -1.0f,  
 -1.0f, -1.0f, 1.0f,  
 1.0f, -1.0f, 1.0f,  
 1.0f, 1.0f, 1.0f,  
 -1.0f, 1.0f, 1.0f  
};  
  
**private static final float**[] ***COLORS*** = {  
 0.0f, 1.0f, 0.0f, 1.0f,  
 0.0f, 1.0f, 0.0f, 1.0f,  
 1.0f, 0.5f, 0.0f, 1.0f,  
 1.0f, 0.5f, 0.0f, 1.0f,  
 1.0f, 0.0f, 0.0f, 1.0f,  
 1.0f, 0.0f, 0.0f, 1.0f,  
 0.0f, 0.0f, 1.0f, 1.0f,  
 1.0f, 0.0f, 1.0f, 1.0f  
};  
  
**private static final byte**[] ***INDICES*** = {  
 0, 4, 5, 0, 5, 1,  
 1, 5, 6, 1, 6, 2,  
 2, 6, 7, 2, 7, 3,  
 3, 7, 4, 3, 4, 0,  
 4, 7, 6, 4, 6, 5,  
 3, 0, 1, 3, 1, 2  
};

Непосредственно за отрисовку данных, отвечает класс OpenGLRenderer. Методы onSurfaceChanged и onDrawFrame, что показано в листинге 2, в основном эквивалентны glutReshapeFunc.glutDisplayFunc. Первый вызывается при изменении размера поверхности, например, когда телефон переключается между альбомным и портретным режимами, второй вызывается каждым кадром, и именно там происходит отрисовка куба

Листинг 2 – класс OpenGLRenderer

@Override  
**public void** onSurfaceCreated(GL10 gl, EGLConfig config) {  
 gl.glClearColor(0.0f, 0.0f, 0.0f, 0.5f);  
  
 gl.glClearDepthf(1.0f);  
 gl.glEnable(GL10.***GL\_DEPTH\_TEST***);  
 gl.glDepthFunc(GL10.***GL\_LEQUAL***);  
  
 gl.glHint(GL10.***GL\_PERSPECTIVE\_CORRECTION\_HINT***,  
 GL10.***GL\_NICEST***);  
}  
  
@Override  
**public void** onDrawFrame(GL10 gl) {  
 gl.glClear(GL10.***GL\_COLOR\_BUFFER\_BIT*** | GL10.***GL\_DEPTH\_BUFFER\_BIT***);  
 gl.glLoadIdentity();  
  
 gl.glTranslatef(0.0f, 0.0f, -10.0f);  
 gl.glRotatef(**cubeRotation**, 1.0f, 1.0f, 1.0f);  
  
 **cube**.draw(gl);  
  
 gl.glLoadIdentity();  
  
 **cubeRotation** -= 0.15f;  
}  
  
@Override  
**public void** onSurfaceChanged(GL10 gl, **int** width, **int** height) {  
 gl.glViewport(0, 0, width, height);  
 gl.glMatrixMode(GL10.***GL\_PROJECTION***);  
 gl.glLoadIdentity();  
 GLU.*gluPerspective*(gl, 45.0f, (**float**)width / (**float**)height, 0.1f, 100.0f);  
 gl.glViewport(0, 0, width, height);  
  
 gl.glMatrixMode(GL10.***GL\_MODELVIEW***);  
 gl.glLoadIdentity();  
}

Для обеспечения вращения куба, в классе имеется глобальная переменная cubeRotation, из которой каждый кадр производится вычитание небольшого числа. Далее, эта переменная используется в методе gl.glRotatef(cubeRotation, 1.0f, 1.0f, 1.0f).

Пример отрисовки куба (результата работы приложения) изображён на рисунке 1.

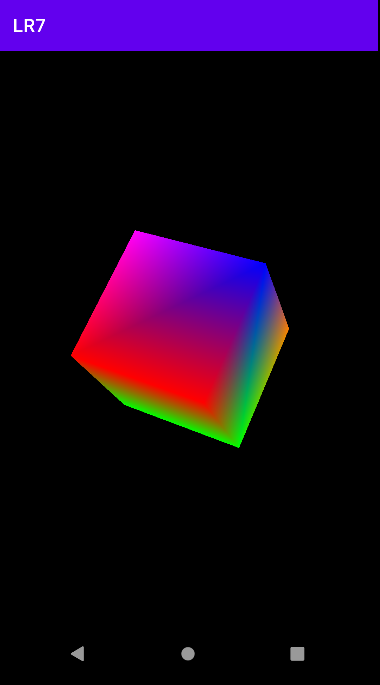


Рисунок 1 – Результат работы приложения

**ВЫВОД**

В результате выполнения данной лабораторной работы были изучены навыки отрисовки 3д-объектов, с использованием библиотеки OpenGL, а также, создано приложение реализующее отрисовку вращающегося куба.

**Приложение А. Листинг кода класса Cube**

**package** com.labs.lr7;  
  
**import** java.nio.ByteBuffer;  
**import** java.nio.ByteOrder;  
**import** java.nio.FloatBuffer;  
  
**import** javax.microedition.khronos.opengles.GL10;  
  
**class** Cube {  
  
 **private final** FloatBuffer **vertexBuffer**;  
 **private final** FloatBuffer **colorBuffer**;  
 **private final** ByteBuffer **indexBuffer**;  
  
 **private static final float**[] ***VERTICES*** = {  
 -1.0f, -1.0f, -1.0f,  
 1.0f, -1.0f, -1.0f,  
 1.0f, 1.0f, -1.0f,  
 -1.0f, 1.0f, -1.0f,  
 -1.0f, -1.0f, 1.0f,  
 1.0f, -1.0f, 1.0f,  
 1.0f, 1.0f, 1.0f,  
 -1.0f, 1.0f, 1.0f  
 };  
  
 **private static final float**[] ***COLORS*** = {  
 0.0f, 1.0f, 0.0f, 1.0f,  
 0.0f, 1.0f, 0.0f, 1.0f,  
 1.0f, 0.5f, 0.0f, 1.0f,  
 1.0f, 0.5f, 0.0f, 1.0f,  
 1.0f, 0.0f, 0.0f, 1.0f,  
 1.0f, 0.0f, 0.0f, 1.0f,  
 0.0f, 0.0f, 1.0f, 1.0f,  
 1.0f, 0.0f, 1.0f, 1.0f  
 };  
  
 **private static final byte**[] ***INDICES*** = {  
 0, 4, 5, 0, 5, 1,  
 1, 5, 6, 1, 6, 2,  
 2, 6, 7, 2, 7, 3,  
 3, 7, 4, 3, 4, 0,  
 4, 7, 6, 4, 6, 5,  
 3, 0, 1, 3, 1, 2  
 };  
  
 **public** Cube() {  
 ByteBuffer byteBuf = ByteBuffer.*allocateDirect*(***VERTICES***.**length** \* 4);  
 byteBuf.order(ByteOrder.*nativeOrder*());  
 **vertexBuffer** = byteBuf.asFloatBuffer();  
 **vertexBuffer**.put(***VERTICES***);  
 **vertexBuffer**.position(0);  
  
 byteBuf = ByteBuffer.*allocateDirect*(***COLORS***.**length** \* 4);  
 byteBuf.order(ByteOrder.*nativeOrder*());  
 **colorBuffer** = byteBuf.asFloatBuffer();  
 **colorBuffer**.put(***COLORS***);  
 **colorBuffer**.position(0);  
  
 **indexBuffer** = ByteBuffer.*allocateDirect*(***INDICES***.**length**);  
 **indexBuffer**.put(***INDICES***);  
 **indexBuffer**.position(0);  
 }  
  
 **public void** draw(GL10 gl) {  
 gl.glFrontFace(GL10.***GL\_CW***);  
  
 gl.glVertexPointer(3, GL10.***GL\_FLOAT***, 0, **vertexBuffer**);  
 gl.glColorPointer(4, GL10.***GL\_FLOAT***, 0, **colorBuffer**);  
  
 gl.glEnableClientState(GL10.***GL\_VERTEX\_ARRAY***);  
 gl.glEnableClientState(GL10.***GL\_COLOR\_ARRAY***);  
  
 gl.glDrawElements(GL10.***GL\_TRIANGLES***, 36, GL10.***GL\_UNSIGNED\_BYTE***,  
 **indexBuffer**);  
  
 gl.glDisableClientState(GL10.***GL\_VERTEX\_ARRAY***);  
 gl.glDisableClientState(GL10.***GL\_COLOR\_ARRAY***);  
 }  
}

**Приложение Б. Листинг кода класса OpenGLRenderer**

**package** com.labs.lr7;  
  
**import** android.opengl.GLSurfaceView;  
**import** android.opengl.GLU;  
  
**import** javax.microedition.khronos.egl.EGLConfig;  
**import** javax.microedition.khronos.opengles.GL10;  
  
**class** OpenGLRenderer **implements** GLSurfaceView.Renderer {  
  
 **private final** Cube **cube** = **new** Cube();  
 **private float cubeRotation**;  
  
 @Override  
 **public void** onSurfaceCreated(GL10 gl, EGLConfig config) {  
 gl.glClearColor(0.0f, 0.0f, 0.0f, 0.5f);  
  
 gl.glClearDepthf(1.0f);  
 gl.glEnable(GL10.***GL\_DEPTH\_TEST***);  
 gl.glDepthFunc(GL10.***GL\_LEQUAL***);  
  
 gl.glHint(GL10.***GL\_PERSPECTIVE\_CORRECTION\_HINT***,  
 GL10.***GL\_NICEST***);  
 }  
  
 @Override  
 **public void** onDrawFrame(GL10 gl) {  
 gl.glClear(GL10.***GL\_COLOR\_BUFFER\_BIT*** | GL10.***GL\_DEPTH\_BUFFER\_BIT***);  
 gl.glLoadIdentity();  
  
 gl.glTranslatef(0.0f, 0.0f, -10.0f);  
 gl.glRotatef(**cubeRotation**, 1.0f, 1.0f, 1.0f);  
  
 **cube**.draw(gl);  
  
 gl.glLoadIdentity();  
  
 **cubeRotation** -= 0.15f;  
 }  
  
 @Override  
 **public void** onSurfaceChanged(GL10 gl, **int** width, **int** height) {  
 gl.glViewport(0, 0, width, height);  
 gl.glMatrixMode(GL10.***GL\_PROJECTION***);  
 gl.glLoadIdentity();  
 GLU.*gluPerspective*(gl, 45.0f, (**float**)width / (**float**)height, 0.1f, 100.0f);  
 gl.glViewport(0, 0, width, height);  
  
 gl.glMatrixMode(GL10.***GL\_MODELVIEW***);  
 gl.glLoadIdentity();  
 }  
}

**Приложение В. Листинг кода основного класса приложения**

**package** com.labs.lr7;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.opengl.GLSurfaceView;  
**import** android.os.Bundle;  
**import** android.view.Window;  
**import** android.view.WindowManager;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 @Override  
 **public void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
  
 *// Go fullscreen* requestWindowFeature(Window.***FEATURE\_NO\_TITLE***);  
 getWindow().setFlags(WindowManager.LayoutParams.***FLAG\_FULLSCREEN***,  
 WindowManager.LayoutParams.***FLAG\_FULLSCREEN***);  
  
 *// Create view for OpenGL with custom OpenGLRenderer class* GLSurfaceView view = **new** GLSurfaceView(**this**);  
 view.setRenderer(**new** OpenGLRenderer());  
  
 *// Start rendering* setContentView(view);  
 }  
}