Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

Кафедра ПМиК

Программирование мобильных устройств

Лабораторная работа №2

Примитивы OpenGL ES 1

Выполнил: студент 4 курса

Ф. ИВТ, группа: ИП-711

Мартасов И. О.

Проверил: доцент кафедры ПМиК

Нечта Иван Васильевич

Новосибирск, 2020

**Содержание**

1. **Постановка задачи..………………………………………………………3**
2. **Исходный код……..……………………………………………………….3**
3. **Результаты работы…..………………………………………………….10**

**Постановка задачи**

Необходимо создать классы прорисовки квадрата, куба, сферы

**Исходный код**

**OpenGLRendererSquare.java**

package com.example.lab1\_opengl;  
  
import android.opengl.GLSurfaceView;  
  
import java.nio.ByteBuffer;  
import java.nio.ByteOrder;  
import java.nio.FloatBuffer;  
  
import javax.microedition.khronos.egl.EGLConfig;  
import javax.microedition.khronos.opengles.GL10;  
  
public class OpenGLRendererSquare implements GLSurfaceView.Renderer  
{  
 float []vertices = new float[]  
 {  
 -0.5f, 0.25f, 0,  
 -0.5f, -0.25f, 0,  
 0.5f, -0.25f, 0,  
 0.5f, 0.25f, 0  
 };  
 FloatBuffer f;  
 public OpenGLRendererSquare()  
 {  
 ByteBuffer b = ByteBuffer.*allocateDirect*(4 \* 3 \* 4);  
 b.order(ByteOrder.*nativeOrder*());  
 f = b.asFloatBuffer();  
 f.put(vertices);  
 f.position(0);  
 }  
 @Override  
 public void onSurfaceCreated(GL10 gl, EGLConfig config)  
 {  
 }  
 @Override  
 public void onSurfaceChanged(GL10 gl, int width, int height)  
 {  
 }  
 @Override  
 public void onDrawFrame(GL10 gl)  
 {  
 gl.glClear(GL10.*GL\_COLOR\_BUFFER\_BIT*);  
 gl.glLoadIdentity();  
  
 gl.glTranslatef(0,0,-1);  
 gl.glColor4f(0,1,1,1);  
  
 gl.glEnableClientState(GL10.*GL\_VERTEX\_ARRAY*);  
 gl.glVertexPointer(3, GL10.*GL\_FLOAT*,0, f);  
 gl.glDrawArrays(GL10.*GL\_TRIANGLE\_FAN*,0,4);  
 gl.glDisableClientState(GL10.*GL\_VERTEX\_ARRAY*);  
 }  
}

**OpenGLRendererCube.java**

package com.example.lab1\_opengl;  
  
import android.opengl.GLSurfaceView;  
import android.opengl.GLU;  
  
import java.nio.ByteBuffer;  
import java.nio.ByteOrder;  
import java.nio.FloatBuffer;  
  
import javax.microedition.khronos.egl.EGLConfig;  
import javax.microedition.khronos.opengles.GL10;  
  
public class OpenGLRendererCube implements GLSurfaceView.Renderer  
{  
 private float mCubeRotation;  
  
 private FloatBuffer mVertexBuffer;  
 private FloatBuffer mColorBuffer;  
 private ByteBuffer mIndexBuffer;  
  
 private 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 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 void create\_a\_cube()  
 {  
 ByteBuffer byteBuf = ByteBuffer.*allocateDirect*(vertices.length \* 4);  
 byteBuf.order(ByteOrder.*nativeOrder*());  
 mVertexBuffer = byteBuf.asFloatBuffer();  
 mVertexBuffer.put(vertices);  
 mVertexBuffer.position(0);  
  
 byteBuf.order(ByteOrder.*nativeOrder*());  
 mColorBuffer = byteBuf.asFloatBuffer();  
 mColorBuffer.position(0);  
  
 mIndexBuffer = ByteBuffer.*allocateDirect*(indices.length);  
 mIndexBuffer.put(indices);  
 mIndexBuffer.position(0);  
 }  
  
 @Override  
 public void onSurfaceCreated(GL10 gl, EGLConfig config)  
 {  
 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*);  
  
 create\_a\_cube();  
 }  
  
 @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.glMatrixMode(GL10.*GL\_MODELVIEW*);  
 gl.glLoadIdentity();  
 }  
  
 @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(mCubeRotation, 0.01f, 1.0f, 0);  
  
 gl.glEnableClientState(GL10.*GL\_VERTEX\_ARRAY*);  
 gl.glEnableClientState(GL10.*GL\_COLOR\_ARRAY*);  
  
 gl.glVertexPointer(3, GL10.*GL\_FLOAT*, 0, mVertexBuffer);  
 gl.glColorPointer(4, GL10.*GL\_FLOAT*, 0, mColorBuffer);  
  
 gl.glDrawElements(GL10.*GL\_TRIANGLES*, 36, GL10.*GL\_UNSIGNED\_BYTE*,  
 mIndexBuffer);  
  
 gl.glDisableClientState(GL10.*GL\_VERTEX\_ARRAY*);  
 gl.glDisableClientState(GL10.*GL\_COLOR\_ARRAY*);  
  
 mCubeRotation -= 1.15f;  
 }  
}

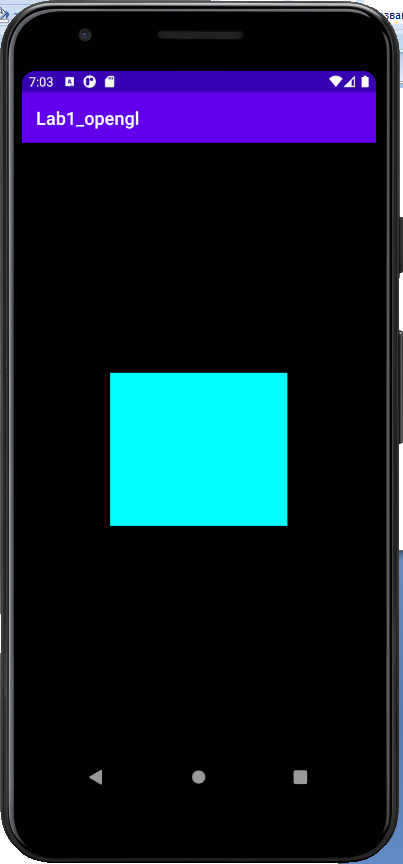
**OpenGLRendererSphere.java**

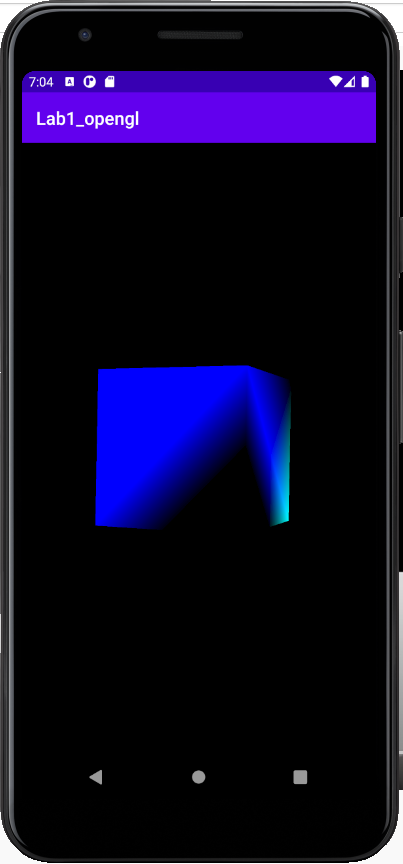
package com.example.lab1\_opengl;  
  
import android.opengl.GLSurfaceView;  
import android.opengl.GLU;  
  
import java.nio.ByteBuffer;  
import java.nio.ByteOrder;  
import java.nio.FloatBuffer;  
  
import javax.microedition.khronos.egl.EGLConfig;  
import javax.microedition.khronos.opengles.GL10;  
  
public class OpenGLRendererSphere implements GLSurfaceView.Renderer  
{  
 private float mSphereRotation;  
  
 private final float[] mat\_ambient =  
 {  
 0.9f, 0.3f, 0.4f, 1.0f  
 };  
 private FloatBuffer mat\_ambient\_buf;  
  
 private final float[] mat\_diffuse =  
 {  
 0.2f, 0.6f, 0.6f, 1.0f  
 };  
 private FloatBuffer mat\_diffuse\_buf;  
  
 private final float[] mat\_specular =  
 {  
 0.2f \* 0.4f, 0.2f \* 0.6f, 0.2f \* 0.8f, 1.0f  
 };  
 private FloatBuffer mat\_specular\_buf;  
  
 private void init\_sphere\_color\_buffers()  
 {  
 ByteBuffer b = ByteBuffer.*allocateDirect*(mat\_ambient.length \* 4);  
 b.order(ByteOrder.*nativeOrder*());  
 mat\_ambient\_buf = b.asFloatBuffer();  
 mat\_ambient\_buf.put(mat\_ambient);  
 mat\_ambient\_buf.position(0);  
  
 b = ByteBuffer.*allocateDirect*(mat\_diffuse.length \* 4);  
 b.order(ByteOrder.*nativeOrder*());  
 mat\_diffuse\_buf = b.asFloatBuffer();  
 mat\_diffuse\_buf.put(mat\_diffuse);  
 mat\_diffuse\_buf.position(0);  
  
 b = ByteBuffer.*allocateDirect*(mat\_specular.length \* 4);  
 b.order(ByteOrder.*nativeOrder*());  
 mat\_specular\_buf = b.asFloatBuffer();  
 mat\_specular\_buf.put(mat\_specular);  
 mat\_specular\_buf.position(0);  
 }  
  
 public void draw\_a\_sphere(GL10 gl)  
 {  
 float angleA, angleB;  
 float cos, sin;  
 float r1, r2;  
 float h1, h2;  
 float step = 30.0f;  
 float[][] v = new float[64][3];  
 ByteBuffer vbb;  
 FloatBuffer vBuf;  
  
 vbb = ByteBuffer.*allocateDirect*(v.length \* v[0].length \* 4);  
 vbb.order(ByteOrder.*nativeOrder*());  
 vBuf = vbb.asFloatBuffer();  
  
 gl.glEnableClientState(GL10.*GL\_VERTEX\_ARRAY*);  
 gl.glEnableClientState(GL10.*GL\_NORMAL\_ARRAY*);  
  
 for (angleA = -90.0f; angleA < 90.0f; angleA += step)  
 {  
 int n = 0;  
  
 r1 = (float)Math.*cos*(angleA \* Math.*PI* / 180.0);  
 r2 = (float)Math.*cos*((angleA + step) \* Math.*PI* / 180.0);  
 h1 = (float)Math.*sin*(angleA \* Math.*PI* / 180.0);  
 h2 = (float)Math.*sin*((angleA + step) \* Math.*PI* / 180.0);  
  
 for (angleB = 0.0f; angleB <= 360.0f; angleB += step)  
 {  
 cos = (float)Math.*cos*(angleB \* Math.*PI* / 180.0);  
 sin = -(float)Math.*sin*(angleB \* Math.*PI* / 180.0);  
  
 v[n][0] = (r2 \* cos);  
 v[n][1] = (h2);  
 v[n][2] = (r2 \* sin);  
 v[n + 1][0] = (r1 \* cos);  
 v[n + 1][1] = (h1);  
 v[n + 1][2] = (r1 \* sin);  
  
 vBuf.put(v[n]);  
 vBuf.put(v[n + 1]);  
  
 n += 2;  
  
 if(n > 63)  
 {  
 vBuf.position(0);  
  
 gl.glVertexPointer(3, GL10.*GL\_FLOAT*, 0, vBuf);  
 gl.glNormalPointer(GL10.*GL\_FLOAT*, 0, vBuf);  
 gl.glDrawArrays(GL10.*GL\_TRIANGLE\_STRIP*, 0, n);  
  
 n = 0;  
 angleB -= step;  
 }  
 }  
 vBuf.position(0);  
  
 gl.glVertexPointer(3, GL10.*GL\_FLOAT*, 0, vBuf);  
 gl.glNormalPointer(GL10.*GL\_FLOAT*, 0, vBuf);  
 gl.glDrawArrays(GL10.*GL\_TRIANGLE\_STRIP*, 0, n);  
 }  
  
 gl.glDisableClientState(GL10.*GL\_VERTEX\_ARRAY*);  
 gl.glDisableClientState(GL10.*GL\_NORMAL\_ARRAY*);  
 }  
  
 @Override  
 public void onSurfaceCreated(GL10 gl, EGLConfig config)  
 {  
 gl.glClearDepthf(1.0f);  
 gl.glEnable(GL10.*GL\_DEPTH\_TEST*);  
 gl.glDepthFunc(GL10.*GL\_LEQUAL*);  
  
 gl.glHint(GL10.*GL\_PERSPECTIVE\_CORRECTION\_HINT*, GL10.*GL\_FASTEST*);  
  
 init\_sphere\_color\_buffers();  
 }  
  
 @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, 65.0f, (float) width / height, 0.1f, 50.0f);  
  
 gl.glMatrixMode(GL10.*GL\_MODELVIEW*);  
 gl.glLoadIdentity();  
 }  
  
 @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, -3.0f);  
 gl.glRotatef(mSphereRotation, 0, 1.0f, 0);  
  
 gl.glEnable(GL10.*GL\_LIGHTING*);  
 gl.glEnable(GL10.*GL\_LIGHT0*);  
  
 gl.glMaterialfv(GL10.*GL\_FRONT\_AND\_BACK*, GL10.*GL\_AMBIENT*, mat\_ambient\_buf);  
 gl.glMaterialfv(GL10.*GL\_FRONT\_AND\_BACK*, GL10.*GL\_DIFFUSE*, mat\_diffuse\_buf);  
 gl.glMaterialfv(GL10.*GL\_FRONT\_AND\_BACK*, GL10.*GL\_SPECULAR*, mat\_specular\_buf);  
  
 draw\_a\_sphere(gl);  
  
 mSphereRotation -= 1.15f;  
 }  
}

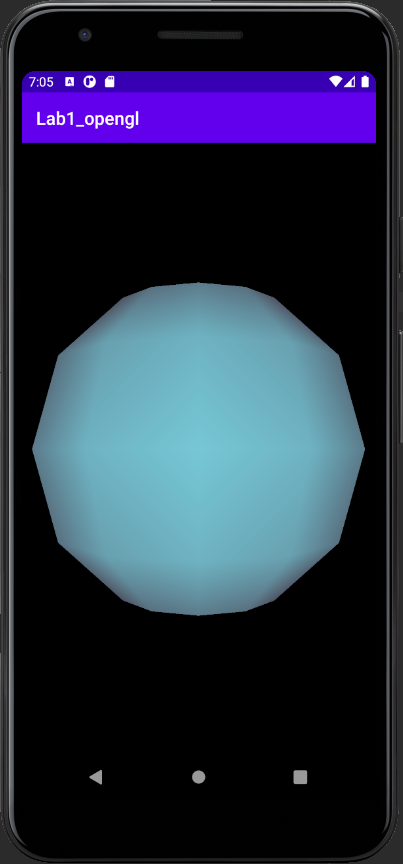
**MainActivity.java**

package com.example.lab1\_opengl;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.opengl.GLSurfaceView;  
import android.os.Bundle;  
  
public class MainActivity extends AppCompatActivity  
{  
 @Override  
 protected void onCreate(Bundle savedInstanceState)  
 {  
 super.onCreate(savedInstanceState);  
 GLSurfaceView g = new GLSurfaceView(this);  
  
 //g.setRenderer(new OpenGLRendererSquare());  
 //g.setRenderer(new OpenGLRendererCube());  
 g.setRenderer(new OpenGLRendererSphere());  
  
 g.setRenderMode(GLSurfaceView.*RENDERMODE\_CONTINUOUSLY*);  
 setContentView(g);  
 }  
}

**Результаты работы**

****

****

****