Федеральное агентство связи  
Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

**Лабораторная работа по теме:**

**«Примитивы OpenGL ES 1»**

Выполнили:

студентки 4 курса

ИВТ, гр. ИП-712

Гервас А.В.

Онищенко А.В.

Новосибирск 2020

Оглавление

[Задание 3](#_Toc51706096)

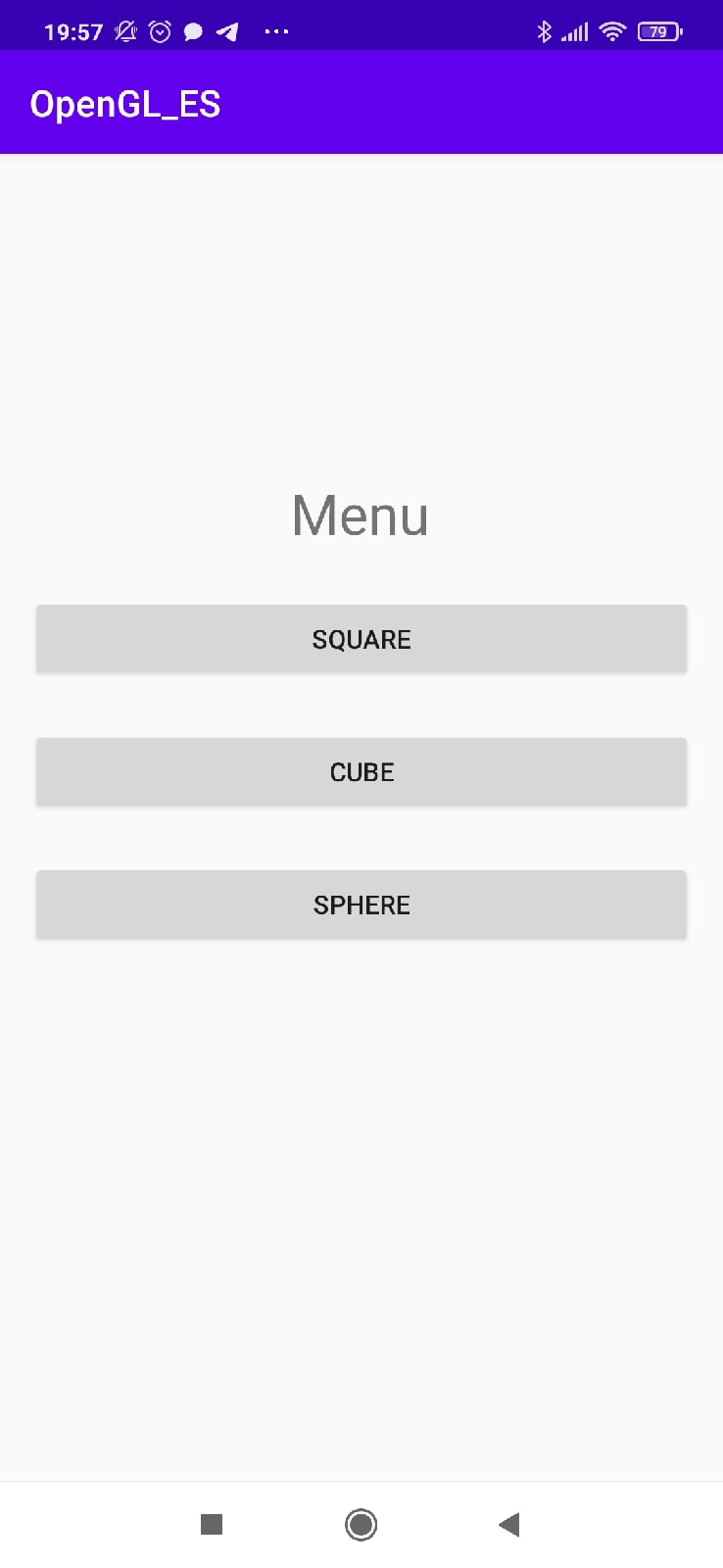
[Скриншоты 3](#_Toc51706097)

[Листинг кода 4](#_Toc51706098)

**Задание**

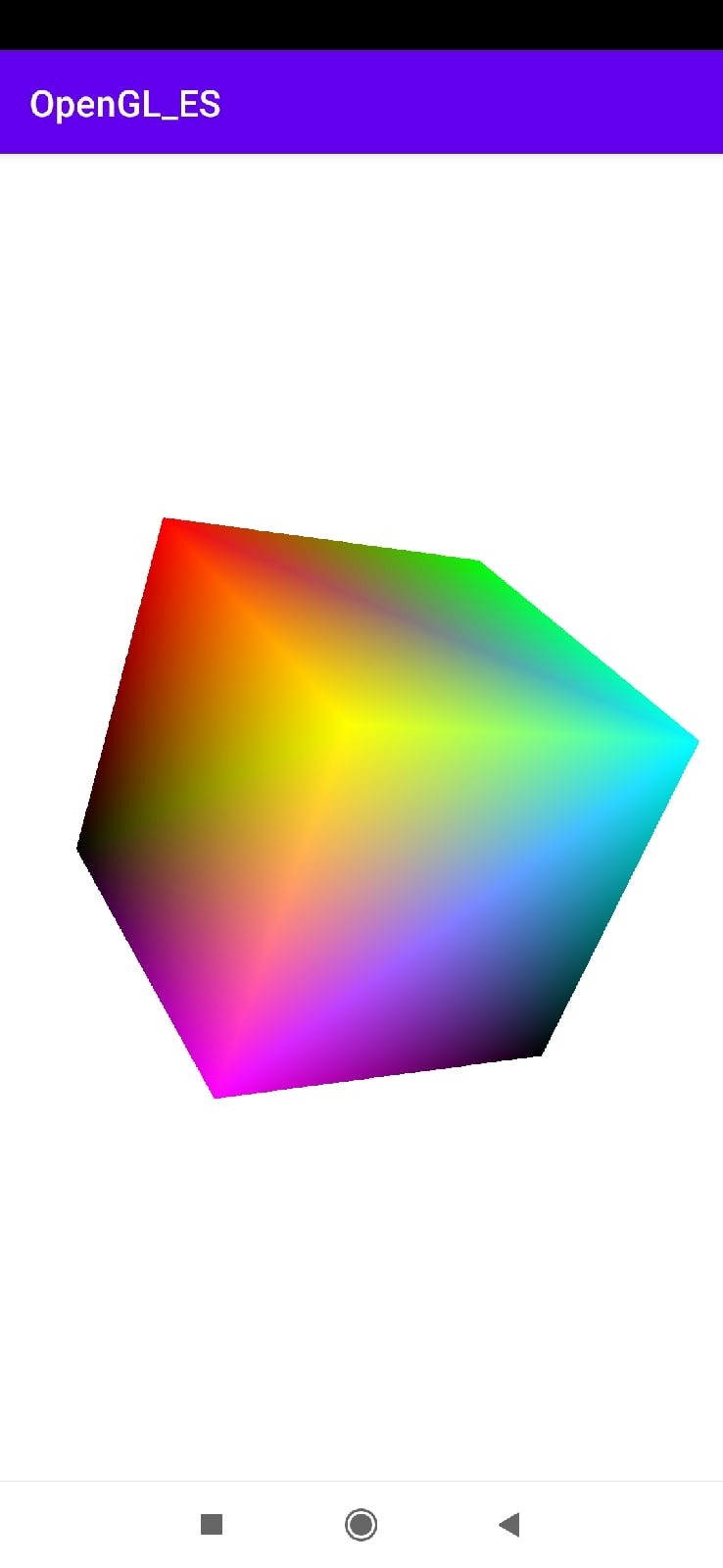
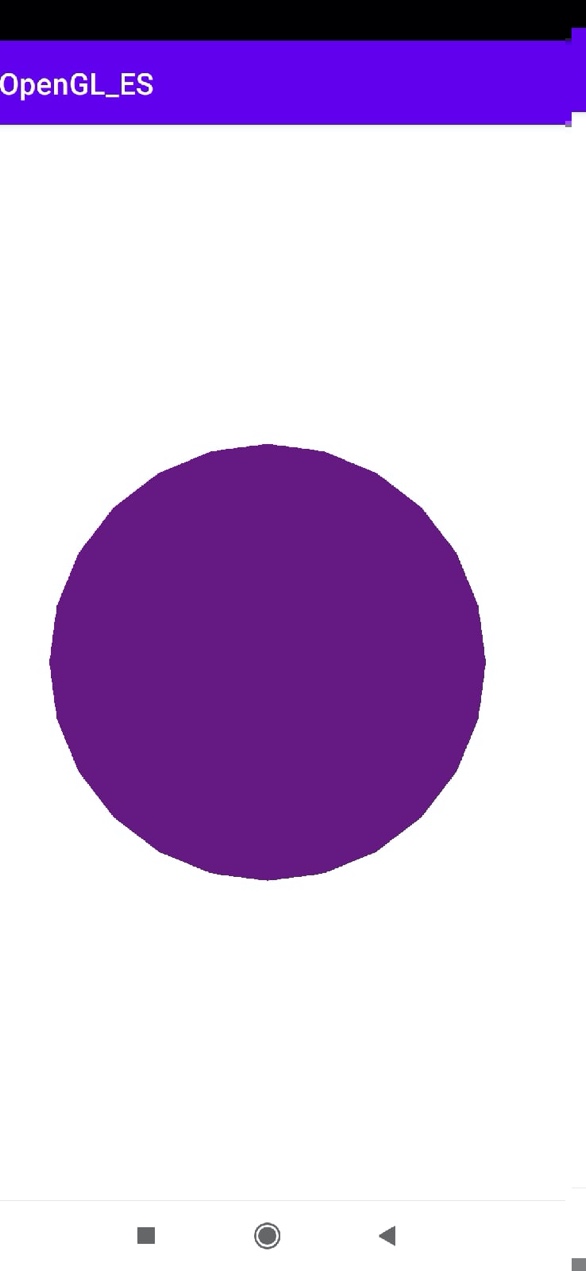
Необходимо создать классы прорисовки квадрата, куба, сферы.

**Скриншоты**



Прорисовка квадрата

Стартовая активность



Прорисовка сферы

Прорисовка куба

**Листинг кода**

Приложение написано на языке Java.

**MainActivity.java**

package ru.sibsutis.opengl\_es.activity;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import ru.sibsutis.opengl\_es.R;

public class MainActivity extends AppCompatActivity {

Button squareButton;

Button cubeButton;

Button sphereButton;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

squareButton = (Button) findViewById(R.id.squareButton);

cubeButton = (Button) findViewById(R.id.cubeButton);

sphereButton = (Button) findViewById(R.id.sphereButton);

squareButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent = new Intent(MainActivity.this, SquareActivity.class);

startActivity(intent);

}

});

cubeButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent = new Intent(MainActivity.this, CubeActivity.class);

startActivity(intent);

}

});

sphereButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent = new Intent(MainActivity.this, SphereActivity.class);

startActivity(intent);

}

});

}

}

**SquareActivity.java**

package ru.sibsutis.opengl\_es.activity;

import androidx.appcompat.app.AppCompatActivity;

import android.opengl.GLSurfaceView;

import android.os.Bundle;

import android.view.WindowManager;

import ru.sibsutis.opengl\_es.entity.Square;

public class SquareActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

getWindow().setFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN, WindowManager.LayoutParams.FLAG\_FULLSCREEN);

GLSurfaceView view = new GLSurfaceView(this);

view.setRenderer(new Square());

setContentView(view);

}

}

Square.java

package ru.sibsutis.opengl\_es.entity;

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;

import javax.microedition.khronos.opengles.GL11;

/\*\*

\* A vertex shaded square.

\*/

public class Square implements GLSurfaceView.Renderer {

private FloatBuffer mFVertexBuffer;

private ByteBuffer mColorBuffer;

private ByteBuffer mIndexBuffer;

private boolean mTranslucentBackground;

private float mTransY;

public Square() {

float[] vertices = {

-1.0f, -1.0f,

1.0f, -1.0f,

-1.0f, 1.0f,

1.0f, 1.0f

};

byte maxColor = (byte) 255;

byte[] colors = {

maxColor, maxColor, 0, maxColor,

0, maxColor, maxColor, maxColor,

0, 0, 0, maxColor,

maxColor, 0, maxColor, maxColor

};

byte[] indices = {

0, 3, 1,

0, 2, 3

};

ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length \* 4);

vbb.order(ByteOrder.nativeOrder());

mFVertexBuffer = vbb.asFloatBuffer();

mFVertexBuffer.put(vertices);

mFVertexBuffer.position(0);

mColorBuffer = ByteBuffer.allocateDirect(colors.length);

mColorBuffer.put(colors);

mColorBuffer.position(0);

mIndexBuffer = ByteBuffer.allocateDirect(indices.length);

mIndexBuffer.put(indices);

mIndexBuffer.position(0);

}

public void draw(GL10 gl) {

gl.glFrontFace(GL11.GL\_CW);

gl.glVertexPointer(2, GL11.GL\_FLOAT, 0, mFVertexBuffer);

gl.glColorPointer(4, GL11.GL\_UNSIGNED\_BYTE, 0, mColorBuffer);

gl.glDrawElements(GL11.GL\_TRIANGLES, 6, GL11.GL\_UNSIGNED\_BYTE, mIndexBuffer);

gl.glFrontFace(GL11.GL\_CCW);

}

@Override

public void onSurfaceCreated(GL10 gl, EGLConfig config) {

gl.glDisable(GL10.GL\_DITHER);

gl.glHint(GL10.GL\_PERSPECTIVE\_CORRECTION\_HINT, GL10.GL\_FASTEST);

if (mTranslucentBackground) {

gl.glClearColor(0, 0, 0, 0);

} else {

gl.glClearColor(1, 1, 1, 1);

gl.glEnable(GL10.GL\_CULL\_FACE);

gl.glShadeModel(GL10.GL\_SMOOTH);

gl.glEnable(GL10.GL\_DEPTH\_TEST);

}

}

@Override

public void onSurfaceChanged(GL10 gl, int width, int height) {

gl.glViewport(0, 0, width, height);

float ratio = (float) width / height;

gl.glMatrixMode(GL10.GL\_PROJECTION);

gl.glLoadIdentity();

gl.glFrustumf(-ratio, ratio, -1, 1, 1, 10);

}

@Override

public void onDrawFrame(GL10 gl) {

gl.glClear(GL10.GL\_COLOR\_BUFFER\_BIT | GL10.GL\_DEPTH\_BUFFER\_BIT);

gl.glMatrixMode(GL10.GL\_MODELVIEW);

gl.glLoadIdentity();

gl.glTranslatef(0.0f, (float) Math.sin(mTransY), -3.0f);

gl.glEnableClientState(GL10.GL\_VERTEX\_ARRAY);

gl.glEnableClientState(GL10.GL\_COLOR\_ARRAY);

draw(gl);

mTransY += .075f;

}

}

**CubeActivity.java**

package ru.sibsutis.opengl\_es.activity;

import androidx.appcompat.app.AppCompatActivity;

import android.opengl.GLSurfaceView;

import android.os.Bundle;

import android.view.WindowManager;

import ru.sibsutis.opengl\_es.R;

import ru.sibsutis.opengl\_es.entity.Cube;

import ru.sibsutis.opengl\_es.entity.Square;

public class CubeActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

getWindow().setFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN, WindowManager.LayoutParams.FLAG\_FULLSCREEN);

GLSurfaceView view = new GLSurfaceView(this);

view.setRenderer(new Cube());

setContentView(view);

}

}

**Cube.java**

package ru.sibsutis.opengl\_es.entity;

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;

import javax.microedition.khronos.opengles.GL11;

public class Cube implements GLSurfaceView.Renderer {

private FloatBuffer mFVertexBuffer;

private ByteBuffer mColorBuffer;

private ByteBuffer mTfan1;

private ByteBuffer mTfan2;

private boolean mTranslucentBackground;

private Cube mCube;

private float mTransY;

private float mAngle;

public Cube() {

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

};

byte maxColor = (byte) 255;

byte[] colors = {

maxColor, maxColor, 0, maxColor,

0, maxColor, maxColor, maxColor,

0, 0, 0, maxColor,

maxColor, 0, maxColor, maxColor,

maxColor, 0, 0, maxColor,

0, maxColor, 0, maxColor,

0, 0, maxColor, maxColor,

0, 0, 0, maxColor

};

byte[] tfan1 = {

1, 0, 3,

1, 3, 2,

1, 2, 6,

1, 6, 5,

1, 5, 4,

1, 4, 0

};

byte[] tfan2 = {

7, 4, 5,

7, 5, 6,

7, 6, 2,

7, 2, 3,

7, 3, 0,

7, 0, 4

};

ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length \* 4);

vbb.order(ByteOrder.nativeOrder());

mFVertexBuffer = vbb.asFloatBuffer();

mFVertexBuffer.put(vertices);

mFVertexBuffer.position(0);

mColorBuffer = ByteBuffer.allocateDirect(colors.length);

mColorBuffer.put(colors);

mColorBuffer.position(0);

mTfan1 = ByteBuffer.allocateDirect(tfan1.length);

mTfan1.put(tfan1);

mTfan1.position(0);

mTfan2 = ByteBuffer.allocateDirect(tfan2.length);

mTfan2.put(tfan2);

mTfan2.position(0);

}

public void draw(GL10 gl) {

gl.glVertexPointer(3, GL11.GL\_FLOAT, 0, mFVertexBuffer);

gl.glColorPointer(4, GL11.GL\_UNSIGNED\_BYTE, 0, mColorBuffer);

gl.glDrawElements(gl.GL\_TRIANGLE\_FAN, 6 \* 3, gl.GL\_UNSIGNED\_BYTE, mTfan1);

gl.glDrawElements(gl.GL\_TRIANGLE\_FAN, 6 \* 3, gl.GL\_UNSIGNED\_BYTE, mTfan2);

}

@Override

public void onSurfaceCreated(GL10 gl, EGLConfig config) {

gl.glDisable(GL11.GL\_DITHER);

gl.glHint(GL11.GL\_PERSPECTIVE\_CORRECTION\_HINT, GL11.GL\_FASTEST);

if (mTranslucentBackground) {

gl.glClearColor(1, 0, 0, 0);

} else {

gl.glClearColor(1, 1, 1, 1);

}

gl.glEnable(GL11.GL\_CULL\_FACE);

gl.glShadeModel(GL11.GL\_SMOOTH);

gl.glEnable(GL11.GL\_DEPTH\_TEST);

}

@Override

public void onSurfaceChanged(GL10 gl, int width, int height) {

gl.glViewport(0, 0, width, height);

float aspectRatio;

float zNear = .1f;

float zFar = 1000;

float fieldOfView = 30.0f / 57.3f;

float size;

gl.glEnable(GL10.GL\_NORMALIZE);

aspectRatio = (float) width / (float) height;

gl.glMatrixMode(GL10.GL\_PROJECTION);

size = zNear \* (float) (Math.tan((double) (fieldOfView / 2.0f)));

gl.glFrustumf(-size, size, -size / aspectRatio,

size / aspectRatio, zNear, zFar);

gl.glMatrixMode(GL10.GL\_MODELVIEW);

}

@Override

public void onDrawFrame(GL10 gl) {

gl.glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

gl.glClear(GL11.GL\_COLOR\_BUFFER\_BIT | GL11.GL\_DEPTH\_BUFFER\_BIT);

gl.glMatrixMode(GL11.GL\_MODELVIEW);

gl.glLoadIdentity();

gl.glTranslatef(0.0f, (float) Math.sin(mTransY), -7.0f);

gl.glRotatef(mAngle, 0.0f, 1.0f, 0.0f);

gl.glRotatef(mAngle, 1.0f, 0.0f, 0.0f);

gl.glEnableClientState(GL11.GL\_VERTEX\_ARRAY);

gl.glEnableClientState(GL11.GL\_COLOR\_ARRAY);

draw(gl);

mTransY += .075f;

mAngle += .4;

}

}

s

**SphereActivity.java**

package ru.sibsutis.opengl\_es.activity;

import androidx.appcompat.app.AppCompatActivity;

import android.opengl.GLSurfaceView;

import android.os.Bundle;

import android.view.WindowManager;

import ru.sibsutis.opengl\_es.entity.Sphere;

public class SphereActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

getWindow().setFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN, WindowManager.LayoutParams.FLAG\_FULLSCREEN);

GLSurfaceView view = new GLSurfaceView(this);

view.setRenderer(new Sphere(0.5f));

setContentView(view);

}

}

**Sphere.java**

package ru.sibsutis.opengl\_es.entity;

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 Sphere implements GLSurfaceView.Renderer {

public FloatBuffer mVertexBuffer;

private ByteBuffer mColorBuffer;

private boolean mTranslucentBackground;

public int n = 0, sz = 0;

public Sphere(float R) {

float i = 0;

int dtheta = 15, dphi = 15;

int theta, phi;

float DTOR = (float) (Math.PI / 180.0f);

ByteBuffer byteBuf = ByteBuffer.allocateDirect(5000 \* 3 \* 4);

byteBuf.order(ByteOrder.nativeOrder());

mVertexBuffer = byteBuf.asFloatBuffer();

byteBuf = ByteBuffer.allocateDirect(5000 \* 2 \* 4);

byteBuf.order(ByteOrder.nativeOrder());

for (theta = -90; theta <= 90 - dtheta; theta += dtheta) {

for (phi = 0; phi <= 360 - dphi; phi += dphi) {

sz++;

mVertexBuffer.put((float) (Math.cos(theta \* DTOR) \* Math.cos(phi \* DTOR)) \* R);

mVertexBuffer.put((float) (Math.cos(theta \* DTOR) \* Math.sin(phi \* DTOR)) \* R);

mVertexBuffer.put((float) (Math.sin(theta \* DTOR)) \* R);

mVertexBuffer.put((float) (Math.cos((theta + dtheta) \* DTOR) \* Math.cos(phi \* DTOR)) \* R);

mVertexBuffer.put((float) (Math.cos((theta + dtheta) \* DTOR) \* Math.sin(phi \* DTOR)) \* R);

mVertexBuffer.put((float) (Math.sin((theta + dtheta) \* DTOR)) \* R);

mVertexBuffer.put((float) (Math.cos((theta + dtheta) \* DTOR) \* Math.cos((phi + dphi) \* DTOR)) \* R);

mVertexBuffer.put((float) (Math.cos((theta + dtheta) \* DTOR) \* Math.sin((phi + dphi) \* DTOR)) \* R);

mVertexBuffer.put((float) (Math.sin((theta + dtheta) \* DTOR)) \* R);

n += 3;

mVertexBuffer.put((float) (Math.cos(theta \* DTOR) \* Math.cos((phi + dphi) \* DTOR)) \* R);

mVertexBuffer.put((float) (Math.cos(theta \* DTOR) \* Math.sin((phi + dphi) \* DTOR)) \* R);

mVertexBuffer.put((float) (Math.sin(theta \* DTOR)) \* R);

n++;

}

}

mVertexBuffer.position(0);

}

public void draw(GL10 gl) {

gl.glFrontFace(GL10.GL\_CCW); // Front face in counter-clockwise orientation

gl.glEnable(GL10.GL\_CULL\_FACE); // Enable cull face

gl.glCullFace(GL10.GL\_BACK); // Cull the back face (don't display)

gl.glEnable(GL10.GL\_BLEND);

gl.glBlendFunc(GL10.GL\_SRC\_ALPHA, GL10.GL\_ONE\_MINUS\_SRC\_ALPHA);

gl.glEnableClientState(GL10.GL\_VERTEX\_ARRAY);

gl.glVertexPointer(3, GL10.GL\_FLOAT, 0, mVertexBuffer);

int i = 0;

for (i = 0; i < n; i += 4) {

gl.glColor4f(0.4f, 0.1f, 0.5f, 1.0f);

gl.glDrawArrays(GL10.GL\_TRIANGLE\_FAN, i, 4);

}

gl.glColor4f(0.4f, 0.1f, 0.5f, 1.0f);

}

@Override

public void onSurfaceCreated(GL10 gl, EGLConfig config) {

gl.glDisable(GL10.GL\_DITHER);

gl.glHint(GL10.GL\_PERSPECTIVE\_CORRECTION\_HINT, GL10.GL\_FASTEST);

if (mTranslucentBackground) {

gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

} else {

gl.glClearColor(1, 1, 1, 1);

gl.glEnable(GL10.GL\_CULL\_FACE);

gl.glShadeModel(GL10.GL\_SMOOTH);

gl.glEnable(GL10.GL\_DEPTH\_TEST);

}

gl.glEnableClientState(GL10.GL\_VERTEX\_ARRAY);

}

@Override

public void onSurfaceChanged(GL10 gl, int width, int height) {

gl.glViewport(0, 0, width, height);

gl.glMatrixMode(GL10.GL\_PROJECTION);

gl.glLoadIdentity();

float ratio = (float) width / height;

GLU.gluPerspective(gl, 45.0f, ratio, 1f, 100f);

}

@Override

public void onDrawFrame(GL10 gl) {

gl.glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

gl.glClear(GL10.GL\_COLOR\_BUFFER\_BIT | GL10.GL\_DEPTH\_BUFFER\_BIT);

gl.glMatrixMode(GL10.GL\_MODELVIEW);

gl.glLoadIdentity();

gl.glTranslatef(0f, 0f, -3.0f);

draw(gl);

}

}