Федеральное агентство связи

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

**Лабораторная работа по теме:**

**«Матрицы модели-вида OpenGL ES1»**

Выполнил:

Студент 4 курса

ИВТ, гр. ИП-712

Алексеев С.В.

Новосибирск 2020 г.

Оглавление

[Задание 2](#_Toc58687998)

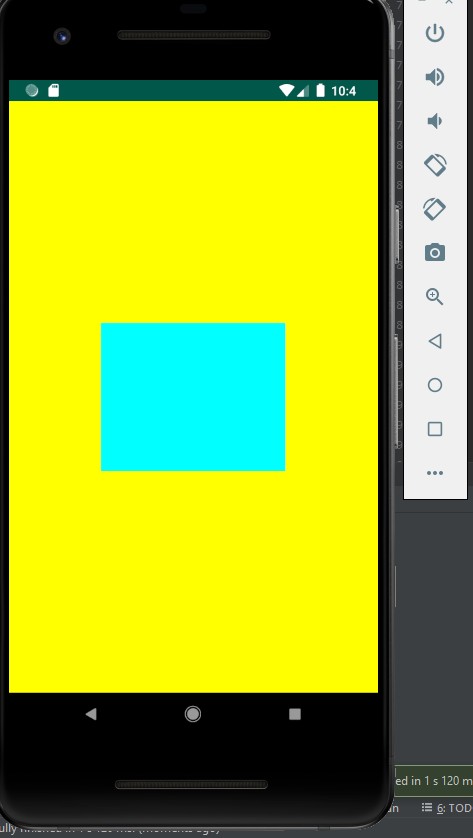
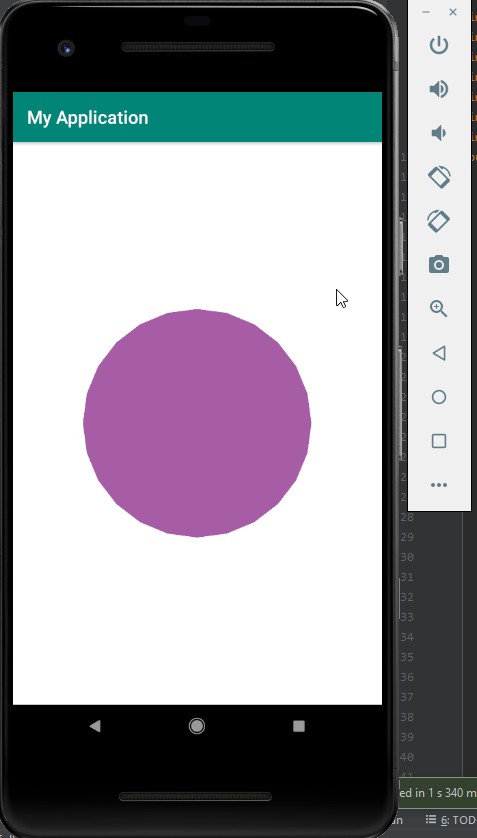
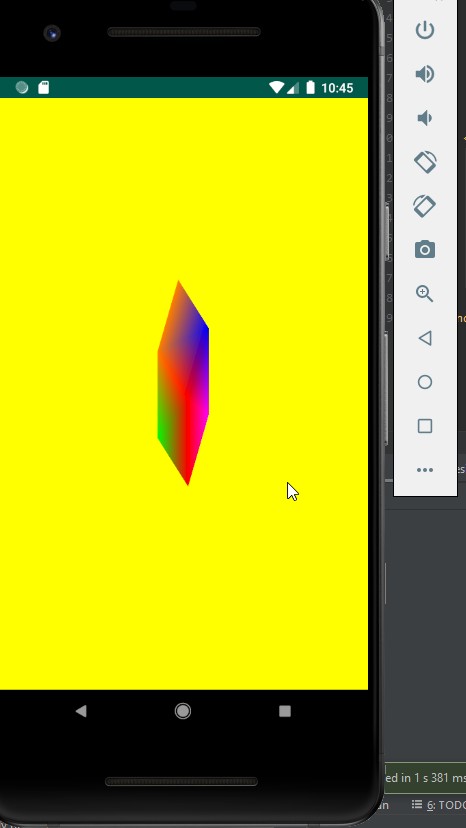
[Скриншоты 3](#_Toc58687999)

[Листинг 6](#_Toc58688000)

# Задание

Лабораторная работа 2 примитивы OpenGL ES 1. Необходимо создать классы прорисовки квадрата, куба, сферы

# Скриншоты



# Листинг

Квадрат, куб

MainActivity.java

package com.example.pmylab1;

import android.app.Activity;

import android.opengl.GLSurfaceView;

import android.os.Bundle;

import android.view.WindowManager;

public class MainActivity extends Activity {

@Override

public void onCreate(Bundle savedInstanceState) {

getWindow().addFlags(WindowManager.LayoutParams.FLAG\_KEEP\_SCREEN\_ON);

super.onCreate(savedInstanceState);

GLSurfaceView g = new GLSurfaceView(this);

g.setEGLConfigChooser(8, 8, 8, 8, 16, 1);

g.setRenderer(new MyRenderer());

g.setRenderMode(GLSurfaceView.RENDERMODE\_CONTINUOUSLY);

setContentView(g);

}

}

// public void onCreate(Bundle savedInstanceState) {

// super.onCreate(savedInstanceState);

// getWindow().addFlags(WindowManager.LayoutParams.FLAG\_KEEP\_SCREEN\_ON);

// GLSurfaceView g=new GLSurfaceView(this);

// g.setRenderer(new com.example.pmylab1.MyKube());

// g.setRenderMode(GLSurfaceView.RENDERMODE\_CONTINUOUSLY);

// setContentView(g);

// }

//}

MyKube.java

package com.example.pmylab1;

import java.nio.ByteBuffer;

import java.nio.ByteOrder;

import java.nio.FloatBuffer;

import javax.microedition.khronos.opengles.GL10;

public class MyKube {

private FloatBuffer mVertexBuffer;

private FloatBuffer mColorBuffer;

private ByteBuffer mIndexBuffer;

private 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 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, 1.0f, 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 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 MyKube() {

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

byteBuf.order(ByteOrder.nativeOrder());

mVertexBuffer = byteBuf.asFloatBuffer();

mVertexBuffer.put(vertices);

mVertexBuffer.position(0);

byteBuf = ByteBuffer.allocateDirect(colors.length \* 4);

byteBuf.order(ByteOrder.nativeOrder());

mColorBuffer = byteBuf.asFloatBuffer();

mColorBuffer.put(colors);

mColorBuffer.position(0);

mIndexBuffer = ByteBuffer.allocateDirect(indices.length);

mIndexBuffer.put(indices);

mIndexBuffer.position(0);

}

public void draw(GL10 gl) {

//gl.glFrontFace(GL10.GL\_CW);

// gl.glEnable(GL10.GL\_BACK);

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

gl.glColorPointer(4, GL10.GL\_FLOAT, 0, mColorBuffer);

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

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

gl.glDrawElements(GL10.GL\_TRIANGLES, 36, GL10.GL\_UNSIGNED\_BYTE,

mIndexBuffer);

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

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

// gl.glDisable(GL10.GL\_CULL\_FACE);

}

}

//

//import java.io.InputStream;

//import java.nio.ByteBuffer;

//import java.nio.ByteOrder;

//import java.nio.FloatBuffer;

//

//import javax.microedition.khronos.opengles.GL10;

//

//import android.content.Context;

//import android.graphics.Bitmap;

//import android.graphics.BitmapFactory;

//import android.graphics.Point;

//import android.opengl.GLUtils;

//

//public class MyKube {

// private FloatBuffer mVertexBuffer;

//

// private FloatBuffer textureBuffer;

//

//

// int n=0;

//

// public MyKube() {

// 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());

// textureBuffer=byteBuf.asFloatBuffer();

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

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

//

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

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

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

//

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

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

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

//

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

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

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

// n+=3;

//

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

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

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

// n++;

// textureBuffer.put((float)(phi/360.0f));

// textureBuffer.put((float)((90+theta)/180.0f));

//

// textureBuffer.put((float)(phi/360.0f));

// textureBuffer.put((float)((90+theta+dtheta)/180.0f));

//

// textureBuffer.put((float)((phi+dphi)/360.0f));

// textureBuffer.put((float)((90+theta+dtheta)/180.0f));

//

// textureBuffer.put((float)((phi+dphi)/360.0f));

// textureBuffer.put((float)((90+theta)/180.0f));

// }

// }

//

//

// mVertexBuffer.position(0);

// textureBuffer.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.glTexEnvf(GL10.GL\_TEXTURE\_ENV, GL10.GL\_TEXTURE\_ENV\_MODE, GL10.GL\_BLEND);

//

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

// gl.glEnableClientState(GL10.GL\_TEXTURE\_COORD\_ARRAY); // Enable texture-coords-array (NEW)

// gl.glEnable(GL10.GL\_TEXTURE\_2D);

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

// gl.glTexCoordPointer(2, GL10.GL\_FLOAT, 0, textureBuffer);

// //gl.glColor4f(1,0,0,0.5f);

// int i=0;

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

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

// }

// gl.glDisableClientState(GL10.GL\_VERTEX\_ARRAY);

// gl.glDisableClientState(GL10.GL\_TEXTURE\_COORD\_ARRAY);

// gl.glDisable(GL10.GL\_TEXTURE\_2D);

// gl.glDisable(GL10.GL\_BLEND);

// //gl.glColor4f(1,1,1,1);

// }

//

//

//}

//

MyRenderer.java

package com.example.pmylab1;

//КВАДРАТ

import android.opengl.GLSurfaceView;

import javax.microedition.khronos.egl.EGLConfig;

import javax.microedition.khronos.opengles.GL10;

import java.nio.ByteBuffer;

import java.nio.ByteOrder;

import java.nio.FloatBuffer;

public class MyRenderer implements GLSurfaceView.Renderer {

float []a=new float[]{

-1,1,0,

-1,-1,0,

1,-1,0,

1,1,0

};

FloatBuffer f;

ByteBuffer b;

public MyRenderer(){

b=ByteBuffer.allocateDirect(4\*3\*4);

b.order(ByteOrder.nativeOrder());

f=b.asFloatBuffer();

f.put(a);

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

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

gl.glLoadIdentity();

gl.glTranslatef(0,0,-1);//сдвиг начала системы координат.

gl.glScalef(0.5f,0.25f,0.5f);

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);

}

}

//КУБ

//import android.opengl.GLES10;

//import android.opengl.GLSurfaceView;

//import android.opengl.GLU;

//

//

//import javax.microedition.khronos.egl.EGLConfig;

//import javax.microedition.khronos.opengles.GL10;

//public class MyRenderer implements GLSurfaceView.Renderer {

// MyKube cube;

// float p;

//

// public MyRenderer(){

// cube=new MyKube();

//

// }

//

//

// @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.glClearColor(1,1,0,0);

//

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

// gl.glMatrixMode(GL10.GL\_MODELVIEW);

// gl.glLoadIdentity();

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

// gl.glOrthof(-10,10, -5,5,5,-5);

// GLU.gluLookAt(gl,0,1,-2,0,2,-1 ,0,1,0);

//

//

// gl.glTranslatef(0,3,0);

// gl.glPushMatrix();

// gl.glRotatef(p,0,1,0);

// cube.draw(gl);

// gl.glPopMatrix();

//

//

// gl.glEnable(GL10.GL\_STENCIL\_TEST);

// gl.glStencilFunc(GL10.GL\_ALWAYS, 1, 0xFF); // Set any stencil to 1

// gl.glStencilOp(GL10.GL\_ZERO, GL10.GL\_ZERO, GL10.GL\_REPLACE);

// gl.glStencilMask(0xff); // Write to stencil buffer

// gl.glDepthMask(false); // Don't write to depth buffer

// gl.glClearStencil(0);

// gl.glClear(GL10.GL\_STENCIL\_BUFFER\_BIT); // Clear stencil buffer (0 by default)

//

//

// gl.glEnable(GL10.GL\_STENCIL\_TEST);

// gl.glStencilFunc(GL10.GL\_EQUAL, 1, 0xFF);

// gl.glStencilMask(0x0);

//

// gl.glColorMask(true,true,true,true);

// gl.glDepthMask(true);

//

//

// gl.glPushMatrix();

// gl.glTranslatef(0,-2,0);

// gl.glRotatef(p,0,1,0);

// cube.draw(gl);

// gl.glPopMatrix();

// p=(p>360)?0:p+1f;

//

//

// gl.glDisable(GL10.GL\_DEPTH\_TEST);

// gl.glDisable(GL10.GL\_STENCIL\_TEST);

// }

//}

Сфера

MainActivity.java

package com.example.sfera;

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 MainActivity 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 com.example.sfera;

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(0.0f, 0.1f, 0.5f, 1.0f);

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

}

gl.glColor4f(0.0f, 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);

}

}

//import java.io.InputStream;

//import java.nio.ByteBuffer;

//import java.nio.ByteOrder;

//import java.nio.FloatBuffer;

//

//import javax.microedition.khronos.opengles.GL10;

//

//import android.content.Context;

//import android.graphics.Bitmap;

//import android.graphics.BitmapFactory;

//import android.graphics.Point;

//import android.opengl.GLUtils;

//

//public class Sphere {

// private FloatBuffer mVertexBuffer;

//

// private FloatBuffer textureBuffer;

//

//

// int n=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());

// textureBuffer=byteBuf.asFloatBuffer();

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

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

//

// 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++;

// textureBuffer.put((float)(phi/360.0f));

// textureBuffer.put((float)((90+theta)/180.0f));

//

// textureBuffer.put((float)(phi/360.0f));

// textureBuffer.put((float)((90+theta+dtheta)/180.0f));

//

// textureBuffer.put((float)((phi+dphi)/360.0f));

// textureBuffer.put((float)((90+theta+dtheta)/180.0f));

//

// textureBuffer.put((float)((phi+dphi)/360.0f));

// textureBuffer.put((float)((90+theta)/180.0f));

// }

// }

//

//

// mVertexBuffer.position(0);

// textureBuffer.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.glTexEnvf(GL10.GL\_TEXTURE\_ENV, GL10.GL\_TEXTURE\_ENV\_MODE, GL10.GL\_BLEND);

//

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

// gl.glEnableClientState(GL10.GL\_TEXTURE\_COORD\_ARRAY); // Enable texture-coords-array (NEW)

// gl.glEnable(GL10.GL\_TEXTURE\_2D);

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

// gl.glTexCoordPointer(2, GL10.GL\_FLOAT, 0, textureBuffer);

// //gl.glColor4f(1,0,0,0.5f);

// int i=0;

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

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

// }

// gl.glDisableClientState(GL10.GL\_VERTEX\_ARRAY);

// gl.glDisableClientState(GL10.GL\_TEXTURE\_COORD\_ARRAY);

// gl.glDisable(GL10.GL\_TEXTURE\_2D);

// gl.glDisable(GL10.GL\_BLEND);

// //gl.glColor4f(1,1,1,1);

// }

//

//

//}