Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

Курсовая работа

Выполнил: студент 4 курса ИВТ,

гр. ИП-713

Михеев Н.А.

Проверил: ассистент кафедры ПМиК,

Павлова У.В.

Новосибирск, 2020 г.

Оглавление

1. Задание 3

2. Решение поставленной задачи 3

3. Демонстрация работы программы 3

4. Листинг программы 5

MainActivity: 5

MyGLRenderer: 5

Cup.java: 11

Sphere.java: 12

1. Задание

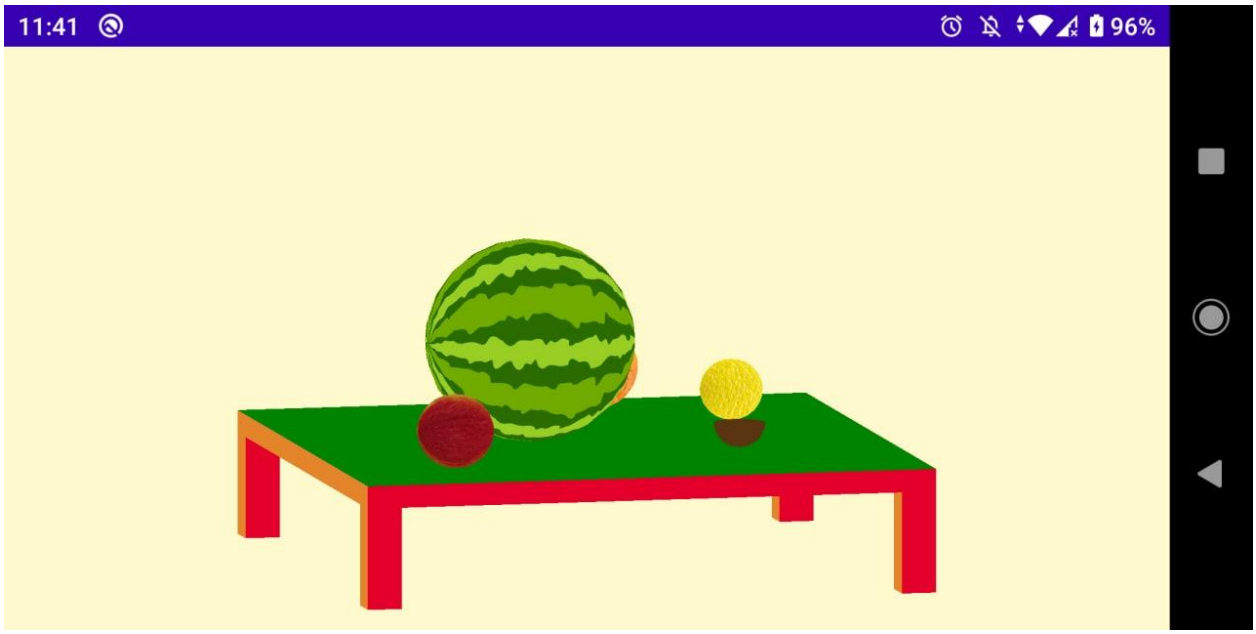
Создайте программу, в которой нарисован стол на OpenGL for Embeded Systems. На столе лежат различные фрукты/овощи (не менее 4 различных), стакан с напитком.

2. Решение поставленной задачи

Для реализации задания на курсовую работу были разработаны классы примитивов: Cube.java – для куба, Cup.java – для чаши, Sphere.java – для сферы. Во всех классах примитивов реализованы методы просчета вершинных буферов для отрисовки, текстурных буферов для текстурирования моделей и методы отрисовки onDrawFrame. Далее был реализован основной класс программы, занимающийся рендером 3D-сцены – MyGLRenderer.java – в нем реализованы методы:

- void loadGLTexture – метод, отвечающий за предзагрузку всех текстур из папки res в массив текстур textures, который позже используется для текстурирования;
- void onSurfaceCreated – метод, отвечающий за положение камеры на нашей 3D сцене;
- void onDrawFrame – основной метод, в котором происходит отрисовка моделей, линковка текстур с помощью glBindTexture(), установка положения фруктов, задание анимации.

3. Демонстрация работы программы



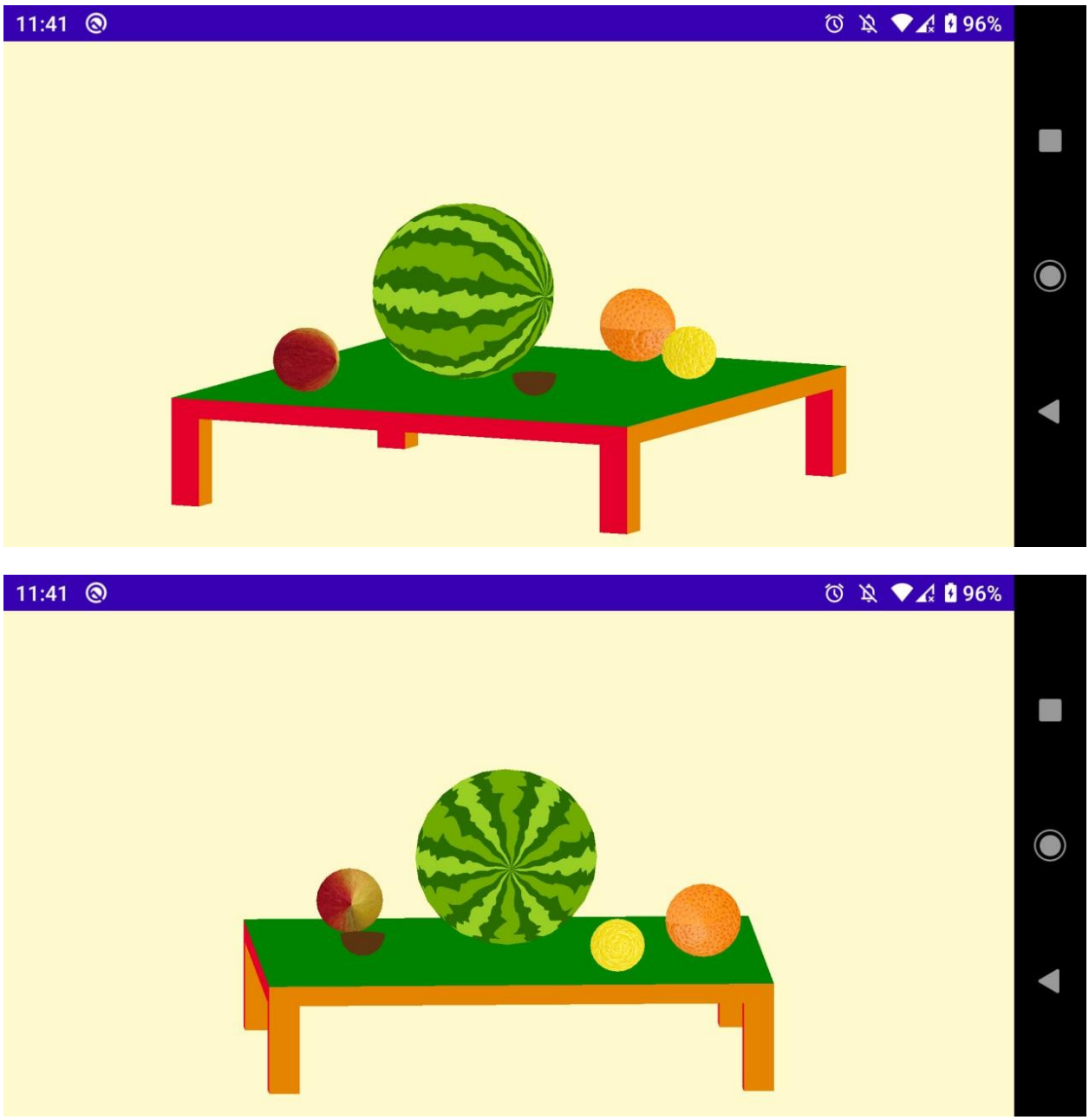


Рис. 1-3 – демонстрация работы программы.

4. Листинг программы

MainActivity:

```
package ru.mikheev_ustenko.courc;

import android.app.Activity;
import android.opengl.GLSurfaceView;
import android.os.Bundle;

public class MainActivity extends Activity {

    private GLSurfaceView gLView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        gLView = new GLSurfaceView(this);
        gLView.setRenderer(new MyGLRenderer(this));
        gLView.setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);
        setContentView(gLView);
    }

    @Override
    protected void onPause() {
        super.onPause();
        gLView.onPause();
    }

    @Override
    protected void onResume() {
        super.onResume();
        gLView.onResume();
    }
}
```

MyGLRenderer:

```
package ru.mikheev_ustenko.courc;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.opengl.GLSurfaceView;
import android.opengl.GLU;
import android.opengl.GLUtils;
import java.io.InputStream;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;

class MyGLRenderer implements GLSurfaceView.Renderer {
```

```
static public int[] texture_name = {
        R.drawable.watermelon,
        R.drawable.apple,
        R.drawable.lemon,
        R.drawable.orange
};

private float x0 = 0f;
private float z0 = 0f;
private float a = 0;
private float angle = 0;
private float speed = 0.1f;
static public int[] textures = new int [texture_name.length];
Context c;
private final Sphere watermelon = new Sphere(3f);
private final Cup cup = new Cup(1f, 0.4f, 0.4f, 0.4f);
private final Cup water = new Cup(1f, 0.1f, 0.3f, 0.7f);
private final Sphere apple = new Sphere(1.5f);
private final Sphere lemon = new Sphere(1.5f);
private final Sphere orange = new Sphere(1.5f);
private final Cube cube = new Cube(new float[]{
        // FRONT
        -1.0f, -1.0f, 1.0f,  // 0. left-bottom-front
        1.0f, -1.0f, 1.0f,  // 1. right-bottom-front
        -1.0f, 1.0f, 1.0f,  // 2. left-top-front
        1.0f, 1.0f, 1.0f,  // 3. right-top-front
        // BACK
        1.0f, -1.0f, -1.0f,  // 6. right-bottom-back
        -1.0f, -1.0f, -1.0f,  // 4. left-bottom-back
        1.0f, 1.0f, -1.0f,  // 7. right-top-back
        -1.0f, 1.0f, -1.0f,  // 5. left-top-back
        // LEFT
        -1.0f, -1.0f, -1.0f,  // 4. left-bottom-back
        -1.0f, -1.0f, 1.0f,  // 0. left-bottom-front
        -1.0f, 1.0f, -1.0f,  // 5. left-top-back
        -1.0f, 1.0f, 1.0f,  // 2. left-top-front
        // RIGHT
        1.0f, -1.0f, 1.0f,  // 1. right-bottom-front
        1.0f, -1.0f, -1.0f,  // 6. right-bottom-back
        1.0f, 1.0f, 1.0f,  // 3. right-top-front
        1.0f, 1.0f, -1.0f,  // 7. right-top-back
        // TOP
        -1.0f, 1.0f, 1.0f,  // 2. left-top-front
        1.0f, 1.0f, 1.0f,  // 3. right-top-front
        -1.0f, 1.0f, -1.0f,  // 5. left-top-back
        1.0f, 1.0f, -1.0f,  // 7. right-top-back
        // BOTTOM
        -1.0f, -1.0f, -1.0f,  // 4. left-bottom-back
        1.0f, -1.0f, -1.0f,  // 6. right-bottom-back
        -1.0f, -1.0f, 1.0f,  // 0. left-bottom-front
```

```
            1.0f, -1.0f, 1.0f   // 1. right-bottom-front
    });

    private final Cube FL = new Cube(new float[]{
            // FRONT
            -1.0f, -1.0f, 1.0f,  // 0. left-bottom-front
            1.0f, -1.0f, 1.0f,  // 1. right-bottom-front
            -1.0f, 1.0f, 1.0f,  // 2. left-top-front
            1.0f, 1.0f, 1.0f,  // 3. right-top-front
            // BACK
            1.0f, -1.0f, -1.0f,  // 6. right-bottom-back
            -1.0f, -1.0f, -1.0f,  // 4. left-bottom-back
            1.0f, 1.0f, -1.0f,  // 7. right-top-back
            -1.0f, 1.0f, -1.0f,  // 5. left-top-back
            // LEFT
            -1.0f, -1.0f, -1.0f,  // 4. left-bottom-back
            -1.0f, -1.0f, 1.0f,  // 0. left-bottom-front
            -1.0f, 1.0f, -1.0f,  // 5. left-top-back
            -1.0f, 1.0f, 1.0f,  // 2. left-top-front
            // RIGHT
            1.0f, -1.0f, 1.0f,  // 1. right-bottom-front
            1.0f, -1.0f, -1.0f,  // 6. right-bottom-back
            1.0f, 1.0f, 1.0f,  // 3. right-top-front
            1.0f, 1.0f, -1.0f,  // 7. right-top-back
            // TOP
            -1.0f, 1.0f, 1.0f,  // 2. left-top-front
            1.0f, 1.0f, 1.0f,  // 3. right-top-front
            -1.0f, 1.0f, -1.0f,  // 5. left-top-back
            1.0f, 1.0f, -1.0f,  // 7. right-top-back
            // BOTTOM
            -1.0f, -1.0f, -1.0f,  // 4. left-bottom-back
            1.0f, -1.0f, -1.0f,  // 6. right-bottom-back
            -1.0f, -1.0f, 1.0f,  // 0. left-bottom-front
            1.0f, -1.0f, 1.0f   // 1. right-bottom-front
    });

    public MyGLRenderer(Context context) {
        c = context;
    }

    private void loadGLTexture(GL10 gl) {
        gl.glGenTextures(3, textures, 0);
        for (int i = 0; i < texture_name.length; ++i) {
            gl.glBindTexture(GL10.GL_TEXTURE_2D, textures[i]);
            gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MIN_FILTER, GL
10.GL_LINEAR);
            InputStream is = c.getResources().openRawResource(texture_name[i]);
            Bitmap bitmap = BitmapFactory.decodeStream(is);
            GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bitmap, 0);
            bitmap.recycle();
```

```
    }
}

@Override
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
    gl.glClearColor(0.0f, 0.0f, 0.0f,1.0f);
    gl.glClearDepthf(1.0f);
    gl.glEnable(GL10.GL_DEPTH_TEST);
    gl.glDepthFunc(GL10.GL_LEQUAL);
    gl.glMatrixMode(GL10.GL_PROJECTION);

    gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT, GL10.GL_NICEST);
    gl.glShadeModel(GL10.GL_SMOOTH);

    gl.glLoadIdentity();
    gl.glOrthof(-10,10, -10, 10, -10, 100);
    GLU.gluPerspective(gl, 45, 1, -10f, 10.f);
    GLU.gluLookAt(gl,
            5.0f, 1.5f, 2.0f,
            0.0f, 0.0f, 0.0f,
            0.0f, 1.0f, 0.0f
    );
    gl.glMatrixMode(GL10.GL_MODELVIEW);
    gl.glLoadIdentity();
    gl.glScalef(1f, 2f, 1);
    loadGLTexture(gl);
}

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

@Override
public void onDrawFrame(GL10 gl) {

    gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
    gl.glEnable(GL10.GL_TEXTURE_2D);

    gl.glClearColor(1.0f, 0.980f, 0.804f, 1.0f);
    gl.glPopMatrix(); // крышка
    gl.glPushMatrix();
    gl.glTranslatef(0.0f, -2.0f, 0f);
    gl.glScalef(5f, 0.2f, 5f);
    gl.glColor4f(1, 1 ,1, 1.0f);
    gl.glRotatef(angle, 0.0f, -1.0f, 0f);
    cube.draw(gl);

    gl.glPopMatrix(); // чаша
    gl.glPushMatrix();
    gl.glTranslatef((float) (3f * Math.cos(a) - -2f * Math.sin(a)),
            -1.15f,
```

```
        (float) (-2f * Math.cos(a) + 3f * Math.sin(a)));
gl.glScalef(.44f, .44f, .44f);
gl.glRotatef(angle, 0.0f, 1.0f, 0.0f);
cup.onDrawFrame(gl);

gl.glPopMatrix(); // ножка передняя слева
gl.glPushMatrix();
gl.glTranslatef((float) (-4.7f * Math.cos(a) - 4.7f * Math.sin(a)),
        -3.0f,
        (float) (4.7f * Math.cos(a) + -4.7f * Math.sin(a)));
gl.glScalef(0.3f, 1f, 0.3f);
gl.glColor4f(1, 1 ,0, 1.0f);
gl.glRotatef(angle, .0f, -1.0f, 0f);
FL.draw(gl);

gl.glPopMatrix(); // ножка передняя справа
gl.glPushMatrix();
gl.glTranslatef((float) (4.7f * Math.cos(a) - 4.7f * Math.sin(a)),
        -3.0f,
        (float) (4.7f * Math.cos(a) + 4.7f * Math.sin(a)));
gl.glScalef(0.3f, 1f, 0.3f);
gl.glColor4f(1, 1 ,0, 1.0f);
gl.glRotatef(angle, 0.0f, -1.0f, 0f);
cube.draw(gl);

gl.glPopMatrix(); // ножка задняя справа
gl.glPushMatrix();
gl.glTranslatef((float) (4.7f * Math.cos(a) - -4.7f * Math.sin(a)),
        -3.0f,
        (float) (-4.7f * Math.cos(a) + 4.7f * Math.sin(a)));
gl.glScalef(0.3f, 1f, 0.3f);
gl.glColor4f(1, 1 ,0, 1.0f);
gl.glRotatef(angle, 0.0f, -1.0f, 0f);
cube.draw(gl);

gl.glPopMatrix(); // ножка задняя слева
gl.glPushMatrix();
gl.glTranslatef((float) (-4.7f * Math.cos(a) - -4.7f * Math.sin(a)),
        -3.0f,
        (float) (-4.7f * Math.cos(a) + -4.7f * Math.sin(a)));
gl.glScalef(0.3f, 1f, 0.3f);
gl.glColor4f(1, 1 ,0, 1.0f);
gl.glRotatef(angle, 0.0f, -1.0f, 0f);
cube.draw(gl);

gl.glPopMatrix(); // арбузик
gl.glPushMatrix();
gl.glTranslatef((float) (0f * Math.cos(a) - 1f * Math.sin(a)),
        0.0f,
        (float) (1f * Math.cos(a) + 0f * Math.sin(a)));
```

```
        gl.glBindTexture(GL10.GL_TEXTURE_2D, textures[0]);
        gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
        gl.glTexCoordPointer(1, GL10.GL_FLOAT, 0, watermelon.textureBuffer);
        gl.glColor4f(1, 1 ,1, 1);
        gl.glScalef(.6f, .6f, .6f);
        gl.glRotatef(angle, 0.0f, -1.0f, 0.0f);
        watermelon.onDrawFrame(gl);

        gl.glPopMatrix(); // яблочко
        gl.glPushMatrix();
        gl.glTranslatef((float) (3f * Math.cos(a) - 3f * Math.sin(a)),
                -1.15f,
                (float) (3f * Math.cos(a) + 3f * Math.sin(a)));
        gl.glScalef(.44f, .44f, .44f);
        gl.glBindTexture(GL10.GL_TEXTURE_2D, textures[1]);
        gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, apple.textureBuffer);
        gl.glColor4f(1, 1 ,1, 1);
        gl.glRotatef(angle, 0.0f, -1.0f, 0.0f);
        apple.onDrawFrame(gl);

        gl.glPopMatrix(); // лимончик
        gl.glPushMatrix();
        gl.glTranslatef((float) (-2f * Math.cos(a) - -3f * Math.sin(a)),
                -1.27f,
                (float) (-3f * Math.cos(a) + -2f * Math.sin(a)));
        gl.glScalef(.36f, .36f, .36f);
        gl.glBindTexture(GL10.GL_TEXTURE_2D, textures[2]);
        gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, lemon.textureBuffer);
        gl.glColor4f(1, 1 ,1, 1);
        gl.glRotatef(angle, 0.0f, -1.0f, 0.0f);
        lemon.onDrawFrame(gl);

        gl.glPopMatrix(); // апельсинчик
        gl.glPushMatrix();
        gl.glTranslatef((float) (-3.8f * Math.cos(a) - -1f * Math.sin(a)),
                -1.05f,
                (float) (-1f * Math.cos(a) + -3.8f * Math.sin(a)));
        gl.glBindTexture(GL10.GL_TEXTURE_2D, textures[3]);
        gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, orange.textureBuffer);
        gl.glScalef(.5f, .5f, .5f);
        gl.glColor4f(1, 1 ,1, 1);
        gl.glRotatef(angle, 0.0f, -1.0f, 0.0f);
        orange.onDrawFrame(gl);

        angle += speed;
        a += 0.001745 % Float.MAX_VALUE;
        gl.glPopMatrix();
        gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
        gl.glDisable(GL10.GL_TEXTURE_2D);
    }
```

```
}
```

Cup.java:

```
package ru.mikheev_ustenko.courc;

import javax.microedition.khronos.opengles.GL10;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.FloatBuffer;

public class Cup {

    public FloatBuffer mVertexBuffer;
    public FloatBuffer textureBuffer;
    public int n = 0, sz = 0;
    private float red;
    private float green;
    private float blue;

    public Cup(float R, float r, float g, float b) {
        this.red = r;
        this.green = g;
        this.blue = b;
        int dtheta = 15, dphi = 15;
        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 (int theta = -90; theta <= 90 - dtheta; theta += dtheta) {
            for (int phi = 180; 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) * Ma
th.cos(phi * DTOR)) * R);
                mVertexBuffer.put((float) (Math.cos((theta + dtheta) * DTOR) * Ma
th.sin(phi * DTOR)) * R);
                mVertexBuffer.put((float) (Math.sin((theta + dtheta) * DTOR)) * R
);
```

```
                mVertexBuffer.put((float) (Math.cos((theta + dtheta) * DTOR) * Ma
th.cos((phi + dphi) * DTOR)) * R);
                mVertexBuffer.put((float) (Math.cos((theta + dtheta) * DTOR) * Ma
th.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);
        textureBuffer.position(0);
    }

    public void onDrawFrame(GL10 gl) {
        gl.glFrontFace(GL10.GL_CCW);
        gl.glEnable(GL10.GL_CULL_FACE);
        gl.glCullFace(GL10.GL_BACK);
        gl.glColor4f(red, green, blue, 1.0f);

        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
        gl.glVertexPointer(3, GL10.GL_FLOAT, 0, mVertexBuffer);
        for (int i = 0; i < n; i += 4) {
            gl.glDrawArrays(GL10.GL_TRIANGLE_FAN, i, 4);
        }
        gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
        gl.glDisable(GL10.GL_CULL_FACE);
    }
}
```

Sphere.java:

```
package ru.mikheev_ustenko.courc;
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;

class Sphere implements GLSurfaceView.Renderer {
    private final FloatBuffer mVertexBuffer;
    public FloatBuffer textureBuffer;
```

```
    private int n = 0;

    private float[][] colors = {  // Colors of the 6 faces
            {1.0f, 0.0f, 1.0f, 1.0f},  // 0. orange
            {0.95f, 0.0f, 1.0f, 1.0f},  // 1. violet
            {0.9f, 0.0f, 1.0f, 1.0f},  // 1. violet
    };

    public Sphere(float R) {
        int dtheta = 15, dphi = 15;
        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 (int theta = -90; theta <= 90 - dtheta; theta += dtheta) {
            for (int phi = 0; phi <= 360 - dphi; phi += dphi){
                mVertexBuffer.put((float)(Math.cos(theta*DTOR) * Math.cos(phi*DTO
R))*R);
                mVertexBuffer.put((float)(Math.cos(theta*DTOR) * Math.sin(phi*DTO
R))*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+dp
hi)*DTOR))*R);
                mVertexBuffer.put((float)(Math.cos(theta*DTOR) * Math.sin((phi+dp
hi)*DTOR))*R);
                mVertexBuffer.put((float)(Math.sin(theta*DTOR))*R);
                n++;

                textureBuffer.put(phi / 360.0f);
                textureBuffer.put((90 + theta) / 180.0f);

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

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

                textureBuffer.put((phi + dphi) / 360.0f);
                textureBuffer.put((90 + theta + dtheta) / 180.0f);

                textureBuffer.put((phi + dphi) / 360.0f);
                textureBuffer.put((90 + theta) / 180.0f);
            }
        }
        mVertexBuffer.position(0);
        textureBuffer.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.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);
        gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, textureBuffer);
        for (int i = 0; i < n; i += 4)
            gl.glDrawArrays(GL10.GL_TRIANGLE_FAN, i,4);

        gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
        gl.glDisable(GL10.GL_BLEND);
    }
}
```