Федеральное агентство связи

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

Лабораторная работа №3

«Матрицы модели-вида OpenGL ES»

Выполнил: студент 4 курса ИВТ,

гр. ИП-713

Михеев Н.А.

Проверил: ассистент кафедры ПМиК,

Павлова У.В.

Новосибирск, 2020 г.

Задание

Необходимо создать модель Солнце и вращающиеся Земля и Луна. Текстуры взять из интеренета.

Решение поставленной задачи

За основу были взяты классы шара и рендерера из лабораторной работы №2 с модификациями для работы с текстурами. Добавлено вращение Солнца, Луны и Земли вокруг своей оси, так же движение Земли вокруг Солнца и Луны вокруг Земли.

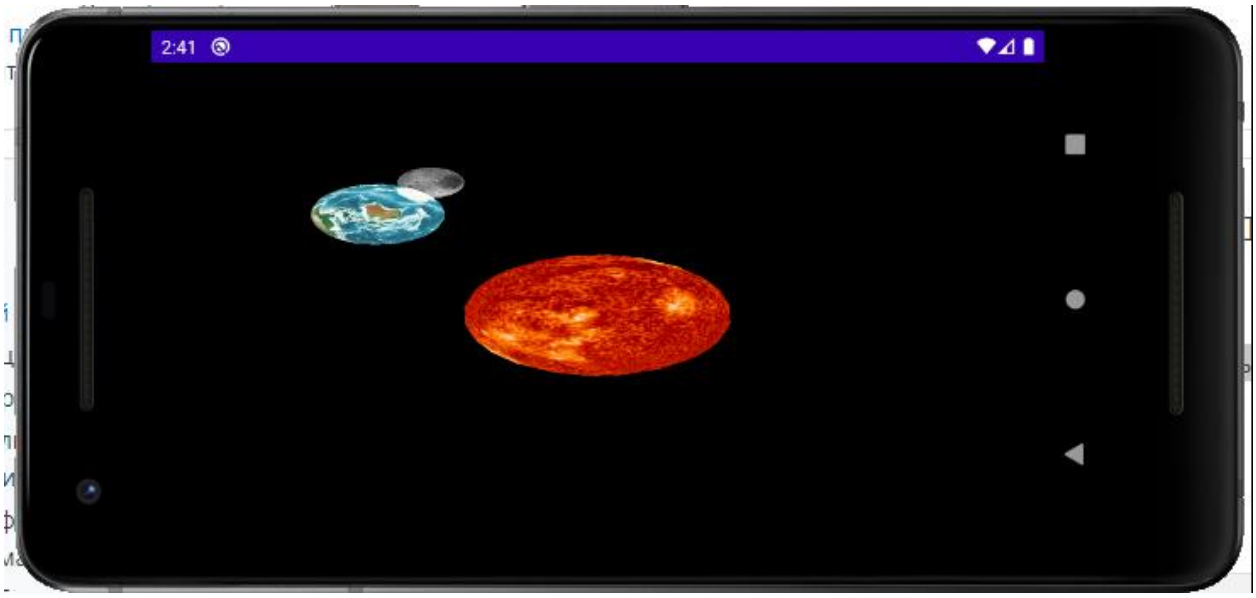

Рис. 1 – демонстрация работы программы.

Листинг программы

MainActivity:

```
package ru.mikheev.solarsystem;

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));
        setContentView(gLView);
    }
```

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

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

MyGLRenderer:

```
package ru.mikheev.solarsystem;

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 java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.FloatBuffer;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import static java.lang.Math.cos;
import static java.lang.Math.sin;

class MyGLRenderer implements GLSurfaceView.Renderer {
    static public int[] texture_name = {
            R.drawable.sun,
            R.drawable.earth,
            R.drawable.moon
    };

    static public int[] textures = new int [texture_name.length];
    Context c;
    private final Sphere Sun = new Sphere(2f);
    private final Sphere Earth = new Sphere(1f);
    private final Sphere Moon = new Sphere(0.5f);

    private float p = 0.0f;
    private float angle = 40.0f;
    private float mTransY = 0f;

    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,
GL10.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.glLoadIdentity();
        gl.glOrthof(-10,10, -10, 10, -10, 10);
        gl.glMatrixMode(GL10.GL_MODELVIEW);
        gl.glLoadIdentity();
        gl.glScalef(1, 0.8f, 1);
        loadGLTexture(gl);
    }

    @Override
    public void onSurfaceChanged(GL10 gl, int width, int height) {
//        if (height == 0) height = 1;
//        float aspect = (float)width / height;
//
//        gl.glViewport(0, 0, width, height);
//
//        gl.glMatrixMode(GL10.GL_PROJECTION);
//        gl.glLoadIdentity();
//
//        GLU.gluPerspective(gl, 90, aspect, 0.1f, 0.5f);
//
//        gl.glMatrixMode(GL10.GL_MODELVIEW);  // Select model-view matrix
//        gl.glLoadIdentity();
    }

    @Override
    public void onDrawFrame(GL10 gl) {
        float RotationOffset;
        float RotationSpeed;

        p = (p == 360) ? 0 : p + 2;
        angle = (angle == 360) ? 0 : angle + 0.5f;

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

        gl.glEnable(GL10.GL_TEXTURE_2D);
        gl.glBindTexture(GL10.GL_TEXTURE_2D, textures[0]);
        gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
```

```
        gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, Sun.textureBuffer);
        gl.glPushMatrix();
        //gl.glRotatef(90, 1, 0, 0);
        //gl.glRotatef(p, 1, 0, 0f);
        gl.glScalef(1.5f, 1.5f, 1.5f);
        gl.glRotatef(p, 1, 0, 1);
        Sun.onDrawFrame(gl);

        float e_x = (float) Math.cos(mTransY) * 7f;
        float e_y = (float) Math.sin(mTransY) * 7f;

        gl.glPopMatrix();
        RotationOffset = 6.0f;
        RotationSpeed = 0.1f;
        gl.glPushMatrix();
        gl.glTranslatef(e_x, e_y, 1.0f);
        //gl.glRotatef(90, 1, 0, 0);
        //gl.glRotatef(p, 0, 0, 2);
        //gl.glTranslatef(e_x, e_y, -1.0f);
        gl.glScalef(1.5f, 1.5f, 1.5f);
        gl.glRotatef(angle, 1, 0, 1);
        gl.glPushMatrix();
        gl.glBindTexture(GL10.GL_TEXTURE_2D, textures[1]);
        gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, Earth.textureBuffer);
        Earth.onDrawFrame(gl);

        gl.glRotatef(-p, 0.3f, 1, 0);
        RotationOffset = 1.5f;
        RotationSpeed = 0.2f;
        gl.glTranslatef(RotationOffset * (float)(cos(1 * RotationSpeed)),0,
                RotationOffset * (float)(sin(1 * RotationSpeed)));
        gl.glRotatef(p, 0, 1, 0);
        gl.glColor4f(0, 0 ,1f , 1);
//gl.glRotatef(angle, 1.0f, 1.0f, 1.0f);
        gl.glBindTexture(GL10.GL_TEXTURE_2D, textures[2]);
        gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, Moon.textureBuffer);
        gl.glColor4f(1, 1 ,1, 1);
        Moon.onDrawFrame(gl);

        gl.glPopMatrix();
        gl.glPopMatrix();
        gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
        gl.glDisable(GL10.GL_TEXTURE_2D);

        mTransY+=0.05f;
    }
}
```

Sphere:

```
package ru.mikheev.solarsystem;

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*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(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);
    }
}
```