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»**

Выполнили:

Студенты 4 курса

ИВТ, гр. ИП-712

Ладышкин А.В

Гольцев Д.С.

Самсонов Д.В.

Новосибирск 2020 г.

Оглавление

***Задание ......................................................................................................................... 3***

***Скриншоты ................................................................................................................. 3***

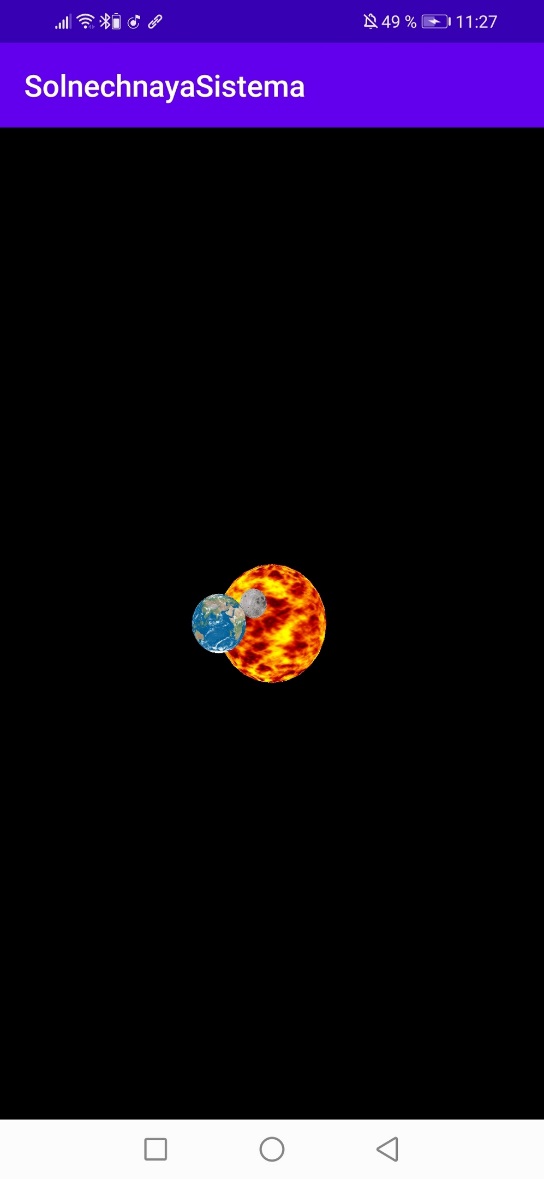
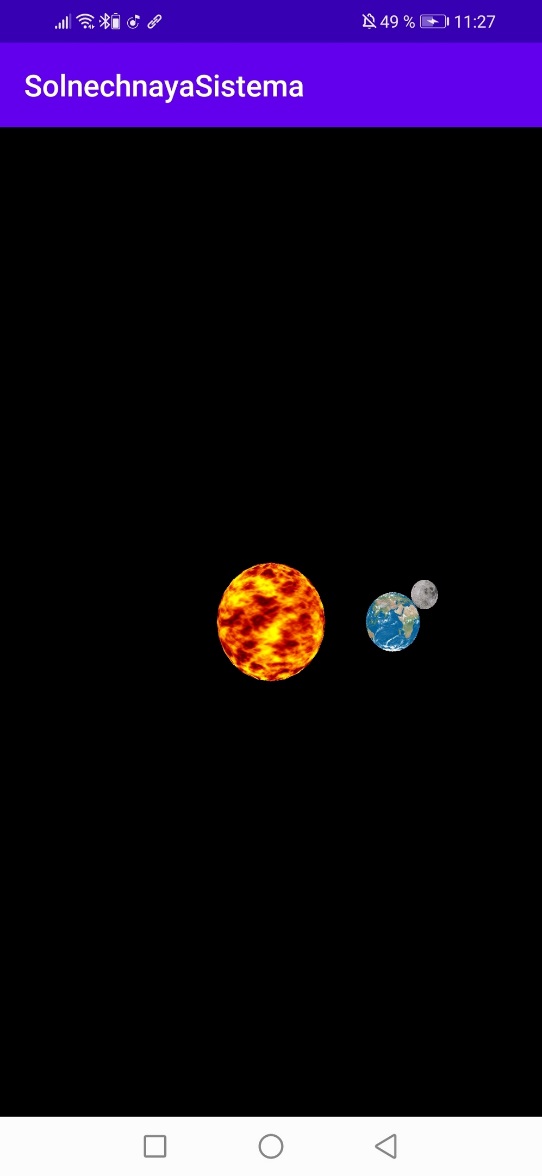
***Листинг кода .............................................................................................................. 5***

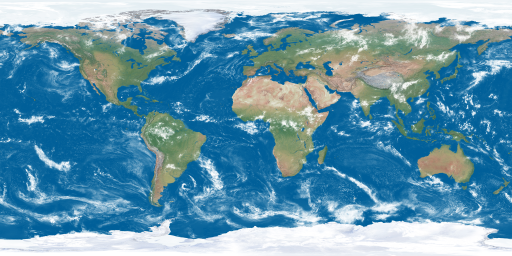
**Задание**

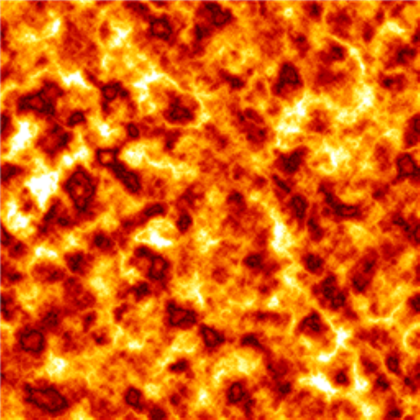
Лабораторная работа "Матрицы модели-вида OpenGL ES 1"

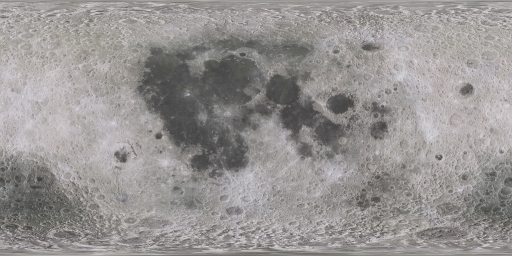
Необходимо создать модель Солнце и вращающиеся Земля и Луна. Текстуры взять из интеренета.

**Скриншоты**







**Листинг проекта**

MainActivity.java

package com.example. solnechnayasistema;

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 OpenGLRenderer(this));

g.setRenderMode(GLSurfaceView.RENDERMODE\_CONTINUOUSLY);

setContentView(g);

}

}

Renderer.java

package com.example. solnechnayasistema;

import android.content.Context;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.opengl.GLSurfaceView;

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 OpenGLRenderer 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 Sphere Sun = new Sphere(2f);

private Sphere Earth = new Sphere(1f);

private Sphere Moon = new Sphere(0.5f);

private float p = 0.0f;

private float angle = 40.0f;

public OpenGLRenderer(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, 0,1.0f);

gl.glClearDepthf(1);

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

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, 0.60f, 1);

loadGLTexture(gl);

}

@Override

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

}

@Override

public void onDrawFrame(GL10 gl) {

float RotationOffset;

float RotationSpeed;

p = (p == 360) ? 0 : p + 2;

angle = (angle == 360) ? 0 : angle + 0.15f;

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, 0, 0, 0.1f);

gl.glColor4f(1, 1 ,0 , 1);

Sun.onDrawFrame(gl);

gl.glPopMatrix();

RotationOffset = 6.0f;

RotationSpeed = 0.1f;

gl.glPushMatrix();

gl.glTranslatef(RotationOffset \* (float)(cos(angle \* RotationSpeed)),

/\*RotationOffset \* (float)(cos(angle \* RotationSpeed))\*/ 0,

gl.glRotatef(90, 1, 0, 0);

gl.glRotatef(p, 0, 0, 2);

gl.glPushMatrix();

gl.glBindTexture(GL10.GL\_TEXTURE\_2D, textures[1]);

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

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

Earth.onDrawFrame(gl);

gl.glRotatef(-p, 0.3f, 1, 0);

RotationOffset = 1.5f;

RotationSpeed = 0.2f;

gl.glTranslatef(RotationOffset \* (float)(cos(1 \* RotationSpeed)),

/\*RotationOffset \* (float)(-0.5f \* cos(angle \* 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);

}

}

Sphere.java

package com.example. solnechnayasistema;

import android.opengl.GLSurfaceView;

import java.nio.ByteBuffer;

import java.nio.ByteOrder;

import java.nio.FloatBuffer;

import java.util.Random;

import javax.microedition.khronos.egl.EGLConfig;

import javax.microedition.khronos.opengles.GL10;

class Sphere implements GLSurfaceView.Renderer {

private FloatBuffer mVertexBuffer;

public FloatBuffer textureBuffer;

private int n;

public Sphere(float R) {

float i = 0;

n = 0;

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);

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 += 4;

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);

}

@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); //разрешить наложение цветов

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

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

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

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

// также

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

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

// рендер примитивов из массива

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

gl.glDisable(GL10.GL\_BLEND);

}

}