Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Кафедра ПМиК

Лабораторная работа №3

по дисциплине

«Программирование мобильных устройств»

Выполнил:   
студент гр. ИП-813

Бурдуковский И.А.

Проверила:

Павлова У.В.

Новосибирск 2021

Оглавление

[Задание 3](#_Toc36477188)

[Выполнение 4](#_Toc36477189)

[Листинг проекта 5](#_Toc36477190)

# Задание

Лабораторная работа "Матрицы модели-вида OpenGL ES 1"

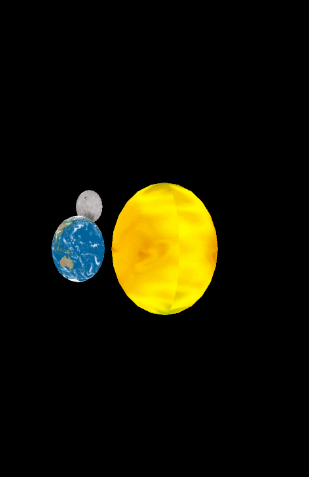
Необходимо создать модель Солнце и вращающиеся Земля и Луна. Текстуры взять из интеренета.

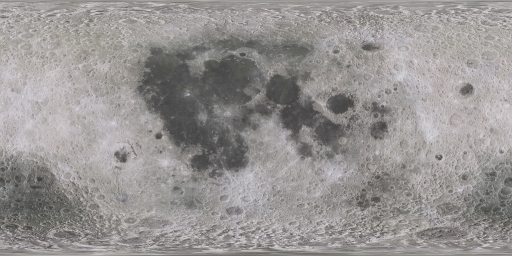
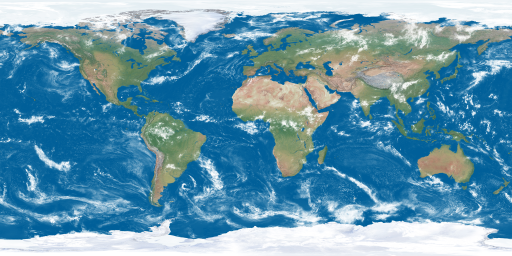
# Выполнение

Для реализации данной лабораторной работы мне было необходимо создать один класс-рендер который рисовал объекты-сферы и имел в себе описание движений их в пространстве.

И так же класс ответственный за описание сферы, постановку вершин объекта в пространстве и наложение текстуры на объект.

Текстуры были позаимствованы из Интернета.







# Листинг проекта

MainActivity.java

package com.example.lab3;  
  
import android.app.Activity;  
import android.opengl.GLSurfaceView;  
import android.os.Bundle;  
import android.view.WindowManager;  
  
public class MainActivity extends Activity {  
 private GLSurfaceView g;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 getWindow().addFlags(WindowManager.LayoutParams.FLAG\_KEEP\_SCREEN\_ON);  
  
 super.onCreate(savedInstanceState);  
  
 g = new GLSurfaceView(this);  
 g.setEGLConfigChooser(8,8,8,8,16,1);  
 g.setRenderer(new MyRenderer(this));  
 g.setRenderMode(GLSurfaceView.RENDERMODE\_CONTINUOUSLY);  
 setContentView(g);  
 }  
  
 @Override  
 protected void onPause() {  
 super.onPause();  
 g.onPause();  
 }  
 @Override  
 protected void onResume() {  
 super.onResume();  
 g.onResume();  
 }  
}

MyRenderer.java

package com.example.lab3;  
  
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 javax.microedition.khronos.egl.EGLConfig;  
import javax.microedition.khronos.opengles.GL10;  
  
  
public class MyRenderer 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 context;  
  
 private Sphere Sun = new Sphere(2f);  
 private Sphere Earth = new Sphere(1f);  
 private Sphere Moon = new Sphere(0.5f);  
  
 private float p = 0f;  
 private float angle = 0;  
  
 public MyRenderer(Context context){  
 this.context = context;  
 Sun = new Sphere(2f);  
 Earth = new Sphere(1f);  
 Moon = new Sphere(0.5f);  
 }  
  
 @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(1.5f, 1f, 1);  
 loadGLTexture(gl);  
 }  
  
 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 = context.getResources().openRawResource(texture\_name[i]);  
 Bitmap bitmap = BitmapFactory.decodeStream(is);  
 GLUtils.texImage2D(GL10.GL\_TEXTURE\_2D, 0, bitmap, 0);  
 bitmap.recycle();  
 }  
 }  
  
  
 @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)(Math.cos(angle \* RotationSpeed)),  
 /\*RotationOffset \* (float)(Math.cos(angle \* RotationSpeed))\*/ 0,  
 RotationOffset \* (float)(Math.sin(angle \* RotationSpeed)));  
 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)(Math.cos(1 \* RotationSpeed)),  
 /\*RotationOffset \* (float)(-0.5f \* Math.cos(angle \* RotationSpeed))\*/ 0,  
 RotationOffset \* (float)(Math.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.lab3;  
  
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;  
  
public class Sphere implements GLSurfaceView.Renderer {  
 public FloatBuffer mVertexBuffer;  
 public FloatBuffer textureBuffer;  
 public int n = 0, sz = 0;  
  
 private float[][] colors = { // Colors of the 6 faces  
 {1.0f, 0.0f, 0.0f, 1.0f}, // 0. orange  
 {0.95f, 0.5f, 0.5f, 1.0f}, // 1. violet  
 {1.0f, 1.0f, 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) {  
 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);  
  
 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(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.glFrontFace(GL10.GL\_CCW);  
 gl.glEnable(GL10.GL\_CULL\_FACE);  
 gl.glCullFace(GL10.GL\_BACK);  
  
 gl.glEnableClientState(GL10.GL\_VERTEX\_ARRAY);  
 gl.glVertexPointer(3, GL10.GL\_FLOAT, 0, mVertexBuffer);  
 for (int i = 0; i < n; i += 4) {  
 // gl.glColor4f(colors[i % 3][0], colors[i % 3][1], colors[i % 3][2], colors[i % 3][3]);  
 gl.glDrawArrays(GL10.GL\_TRIANGLE\_FAN, i, 4);  
 }  
 gl.glDisableClientState(GL10.GL\_VERTEX\_ARRAY);  
 gl.glDisable(GL10.GL\_CULL\_FACE);  
 }  
}