Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

Кафедра ПМиК

Курсовая работа по дисциплине

Программирование мобильных устройств

Выполнил: студент 4 курса

Ф. ИВТ, группа: ИП-711

Мартасов И. О.

Проверил: доцент кафедры ПМиК

Нечта Иван Васильевич

Новосибирск, 2020

**Содержание**

1. **Постановка задачи..………………………………………………………3**
2. **Исходный код……..……………………………………………………….3**
3. **Результаты работы…..………………………………………………….31**

**Постановка задачи**

Создайте программу в которой нарисован стол на OpenGL ES 2.0.

На столе лежат различные фрукты/овощи (не менее 4 различных), стакан с напитком. Имеется свеча, дающее освещение (по модели Фонга).

**Исходный код**

**MyRenderer.java – основной класс работы**

package com.example.curs\_work;

import javax.microedition.khronos.egl.EGLConfig;

import javax.microedition.khronos.opengles.GL10;

import android.opengl.GLES20;

import android.opengl.Matrix;

import android.content.Context;

import android.opengl.GLSurfaceView;

public class MyRenderer implements GLSurfaceView.Renderer

{

private final MainActivity mShadowsActivity;

private ShadersWork mDepthMapProgram;

private ShadersWork mSimpleShadowProgram;

private int mActiveProgram;

private final float[] mMVMatrix = new float[16];

private final float[] mMVPMatrix = new float[16];

private final float[] mViewMatrix = new float[16];

private final float[] mModelMatrix = new float[16];

private final float[] mNormalMatrix = new float[16];

private final float[] mLightMvpMatrix = new float[16];

private final float[] mLightViewMatrix = new float[16];

private final float[] mProjectionMatrix = new float[16];

private final float[] mLightPosInEyeSpace = new float[16];

private final float[] mActualLightPosition = new float[4];

private final float[] mLightProjectionMatrix = new float[16];

private final float[] mLightPosModel = new float[]

{

0.1f, 10.0f, 0.1f, 1.0f

};

private float s = 0;

private int mDisplayWidth;

private int mDisplayHeight;

private int mShadowMapWidth;

private int mShadowMapHeight;

private int[] fboId;

private int scene\_mvpMatrixUniform;

private int scene\_mvMatrixUniform;

private int scene\_normalMatrixUniform;

private int scene\_lightPosUniform;

private int scene\_shadowProjMatrixUniform;

private int scene\_mapStepXUniform;

private int scene\_mapStepYUniform;

private int shadow\_mvpMatrixUniform;

private int scene\_positionAttribute;

private int scene\_normalAttribute;

private int scene\_colorAttribute;

private int shadow\_positionAttribute;

private final Context c;

private TableObjects Cup;

private TableObjects Table;

private TableObjects Torch;

private TableObjects Beets;

private TableObjects Apple;

private TableObjects Orange;

private TableObjects Cabbage;

MyRenderer(final MainActivity shadowsActivity, Context c)

{

mShadowsActivity = shadowsActivity;

this.c = c;

}

@Override

public void onSurfaceCreated(GL10 unused, EGLConfig config)

{

GLES20.glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

GLES20.glEnable(GLES20.GL\_DEPTH\_TEST);

GLES20.glEnable(GLES20.GL\_CULL\_FACE);

/\*создание объектов\*/

Apple = new TableObjects(c, new float[]{1f, 0f, 0f, 1.0f}, "orange.obj");

Cup = new TableObjects(c, new float[]{0.0f, 0.2f, 0.3f, 1.0f}, "cup.obj");

Torch = new TableObjects(c, new float[]{0.9f, 1f, 0f, 1.0f}, "torch.obj");

Table = new TableObjects(c, new float[]{0.1f, 0f, 0.6f, 1.0f}, "Table.obj");

Beets = new TableObjects(c, new float[]{0.9f, 0f, 0.2f, 1.0f}, "beets.obj");

Orange = new TableObjects(c, new float[]{0.9f, 0.5f, 0f, 1.0f}, "orange.obj");

Cabbage = new TableObjects(c, new float[]{0.5f, 0.9f, 0.5f, 1.0f}, "cabbage.obj");

/\*Загрузка шейдеров\*/

String depth\_tex\_f\_depth\_map\_shader = "precision highp float;\n" +

"\n" +

"void main() {\n" +

"}";

String depth\_tex\_v\_depth\_map\_shader = "precision highp float;\n" +

"uniform mat4 uMVPMatrix;\n" +

"attribute vec4 aShadowPosition;\n" +

"\n" +

"void main() {\n" +

"\tgl\_Position = uMVPMatrix \* aShadowPosition;\n" +

"}";

mDepthMapProgram = new ShadersWork(depth\_tex\_v\_depth\_map\_shader, depth\_tex\_f\_depth\_map\_shader, mShadowsActivity);

String depth\_tex\_v\_with\_shadow\_shader = "uniform mat4 uMVPMatrix;\n" +

"uniform mat4 uMVMatrix;\n" +

"uniform mat4 uNormalMatrix;\n" +

"uniform mat4 uShadowProjMatrix;\n" +

"attribute vec4 aPosition;\n" +

"attribute vec4 aColor;\n" +

"attribute vec3 aNormal;\n" +

"varying vec3 vPosition; \t\t\n" +

"varying vec4 vColor; \t\t\n" +

"varying vec3 vNormal;\n" +

"varying vec4 vShadowCoord;\n" +

"\n" +

"void main() {\n" +

"\tvPosition = vec3(uMVMatrix \* aPosition);\n" +

"\tvColor = aColor;\n" +

"\tvNormal = vec3(uNormalMatrix \* vec4(aNormal, 0.0));\n" +

"\tvShadowCoord = uShadowProjMatrix \* aPosition;\n" +

"\tgl\_Position = uMVPMatrix \* aPosition; \n" +

"}";

String depth\_tex\_f\_with\_simple\_shadow\_shader = "precision mediump float;\n" +

"\n" +

"uniform vec3 uLightPos;\n" +

"uniform sampler2D uShadowTexture;\n" +

"uniform float uxPixelOffset;\n" +

"uniform float uyPixelOffset;\n" +

"varying vec3 vPosition;\n" +

"varying vec4 vColor;\n" +

"varying vec3 vNormal;\n" +

"varying vec4 vShadowCoord;\n" +

"\n" +

"float shadowSimple(){\n" +

"\tvec4 shadowMapPosition = vShadowCoord / vShadowCoord.w;\n" +

"\tfloat distanceFromLight = texture2D(uShadowTexture, shadowMapPosition.st).z;\n" +

"\tfloat bias = 0.001;\n" +

"\treturn float(distanceFromLight > shadowMapPosition.z - bias);\n" +

"}\n" +

" \n" +

"void main() {\n" +

"\tvec3 lightVec = uLightPos - vPosition;\n" +

"\tlightVec = normalize(lightVec);\n" +

"\tfloat specular = pow(max(dot(vNormal, lightVec), 0.0), 5.0);\n" +

"\tfloat diffuse = max(dot(vNormal, lightVec), 0.1);\n" +

"\tfloat ambient = 0.3;\n" +

" \tfloat shadow = 1.0;\n" +

"\t\tif (vShadowCoord.w > 0.0) {\n" +

"\t\t\tshadow = shadowSimple();\n" +

"\t\t\tshadow = (shadow \* 0.9) + 0.9;\n" +

"\t\t}\n" +

" gl\_FragColor = (vColor \* (diffuse + ambient + specular) \* shadow);\n" +

"} ";

mSimpleShadowProgram = new ShadersWork(depth\_tex\_v\_with\_shadow\_shader, depth\_tex\_f\_with\_simple\_shadow\_shader, mShadowsActivity);

mActiveProgram = mSimpleShadowProgram.getProgram();

}

@Override

public void onSurfaceChanged(GL10 unused, int width, int height)

{

mDisplayWidth = width;

mDisplayHeight = height;

GLES20.glViewport(0, 0, mDisplayWidth, mDisplayHeight);

generate\_shadows();

float ratio = (float) mDisplayWidth / mDisplayHeight;

Matrix.frustumM(mProjectionMatrix, 0, -ratio, ratio, -1.0f, 1.0f, 1.0f, 100.0f);

Matrix.frustumM(mLightProjectionMatrix, 0, -1.1f \* ratio, 1.1f \* ratio, 1.1f \* -1.0f, 1.1f \* 1.0f, 1.0f, 100.0f);

}

@Override

public void onDrawFrame(GL10 unused)

{

mActiveProgram = mSimpleShadowProgram.getProgram();

Matrix.setLookAtM(mViewMatrix, 0,

3, 5, 0,

0, 0, 0,

-1,0,0);

scene\_mvpMatrixUniform = GLES20.glGetUniformLocation(mActiveProgram, "uMVPMatrix");

scene\_mvMatrixUniform = GLES20.glGetUniformLocation(mActiveProgram, "uMVMatrix");

scene\_normalMatrixUniform = GLES20.glGetUniformLocation(mActiveProgram, "uNormalMatrix");

scene\_lightPosUniform = GLES20.glGetUniformLocation(mActiveProgram, "uLightPos");

scene\_shadowProjMatrixUniform = GLES20.glGetUniformLocation(mActiveProgram, "uShadowProjMatrix");

scene\_positionAttribute = GLES20.glGetAttribLocation(mActiveProgram, "aPosition");

scene\_normalAttribute = GLES20.glGetAttribLocation(mActiveProgram, "aNormal");

scene\_colorAttribute = GLES20.glGetAttribLocation(mActiveProgram, "aColor");

scene\_mapStepXUniform = GLES20.glGetUniformLocation(mActiveProgram, "uxPixelOffset");

scene\_mapStepYUniform = GLES20.glGetUniformLocation(mActiveProgram, "uyPixelOffset");

int shadowMapProgram = mDepthMapProgram.getProgram();

shadow\_mvpMatrixUniform = GLES20.glGetUniformLocation(shadowMapProgram, "uMVPMatrix");

shadow\_positionAttribute = GLES20.glGetAttribLocation(shadowMapProgram, "aShadowPosition");

float[] basicMatrix = new float[16];

Matrix.setIdentityM(basicMatrix, 0);

Matrix.multiplyMV(mActualLightPosition, 0, basicMatrix, 0, mLightPosModel, 0);

Matrix.setIdentityM(mModelMatrix, 0);

Matrix.setLookAtM(mLightViewMatrix, 0,

mActualLightPosition[0], mActualLightPosition[1], mActualLightPosition[2],

mActualLightPosition[0], -mActualLightPosition[1], mActualLightPosition[2],

-mActualLightPosition[0], 0, -mActualLightPosition[2]);

GLES20.glCullFace(GLES20.GL\_FRONT);

s+=1f;

if (s >= 360)

{

s -= 360;

}

Matrix.rotateM(mModelMatrix, 0, s, 0,1,0);

draw\_shadows();

GLES20.glCullFace(GLES20.GL\_BACK);

draw\_picture();

}

/\*создание теней\*/

private void generate\_shadows()

{

mShadowMapWidth = Math.round(mDisplayWidth);

mShadowMapHeight = Math.round(mDisplayHeight);

fboId = new int[1];

int[] depthTextureId = new int[1];

int[] renderTextureId = new int[1];

GLES20.glGenFramebuffers(1, fboId, 0);

GLES20.glGenRenderbuffers(1, depthTextureId, 0);

GLES20.glBindRenderbuffer(GLES20.GL\_RENDERBUFFER, depthTextureId[0]);

GLES20.glRenderbufferStorage(GLES20.GL\_RENDERBUFFER, GLES20.GL\_DEPTH\_COMPONENT16, mShadowMapWidth, mShadowMapHeight);

GLES20.glGenTextures(1, renderTextureId, 0);

GLES20.glBindTexture(GLES20.GL\_TEXTURE\_2D, renderTextureId[0]);

GLES20.glTexParameteri(GLES20.GL\_TEXTURE\_2D, GLES20.GL\_TEXTURE\_MIN\_FILTER, GLES20.GL\_NEAREST);

GLES20.glTexParameteri(GLES20.GL\_TEXTURE\_2D, GLES20.GL\_TEXTURE\_MAG\_FILTER, GLES20.GL\_NEAREST);

GLES20.glTexParameteri(GLES20.GL\_TEXTURE\_2D, GLES20.GL\_TEXTURE\_WRAP\_S, GLES20.GL\_CLAMP\_TO\_EDGE);

GLES20.glTexParameteri(GLES20.GL\_TEXTURE\_2D, GLES20.GL\_TEXTURE\_WRAP\_T, GLES20.GL\_CLAMP\_TO\_EDGE);

GLES20.glBindFramebuffer(GLES20.GL\_FRAMEBUFFER, fboId[0]);

GLES20.glTexImage2D(GLES20.GL\_TEXTURE\_2D, 0, GLES20.GL\_DEPTH\_COMPONENT, mShadowMapWidth, mShadowMapHeight, 0, GLES20.GL\_DEPTH\_COMPONENT, GLES20.GL\_UNSIGNED\_INT, null);

GLES20.glFramebufferTexture2D(GLES20.GL\_FRAMEBUFFER, GLES20.GL\_DEPTH\_ATTACHMENT, GLES20.GL\_TEXTURE\_2D, renderTextureId[0], 0);

}

/\*отрисовка теней\*/

private void draw\_shadows()

{

GLES20.glBindFramebuffer(GLES20.GL\_FRAMEBUFFER, fboId[0]);

GLES20.glViewport(0, 0, mShadowMapWidth, mShadowMapHeight);

GLES20.glClearColor(1f, 1f, 1f, 1.0f);

GLES20.glClear(GLES20.GL\_DEPTH\_BUFFER\_BIT | GLES20.GL\_COLOR\_BUFFER\_BIT);

GLES20.glUseProgram(mDepthMapProgram.getProgram());

float[] tempResultMatrix = new float[16];

Matrix.multiplyMM(mLightMvpMatrix, 0, mLightViewMatrix, 0, mModelMatrix, 0);

Matrix.multiplyMM(tempResultMatrix, 0, mLightProjectionMatrix, 0, mLightMvpMatrix, 0);

System.arraycopy(tempResultMatrix, 0, mLightMvpMatrix, 0, 16);

GLES20.glUniformMatrix4fv(shadow\_mvpMatrixUniform, 1, false, mLightMvpMatrix, 0);

Cup.render(shadow\_positionAttribute, 0, 0, true);

Orange.render(shadow\_positionAttribute, 0, 0, true);

Apple.render(shadow\_positionAttribute, 0, 0, true);

Torch.render(shadow\_positionAttribute, 0, 0, true);

Cabbage.render(shadow\_positionAttribute, 0, 0, true);

Table.render(shadow\_positionAttribute, 0, 0, true);

Beets.render(shadow\_positionAttribute, 0, 0, true);

}

/\*отрисовка объектов\*/

private void draw\_picture()

{

GLES20.glBindFramebuffer(GLES20.GL\_FRAMEBUFFER, 0);

GLES20.glClear(GLES20.GL\_COLOR\_BUFFER\_BIT | GLES20.GL\_DEPTH\_BUFFER\_BIT);

GLES20.glUseProgram(mActiveProgram);

GLES20.glViewport(0, 0, mDisplayWidth, mDisplayHeight);

GLES20.glUniform1f(scene\_mapStepXUniform, (float) (1.0 / mShadowMapWidth));

GLES20.glUniform1f(scene\_mapStepYUniform, (float) (1.0 / mShadowMapHeight));

float[] tempResultMatrix = new float[16];

float[] bias = new float[]

{

0.5f, 0.0f, 0.0f, 0.0f,

0.0f, 0.5f, 0.0f, 0.0f,

0.0f, 0.0f, 0.5f, 0.0f,

0.5f, 0.5f, 0.5f, 1.0f

};

float[] depthBiasMVP = new float[16];

Matrix.multiplyMM(tempResultMatrix, 0, mViewMatrix, 0, mModelMatrix, 0);

System.arraycopy(tempResultMatrix, 0, mMVMatrix, 0, 16);

GLES20.glUniformMatrix4fv(scene\_mvMatrixUniform, 1, false, mMVMatrix, 0);

Matrix.invertM(tempResultMatrix, 0, mMVMatrix, 0);

Matrix.transposeM(mNormalMatrix, 0, tempResultMatrix, 0);

GLES20.glUniformMatrix4fv(scene\_normalMatrixUniform, 1, false, mNormalMatrix, 0);

Matrix.multiplyMM(tempResultMatrix, 0, mProjectionMatrix, 0, mMVMatrix, 0);

System.arraycopy(tempResultMatrix, 0, mMVPMatrix, 0, 16);

GLES20.glUniformMatrix4fv(scene\_mvpMatrixUniform, 1, false, mMVPMatrix, 0);

Matrix.multiplyMV(mLightPosInEyeSpace, 0, mViewMatrix, 0, mActualLightPosition, 0);

GLES20.glUniform3f(scene\_lightPosUniform, mLightPosInEyeSpace[0], mLightPosInEyeSpace[1], mLightPosInEyeSpace[2]);

Matrix.multiplyMM(depthBiasMVP, 0, bias, 0, mLightMvpMatrix, 0);

System.arraycopy(depthBiasMVP, 0, mLightMvpMatrix, 0, 16);

GLES20.glUniformMatrix4fv(scene\_shadowProjMatrixUniform, 1, false, mLightMvpMatrix, 0);

Cup.render(scene\_positionAttribute, scene\_normalAttribute, scene\_colorAttribute, false);

Orange.render(scene\_positionAttribute, scene\_normalAttribute, scene\_colorAttribute, false);

Apple.render(scene\_positionAttribute, scene\_normalAttribute, scene\_colorAttribute, false);

Cabbage.render(scene\_positionAttribute, scene\_normalAttribute, scene\_colorAttribute, false);

Table.render(scene\_positionAttribute, scene\_normalAttribute, scene\_colorAttribute, false);

Torch.render(scene\_positionAttribute, scene\_normalAttribute, scene\_colorAttribute, false);

Beets.render(scene\_positionAttribute, scene\_normalAttribute, scene\_colorAttribute, false);

}

}

**TableObjects.java – класс объектов**

package com.example.curs\_work;

import java.util.List;

import java.util.Scanner;

import java.nio.ByteOrder;

import java.io.IOException;

import java.nio.ByteBuffer;

import java.nio.FloatBuffer;

import java.nio.ShortBuffer;

import java.util.ArrayList;

import android.opengl.GLES20;

import android.content.Context;

class TableObjects

{

private FloatBuffer normalBuffer;

private FloatBuffer verticesBuffer;

private final FloatBuffer colorBuffer;

private ShortBuffer facesVertexBuffer;

private ShortBuffer facesNormalBuffer;

private final List<String> facesList;

/\*создание матриц объектов\*/

TableObjects(Context c, float[] color, String ObjName)

{

facesList = new ArrayList<>();

List<String> verticesList = new ArrayList<>();

List<String> normalList = new ArrayList<>();

try

{

Scanner scanner = new Scanner(c.getAssets().open(ObjName));

while (scanner.hasNextLine())

{

String line = scanner.nextLine();

if (line.startsWith("v "))

{

verticesList.add(line);

}

else if (line.startsWith("f "))

{

facesList.add(line);

}

else if (line.startsWith("vn "))

{

normalList.add(line);

}

}

ByteBuffer buffer1 = ByteBuffer.allocateDirect(verticesList.size() \* 3 \* 4);

buffer1.order(ByteOrder.nativeOrder());

verticesBuffer = buffer1.asFloatBuffer();

ByteBuffer buffer2 = ByteBuffer.allocateDirect(normalList.size() \* 3 \* 4);

buffer2.order(ByteOrder.nativeOrder());

normalBuffer = buffer2.asFloatBuffer();

ByteBuffer buffer3 = ByteBuffer.allocateDirect(facesList.size() \* 3 \* 2);

buffer3.order(ByteOrder.nativeOrder());

facesVertexBuffer = buffer3.asShortBuffer();

ByteBuffer buffer4 = ByteBuffer.allocateDirect(facesList.size() \* 3 \* 2);

buffer4.order(ByteOrder.nativeOrder());

facesNormalBuffer = buffer4.asShortBuffer();

for (String vertex : verticesList)

{

String[] coords = vertex.split(" ");

float x = Float.parseFloat(coords[1]);

float y = Float.parseFloat(coords[2]);

float z = Float.parseFloat(coords[3]);

verticesBuffer.put(x);

verticesBuffer.put(y);

verticesBuffer.put(z);

}

verticesBuffer.position(0);

for (String vertex : normalList)

{

String[] coords = vertex.split(" ");

float x = Float.parseFloat(coords[1]);

float y = Float.parseFloat(coords[2]);

float z = Float.parseFloat(coords[3]);

normalBuffer.put(x);

normalBuffer.put(y);

normalBuffer.put(z);

}

normalBuffer.position(0);

for (String face : facesList)

{

String[] vertexIndices = face.split(" ");

String[] coord1 = vertexIndices[1].split("//");

String[] coord2 = vertexIndices[2].split("//");

String[] coord3 = vertexIndices[3].split("//");

short vertex1 = Short.parseShort(coord1[0]);

short vertex2 = Short.parseShort(coord2[0]);

short vertex3 = Short.parseShort(coord3[0]);

facesVertexBuffer.put((short) (vertex1 - 1));

facesVertexBuffer.put((short) (vertex2 - 1));

facesVertexBuffer.put((short) (vertex3 - 1));

vertex1 = Short.parseShort(coord1[1]);

vertex2 = Short.parseShort(coord2[1]);

vertex3 = Short.parseShort(coord3[1]);

facesNormalBuffer.put((short) (vertex1 - 1));

facesNormalBuffer.put((short) (vertex2 - 1));

facesNormalBuffer.put((short) (vertex3 - 1));

}

facesVertexBuffer.position(0);

facesNormalBuffer.position(0);

verticesList.clear();

normalList.clear();

scanner.close();

}

catch (IOException e)

{

e.printStackTrace();

}

float[] colorData = new float[facesList.size() \* 4];

for (int v = 0; v < facesList.size(); v++)

{

colorData[4 \* v] = color[0];

colorData[4 \* v + 1] = color[1];

colorData[4 \* v + 2] = color[2];

colorData[4 \* v + 3] = color[3];

}

ByteBuffer bColor = ByteBuffer.allocateDirect(colorData.length \* 4);

bColor.order(ByteOrder.nativeOrder());

colorBuffer = bColor.asFloatBuffer();

colorBuffer.put(colorData).position(0);

}

/\*функция отрисовки объекта\*/

void render(int positionAttribute, int normalAttribute, int colorAttribute, boolean onlyPosition)

{

colorBuffer.position(0);

normalBuffer.position(0);

verticesBuffer.position(0);

facesVertexBuffer.position(0);

facesNormalBuffer.position(0);

GLES20.glVertexAttribPointer(positionAttribute, 3, GLES20.GL\_FLOAT, false,

0, verticesBuffer);

GLES20.glEnableVertexAttribArray(positionAttribute);

if (!onlyPosition)

{

GLES20.glVertexAttribPointer(normalAttribute, 3, GLES20.GL\_FLOAT, false,

0, normalBuffer);

GLES20.glEnableVertexAttribArray(normalAttribute);

GLES20.glVertexAttribPointer(colorAttribute, 4, GLES20.GL\_FLOAT, false,

0, colorBuffer);

GLES20.glEnableVertexAttribArray(colorAttribute);

}

GLES20.glDrawElements(GLES20.GL\_TRIANGLES, facesList.size() \* 3,

GLES20.GL\_UNSIGNED\_SHORT, facesVertexBuffer);

}

}

**ShaderWorks.java – класс работы с шейдерами**

package com.example.curs\_work;

import android.content.Context;

import android.opengl.GLES20;

class ShadersWork

{

private int mProgram;

private String mVertexS, mFragmentS;

ShadersWork(String vID, String fID, Context context)

{

this.mVertexS = vID;

this.mFragmentS = fID;

if (createProgram() != 1)

{

throw new RuntimeException("Error at creating shaders");

}

}

/\*создание шейдеров\*/

private int createProgram()

{

int mVertexShader = loadShader(GLES20.GL\_VERTEX\_SHADER, mVertexS);

if (mVertexShader == 0)

{

return 0;

}

int mPixelShader = loadShader(GLES20.GL\_FRAGMENT\_SHADER, mFragmentS);

if (mPixelShader == 0)

{

return 0;

}

mProgram = GLES20.glCreateProgram();

if (mProgram != 0)

{

GLES20.glAttachShader(mProgram, mVertexShader);

GLES20.glAttachShader(mProgram, mPixelShader);

GLES20.glLinkProgram(mProgram);

int[] linkStatus = new int[1];

GLES20.glGetProgramiv(mProgram, GLES20.GL\_LINK\_STATUS, linkStatus, 0);

if (linkStatus[0] != GLES20.GL\_TRUE)

{

GLES20.glDeleteProgram(mProgram);

mProgram = 0;

return 0;

}

}

else

{

return -1;

}

return 1;

}

/\*загрузка шейдеров\*/

private int loadShader(int shaderType, String source)

{

int shader = GLES20.glCreateShader(shaderType);

if (shader != 0)

{

GLES20.glShaderSource(shader, source);

GLES20.glCompileShader(shader);

int[] compiled = new int[1];

GLES20.glGetShaderiv(shader, GLES20.GL\_COMPILE\_STATUS, compiled, 0);

if (compiled[0] == 0)

{

GLES20.glDeleteShader(shader);

shader = 0;

}

}

return shader;

}

int getProgram()

{

return mProgram;

}

}

**Пример файла объекта (orange.obj):**

# Blender v2.91.0 OBJ File: 'obj.blend'

# www.blender.org

o Cube.001\_Cube.000

v -0.155856 2.295681 -0.820042

v 0.011351 2.155632 -0.872294

v -0.109724 2.134090 -0.776965

v 0.197406 2.295681 -0.848369

v 0.233988 2.155632 -0.664387

v 0.150186 2.134577 -0.797807

v 0.193287 2.295681 -0.493998

v 0.026080 2.155632 -0.441750

v 0.147156 2.134090 -0.537079

v -0.159974 2.295681 -0.465672

v -0.196556 2.155632 -0.649657

v -0.112755 2.129674 -0.516237

v 0.018716 2.124851 -0.657021

v 0.075705 2.110069 -0.718048

v 0.069189 2.108125 -0.609888

v 0.018920 2.443954 -0.657238

v 0.091189 2.475540 -0.736916

v -0.060753 2.477057 -0.731843

v -0.196556 2.419202 -0.649655

v -0.112755 2.459783 -0.516235

v 0.026080 2.419202 -0.441748

v 0.147156 2.457272 -0.537077

v 0.233988 2.419202 -0.664385

v 0.150186 2.451723 -0.797805

v 0.011351 2.419202 -0.872293

v -0.109724 2.457272 -0.776963

v -0.231678 2.295681 -0.648455

v 0.269110 2.295681 -0.665587

v -0.040190 2.108503 -0.593942

v -0.037833 2.110148 -0.709830

v -0.057987 2.479521 -0.577172

v 0.106077 2.474906 -0.576051

v 0.010150 2.295681 -0.907415

v 0.027282 2.295681 -0.406626

v 0.012119 2.510610 -0.666511

v 0.020032 2.445387 -0.668900

v 0.014351 2.438457 -0.664427

v 0.028831 2.510610 -0.667852

v 0.030478 2.445387 -0.659145

v 0.026546 2.433460 -0.665406

v 0.028636 2.510610 -0.651088

v 0.020723 2.445387 -0.648699

v 0.026404 2.438457 -0.653172

v 0.011925 2.510610 -0.649749

v 0.010277 2.445387 -0.658454

v 0.014209 2.433460 -0.652194

v 0.020378 2.424695 -0.658800

v 0.024993 2.579627 -0.685686

v 0.007297 2.579590 -0.684334

v 0.014779 2.593830 -0.681855

v 0.000860 2.578928 -0.674761

v 0.005043 2.597789 -0.673409

v 0.014725 2.584405 -0.661815

v 0.023294 2.585910 -0.669396

v 0.030500 2.570044 -0.671522

v 0.016635 2.564568 -0.684469

v 0.006536 2.510610 -0.658327

v 0.034219 2.510610 -0.659274

v 0.019904 2.510610 -0.672642

v 0.020851 2.510610 -0.644958

vn -0.5378 -0.3571 -0.7637

vn 0.8096 -0.3782 -0.4489

vn 0.5378 -0.3571 0.7637

vn -0.8181 -0.3674 0.4424

vn -0.2853 -0.9578 -0.0344

vn 0.0221 0.9224 0.3857

vn -0.8339 0.3670 0.4122

vn 0.5072 0.3542 0.7857

vn 0.8209 0.3851 -0.4217

vn -0.5072 0.3542 -0.7857

vn -0.8861 0.2481 -0.3915

vn 0.8861 0.2481 0.3915

vn -0.3851 0.9228 0.0133

vn 0.0534 -0.9573 0.2842

vn 0.8922 -0.2204 0.3942

vn -0.8922 -0.2204 -0.3942

vn -0.2868 -0.9580 -0.0070

vn -0.2843 -0.9586 -0.0130

vn -0.2822 -0.9592 -0.0194

vn 0.0231 0.9185 0.3947

vn 0.0214 0.9189 0.3939

vn 0.0220 0.9188 0.3942

vn 0.2905 0.2591 -0.9211

vn -0.2905 0.2591 0.9211

vn 0.3807 0.9242 -0.0316

vn -0.0206 -0.9611 -0.2753

vn -0.2926 -0.2302 0.9281

vn 0.2926 -0.2302 -0.9281

vn -0.5864 -0.0415 -0.8090

vn 0.8795 -0.0411 -0.4742

vn 0.5864 -0.0415 0.8090

vn -0.8795 -0.0411 0.4742

vn 0.8656 -0.4600 0.1979

vn -0.0750 0.2091 -0.9750

vn -0.7467 0.1187 0.6545

vn 0.6228 0.2263 0.7489

vn 0.9190 -0.0500 -0.3911

vn -0.4218 -0.2543 -0.8703

vn -0.8086 -0.2001 -0.5532

vn -0.8073 -0.2001 -0.5551

vn -0.8094 -0.2001 -0.5521

vn 0.8147 0.1655 0.5557

vn -0.8166 0.2194 -0.5339

vn 0.2303 -0.3874 0.8927

vn 0.8252 -0.0462 0.5629

vn -0.8252 -0.0462 -0.5629

vn -0.9064 -0.3874 -0.1687

vn 0.3629 0.7461 0.5583

vn 0.4666 -0.1623 -0.8695

vn -0.4668 0.1599 0.8698

vn 0.7828 0.6037 -0.1512

vn -0.2567 -0.4599 -0.8500

vn -0.4723 -0.0496 0.8801

vn 0.4722 -0.0496 -0.8801

vn 0.2853 -0.9580 -0.0280

vn 0.2590 -0.9658 -0.0084

vn -0.0456 -0.9558 -0.2903

vn 0.0175 -0.9679 0.2508

vn 0.0286 0.9234 -0.3827

vn -0.3652 0.9302 0.0375

vn 0.4009 0.9159 -0.0218

vn -0.0037 0.9241 -0.3822

vn -0.0149 0.9262 -0.3767

vn -0.0168 0.9236 -0.3830

vn -0.4543 -0.2204 -0.8632

vn 0.9059 -0.2302 -0.3554

vn 0.4543 -0.2204 0.8632

vn -0.9059 -0.2302 0.3554

vn -0.8991 0.2591 0.3527

vn 0.4512 0.2481 0.8572

vn 0.8991 0.2591 -0.3527

vn -0.4512 0.2481 -0.8572

vn -0.8185 0.3542 -0.4523

vn 0.8185 0.3542 0.4523

vn -0.3801 0.9240 -0.0423

vn -0.3958 0.9183 -0.0076

vn 0.0317 -0.9586 0.2830

vn 0.0260 -0.9581 0.2853

vn 0.0263 -0.9583 0.2847

vn 0.0376 -0.9591 0.2806

vn 0.7987 -0.3571 0.4843

vn -0.7987 -0.3571 -0.4843

vn -0.2865 -0.9581 -0.0065

vn -0.2872 -0.9573 -0.0339

vn 0.0741 0.9273 0.3670

vn 0.0210 0.9190 0.3937

vn 0.3647 0.3851 -0.8478

vn -0.3542 0.3670 0.8602

vn 0.3972 0.9177 -0.0035

vn 0.3694 0.9292 -0.0055

vn -0.0534 -0.9573 -0.2842

vn 0.0084 -0.9580 -0.2865

vn -0.3854 -0.3674 0.8464

vn 0.3925 -0.3782 -0.8384

vn -0.6180 -0.0462 -0.7848

vn 0.8463 -0.0500 -0.5303

vn 0.8460 -0.0562 -0.5302

vn 0.8458 -0.0496 -0.5313

vn 0.6180 -0.0462 0.7848

vn -0.8457 -0.0496 0.5313

vn 0.9063 -0.3874 0.1688

vn -0.0761 -0.0383 -0.9964

vn -0.8454 0.0575 0.5311

vn 0.6028 0.2249 0.7655

vn 0.8454 -0.0565 -0.5311

vn -0.6055 -0.2052 -0.7689

vn -0.8066 -0.2001 -0.5562

vn 0.6846 0.2177 0.6956

vn -0.5034 0.3980 -0.7669

vn 0.2567 -0.4600 0.8500

vn 0.8472 -0.0415 0.5297

vn -0.8472 -0.0415 -0.5297

vn -0.8656 -0.4600 -0.1980

vn 0.5061 0.8415 0.1890

vn 0.2720 -0.2266 -0.9352

vn -0.6712 0.1444 0.7271

vn 0.9050 0.4201 -0.0676

vn -0.2303 -0.3874 -0.8927

vn -0.4130 -0.0411 0.9098

vn 0.4129 -0.0411 -0.9098

vn 0.2872 -0.9573 0.0339

vn 0.2768 -0.9609 -0.0006

vn -0.0153 0.9217 -0.3876

vn -0.0040 0.9240 -0.3823

s 1

f 1//1 2//1 3//1

f 4//2 5//2 6//2

f 7//3 8//3 9//3

f 10//4 11//4 12//4

f 13//5 14//5 15//5

f 16//6 17//6 18//6

f 19//7 10//7 20//7

f 21//8 7//8 22//8

f 23//9 4//9 24//9

f 25//10 1//10 26//10

f 19//11 1//11 27//11

f 23//12 7//12 28//12

f 18//13 26//13 19//13

f 15//14 9//14 8//14

f 7//15 5//15 28//15

f 1//16 11//16 27//16

f 29//17 11//18 30//19

f 31//20 21//21 32//22

f 25//23 4//23 33//23

f 21//24 10//24 34//24

f 23//25 17//25 32//25

f 2//26 14//26 30//26

f 10//27 8//27 34//27

f 4//28 2//28 33//28

f 35//29 36//29 37//29

f 38//30 39//30 40//30

f 41//31 42//31 43//31

f 44//32 45//32 46//32

f 40//33 43//33 47//33

f 48//34 49//34 50//34

f 51//35 44//35 52//35

f 53//36 41//36 54//36

f 55//37 38//37 48//37

f 56//38 35//38 49//38

f 51//39 35//40 57//41

f 55//42 41//42 58//42

f 52//43 49//43 51//43

f 46//44 43//44 42//44

f 41//45 39//45 58//45

f 35//46 45//46 57//46

f 37//47 46//47 45//47

f 54//48 52//48 53//48

f 56//49 38//49 59//49

f 53//50 44//50 60//50

f 54//51 48//51 50//51

f 37//52 40//52 47//52

f 44//53 42//53 60//53

f 38//54 36//54 59//54

f 14//55 6//55 5//55

f 29//56 30//56 13//56

f 15//57 29//57 13//57

f 30//58 14//58 13//58

f 31//59 32//59 16//59

f 32//60 17//60 16//60

f 18//61 31//61 16//61

f 17//62 24//63 25//64

f 1//65 33//65 2//65

f 4//66 28//66 5//66

f 7//67 34//67 8//67

f 10//68 27//68 11//68

f 19//69 27//69 10//69

f 21//70 34//70 7//70

f 23//71 28//71 4//71

f 25//72 33//72 1//72

f 19//73 26//73 1//73

f 23//74 22//74 7//74

f 19//75 20//75 31//75

f 31//76 18//76 19//76

f 8//77 12//78 29//79

f 29//79 15//80 8//77

f 7//81 9//81 5//81

f 1//82 3//82 11//82

f 12//83 11//18 29//17

f 11//84 3//84 30//84

f 20//85 21//85 31//85

f 21//21 22//86 32//22

f 25//87 24//87 4//87

f 21//88 20//88 10//88

f 32//89 22//89 23//89

f 23//90 24//90 17//90

f 30//91 3//91 2//91

f 2//92 6//92 14//92

f 10//93 12//93 8//93

f 4//94 6//94 2//94

f 35//95 59//95 36//95

f 38//96 58//97 39//98

f 41//99 60//99 42//99

f 44//100 57//100 45//100

f 40//101 39//101 43//101

f 48//102 56//102 49//102

f 51//103 57//103 44//103

f 53//104 60//104 41//104

f 55//105 58//97 38//96

f 56//106 59//106 35//106

f 51//39 49//107 35//40

f 55//108 54//108 41//108

f 52//109 50//109 49//109

f 46//110 47//110 43//110

f 41//111 43//111 39//111

f 35//112 37//112 45//112

f 37//113 47//113 46//113

f 54//114 50//114 52//114

f 56//115 48//115 38//115

f 53//116 52//116 44//116

f 54//117 55//117 48//117

f 37//118 36//118 40//118

f 44//119 46//119 42//119

f 38//120 40//120 36//120

f 9//121 15//121 5//121

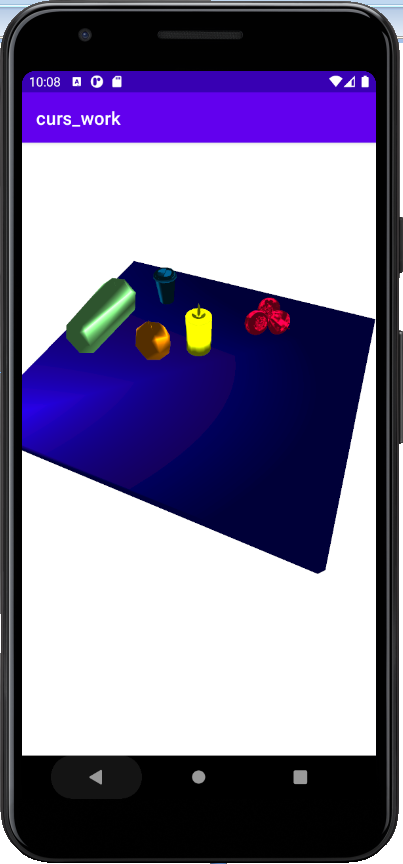
f 15//122 14//122 5//122

f 26//123 18//124 25//64

f 18//124 17//62 25//64

**Результаты работы**

На столе находятся свеча, кабачок, апельсин, три свеклы и стакан

****