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 №6

по дисциплине

«Программирование мобильных устройств»

Выполнил:   
студент гр. ИП-813

Бурдуковский И.А.

Проверила:

Павлова У.В.

Новосибирск 2021

Оглавление

[Задание 3](#_Toc36477188)

[Выполнение 3](#_Toc36477189)

[Листинг проекта 6](#_Toc36477190)

# Задание

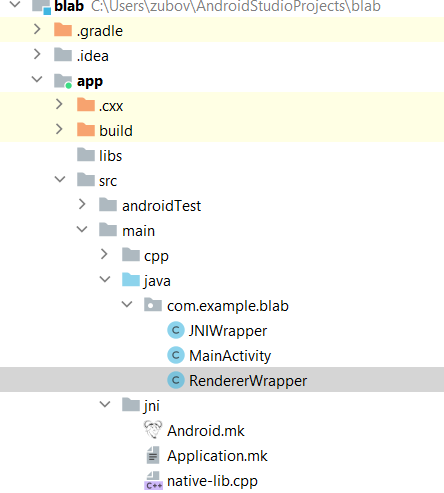
Написать программу, рисующую куб с текстурой. Вся прорисовка должна быть реализована в JNI.

# Выполнение

1. Скачиваем и распаковываем NDK (например, в папку d:\NDK\android-ndk-r11c\).
2. Создаем sdk проект, выбирая максимальный sdkVersion.



1. Добавляем ему папку jni и файл native-lib.c, Android.mk,Application.mk



1. Заполняем файлы:

**Android.mk**

LOCAL\_PATH:= **$**(**call** my-dir)   
  
**include $**(CLEAR\_VARS)  
  
LOCAL\_MODULE := native-lib  
LOCAL\_SRC\_FILES := native-lib.cpp  
LOCAL\_LDLIBS := -ldl -lGLESv1\_CM -llog  
  
**include $**(BUILD\_SHARED\_LIBRARY)

**Application.cpp**

APP\_OPTIM := release  
APP\_PLATFORM := android-16  
APP\_CPPFLAGS += -frtti  
APP\_CPPFLAGS += -fexceptions  
APP\_ABI := all  
APP\_MODULES := native-lib

**Native-lib.cpp**

#include **<jni.h>**#include **<string>**#include **<GLES2/gl2.h>**#include **<GLES/gl.h>****extern "C"****JNIEXPORT void JNICALL**Java\_com\_example\_blab\_JNIWrapper\_onsurfacecreated(**\_\_unused** JNIEnv \*env, **\_\_unused** jclass cls) {  
 glMatrixMode(**GL\_PROJECTION**);  
 glLoadIdentity();  
 glOrthof(-8, 8, -8, 8, -8, 8);  
 glEnable(**GL\_DEPTH\_TEST**);  
 glClearColor(1, 1, 0, 0);  
 glClearDepthf(1);  
 glMatrixMode(**GL\_MODELVIEW**);  
 glLoadIdentity();  
}  
  
**extern "C"  
JNIEXPORT void JNICALL**Java\_com\_example\_blab\_JNIWrapper\_onsurfacechanged(**\_\_unused** JNIEnv \*env, **\_\_unused** jclass cls,  
 **\_\_unused** jint width,  
 **\_\_unused** jint height) {  
}  
  
GLfloat a[12] = {  
 -1, 1, 0,  
 -1, -1, 0,  
 1, -1, 0,  
 1, 1, 0  
};  
  
GLfloat texCoords[8] = {  
 0.0f, 1.0f,  
 0.0f, 0.0f,  
 1.0f, 0.0f,  
 1.0f, 1.0f  
};  
**int** angle = 0;  
**extern "C"  
  
JNIEXPORT void JNICALL**Java\_com\_example\_blab\_JNIWrapper\_ondrawframe(**\_\_unused** JNIEnv \*env, **\_\_unused** jclass cls) {  
 glClearColor(0.4, 0.4, 0.9, 1);  
 glClear(**GL\_COLOR\_BUFFER\_BIT** | **GL\_DEPTH\_BUFFER\_BIT**);  
 glLoadIdentity();  
 glTranslatef(0, 0, -1);  
 glScalef(4, 2.5, 1);  
 glColor4f(1, 1, 1, 1);  
 glEnableClientState(**GL\_VERTEX\_ARRAY**);  
 glEnableClientState(**GL\_TEXTURE\_COORD\_ARRAY**);  
 glEnable(**GL\_TEXTURE\_2D**);  
angle = (angle == 360) ? 0 : angle + 1;  
 glRotatef(angle, 1, 0.5, 0.1);

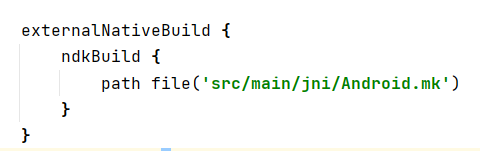
glPushMatrix();glTranslatef(0, 0, 1);  
 glVertexPointer(3, **GL\_FLOAT**, 0, a);  
 glTexCoordPointer(2, **GL\_FLOAT**, 0, texCoords);  
 glDrawArrays(**GL\_TRIANGLE\_FAN**, 0, 4);  
 glPopMatrix();  
glPushMatrix();glTranslatef(0, 0, -1);  
 glVertexPointer(3, **GL\_FLOAT**, 0, a);  
 glTexCoordPointer(2, **GL\_FLOAT**, 0, texCoords);  
 glDrawArrays(**GL\_TRIANGLE\_FAN**, 0, 4);  
 glPopMatrix();  
glPushMatrix();glTranslatef(0, 1, 0);  
 glRotatef(90, 1, 0, 0);  
 glVertexPointer(3, **GL\_FLOAT**, 0, a);  
 glTexCoordPointer(2, **GL\_FLOAT**, 0, texCoords);  
 glDrawArrays(**GL\_TRIANGLE\_FAN**, 0, 4);  
 glPopMatrix();  
  
glPushMatrix();glRotatef(90, 1, 0, 0);  
 glTranslatef(0, 0, 1);  
 glVertexPointer(3, GL\_FLOAT, 0, a);  
 glTexCoordPointer(2, GL\_FLOAT, 0, texCoords);  
 glDrawArrays(GL\_TRIANGLE\_FAN, 0, 4);  
 glPopMatrix();  
glPushMatrix();glRotatef(90, 0, 1, 0);  
 glTranslatef(0, 0, -1);  
 glVertexPointer(3, **GL\_FLOAT**, 0, a);  
 glTexCoordPointer(2, **GL\_FLOAT**, 0, texCoords);  
 glDrawArrays(**GL\_TRIANGLE\_FAN**, 0, 4);  
 glPopMatrix();  
glPushMatrix();glRotatef(90, 0, 1, 0);  
 glTranslatef(0, 0, 1);  
 glVertexPointer(3, **GL\_FLOAT**, 0, a);  
 glTexCoordPointer(2, **GL\_FLOAT**, 0, texCoords);  
 glDrawArrays(**GL\_TRIANGLE\_FAN**, 0, 4);  
 glPopMatrix();  
 glDisable(**GL\_TEXTURE\_2D**);  
 glDisableClientState(**GL\_TEXTURE\_COORD\_ARRAY**);  
 glDisableClientState(**GL\_VERTEX\_ARRAY**);}

1. Исправим\добавим в файле local.properties



1. Исправим в файле build.gradle



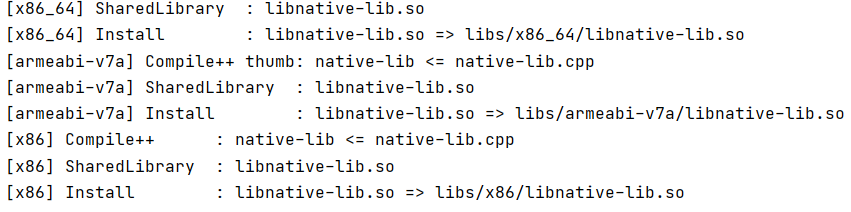


1. Создадим файл в папке проекта a.bat(В одну строку)

C:\Users\zubov\AndroidStudioProjects\NDK\android-ndk-r23b\ndk-build.cmd NDK\_APPLICATION\_MK=C:\Users\zubov\AndroidStudioProjects\blab\app\src\main\jni\Application.mk NDK\_PROJECT\_PATH=C:\Users\zubov\AndroidStudioProjects\blab\app\src\main APP\_BUILD\_SCRIPT=C:\Users\zubov\AndroidStudioProjects\blab\app\src\main\jni\Android.mk

1. Запустим a.bat, если исходники верны, то библиотеки создадутся в папке src/main/libs

Примечание: в терминале надо перейти в папку, где находится a.bat( Компиляция a.bat)



1. Оставляем одну библиотек и компилируем проект.

**Результат**:



# Листинг проекта

MainActivity.java

package com.example.lab6;  
  
import android.app.Activity;  
  
import android.app.ActivityManager;  
  
import android.content.Context;  
  
import android.content.pm.ConfigurationInfo;  
  
import android.opengl.GLSurfaceView;  
  
import android.os.Build;  
  
import android.os.Bundle;  
  
import android.widget.Toast;  
  
public class MainActivity extends Activity {  
 private GLSurfaceView glSurfaceView;  
 private boolean rendererSet;  
 private boolean isProbablyEmulator() {  
 return Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.ICE\_CREAM\_SANDWICH\_MR1  
 && (Build.FINGERPRINT.startsWith("generic")  
 || Build.FINGERPRINT.startsWith("unknown")  
 || Build.MODEL.contains("google\_sdk")  
 || Build.MODEL.contains("Emulator")  
 || Build.MODEL.contains("Android SDK built for x86"));  
  
 }  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 ActivityManager activityManager  
 = (ActivityManager) getSystemService(Context.ACTIVITY\_SERVICE);  
 ConfigurationInfo configurationInfo = activityManager.getDeviceConfigurationInfo();  
 final boolean supportsEs2 =  
 configurationInfo.reqGlEsVersion >= 0x20000 || isProbablyEmulator();  
 if (supportsEs2) {  
 glSurfaceView = new GLSurfaceView(this);  
 if (isProbablyEmulator()) {  
  
// Avoids crashes on startup with some emulator images.  
 glSurfaceView.setEGLConfigChooser(8, 8, 8, 8, 16, 0);  
 }  
  
 glSurfaceView.setRenderer(new RendererWrapper(this));  
 rendererSet = true;  
 setContentView(glSurfaceView);  
  
 } else {  
  
// Should never be seen in production, since the manifest filters  
// unsupported devices.  
 Toast.makeText(this, "This device does not support OpenGL ES 2.0.",  
 Toast.LENGTH\_LONG).show();  
 return;  
 }  
 }  
  
 @Override  
 protected void onPause() {  
 super.onPause();  
 if (rendererSet) {  
 glSurfaceView.onPause();  
 }  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 if (rendererSet) {  
 glSurfaceView.onResume();  
 }  
 }  
}

RendererWrapper.java

package com.example.lab6;  
  
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;  
  
class RendererWrapper implements GLSurfaceView.Renderer {  
  
 static public int[] texture\_name = {  
 R.drawable.img  
 };  
  
 Context c;  
 public RendererWrapper(Context context) {  
 c = context;  
 }  
  
 static public int[] textures = new int [texture\_name.length];  
 private void loadGLTexture(GL10 gl) {  
 gl.glGenTextures(1, 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) {  
 loadGLTexture(gl);  
 JNIWrapper.onsurfacecreated();  
 }  
  
 @Override  
 public void onSurfaceChanged(GL10 gl, int width, int height) {  
 JNIWrapper.onsurfacechanged(width, height);  
 }  
  
 @Override  
 public void onDrawFrame(GL10 gl) {  
 JNIWrapper.ondrawframe();  
 }  
}

JNIWrapper.java

package com.example.lab6;  
  
public class JNIWrapper {  
//System.LoadLibrary(String libname) – статический метод, который загружает общую библиотеку  
// из файловой системы в память  
// и делает ее экспортированные функции доступными для нашего кода Java.  
 static {  
 System.loadLibrary("native-lib");  
 }  
 //любой метод, помеченный как native, должен быть реализован в собственной общей библиотеке.  
 public static native void onsurfacecreated();  
 public static native void onsurfacechanged(int width, int height);  
 public static native void ondrawframe();  
}

Android.mk

LOCAL\_PATH:= $(call my-dir)   
  
include $(CLEAR\_VARS)  
  
  
LOCAL\_MODULE := native-lib  
LOCAL\_SRC\_FILES := native-lib.cpp  
LOCAL\_LDLIBS := -ldl -lGLESv1\_CM -llog  
  
include $(BUILD\_SHARED\_LIBRARY)  
#файл Android.mk, который будет собирать проект;  
#LOCAL\_PATH := $(call my-dir) – функция call my-dir возвращает путь папки в которой вызывается файл;  
#include $(CLEAR\_VARS) – очищает переменные которые использовались до этого кроме LOCAL\_PATH.  
#Это необходимо так как все переменные являются глобальными, потому что сборка происходит в контексте одного GNU Make;  
#LOCAL\_MODULE – имя выходного модуля.

Application.mk

APP\_OPTIM := release  
APP\_PLATFORM := android-16  
APP\_CPPFLAGS += -frtti  
APP\_CPPFLAGS += -fexceptions  
APP\_ABI := all  
APP\_MODULES := native-lib

native-lib.cpp

#include <jni.h>  
#include <string>  
#include <GLES2/gl2.h>  
#include <GLES/gl.h>  
//JNIEXPORT – помечает функцию в общей библиотеке как экспортируемую, чтобы она была включена в таблицу функций, и,  
//таким образом, JNI может найти ее  
//JNICALL – в сочетании с JNIEXPORT это гарантирует , что наши методы доступны для фреймворка JNI  
// JNIEnv – структура, содержащая методы, которые мы можем использовать наш собственный код для доступа к элементам Java  
extern "C"  
//указатель на текущий JNIEnv; , а также объект Java, к которому прикреплен метод  
JNIEXPORT void JNICALL  
  
Java\_com\_example\_lab6\_JNIWrapper\_onsurfacecreated(\_\_unused JNIEnv \*env, \_\_unused jclass cls) {  
 glMatrixMode(GL\_PROJECTION);  
 glLoadIdentity();  
 glOrthof(-8, 8, -8, 8, -8, 8);  
 glEnable(GL\_DEPTH\_TEST);  
 glClearColor(1, 1, 0, 0);  
 glClearDepthf(1);  
 glMatrixMode(GL\_MODELVIEW);  
 glLoadIdentity();  
}  
  
extern "C"  
JNIEXPORT void JNICALL  
  
Java\_com\_example\_lab6\_JNIWrapper\_onsurfacechanged(\_\_unused JNIEnv \*env, \_\_unused jclass cls,  
 \_\_unused jint width,  
 \_\_unused jint height) {  
}  
  
GLfloat a[12] = {  
 -1, 1, 0,  
 -1, -1, 0,  
 1, -1, 0,  
 1, 1, 0  
};  
  
GLfloat texCoords[8] = {  
 0.0f, 1.0f,  
 0.0f, 0.0f,  
 1.0f, 0.0f,  
 1.0f, 1.0f  
};  
int angle = 0;  
extern "C"  
  
JNIEXPORT void JNICALL  
Java\_com\_example\_lab6\_JNIWrapper\_ondrawframe(\_\_unused JNIEnv \*env, \_\_unused jclass cls) {  
 glClearColor(0.4, 0.4, 0.9, 1);  
 glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);  
 glLoadIdentity();  
 glTranslatef(0, 0, -1);  
 glScalef(4, 2.5, 1);  
 glColor4f(1, 1, 1, 1);  
 glEnableClientState(GL\_VERTEX\_ARRAY);  
 glEnableClientState(GL\_TEXTURE\_COORD\_ARRAY);  
 glEnable(GL\_TEXTURE\_2D);  
//glEnableClientState(GL\_COLOR\_ARRAY);  
 angle = (angle == 360) ? 0 : angle + 1;  
 glRotatef(angle, 1, 0.5, 0.1);  
  
//лицевая грань  
 glPushMatrix();  
//glColor4f(1, 1, 0, 1);  
 glTranslatef(0, 0, 1);  
 glVertexPointer(3, GL\_FLOAT, 0, a);  
 glTexCoordPointer(2, GL\_FLOAT, 0, texCoords);  
 glDrawArrays(GL\_TRIANGLE\_FAN, 0, 4);  
 glPopMatrix();  
  
//задняя  
 glPushMatrix();  
//glColor4f(1, 0, 1, 1);  
 glTranslatef(0, 0, -1);  
 glVertexPointer(3, GL\_FLOAT, 0, a);  
 glTexCoordPointer(2, GL\_FLOAT, 0, texCoords);  
 glDrawArrays(GL\_TRIANGLE\_FAN, 0, 4);  
 glPopMatrix();  
  
//верхняя  
 glPushMatrix();  
//glColor4f(1, 1, 1, 1)  
 glTranslatef(0, 1, 0);  
 glRotatef(90, 1, 0, 0);  
 glVertexPointer(3, GL\_FLOAT, 0, a);  
 glTexCoordPointer(2, GL\_FLOAT, 0, texCoords);  
 glDrawArrays(GL\_TRIANGLE\_FAN, 0, 4);  
 glPopMatrix();  
  
//нижняя  
 glPushMatrix();  
//glColor4f(0, 1, 1, 1);  
 glRotatef(90, 1, 0, 0);  
 glTranslatef(0, 0, 1);  
 glVertexPointer(3, GL\_FLOAT, 0, a);  
 glTexCoordPointer(2, GL\_FLOAT, 0, texCoords);  
 glDrawArrays(GL\_TRIANGLE\_FAN, 0, 4);  
 glPopMatrix();  
  
//левая  
 glPushMatrix();  
//glColor4f(1, 0, 0, 1);  
 glRotatef(90, 0, 1, 0);  
 glTranslatef(0, 0, -1);  
 glVertexPointer(3, GL\_FLOAT, 0, a);  
 glTexCoordPointer(2, GL\_FLOAT, 0, texCoords);  
 glDrawArrays(GL\_TRIANGLE\_FAN, 0, 4);  
 glPopMatrix();  
  
//правая  
 glPushMatrix();  
//glColor4f(0, 0, 1, 1);  
 glRotatef(90, 0, 1, 0);  
 glTranslatef(0, 0, 1);  
 glVertexPointer(3, GL\_FLOAT, 0, a);  
 glTexCoordPointer(2, GL\_FLOAT, 0, texCoords);  
 glDrawArrays(GL\_TRIANGLE\_FAN, 0, 4);  
 glPopMatrix();  
 glDisable(GL\_TEXTURE\_2D);  
 glDisableClientState(GL\_TEXTURE\_COORD\_ARRAY);  
 glDisableClientState(GL\_VERTEX\_ARRAY);  
//glDisableClientState(GL\_COLOR\_ARRAY);  
}