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

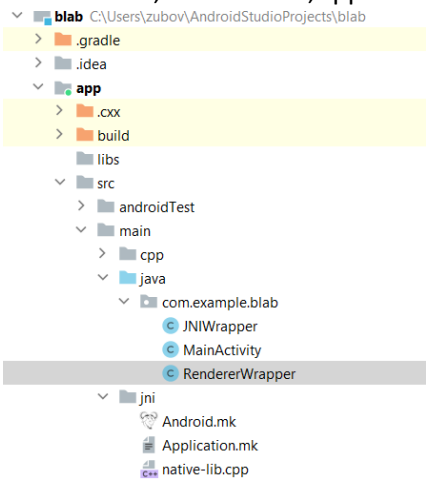
Написать программу, рисующую куб с текстурой. Вся прорисовка должна быть реализована в JNI.

Выполнение

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

Native C++

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



4. 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);
    }
```

5. Исправим\добавим в файле local.properties

```
ndk.dir=C\:\\Users\\zubov\\AndroidStudioProjects\\NDK\\android-ndk-r23b
```

6. Исправим в файле build.gradle

```
    minSdk 16

  externalNativeBuild {
      ndkBuild {
          path file('src/main/jni/Android.mk')
      }
  }
```

7. 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\j
ni\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
```

8. 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)

```
[x86_64] SharedLibrary  : libnative-lib.so
[x86_64] Install        : libnative-lib.so => libs/x86_64/libnative-lib.so
[armeabi-v7a] Compile++ thumb: native-lib <= native-lib.cpp
[armeabi-v7a] SharedLibrary  : libnative-lib.so
[armeabi-v7a] Install        : libnative-lib.so => libs/armeabi-v7a/libnative-lib.so
[x86] Compile++      : native-lib <= native-lib.cpp
[x86] SharedLibrary  : libnative-lib.so
[x86] Install        : libnative-lib.so => libs/x86/libnative-lib.so
```

9. 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);
}
```