HUST

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

TRƯỜNG ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

LẬP TRÌNH ỨNG DỤNG DI ĐỘNG

Mobile Application Programming

ET4710

PGS. TS. Đỗ Trọng Tuấn
Viện Điện tử Viễn thông * Đại học Bách Khoa Hà Nội

ONE LOVE. ONE FUTURE.

CHƯƠNG 5.

Lập trình đồ họa và hoạt hình

(Graphics, Multimedia, Gaming and Animation)

Chương 5

Lập trình đồ họa và hoạt hình
(Graphics, Multimedia, Gaming and Animation)

5.1. Cơ bản về đồ họa (Graphics Introduction)

5.2. Lập trình đa phương tiện (Multimedia programming)

5.3 Lập trình đồ họa hoạt hình và trò chơi (Animation and Game Programming)

Cơ bản về đồ họa (Graphics Introduction)

- Computer graphics are used for any kind of display for which there isn't a GUI component: charting, displaying pictures, and so on.
- Graphics tools are used to create GUI components as well as to draw shapes, lines, pictures, etc.
- Android is well provisioned for graphics, including a full implementation of OpenGL ES, a subset of OpenGL intended for smaller devices.

Source: "*Android Cookbook, Problems and Solutions for Android Developers*", Second Edition, by Ian F. Darwin

The Standard for Embedded Accelerated 3D Graphics

OpenGL® ES is a royalty-free, cross-platform API for rendering advanced 2D and 3D graphics on embedded and mobile systems - including consoles, phones, appliances and vehicles. It consists of a well-defined subset of desktop OpenGL suitable for low-power devices, and provides a flexible and powerful interface between software and graphics acceleration hardware.

Using a Custom Font

Problem

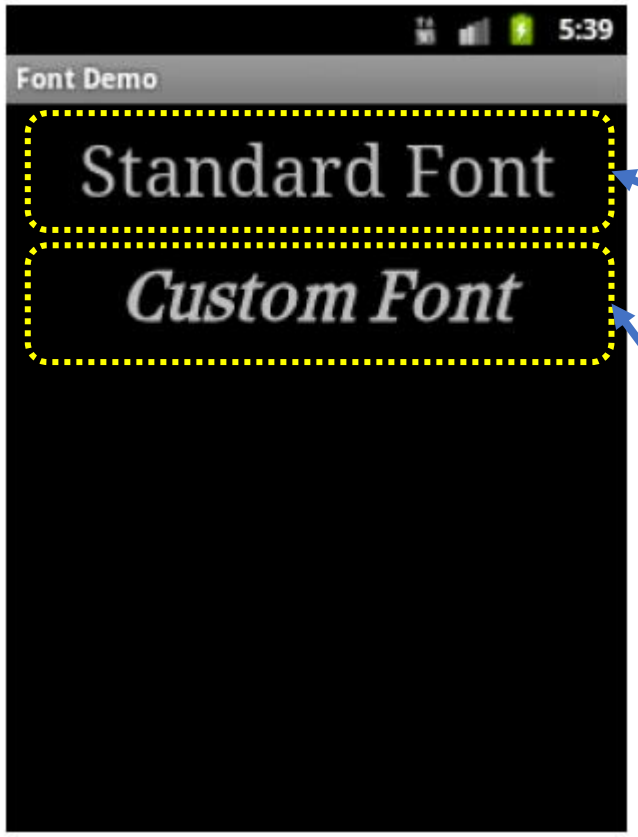
The range of fonts on Android devices is pretty small. You want something better.

Solution

Install a TTF or OTF version of your font in assets/fonts (creating this directory if necessary). In your code, create a typeface from the “asset” and call the View’s setTypeface() method. You’re done!

Source: "*Android Cookbook, Problems and Solutions for Android Developers”*, Second Edition, by Ian F. Darwin

XML layout with font specification




```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:id="@+id/PlainTextView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/plain"
        android:textSize="36sp"
        android:typeface="serif"
        android:padding="10sp"
        android:gravity="center"
        />
    <TextView
        android:id="@+id/FontView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/nicer"
        android:textSize="36sp"
        android:typeface="normal"
        android:padding="10sp"
        android:gravity="center"
        />
</LinearLayout>
```

Setting a custom font

```
public class FontDemo extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        TextView v = (TextView) findViewById(R.id.FontView); ❶
        Typeface t = Typeface.createFromAsset(getAssets(),  ❷
                "fonts/fontdemo.ttf");
        v.setTypeface(t, Typeface.BOLD_ITALIC);             ❸
    }
}
```

❶ Find the `View` you want to use your font in.

❷ Create a `Typeface` object from one of the `Typeface` class's static `create()` methods.

❸ Message the `Typeface` into the `View`'s `setTypeface()` method.

Drawing a Spinning Cube with OpenGL ES

Problem

You want to create a basic OpenGL ES application.

Solution

Create a GLSurfaceView and a custom Renderer that will draw a spinning cube.

Discussion

Android supports 3D graphics via the OpenGL ES API, a flavor of OpenGL specifically designed for embedded devices.

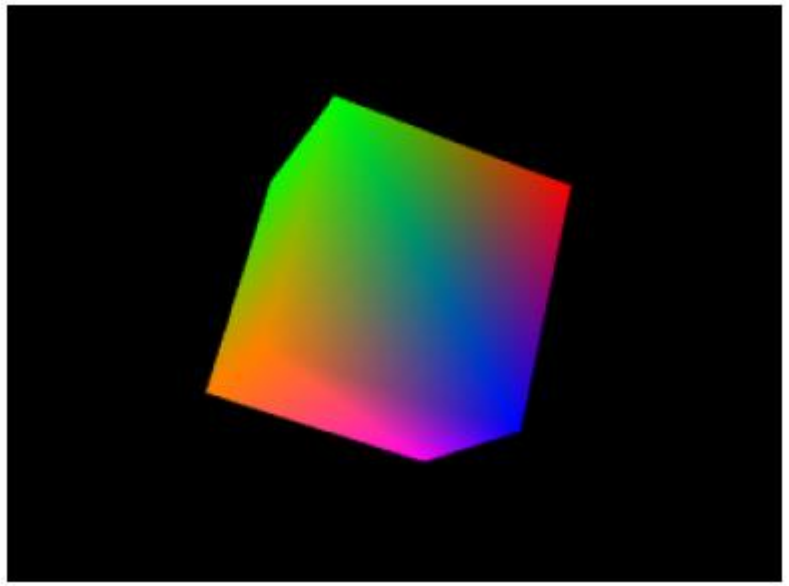
Source: "*Android Cookbook, Problems and Solutions for Android Developers*", Second Edition, by Ian F. Darwin

OpenGL demo Activity

```
public class OpenGLDemoActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Go fullscreen
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                             WindowManager.LayoutParams.FLAG_FULLSCREEN);

        GLSurfaceView view = new GLSurfaceView(this);
        view.setRenderer(new OpenGLRenderer());
        setContentView(view);
    }
}
```

- First, we write a new Activity, and in the onCreate() method we create the two fundamental objects we need to use the OpenGL API: a GLSurfaceView and a Renderer

The renderer implementation



```
class OpenGLRenderer implements Renderer {

    private Cube mCube = new Cube();
    private float mCubeRotation;

    @Override
    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        gl.glClearColor(0.0f, 0.0f, 0.0f, 0.5f);
        gl.glClearDepthf(1.0f);
        gl.glEnable(GL10.GL_DEPTH_TEST);
        gl.glDepthFunc(GL10.GL_LEQUAL);

        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,
                  GL10.GL_NICEST);

    }
```

- The code for our Renderer; it uses a simple Cube object.

The renderer implementation

```
@Override
public void onDrawFrame(GL10 gl) {
    gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
    gl.glLoadIdentity();

    gl.glTranslatef(0.0f, 0.0f, -10.0f);
    gl.glRotatef(mCubeRotation, 1.0f, 1.0f, 1.0f);

    mCube.draw(gl);

    gl.glLoadIdentity();

    mCubeRotation -= 0.15f;
}
```

- onDrawFrame() method is called at every frame, and that's where we put the code to draw our cube.

The renderer implementation

```
    @Override
    public void onSurfaceChanged(GL10 gl, int width, int height) {
        gl.glViewport(0, 0, width, height);
        gl.glMatrixMode(GL10.GL_PROJECTION);
        gl.glLoadIdentity();
        GLU.gluPerspective(gl, 45.0f, (float)width / (float)height, 0.1f, 100.0f);
        gl.glViewport(0, 0, width, height);

        gl.glMatrixMode(GL10.GL_MODELVIEW);
        gl.glLoadIdentity();
    }
}
```

- onSurfaceChanged() method is called when the surface is resized—for instance, when the phone switches between landscape and portrait modes.

The Cube class

```
class Cube {

    private FloatBuffer mVertexBuffer;
    private FloatBuffer mColorBuffer;
    private ByteBuffer  mIndexBuffer;

    private float vertices[] = {
                                    -1.0f, -1.0f, -1.0f,
                                     1.0f, -1.0f, -1.0f,
                                     1.0f,  1.0f, -1.0f,
                                    -1.0f, 1.0f, -1.0f,
                                    -1.0f, -1.0f,  1.0f,
                                    1.0f, -1.0f,  1.0f,
                                    1.0f,  1.0f,  1.0f,
                                    -1.0f,  1.0f,  1.0f
                                    };
```

- The Cube uses two FloatBuffer objects to store vertex and color information and a ByteBuffer to store the face indexes.

The Cube class

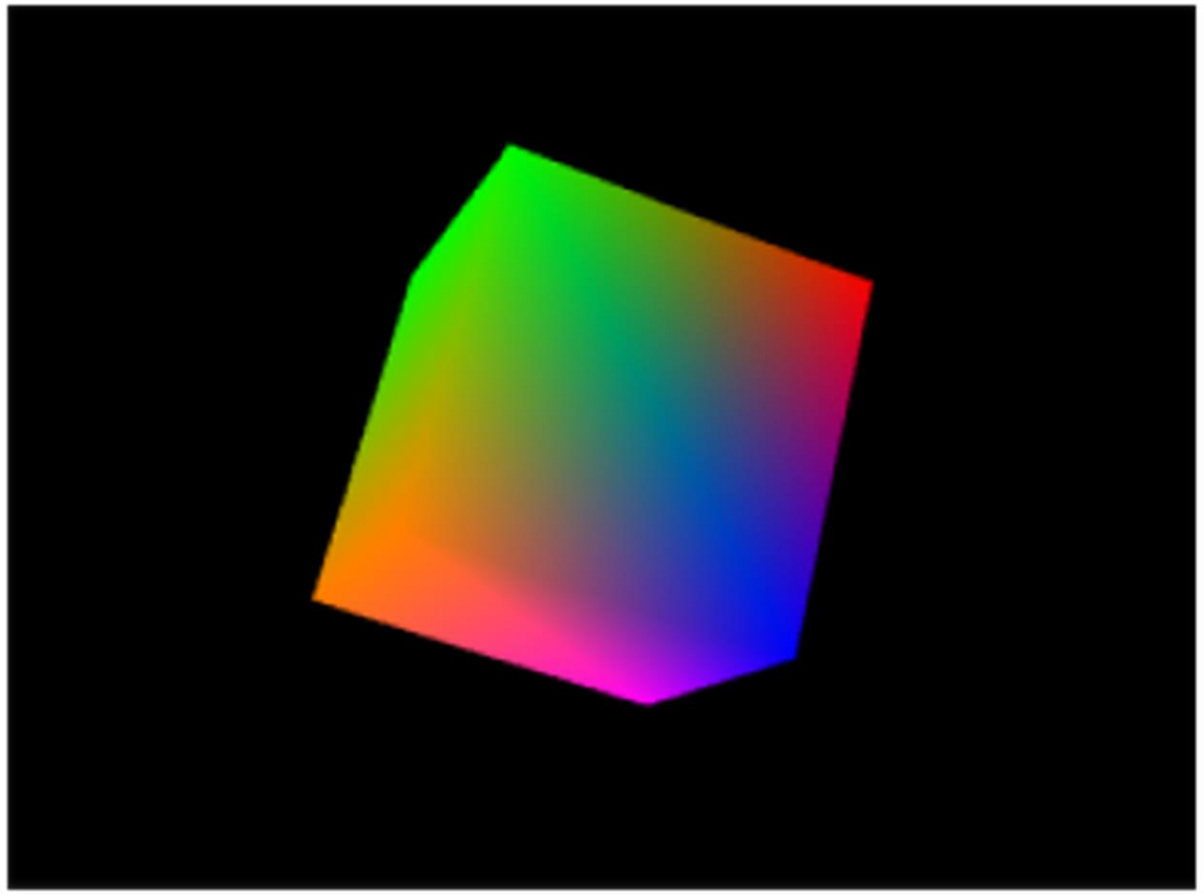
```
private float colors[] = {
                          0.0f,  1.0f,  0.0f,  1.0f,
                          0.0f,  1.0f,  0.0f,  1.0f,
                          1.0f,  0.5f,  0.0f,  1.0f,
                          1.0f,  0.5f,  0.0f,  1.0f,
                          1.0f,  0.0f,  0.0f,  1.0f,
                          1.0f,  0.0f,  0.0f,  1.0f,
                          0.0f,  0.0f,  1.0f,  1.0f,
                          1.0f,  0.0f,  1.0f,  1.0f
                        };

private byte indices[] = {
                          0, 4, 5, 0, 5, 1,
                          1, 5, 6, 1, 6, 2,
                          2, 6, 7, 2, 7, 3,
                          3, 7, 4, 3, 4, 0,
                          4, 7, 6, 4, 6, 5,
                          3, 0, 1, 3, 1, 2
                          };
```

The Cube class

```
public Cube() {
        ByteBuffer byteBuf = ByteBuffer.allocateDirect(vertices.length * 4);
        byteBuf.order(ByteOrder.nativeOrder());
        mVertexBuffer = byteBuf.asFloatBuffer();
        mVertexBuffer.put(vertices);
        mVertexBuffer.position(0);

        byteBuf = ByteBuffer.allocateDirect(colors.length * 4);
        byteBuf.order(ByteOrder.nativeOrder());
        mColorBuffer = byteBuf.asFloatBuffer();
        mColorBuffer.put(colors);
        mColorBuffer.position(0);

        mIndexBuffer = ByteBuffer.allocateDirect(indices.length);
        mIndexBuffer.put(indices);
        mIndexBuffer.position(0);
}
```

The Cube class

```
public void draw(GL10 gl) {
    gl.glFrontFace(GL10.GL_CW);

    gl.glVertexPointer(3, GL10.GL_FLOAT, 0, mVertexBuffer);
    gl.glColorPointer(4, GL10.GL_FLOAT, 0, mColorBuffer);

    gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
    gl.glEnableClientState(GL10.GL_COLOR_ARRAY);

    gl.glDrawElements(GL10.GL_TRIANGLES, 36, GL10.GL_UNSIGNED_BYTE,
                      mIndexBuffer);

        gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
        gl.glDisableClientState(GL10.GL_COLOR_ARRAY);
    }
}
```



Drawing a Spinning Cube with OpenGL ES

Lập trình đa phương tiện (Multimedia programming)

- **Android is a rich multimedia environment.**
- **The standard Android load includes music and video players, and most commercial devices ship with these or fancier versions as well as YouTube players and more.**

Source: "*Android Cookbook, Problems and Solutions for Android Developers*", Second Edition, by Ian F. Darwin

Playing a YouTube Video

Problem You want to play a video from YouTube on your device.

Solution Given a URI to play the video, create an ACTION_VIEW Intent with it and start a new Activity.

Discussion The code required to start a YouTube video with an Intent.

Source: "*Android Cookbook, Problems and Solutions for Android Developers*", Second Edition, by Ian F. Darwin

Starting a YouTube video with an Intent

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    String video_path = "http://www.youtube.com/watch?v=opZ69P-0Jbc";
    Uri uri = Uri.parse(video_path);

    // With this line the YouTube application, if installed, will launch immediately.
    // Without it you will be prompted with a list of applications to choose from.
    uri = Uri.parse("vnd.youtube:"  + uri.getQueryParameter("v"));

    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    startActivity(intent);
}
```

- The example uses a standard YouTube.com URL. The uri.getQueryParameter("v") is used to extract the video ID from the URI itself; in this example, the ID is opZ69P-0Jbc.

Capturing Video Using MediaRecorder

Problem

You want to capture video using the built-in device camera and save it to disk.

Solution

Capture a video and record it on the phone by using the c class provided by the Android framework.

Discussion

The MediaRecorder is normally used to perform audio and/or video recording. The class has a straightforward API, but because it is based on a simple state machine, the methods must be called in the proper order to avoid IllegalStateExceptions from popping up.

Source: "*Android Cookbook, Problems and Solutions for Android Developers*", Second Edition, by Ian F. Darwin

The onCreate() method of the main Activity

- The preview frames from the camera will be displayed on a SurfaceView.
- Recording is controlled by a toggle button.
- After the recording is over, we stop the MediaRecorder.
- Since the stop() method resets all the state machine variables in order to be able to grab another video, we reset the state machine and call our initRecorder() method once more.
- initRecorder() is where we configure the MediaRecorder and the camera

```
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.media_recorder_recipe);
// We shall take the video in landscape orientation
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

mSurfaceView = (SurfaceView) findViewById(R.id.surfaceView);
    mHolder = mSurfaceView.getHolder();
    mHolder.addCallback(this);
    mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

    mToggleButton = (ToggleButton) findViewById(R.id.toggleRecordingButton);
    mToggleButton.setOnClickListener(new OnClickListener() {
            @Override
            // Toggle video recording
            public void onClick(View v) {
                    if (((ToggleButton)v).isChecked())
                            mMediaRecorder.start();
                    else {
                            mMediaRecorder.stop();
                            mMediaRecorder.reset();
                            try {
                                    initRecorder(mHolder.getSurface());
                            } catch (IOException e) {
                                    e.printStackTrace();
                            }
                    }
            }
    });
}
```

Setting up the MediaRecorder

```
/* Init the MediaRecorder. The order the methods are called in is vital to
 * its correct functioning.
 */
private void initRecorder(Surface surface) throws IOException {
    // It is very important to unlock the camera before calling setCamera(),
    // or it will result in a black preview
    if(mCamera == null) {
        mCamera = Camera.open();
        mCamera.unlock();
    }

    if(mMediaRecorder == null)
        mMediaRecorder = new MediaRecorder();

    mMediaRecorder.setPreviewDisplay(surface);
```

Setting up the MediaRecorder

- It is important to create and unlock a Camera object before the creation of a MediaRecorder. setPreviewDisplay, and setCamera() must be called immediately after the creation of the MediaRecorder.
- The choice of the format and output file is obligatory.

```
mMediaRecorder.setCamera(mCamera);

mMediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
mMediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.DEFAULT);
File file = createFile();

mMediaRecorder.setOutputFile(file.getAbsolutePath());

// No limit. Don't forget to check the space on disk.
mMediaRecorder.setMaxDuration(-1);
mMediaRecorder.setVideoFrameRate(15);

mMediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.DEFAULT);

try {
    mMediaRecorder.prepare();
} catch (IllegalStateException e) {
    // This is thrown if the previous calls are not made in the
    // proper order
    e.printStackTrace();
}

mInitSuccesful = true;
}
```

Setting up the MediaRecorder

- The MediaRecorder is best initialized when the surface has been created.
  - We register our Activity as a SurfaceHolder.Callback listener in order to be notified of this and override the surfaceCreated() method to call our initialization code

```
@Override
public void surfaceCreated(SurfaceHolder holder) {
    try {
        if(!mInitSuccessful)
            initRecorder(mHolder.getSurface());
    } catch (IOException e) {
        e.printStackTrace();    // Better error handling?
    }
}
```

Setting up the MediaRecorder

- When you're done with the surface, don't forget to release the resources, as the Camera is a shared object and may be used by other applications

```
private void shutdown() {
    // Release MediaRecorder and especially the Camera as it's a shared
    // object that can be used by other applications
    mMediaRecorder.reset();
    mMediaRecorder.release();
    mCamera.release();
    // Once the objects have been released they can't be reused
    mMediaRecorder = null;
    mCamera = null;
}
```

- Override the surfaceDestroyed() method so that the preceding code can be called automatically when the user is done with the Activity

```
@Override
public void surfaceDestroyed(SurfaceHolder holder) {
    shutdown();
}
```

- Gaming is an important activity that people used to perform on “computers” and now perform on mobile devices.
- Android is a perfectly capable contender in the graphics arena, providing support for OpenGL ES.

Source: "*Android Cookbook, Problems and Solutions for Android Developers*”, Second Edition, by Ian F. Darwin

...

***

~~~
~~~

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !

Lập trình ứng dụng di động

Mobile Application Programming

ET4710

PGS. TS. Đỗ Trọng Tuấn
*Viện Điện tử Viễn thông * Đại học Bách Khoa Hà Nội*

ONE LOVE. ONE FUTURE.