Android Development

A Practical Approach

Fisher Yu
What is Android

• Android is a software stack for mobile devices that includes an operating system, middleware and key applications.

• Features

  • Application framework, Dalvik virtual machine, Integrated browser, Optimized graphics, Media support, GSM, Telephony, Bluetooth, EDGE, 3G, and WiFi, Camera, GPS, compass, and accelerometer, Rich development environment
Android Architecture
Application

• Four kinds of components
  • Activities
  • Services
  • Broadcast receivers
  • Content providers

• For standalone application, we only consider activities
Activity Lifecycle
Outline

- Create An Application
- Menu
- Layout
- Intent
- State
- JNI
- OpenCV
- OpenGL
- Server
- Useful Tips
Plan

• Dive in specific API and interfaces
  • Help you quickly run computer vision algorithm on Android.
  • How to create a great application is not the purpose
• Show sample code for each topic
• Show how it works
Creation
Hello World

- Install a platform
- Create an AVD
- Create a new Android Project
Menu
Menu

- Two kinds of menu
  - Option Menu
  - Context Menu

- Add an item
  - `public abstract MenuItem add (int groupId, int itemId, int order, int titleRes)`

- You can also add submenu

- Get the selected item
  - `public boolean onOptionsItemSelected(MenuItem item)`
Menu

- You can also define Menu in XML
- More commonly used in layout
- `MenuInflater` is used to inflate the menu
Menu

• XML Example

```xml
<menu
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/select_pose"
        android:title="@string/cmenu_pose">
        <menu>
            <item android:id="@+id/select_pose_front" android:title="@string/cmenu_pose_front"/>
            <item android:id="@+id/select_pose_back" android:title="@string/cmenu_pose_back"/>
            <item android:id="@+id/select_pose_above" android:title="@string/cmenu_pose_above"/>
            <item android:id="@+id/select_pose_below" android:title="@string/cmenu_pose_below"/>
            <item android:id="@+id/select_pose_left" android:title="@string/cmenu_pose_left"/>
            <item android:id="@+id/select_pose_right" android:title="@string/cmenu_pose_right"/>
        </menu>
    </item>
    <item android:id="@+id/select_category"
        android:title="@string/cmenu_category">
        <menu>
            <item android:id="@+id/select_category_stapler" android:title="@string/cmenu_category_stapler"/>
            <item android:id="@+id/select_category_mouse" android:title="@string/cmenu_category_mouse"/>
            <item android:id="@+id/select_category_cup" android:title="@string/cmenu_category_cup"/>
        </menu>
    </item>
    <item android:id="@+id/select_bbox"
        android:title="@string/cmenu_bbox"></item>
</menu>
```
Menu

• XML Example

@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.image_view_cmenu, menu);
}
Demo
Layout
Layout

- Two ways to declare layout
  - Declare UI elements in XML
  - Instantiate layout elements at runtime
Layout

- Advantage of XML
  - enables you to better separate the presentation of your application from the code that controls its behavior.
  - XML is external to your application code
  - different screen orientations, different device screen sizes, and different languages
Write the XML

- Exactly one root element
- Other objects or widgets could be added as child elements
- Save the file with .xml extension and put in the res/layout/ directory
Attributes

• ID

```
Button myButton = (Button) findViewById(R.id.my_button);
```

• Layout Parameters

• @ at the beginning means something special
  • “@string/<name>”: The string is in R.string.<name>
  • “@id/<name>”: The id of the object
Attributes

• Example

<Button android:id="@+id/my_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/my_button_text"/>

Write the XML

• Example

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
```
Load XML

- Load the layout resource in `onCreate()`
- Set layout by `setContentView()`
- Reference to the layout by `R.layout.file_name`
Load XML

• Example

```java
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout);
}
```
Demo
Intent
Intent

• Intent messaging is a facility for late run-time binding between components in the same or different applications.

• Three things come first
  • Context
  • Parcel
  • Parcelable
  • Bundle
Context

- **Definition:**
  - `public abstract class Context extends Object`

- **Importance**
  - Interface to global information about an application environment.
  - The famous `Activity` is its indirect class
  - Allow access to application-specific resources and classes, up-calls for application-level operations (such as launching activities, broadcasting and receiving intents, etc).
Parcel

- Container for a message (data and object reference) that can be sent.
- It is mainly used by the system
- You can use it in some sophisticated applications.
- The bulk of the Parcel API revolves around reading and writing data of various types.
There are six major classes of such functions available.

- Primitives (byte, int, double, etc.)
- Primitive Arrays
- Parcelables
- Bundles
- Active Objects
- Untyped Containers
Parcelable

• Interface for classes whose instances can be written to and restored from a Parcel.

• Classes implementing the Parcelable interface must also have a static field called CREATOR, which is an object implementing the Parcelable.Creator interface.
public class MyParcelable implements Parcelable {
    private int mData;

    public int describeContents() {
        return 0;
    }

    public void writeToParcel(Parcel out, int flags) {
        out.writeInt(mData);
    }

    public static final Parcelable.Creator<MyParcelable> CREATOR
        = new Parcelable.Creator<MyParcelable>() {
            public MyParcelable createFromParcel(Parcel in) {
                return new MyParcelable(in);
            }

            public MyParcelable[] newArray(int size) {
                return new MyParcelable[size];
            }
        };

    private MyParcelable(Parcel in) {
        mData = in.readInt();
    }
}
Bundle

- A mapping from String values to various Parcelable types.
- Use a series of put* and get* API to set the content.
Intent Object  In Depth

• An Intent object is a bundle of information

• Two kinds of information
  • For component: Action & Data
  • For Android system: Component name & Category

• Intent filter is used to resolve the intent matching
Different Activities have different functions

We use intent to connect activities with in our application

Switch between applications doesn’t hurt performance a lot. But choose what to pass wisely.
• Only three aspects of an Intent object are consulted when the object is tested against an intent filter
  • action
  • data (both URI and data type)
  • category
• However, if you only want to create a standalone application, you don’t have to deal with the messy aspect.
Intent Object  In Practice

• Create An Intent

• Most of the time, We want to connect to another component (activity) of the same application

• Information about the context (where to find) and the component name (who to find) is enough
Intent Object

In Practice

• NotePad example (Send intent)

```
// NoteEdit is the target of this intent
Intent i = new Intent(this, NoteEdit.class);
// Extra information are inserted as pairs in extra
i.putExtra(NotesDbAdapter.KEY_ROWID, id);
i.putExtra(NotesDbAdapter.KEY_TITLE, c.getString(
    c.getColumnIndexOrThrow(
        NotesDbAdapter.KEY_TITLE)));
i.putExtra(NotesDbAdapter.KEY_BODY, c.getString(
    c.getColumnIndexOrThrow(
        NotesDbAdapter.KEY_BODY)));
startActivityForResult(i, ACTIVITY_EDIT);
```
Intent Object  In Practice

• NotePad example (Send intent back)
  // When NoteEdit finishes, it pass back the results
  // using a bundle to contain the information
  // The bundle could be put as extra in intent directly
  Bundle bundle = new Bundle();

  bundle.putString(NotesDbAdapter.KEY_TITLE, mTitleText.getText().toString());
  bundle.putString(NotesDbAdapter.KEY_BODY, mBodyText.getText().toString());
  if (mRowId != null) {
    bundle.putLong(NotesDbAdapter.KEY_ROWID, mRowId);
  }

  Intent mIntent = new Intent();
  mIntent.putExtras(bundle);
  // When the intent is ready, set the result and
  // call finish() to finish the current activity
  setResult(RESULT_OK, mIntent);
  finish();
@Override
// When the result is send back, onActivityResult is called
// by the Android system
protected void onActivityResult(int requestCode, int resultCode,
Intent intent) {
    super.onActivityResult(requestCode, resultCode, intent);
    // Get the data out of the intent
    Bundle extras = intent.getExtras();

    ...

}
Intent Object

In Practice

• Camera Example (Launch Camera)

Intent intent = new Intent("android.media.action.IMAGE_CAPTURE");
//mImageUri contains the image uri, which identifies the image
intent.putExtra(MediaStore.EXTRA_OUTPUT, mImageUri);
//DEFAULT_CAMERA is a constant defined by the current class
//It can be used when the activity returns the result.
startActivityForResult(intent, DEFAULT_CAMERA);
Intent Object  In Practice

• Camera Example (Receive Result)

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    ...

Demo
State
State

Do you still remember this?
State

- Before the system kill your application
- Save your state
- Recover the state in `onCreate()`
Demo
JNI
JNI

- Java Native Interface
- Applications that use the JNI can incorporate native code written in programming languages such as C and C++, as well as code written in the Java programming language
JNI Tradeoff

• Pros

• Better performance (probably)

• Portable code to other platform (iOS, Brew)

• A huge library in C is ready to use (OpenCV)
JNI Tradeoff

- Cons
  - C code is harder to write than Java
  - Hard to debug
  - The current standard C library support is not stable
  - Android Emulator doesn’t support it very well
Role of JNI
How to Use JNI

- Create a Java application as usual
- Use `javac` to compile the source file, resulting in the class file.
  - If you develop Android Application in Eclipse, the `.class` file is generated automatically
- Use `javah -jni` to generate a C header file containing the function prototype for the native method implementation.
  - `javah` and `javac` are provided in JDK
How to Use JNI

- Write the C implementation of the native method.
  - The native source should be in `<project>/jni/...`
- Create `<project>/jni/Android.mk` to describe your native sources to the NDK build system
- Compile the C implementation into a native library, creating `<name>.dll` or `lib<name>.so`.
  - We can use `ndk-build` to compile the C code.
  - `ndk-build` would do everything for you if your source is correct.
How to Use JNI

• Finally, compile your application using the SDK tools in the usual way. The SDK build tools will package the shared libraries in the application’s deployable .apk file.
Demo
OpenCV

• It should exist theoretically
• NDK doesn’t support C++ very well
• Hackers do find a solution: crystax ndk
  • Support of C++ exceptions, RTTI and Standard C++ Library added.
Demo
OpenGL ES

• An interface to the graphics hardware on the device

• OpenGL ES 1.1 and 2.0 are supported on Android
Why OpenGL ES hardware

- CPU is not alone on the chipset
- GPU is common on smartphone
- Better performance
- GPU is being improved fast
Why OpenGL ES hardware

Fragment Shader Performance Test

- 8650 Android
- 7x30 Android

Time in millisecond

Number of fragment shaders in thousand
Why OpenGL Application

- Augmented Reality
- 3D Model
- Image Processing
Why OpenGL Application

- Image Processing
- Edge Detector
- Gaussian Blur
- Image Enhancement
Demo
Server

- Computer Vision is hard
- More computation is needed
- The smartphone client need to interact with a remote powerful server
- Many technology options
  - Servlet, CGI, etc.
Servlet

- Easy to use
- More efficient than script language
- The client also uses Java
  - Easy to send data between Server and Client
HelloServlet

• Create a servlet project in Eclipse
• Switch on the server (Tomcat)
• Deal with the request and response properly
• Access by URL
Demo
Useful Tips
Camera

- Idea: Lay the actual view on top of the camera preview.
Demo
Manipulate Image

- Idea: Use Bitmap API on Android
- The format of default color is ARGB
- Raw data could be extracted from Bitmap
Demo
Questions?