Fragments

Fragments

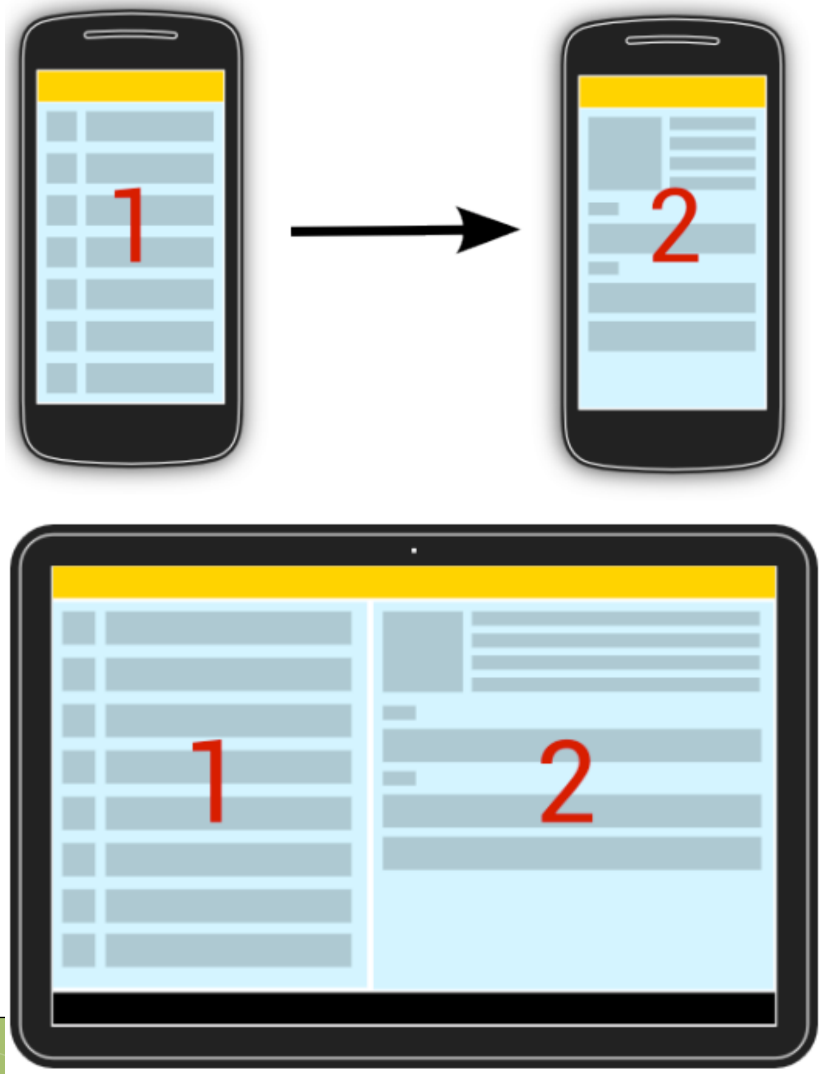
A fragment is a self-contained, modular section of an application's user interface and corresponding behavior that can be embedded within an activity.

Fragments can be assembled to create an activity during the application design phase, and added to, or removed from an activity during application runtime to create a dynamically changing user interface.

Fragments may only be used as part of an activity and cannot be instantiated as standalone application elements.

A fragment can be thought of as a functional "sub-activity" with its own lifecycle similar to that of a full activity.

Using fragments



Fragments lifecycle

Method	**Description**
onAttach()	The fragment instance is associated with an activity instance.The activity is not yet fully initialized
onCreate()	Fragment is created
onCreateView()	The fragment instance creates its view hierarchy. The inflated views become part of the view hierarchy of its containing activity.
onActivityCreated()	Activity and fragment instance have been created as well as thier view hierarchy. At this point, view can be accessed with the `findViewById()` method. example.
onResume()	Fragment becomes visible and active.
onPause()	Fragment is visibile but becomes not active anymore, e.g., if another activity is animating on top of the activity which contains the fragment.
onStop()	Fragment becomes not visible.

Defining a new fragment (from code)

To define a new fragment you either extend the android.app.Fragment class or one of its subclasses, for example,

- ListFragment,
- DialogFragment,
- PreferenceFragment
- WebViewFragment.

Defining a new fragment (from code)

```
public class DetailFragment extends Fragment {
 @Override
 public View onCreateView(LayoutInflater inflater,
        ViewGroup container, Bundle savedInstanceState) {
    View view=inflater.inflate(
        R.layout.fragment_rssitem_detail,
        container, false);
   return view;
 }
 public void setText(String item) {
    TextView view = (TextView)
                getView().findViewById(R.id.detailsText);
    view.setText(item);
 }
}
```

XML-based fragments

```
<RelativeLayout xmlns:android="http://schemas.android.com/
apk/res/android" xmlns:tools="http://schemas.android.com/
tools" android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".FragmentDemoActivity" >
<fragment android:id="@+id/fragment_one"
android:name="com.example.myfragmentdemo.FragmentOne"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_centerVertical="true" tools:layout="@layout/
fragment_one_layout" />
</RelativeLayout>
```

Adding-removing fragments at runtime

The FragmentManager class and the FragmentTransaction class allow you to add, remove and replace fragments in the layout of your *activity*.

Fragments can be dynamically modified via transactions. To dynamically add fragments to an existing layout you typically define a container in the XML layout file in which you add a *Fragment*.

```
FragmentTransaction ft =
getFragmentManager().beginTransaction();
ft.replace(R.id.your_placehodler, new
YourFragment());
ft.commit();
```

A new *Fragment* will replace an existing *Fragment* that was previously added to the container.

Finding if a fragment is already part of your Activity

```
DetailFragment fragment = (DetailFragment)
        getFragmentManager().
        findFragmentById(R.id.detail_frag);

if (fragment==null) {
                // start new Activity
} else {
                fragment.update(...);
}
```

Communication: activity -> fragment

In order for an activity to communicate with a fragment, the activity must identify the fragment object via the ID assigned to it using the findViewById() method.

Once this reference has been obtained, the activity can simply call the public methods of the fragment object.

Communication: fragment-> activity

Communicating in the other direction (from fragment to activity) is a little more complicated.

A) the fragment must define a listener interface, which is then implemented within the activity class.

```
public class MyFragment extends Fragment {
  AListener activityCallback;
  public interface AListener {
                public void someMethod(int par1, String par2);
  }
  …
```

Communication: fragment-> activity

B. the onAttach() method of the fragment class needs to be overridden and implemented. The method is passed a reference to the activity in which the fragment is contained. The method must store a local reference to this activity and verify that it implements the interface.

```
public void onAttach(Activity activity) {
   super.onAttach(activity);
   try { activityCallback = (AListener) activity;
   } catch (ClassCastException e) {
        throw new ClassCastException(
                activity.toString()
                + " must implement ToolbarListener");
} }
```

Communication: fragment-> activity

C. The next step is to call the callback method of the activity from within the fragment. For example, the following code calls the callback method on the activity when a button is clicked:

```
public void buttonClicked(View view) {
   activityCallback.someMethod(arg1, arg2);
}
```

Communication: fragment-> activity

All that remains is to modify the activity class so that it implements theToolbarListener interface.

```
public class MyActivity extends
FragmentActivity implements
MyFragment.AListener {
 public void someMethod(String arg1, int arg2)
      {
      // Implement code for callback method
      }
.
.
}
```

Esempio

vedi
http://www.vogella.com/tutorials/AndroidFragments/article.html

sez. 10

Content Providers

Marco Ronchetti
Università degli Studi di Trento

Content Provider

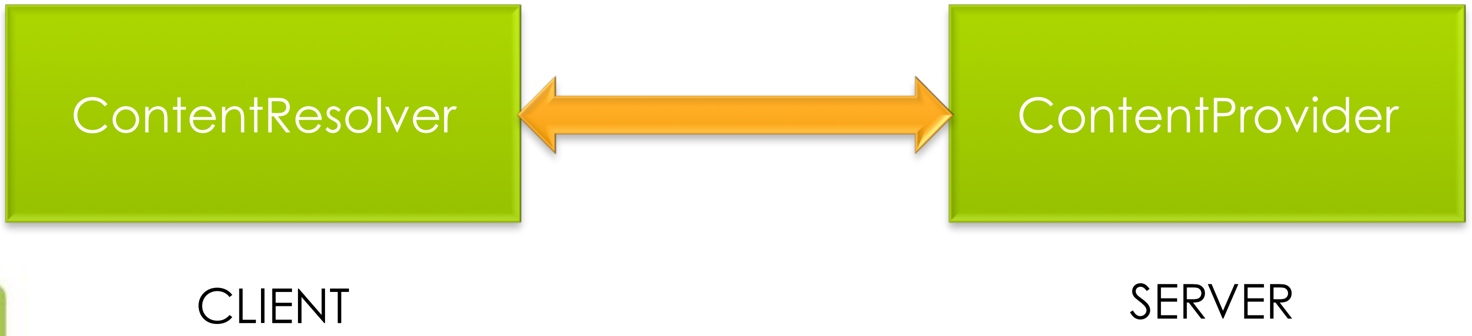
A standard interface connecting a running process with data in another process

Manages access to a structured set of data:

- encapsulate the data
- provide mechanisms for defining data security.

To access data in a content provider, use the **ContentResolver** object in your Context

The ContentResolver object communicates with the provider object, an instance of a class that implements **ContentProvider**. The provider object receives data requests from clients, performs the requested action, and returns the results.



Content Provider

Android includes content providers that manage data such as audio, video, images, and personal contact information..

You can create your own custom content provider to share your data with other packages that works just like the built-in providers.

You need to develop your own provider if

- you intend to share your data with other applications.
- you want to to provide custom search suggestions in your own application.
- you want to copy and paste complex data or files from your application to other applications.

Default content providers

- ContactsContract
  - Stores all contacts information. etc
- Call Log Stores
  - call logs, for example: missed calls, answered calls. etc.
- Browser
  - Use by browser for history, favorites. etc.
- Media Store
  - Media files for Gallery, from SD Card. etc.
- Setting
  - Phone device settings. etc.

Querying a Content Provider

To query a content provider, you provide a query string in the form of a URI, with an optional specifier for a particular row, using the following syntax:

<standard_prefix>://<authority>/<data_path>/<id>

For example, **to retrieve all the bookmarks** stored by your web browsers (in Android), you would use the following content URI:

content://browser/bookmarks

Similarly, **to retrieve all the contacts** stored by the Contacts application, the URI would look like this:

content://contacts/people

To retrieve a particular contact, you can specify the URI with a specific ID:

content://contacts/people/3

Accessing calls

package ...
import ...

```
package ...
import ...
public class ContentProviderActivity extends Activity {
  /** Called when the activity is first created. */
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      Uri allCalls = Uri.parse("content://call_log/calls");
      Cursor c = managedQuery(allCalls, null, null, null, null);
```

```
if (c.moveToFirst()) {
        do{
          String callType = "";
          switch (Integer.parseInt(c.getString(
            c.getColumnIndex(Calls.TYPE))))
          {
              case 1: callType = "Incoming";
                break;
              case 2: callType = "Outgoing";
                break;
              case 3: callType = "Missed";
          }
          Log.v("Content Providers",
              c.getString(c.getColumnIndex(Calls._ID))
              + ", " +
              c.getString(c.getColumnIndex(Calls.NUMBER))
              + ", " +
              callType) ;
        } while (c.moveToNext());
      }
   }
}
```

Error!

```
E/AndroidRuntime(541): java.lang.RuntimeException:
Unable to start activity ComponentInfo{it.unitn.science.latemar/
it.unitn.science.latemar.ContentProviderActivity}:
 java.lang.SecurityException: Permission Denial: opening provider
com.android.providers.contacts.CallLogProvider from ProcessRecord{41475a28
541:it.unitn.science.latemar/10041} (pid=541, uid=10041)
requires
android.permission.READ_CONTACTS or
android.permission.WRITE_CONTACTS
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="it.unitn.science.latemar"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk android:minSdkVersion="13" />

   <application
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name" >
      <activity
         android:name=".ContentProviderActivity"
         android:label="@string/app_name" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>
    <uses-permission
     android:name="android.permission.READ_CONTACTS">
    </uses-permission>
</manifest>
```

Building your own Content Provider

Marco Ronchetti
Università degli Studi di Trento

ContentProvider

The primary methods that need to be implemented are:

- onCreate()
  - is called to initialize the provider
- query(Uri, String[], String, String[], String)
  - returns data to the caller
- insert(Uri, ContentValues)
  - inserts new data into the content provider
- update(Uri, ContentValues, String, String[])
  - updates existing data in the content provider
- delete(Uri, String, String[])
  - deletes data from the content provider
- getType(Uri)
  - returns the MIME type of data in the content provider

Normally these are just wrapper functions around the raw SQL queries, e.g.:
delete(tableName, where, whereArgs);
is simply just a wrapper around the SQL query that looks something like:
"delete from " + tableName + " where " + where + " ? " + whereArgs”

```
public int delete(Uri uri, String where, String[] whereArgs) {
  SQLiteDatabase db = dbHelper.getWritableDatabase();
  int count;
  switch (sUriMatcher.match(uri)) {
  case NOTES:
       count = db.delete(NOTES_TABLE_NAME, where, whereArgs);
       break;
 default:
       throw new IllegalArgumentException("Unknown URI " + uri);
  }
  getContext().getContentResolver().notifyChange(uri, null);
  return count;
}
```

Register your ContentProvider

```
<application>
…
<provider android:name=".contentprovider.MyTodoContentProvider"
android:authorities="MyUniquePath.todos.contentprovider" >
</provider>
</application>
```

URIMatcher

```
 private static final int PEOPLE = 1;
 private static final int PEOPLE_ID = 2;
 private static final int PEOPLE_PHONES = 3;
 private static final int PEOPLE_PHONES_ID = 4;
 private static final int CALLS = 11;
 private static final int CALLS_ID = 12;
 private static final int CALLS_FILTER = 15;

private static final UriMatcher sURIMatcher = new UriMatcher(UriMatcher.NO_MATCH);
static
 {
   sURIMatcher.addURI("contacts", "people", PEOPLE);
   sURIMatcher.addURI("contacts", "people/#", PEOPLE_ID);
   sURIMatcher.addURI("contacts", "people/#/phones", PEOPLE_PHONES);
   sURIMatcher.addURI("contacts", "people/#/phones/#", PEOPLE_PHONES_ID);
   sURIMatcher.addURI("call_log", "calls", CALLS);
   sURIMatcher.addURI("call_log", "calls/filter/*", CALLS_FILTER);
   sURIMatcher.addURI("call_log", "calls/#", CALLS_ID);
 }
       sUriMatcher.addURI(AUTHORITY, PATH, CODE);
```

Example

```
 static {
    sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
    sUriMatcher.addURI(AUTHORITY, NOTES_TABLE_NAME, NOTES);
}
```

Later, given a uri, in the code of your ContentProvider:

```
switch (sUriMatcher.match(uri)) {
   case NOTES:
         count = db.delete(NOTES_TABLE_NAME, where, whereArgs);
         break;
  default:
         throw new IllegalArgumentException("Unknown URI " + uri);
   }
```

See a full example here:
http://thinkandroid.wordpress.com/2010/01/13/writing-your-own-contentprovider/

Accessing a Content Provider

When you want to access data in a content provider, you use the ContentResolver object in your application's Context to communicate with the provider as a client.

The ContentResolver object communicates with the provider object, an instance of a class that implements ContentProvider. The provider object receives data requests from clients, performs the requested action, and returns the results.

ContentResolver

You get a ContentResolver from your Context

ContentResolver cr=getContentResolver();

Then you typically use

- cr.query()
- cr.insert()
- cr.update()
- cr.delete()
- cr.getType()

See http://developer.android.com/reference/android/content/ContentResolver.html

Change in Orientation

Marco Ronchetti
Università degli Studi di Trento

Change in orientation

Change in orientation

For devices that support multiple orientations, Android detects a change in orientation:

the display is "landscape" or "portrait".

When Android detects a change in orientation, its default behavior is to destroy and then re-start the foreground Activity.

- Is the screen re-drawn correctly? Any custom UI code you have should handle changes in the orientation.
- Does the application maintain its state? The Activity should not lose anything that the user has already entered into the UI.

Change in configuration

e.g. a change in the availability of a keyboard or a change in system language.

A change in configuration also triggers the default behavior of destroying and then restarting the foreground Activity.

Besides testing that the application maintains the UI and its transaction state, you should also test that the application updates itself to respond correctly to the new configuration.

Save and restore the application state

```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    // Save UI state changes to the savedInstanceState.
    // This bundle will be passed to onCreate if the process is
    // killed and restarted.
    savedInstanceState.putCharSequence("text", button.getText());
    savedInstanceState.putInt("count", count);
    super.onSaveInstanceState(savedInstanceState);
}
@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    button = (Button) findViewById(R.id.button1);
    button.setText(savedInstanceState.getCharSequence("text"));
    count=savedInstanceState.getInt("count");
}
```

What can we save in a bundle?

- Primitive data types – arrays of p.d.t.
- String – StringArray - StringArrayList
- CharSequence – CharSequenceArray – CharSequenceArrayList
- Parcelable - ParcelableSequenceArray – ParcelableSequenceArrayList
- Serializable

How to rotate the emulator:

Ctrl-F11

On Mac: ctrl-fn F12

See:

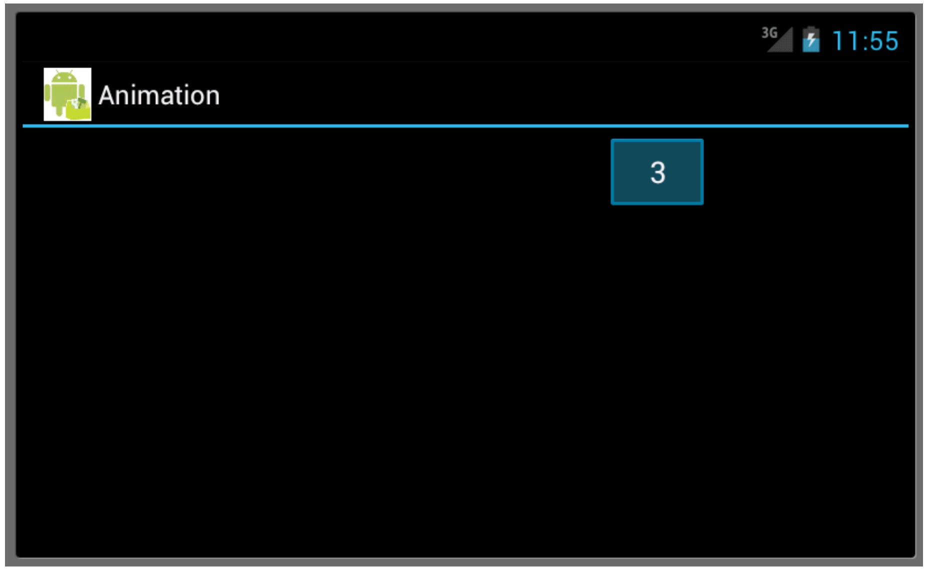
http://developer.android.com/guide/developing/tools/emulator.html#KeyMapping

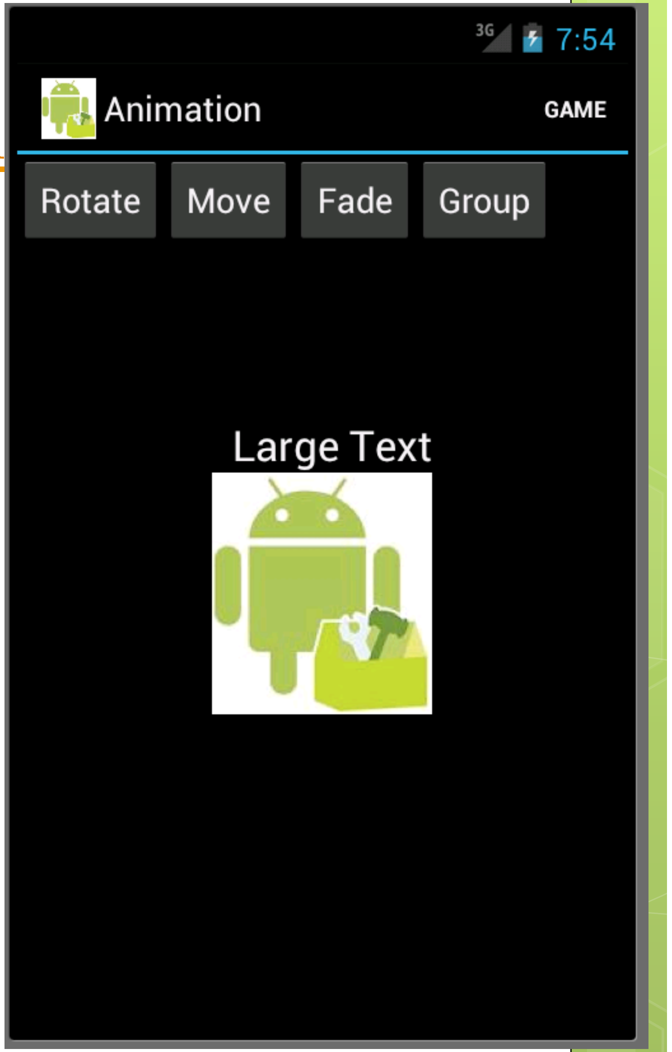
Basic Animation

Marco Ronchetti
Università degli Studi di Trento

An example

Modified from an example by Vogella





Download the source from:

http://latemar.science.unitn.it/segue_userFiles/2012Mobile/Animation.zip

3 ways to do animation

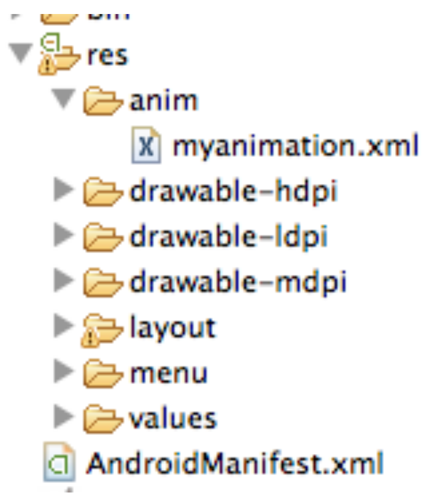
- Property Animation (Android >= 3.0, API level 11)
  - lets you animate properties of any object, including ones that are not rendered to the screen. The system is extensible and lets you animate properties of custom types as well.
- View Animation
  - An older system, can only be used for Views. It is relatively easy to setup and offers enough capabilities to meet many application's needs.
- Drawable Animation
  - involves displaying Drawable resources one after another, like a roll of film.
  - useful if you want to animate things that are easier to represent with Drawable resources, such as a progression of bitmaps

See http://developer.android.com/guide/topics/graphics/animation.html

Example 1 – loading animation from xml

```
ImageView aniView = (ImageView) findViewById(R.id.imageView1);
Animation animation1 = AnimationUtils.loadAnimation
            (this,R.anim.myanimation);
aniView.startAnimation(animation1);
```

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="true">
    <rotate android:fromDegrees="0”
        android:toDegrees="360"
        android:duration="5000"
        android:pivotX="50%”
        android:pivotY="90%"
        android:startOffset="0">
    </rotate>
</set>
```



```
▼ res
  ▼ anim
      myanimation.xml
  ▶ drawable-hdpi
  ▶ drawable-ldpi
  ▶ drawable-mdpi
  ▶ layout
  ▶ menu
  ▶ values
  AndroidManifest.xml
```

Example 2 –animating from code

```
ImageView aniView = (ImageView) findViewById(R.id.imageView1);
ObjectAnimator animation1 = ObjectAnimator.ofFloat(aniView,
                        "rotation", dest);
animation1.setDuration(2000);
animation1.start();

// Example of animation lifecycle trapping
animation1.setAnimationListener(new AnimationListener(){
  @Override
   public void onAnimationEnd(Animation animation) {...}
 @Override
   public void onAnimationRepeat (Animation animation) {...}
 @Override
   public void onAnimationStart (Animation animation) {...}
});
```

ObjectAnimator

static ObjectAnimator ofFloat(Object target, String propertyName, float... values)

- Constructs and returns an ObjectAnimator that animates between float values.

public static ObjectAnimator ofFloat(T target, Property<T, Float> property, float... values)

- Animate a given (float) property in object target

ofInt is similar

Fading demo

```
float dest = 1;
if (aniView.getAlpha() > 0) {
    dest = 0;
}
ObjectAnimator animation3 = ObjectAnimator.ofFloat(aniView,"alpha", dest);
animation3.setDuration(2000);
animation3.start();
```

TypeEvaluator

Interface that implements:
public abstract T evaluate (float fraction, T startValue, T endValue)

- This function should return a linear interpolation between the start and end values, given the fraction parameter.
- The calculation is expected to be simple parametric calculation: result = x0 + t * (x1 - x0), where x0 is startValue, x1 is endValue, and t is fraction.

ObjectAnimator

static ObjectAnimator ofObject(Object target, String propertyName, TypeEvaluator evaluator, Object... values)

- Constructs and returns an ObjectAnimator that animates between Object values.
- .

static <T, V> ObjectAnimator ofObject(T target, Property<T, V> property, TypeEvaluator<V> evaluator, V... values)

Constructs and returns an ObjectAnimator that animates a given property between Object values.

AnimatorSet: combining animations

Since Android 3.0

void playSequentially(Animator... items)
void playTogether(Animator... items)
void start()
void end()

AnimatorSet.Builder play(Animator anim)

- This method creates a Builder object, which is used to set up playing constraints.

AnimatorSet.Builder

AnimatorSet.Builder after(Animator anim)

- anim starts when player ends.

AnimatorSet.Builder after(long delay)

- Start player after specified delay.

AnimatorSet.Builder before(Animator anim)

- start player when anim ends.

AnimatorSet.Builder with(Animator anim)

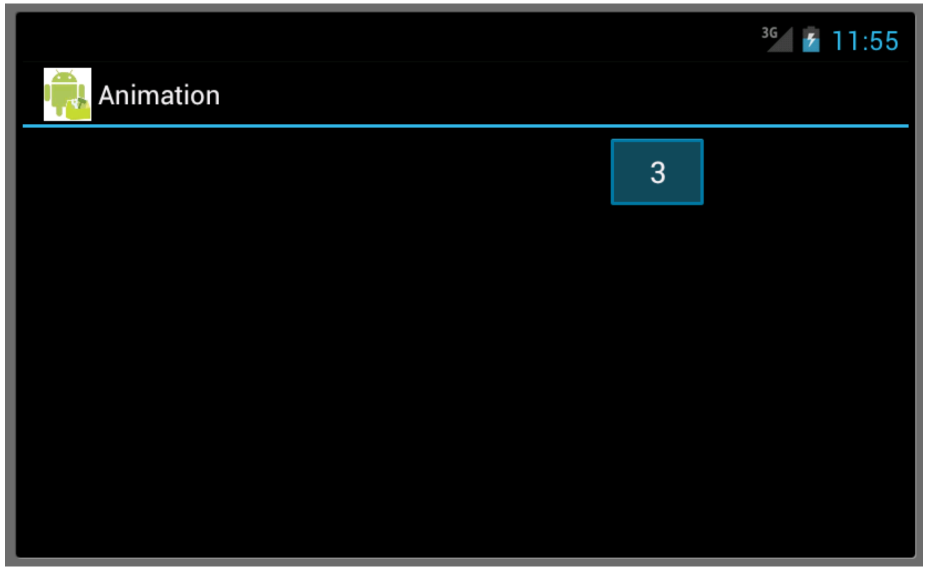
- Starts player and anim at the same time.

AnimatorSet: coreography

```
ObjectAnimator fadeOut = ObjectAnimator.ofFloat(aniView, "alpha", 0f);
fadeOut.setDuration(2000);

ObjectAnimator mover = ObjectAnimator.ofFloat(aniView,
          "translationX", -500f, 0f);
mover.setDuration(2000);

ObjectAnimator fadeIn = ObjectAnimator.ofFloat(aniView, "alpha",1f);
fadeIn.setDuration(2000);

AnimatorSet animatorSet = new AnimatorSet();
animatorSet.play(mover).with(fadeIn).after(fadeOut);
animatorSet.start();
```

Example

See the example in the zip file connected with this lecture:



See how the game behaves when rotating the device with and without **onSaveInstanceState** and **onRestoreInstanceState**

What to test

Marco Ronchetti
Università degli Studi di Trento

What to test

Change in orientation

Battery life

Techniques for minimizing battery usage were presented at the 2010 Google I/O conference in the presentation Coding for Life -- Battery Life, That Is. This presentation describes the impact on battery life of various operations, and the ways you can design your application to minimize these impacts. When you code your application to reduce battery usage, you also write the appropriate unit tests.

What to test

Dependence on external resources

If your application depends on network access, SMS, Bluetooth, or GPS, then you should test what happens when the resource or resources are not available.

For example, if your application uses the network,it can notify the user if access is unavailable, or disable network-related features, or do both. For GPS, it can switch to IP-based location awareness. It can also wait for WiFi access before doing large data transfers, since WiFi transfers maximize battery usage compared to transfers over 3G or EDGE.

You can use the emulator to test network access and bandwidth. To learn more, please see Network Speed Emulation. To test GPS, you can use the emulator console and LocationManager. To learn more about the emulator console, please see Using the Emulator Console.

Support of multiple versions

Marco Ronchetti
Università degli Studi di Trento

http://android-developers.blogspot.it/2010/07/how-to-have-your-cupcake-and-eat-it-too.html

http://android-developers.blogspot.it/2010/06/future-proofing-your-app.html

Best practices

Marco Ronchetti
Università degli Studi di Trento

Performance tips

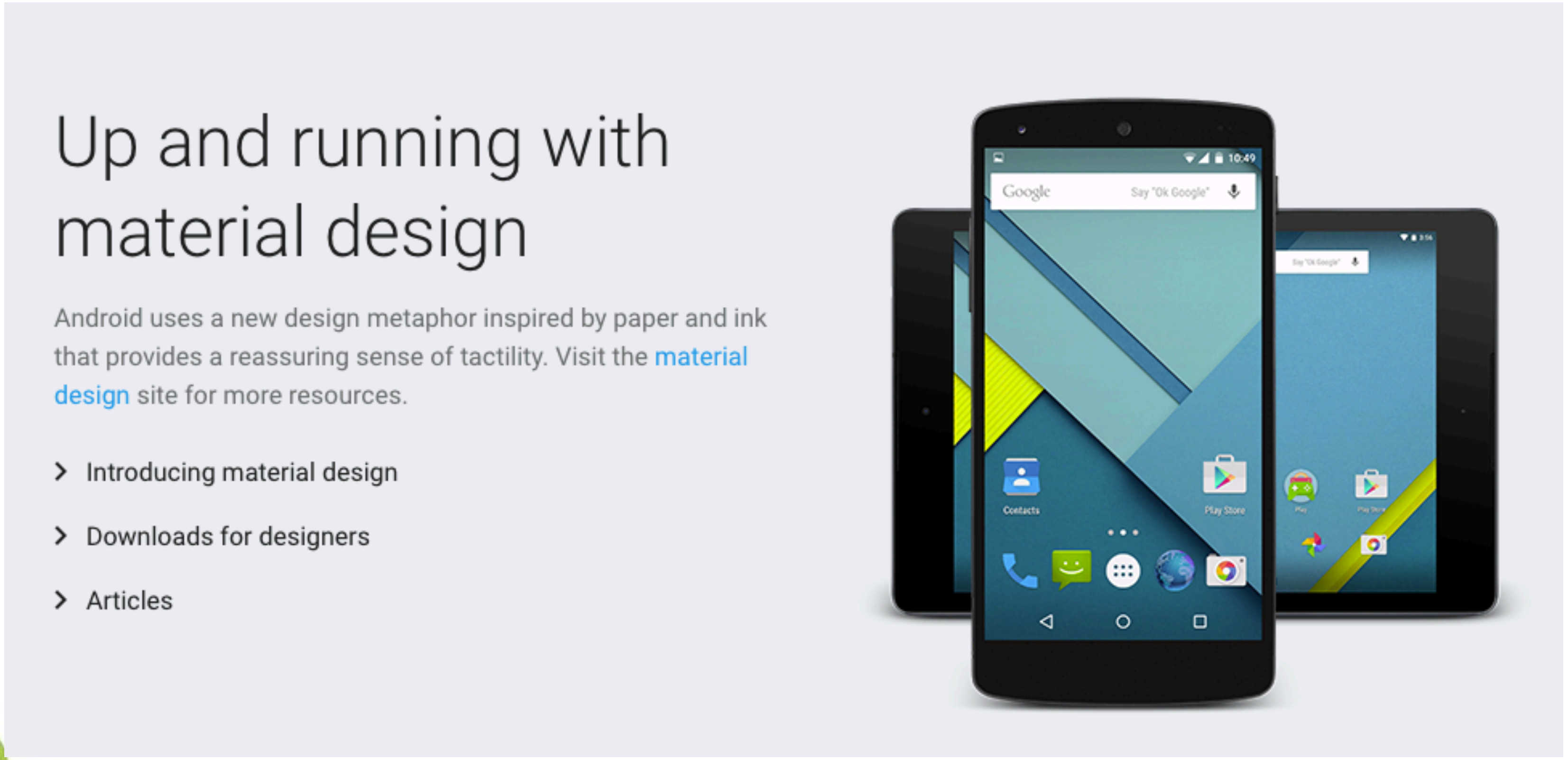
http://developer.android.com/training/articles/perf-tips.html

Design

Marco Ronchetti
Università degli Studi di Trento

Android design

http://developer.android.com/design/index.html



Up and running with material design

Android uses a new design metaphor inspired by paper and ink that provides a reassuring sense of tactility. Visit the material design site for more resources.

- Introducing material design
- Downloads for designers
- Articles