Programming with Android:
Animations, Menu, Toast and Dialogs

Luca Bedogni   **Marco Di Felice**

Dipartimento di Scienze dell'Informazione

Università di Bologna

❖Make the components move/shrink/color

❖Mainly two methods:

– Subsequent images (frame-by-frame)

– Initial state, final state, time, transition (tween)

❖Animation are expensive in terms of memory

– Be sure to manage them correctly

❖ Define a set of frame

- Each Drawable is a frame of the animation

❖ Usage of AnimationDrawable

- An Animation specialization

❖ Could be defined via XML or in Java

```
<animation-list android:id="selected" android:oneshot="false">
      <item android:drawable="@drawable/anim0" android:duration="10" />
      <item android:drawable="@drawable/anim1" android:duration="10" />
      <item android:drawable="@drawable/anim2" android:duration="10" />
      <item android:drawable="@drawable/anim3" android:duration="10" />
      <item android:drawable="@drawable/anim4" android:duration="10" />
      <item android:drawable="@drawable/anim5" android:duration="10" />

</animation-list>
```

```java
public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);

    ImageView imageView = (ImageView) findViewById(R.id.animationView);

    animationDrawable = (AnimationDrawable) imageView.getBackground();

    btnStart = (Button)findViewById(R.id.btnStart);

    btnStop = (Button)findViewById(R.id.btnStop);

    btnStart.setOnClickListener(this);    btnStop.setOnClickListener(this);

}

public void onClick(View v) {

    if (v == btnStart)  animationDrawable.start();

    else animationDrawable.stop();

}
```

❖Not so easy to use

❖If you want to change something in the middle of the animation, you may have to change the entire animation

❖Coupled with a set of images

– Same animation on different images?

→ Define another animation

❖You have to manually create every image

❖The .apk will become larger
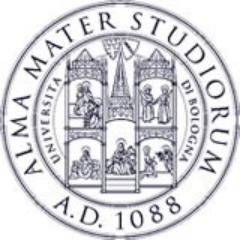
Animations: tween

❖ Define the skeleton of an animation

❖ Define the transitions in the form of *"when it starts, it's like this, when it ends, it's like that, and it lasts x seconds"*

❖ One could define an animation and apply it to multiple objects, so animations are not coupled with objects

– Reuse it!

❖Let's start by creating a TextView

❖Create a *anim* directory under *res*

❖Create a *animation.xml* file

```xml
<set
    xmlns:android="http://schemas.android.com/apk/res/android">
    <alpha
        android:fromAlpha="0.0"
        android:toAlpha="1.0"
        android:duration="1500"
        />
</set>
```
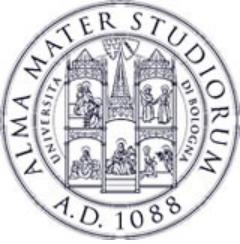
❖Meanings:

   ❖fromAlpha: initial opacity. 0 is invisible,1 is visible.

   ❖toAlpha: final opacity. 0 is invisible, 1 is visible.

   ❖duration: the duration of the animation, in milliseconds.

Tween: Inside the code

- We need a function, like *startAnimation()* inside our activity
  - We need to get the TextView with *findViewById()*
  - Create the animation by calling it
  - Apply the animation to the TextView
- (Nearly) the same for *stopAnimation()*
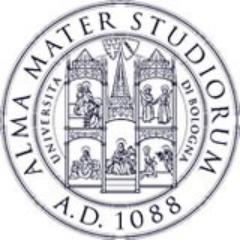
```
public void startAnimation() {

        TextView title = (TextView)findViewById(R.id.title);

        Animation fade = AnimationUtils.loadAnimation(this, R.anim.animation);

        title.startAnimation(fade);

}


public void stopAnimation() {

        TextView title = (TextView)findViewById(R.id.title);

        title.clearAnimation();

}
```

❖ The offset if used if you want to start an animation after a certain amount of time

❖ Not so useful with animations composed by a single View

❖ Could be useful with 2 or more Views

– Start an animation after x seconds of another animation

Tween: AnimationListener

❖ AnimationListener class, to be warned about animations events

❖ Attach it to your animation

❖ Implement the code in the listener

❖ Methods contained are:

  ❖ onAnimationEnd()

  ❖ onAnimationRepeat()

  ❖ onAnimationStart()
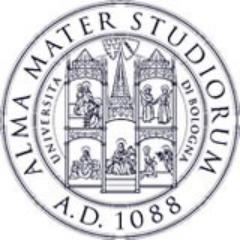
Adding an offset and a listener

```
public void startAnimation() {

        TextView title = (TextView)findViewById(R.id.title);

        Animation fade = AnimationUtils.loadAnimation(this, R.anim.animation);

        fade.setAnimationListener(this);

        title.startAnimation(fade);


        TextView subtitle = (TextView)findViewById(R.id.subtitle);

        Animation fade2 = AnimationUtils.loadAnimation(this, R.anim.animation);

        fade2.setStartOffset(500);

        subtitle.startAnimation(fade2);

}
```

- Of course there isn't only the alpha parameter to set

- One can edit the rotation of an object, the dimension of an image and the position on the screen

- Beware: animation are cool, but too many of them could confuse the user

- Use animations as a support for your application, not as a main purpose

- They appear whenever the user presses the menu button
- Useful for giving different options without leaving the current Activity
- Don't make too big menus, or they'll cover entirely the Activity
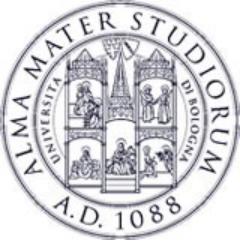
❖ Two methods (again):

   ❖ XML

      ❖ Place a file inside res/menu/

      ❖ Inflate the menu inside the Activity

      ❖ Useful if you want to create the same menu inside different activities

   ❖ Java

      ❖ Create the menu directly inside the activity

❖ Create res/menu/menu.xml

❖ We need:

  ❖ IDs of menu's elements

  ❖ Title of each element

  ❖ Icon of each element

❖ Inside the Activity, create onCreateOptionsMenu()

  ❖ Inflate the menu

  ❖ Add functionality to the buttons
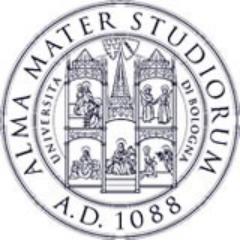
Menu: menu.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/item1" android:title="First Option"></item>
    <item android:id="@+id/item2" android:title="Second Option">
        <menu>
            <item android:id="@+id/item3" android:title="Third Option"/>
            <item android:id="@+id/item4" android:title="Fourth Option"/>
        </menu>
    </item>
</menu>
```
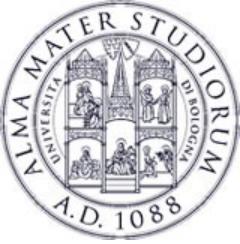
Menu: inflate the menu

```
public boolean onCreateOptionsMenu(Menu menu) {

    super.onCreateOptionsMenu(menu);


    getMenuInflater().inflate(R.menu.myMenu, menu);


    menu.findItem(R.id.menu_first).setIntent(new Intent(this, First.class));


    return true;

}
```

Toast: making a toast

❖ Tiny messages over the Activity

❖ Used to signal to the user confirmation, little errors

❖ Can control the duration of the Toast

❖ As simple as:

Toast msg = Toast.makeText(this, "Toast!", Toast.LENGTH_SHORT).show();

❖ Used to interact with the user

❖ Little messages, easy answers

❖ Different kinds:
  - ❖ AlertDialog
  - ❖ ProgressDialog
  - ❖ DatePickerDialog
  - ❖ TimePickerDialog
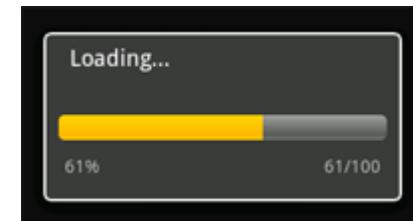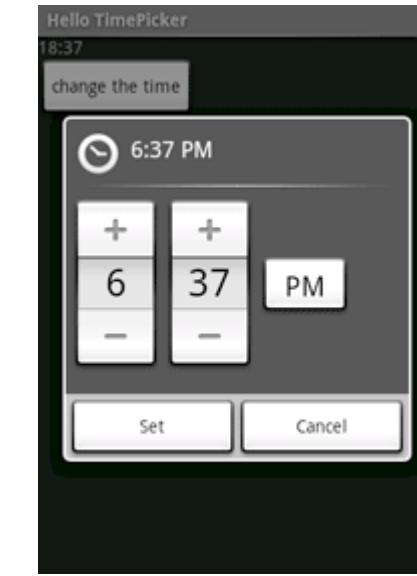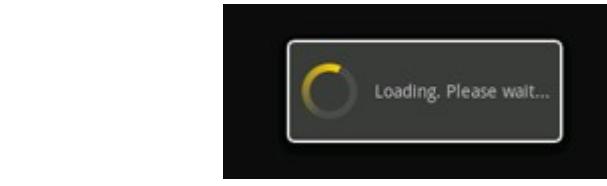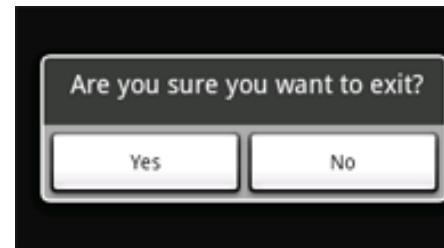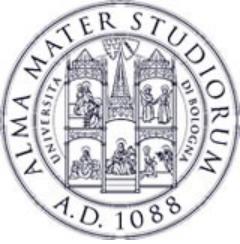
Dialog: AlertDialog

```java
AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setMessage("Are you sure you want to exit?").setCancelable(false);

builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {

        public void onClick(DialogInterface dialog, int id) {

                MenuExampleActivity.this.finish();

        }

});

builder.setNegativeButton("No", new DialogInterface.OnClickListener() {

        public void onClick(DialogInterface dialog, int id) {

                dialog.cancel();

        }

});

AlertDialog alert = builder.create();          alert.show();
```
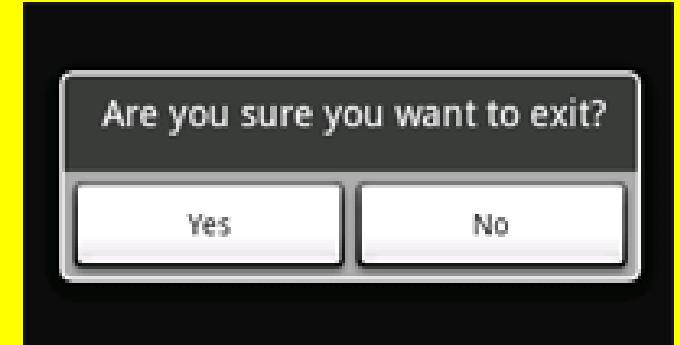
Dialog: AlertDialog with a list

```
final CharSequence[] items = {"Red", "Green", "Blue"};
AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle("Pick a color");

builder.setItems(items, new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int item) {
        Toast.makeText(getApplicationContext(), items[item],
Toast.LENGTH_SHORT).show();
    }
});// OR

builder.setSingleChoiceItems(items, -1, new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int item) {
        Toast.makeText(getApplicationContext(), items[item],
            Toast.LENGTH_SHORT).show();
    }
});
AlertDialog alert = builder.create();
```