MMI 2: Mobile Human-Computer Interaction

Android (3)

Prof. Dr. Michael Rohs

michael.rohs@ifi.lmu.de

Mobile Interaction Lab, LMU München

Review

- What is an “Activity”
- What is an “Intent”?
- What is “Intent Resolution”?

BASIC GRAPHICS

Basic Graphics: Drawing

- Screen drawing in View.onDraw
- Canvas class for “draw” calls
  - drawRect, drawLines, drawCircle, drawText, etc.
  - Transformation matrix
  - Clipping
- Paint class
  - Describes colors and drawing styles
  - Examples: anti-aliasing, stroke width, text size, etc.
- Bitmap class for offscreen drawing
  - Explicit creation of canvas and bitmap
  - Canvas draws into the bitmap

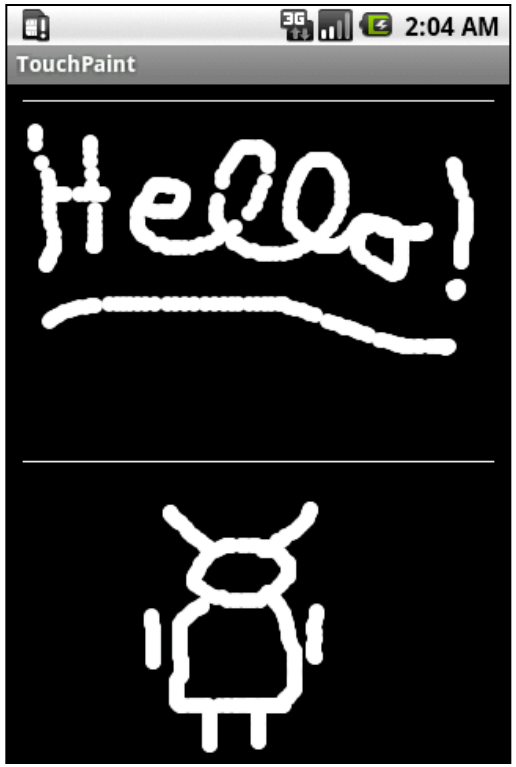
Touch Input Painting (Off-Screen Image)

```
public class MyView extends View {
...
private Bitmap bitmap;
private Canvas canvas;

protected void onSizeChanged(int w, int h, int oldw, int oldh) {
    bitmap = Bitmap.createBitmap(w, h, Bitmap.Config.RGB_565);
    canvas = new Canvas(bitmap);
}

protected void onDraw(Canvas c) {
    if (bitmap != null) c.drawBitmap(bitmap, 0, 0, null);
}

public boolean onTouchEvent(MotionEvent e) {
    if (canvas != null) {
        int x = (int)e.getX(); int y = (int)e.getY();
        canvas.drawCircle(x, y, 3, paint);
        invalidate();
    }
    return true;
}   }
```



Periodic Events

```
public class MainActivity extends Activity {

private static final int TICK_MSG = 1;
private static final int TICK_DELAY = 300; // ms

public void onCreate(Bundle savedInstanceState) {
    ...
    tickHandler.removeMessages(TICK_MSG);
    tickHandler.sendMessageDelayed(tickHandler.obtainMessage(TICK_MSG), TICK_DELAY);
}

private Handler tickHandler = new Handler() {
    public void handleMessage(Message msg) {
        switch (msg.what) {
        case TICK_MSG: {
            gameView.tick();
            tickHandler.sendMessageDelayed(tickHandler.obtainMessage(TICK_MSG), TICK_DELAY);
            break;
        }
        default: super.handleMessage(msg);
} } };
```

Process Event

```
private int xx = -1, yy = -1, radius = 1;
private Random rnd = new Random();

public void tick() {
    if (xx < 0) {
        xx = rnd.nextInt(320);
        yy = rnd.nextInt(430) + 20;
        radius = 1;
    }
    paint.setARGB(255, 255, 255, 255);
    canvas.drawCircle(xx, yy, radius, paint);
    invalidate();
    radius++;
}
```

UI Components

- Common Controls
- Layout Managers
- Menus
- Dialogs

Common Controls

- Predefined user interface elements (“controls”, “widgets”)
  - Define basic interaction patterns
  - Semantics known to users
- Standard widgets
  - Text fields, buttons, lists, grids, date & time controls
- Android-specific controls
  - MapView (display a geographic map)
  - Gallery (display a list of photos)

Handling Button Click Events

- XML

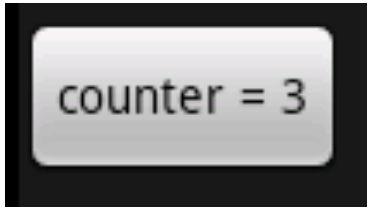
```
<Button android:id="@+id/button1" android:text="Basic Button"
android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

- Java

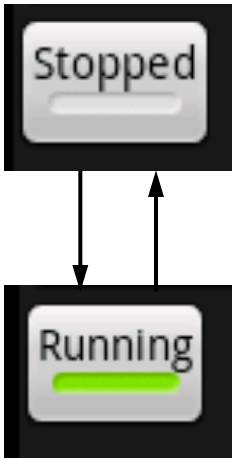
```
public class MainActivity extends Activity implements
    View.OnClickListener {
    public void onCreate(Bundle savedInstanceState) {
        ...
        Button b = (Button) findViewById(R.id.button1);
        b.setOnClickListener(this);
    }

    private int counter = 0;

    public void onClick(View v) {
        Button b = (Button)v;
        b.setText("counter = " + (++counter));
    }
}
```



ToggleButton: Two States

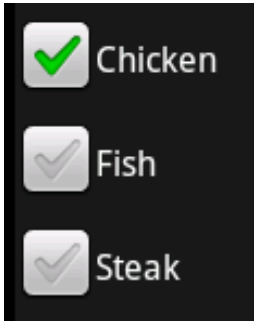


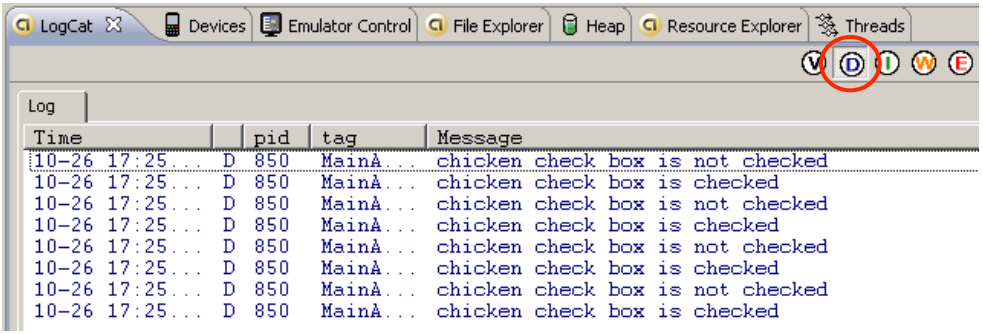
- XML

```
<ToggleButton android:id="@+id/cctglBtn"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textOn="Running" android:textOff="Stopped" />
```

- Default text
  - “On” for state on
  - “Off” for state off

CheckBox





- XML

```
<LinearLayout android:orientation="vertical" ... >
    <CheckBox android:id="@+id/chicken" android:text="Chicken" ... />
    <CheckBox android:id="@+id/fish" android:text="Fish" ... />
    <CheckBox android:id="@+id/steak" android:text="Steak" ... />
</LinearLayout>
```

- Java

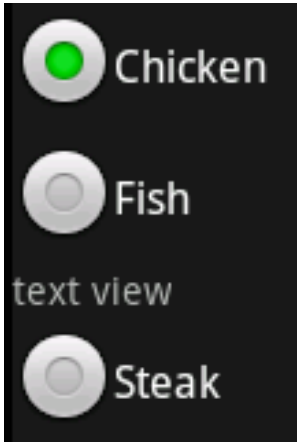
```
CheckBox cb = (CheckBox) findViewById(R.id.chicken);
cb.setChecked(true);
cb.setOnCheckedChangeListener(new OnCheckedChangeListener() {
  public void onCheckedChanged(CompoundButton b, boolean isChecked) {
    Log.d("MainActivity", "chicken check box is " +
      (isChecked ? "" : "not ") + "checked");
  }
});
```

Radio Button

- XML

```
<LinearLayout android:orientation="vertical"
        android:layout_width="wrap_content“
        android:layout_height="wrap_content">
    <RadioGroup android:layout_width="wrap_content“
            android:layout_height="wrap_content">
        <RadioButton android:text="Chicken"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <RadioButton android:text="Fish"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        ...
    </RadioGroup>
</LinearLayout>
```

- Radio groups can contain arbitrary views



Text Controls

- TextView
  - Display text, no editing
  - Automatic link creation if text contains URLs

    `android:autoLink="all"`
- EditText
  - Text editing
  - Expands as needed
  - Correct spelling errors

    `android:autoText="true"`
- AutoCompleteTextView
  - Displays suggestions for word completion
- MultiCompleteTextView
  - Displays suggestions for each word

TextView Automatic Link Creation

- XML

```
<TextView android:id="@+id/nameValue" ... android:autoLink="all" />
```

- Java

```
setContentView(R.layout.test2);
TextView nameValue = (TextView)findViewById(R.id.nameValue);
nameValue.setText("Visit www.tu-berlin.de or email info@tu-berlin.de");
```

Visit www.tu-berlin.de or email info@tu-berlin.de

- Using class Linkify

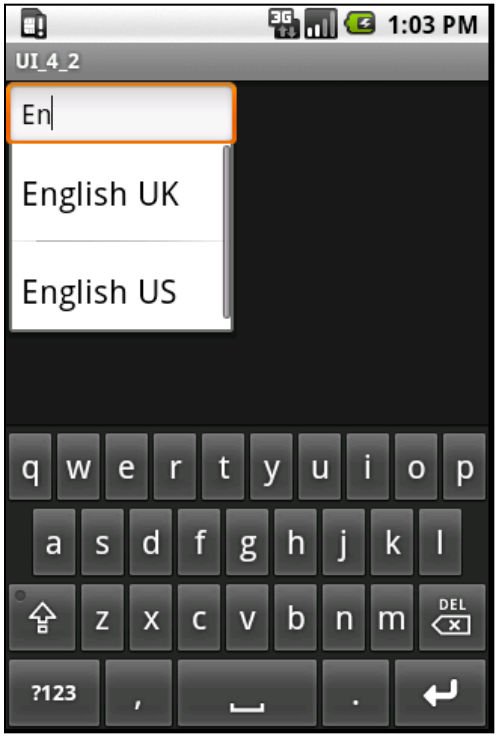
```
Linkify.addLinks(nameValue, Linkify.ALL);
```

EditView Input Type

- android:inputType="*textEmailAddress*"

- android:inputType=*“phone“*

AutoCompleteTextView



- XML

```
<AutoCompleteTextView
    android:id="@+id/auto" ... />
```

- Java

```
AutoCompleteTextView actv =
    (AutoCompleteTextView) findViewById(R.id.auto);
ArrayAdapter<String> aa = new ArrayAdapter<String>(this,
        android.R.layout.simple_dropdown_item_1line,
        new String[] {"English UK", "English US", "Hebrew", "Hindi", ... });
    actv.setAdapter(aa);
```

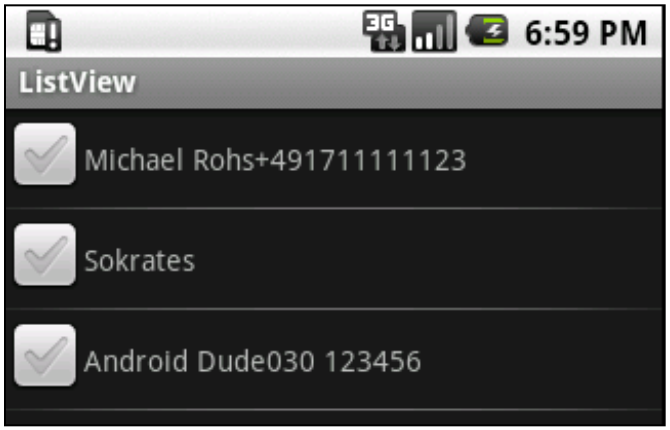
- Adapter
  - Resource ID for showing a single item
  - The data to use

List Controls

- Vertical list of items
- Usage
  - Derive from android.app.ListActivity.ListActivity
  - Set a ListView
  - Setting data for the list view via setListAdapter: SimpleAdapter, SimpleCursorAdapter
- Definition of list item in list_item.xml

```
<LinearLayout ...>
    <CheckBox android:id="@+id/checkbox" ... />
    <TextView android:id="@+id/textview1" ... />
    <TextView android:id="@+id/textview2" ... />
    ...
</LinearLayout>
```

List Controls



- Showing names and numbers from contacts database

```
public class ListDemoActivity extends ListActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Cursor c = getContentResolver().query(People.CONTENT_URI,
                                              null, null, null, null);
        startManagingCursor(c);
        String[] cols = new String[] { People.NAME, People.NUMBER };
        int[] colIds = new int[] { R.id.textview1, R.id.textview2 };
        SimpleCursorAdapter adapter = new
            SimpleCursorAdapter(this, R.layout.list_item, c, cols, colIds);
        setListAdapter(adapter);
    }
}
```

AndroidManifest.xml needs:
<uses-permission android:name="android.permission.READ_CONTACTS" />

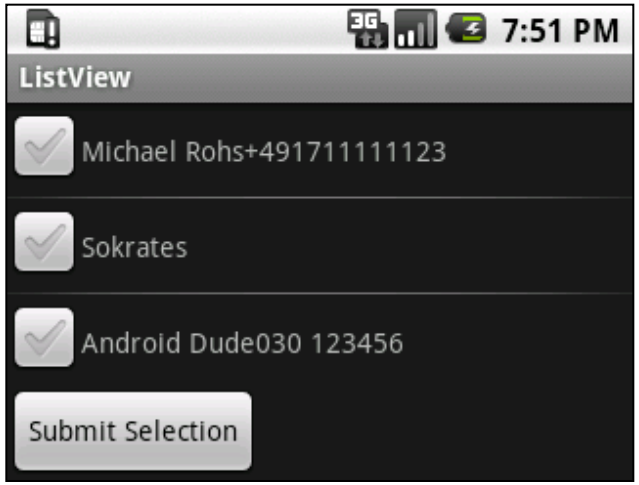
Using a Custom List View

- /res/layout/list.xml

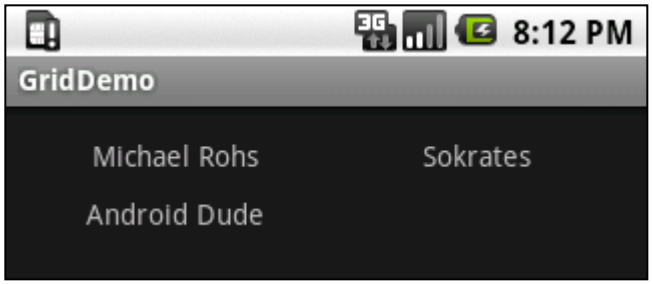
```
<LinearLayout android:orientation="vertical" ...>
    <LinearLayout android:orientation="vertical" ...>
    <ListView android:id="@android:id/list"
            android:layout_width="fill_parent"
            android:layout_height="0dip" android:layout_weight="1"
            android:stackFromBottom="true"
            android:transcriptMode="normal" />
    </LinearLayout>
    <Button android:text="Submit Selection"
    ... />
</LinearLayout>
```

- Java

```
setContentView(R.layout.list);
```



GridView



- XML

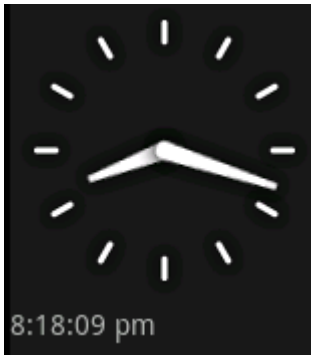
```
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/dataGrid" android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:padding="10px"
    android:verticalSpacing="10px" android:horizontalSpacing="10px"
    android:numColumns="auto_fit" android:columnWidth="100px"
    android:stretchMode="columnWidth" android:gravity="center" />
```

- Java

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.gridview);
    GridView gv = (GridView) this.findViewById(R.id.dataGrid);
    Cursor c = getContentResolver().query(People.CONTENT_URI, null, null, null, null);
    startManagingCursor(c);
    String[] cols = new String[] { People.NAME };
    int[] colIDs = new int[] { R.id.textview };
    SimpleCursorAdapter adapter = new SimpleCursorAdapter(
                                        this, R.layout.grid_item, c, cols, colIDs);
    gv.setAdapter(adapter);
}
```

Android Specific Controls

- DatePicker and TimePicker
- AnalogClock and DigitalClock
- MapView
- Gallery



LAYOUT MANAGERS

LayoutManagers

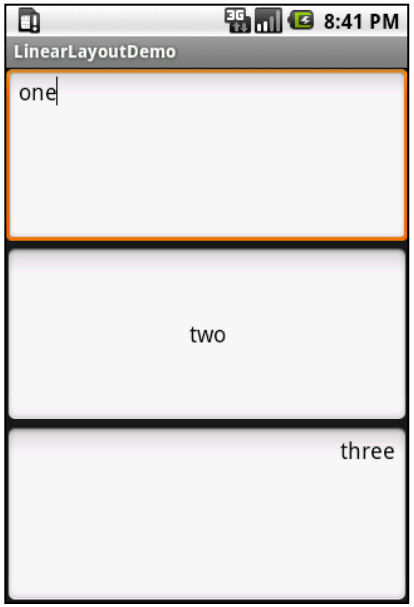
- LayoutManagers
  - Are containers for views (children)
  - Have specific strategy for controlling children's size and position
- Layout Managers in Android
  - LinearLayout: horizontal or vertical arrangement
  - TableLayout: tabular form
  - RelativeLayout: arrange children relative to one another or parent
  - AbsoluteLayout: absolute coordinates
  - FrameLayout: dynamically change controls
  - GridLayout: rectangular grid of child views
- Layout_width and layout_height
  - fill_parent: child wants to fill available space within the parent
  - wrap_content: child wants to be large enough to fit its content

Screen Configurations

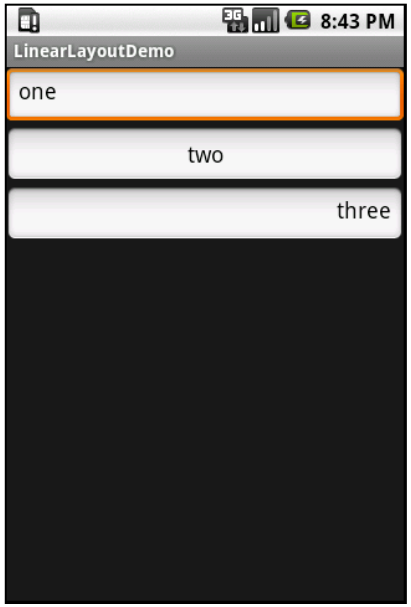
- Configurations
  - Portrait
  - Landscape
  - Square
- Different layouts for different configurations
  - Screen resolutions
- Configuration-specific resource subdirectories
  - /res/layout-port /res/drawable-port
  - /res/layout-land /res/drawable-land
  - /res/layout-square /res/drawable-square
  - /res/layout /res/drawable (default)

LinearLayout

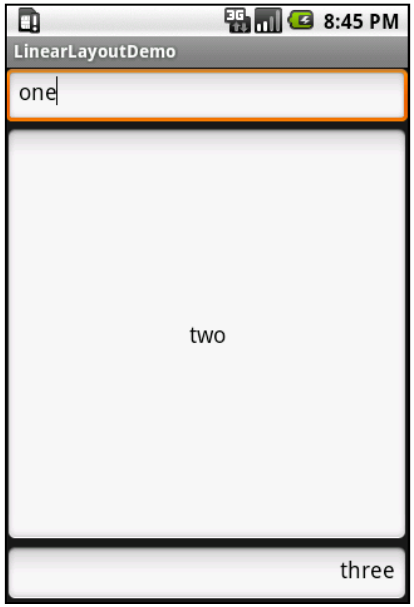
- Orientation: horizontal or vertical
- Gravity: alignment (left, right, center, top, etc.)
- Weight: size importance of one child relative to others



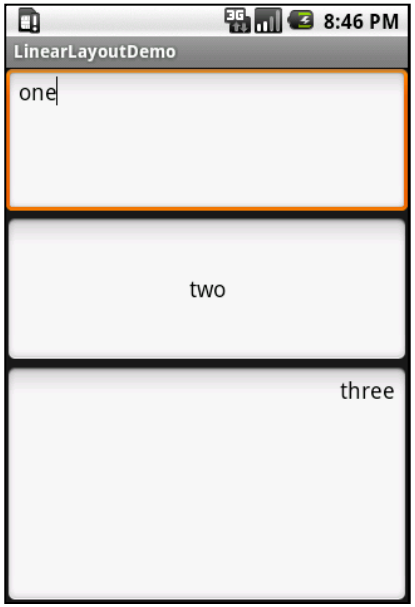
Weights:
1.0, 1.0, 1.0



Weights:
0.0, 0.0, 0.0



Weights:
0.0, 1.0, 0.0

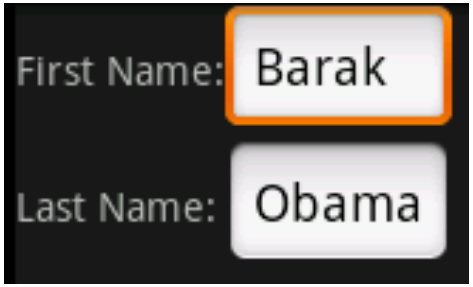


Weights:
0.5, 0.5, 1.0

Example LinearLayout with Weights

```
<LinearLayout android:orientation="vertical"
android:layout_width="fill_parent“ android:layout_height="fill_parent">
   <EditText android:layout_width="fill_parent"
android:layout_weight="0.5" android:layout_height="wrap_content"
   android:text="one" android:gravity="left" />
   <EditText android:layout_width="fill_parent"
   android:layout_weight="0.5" android:layout_height="wrap_content"
   android:text="two" android:gravity="center" />
   <EditText android:layout_width="fill_parent"
   android:layout_weight="1.0" android:layout_height="wrap_content"
   android:text="three" android:gravity="right" />
</LinearLayout>
```

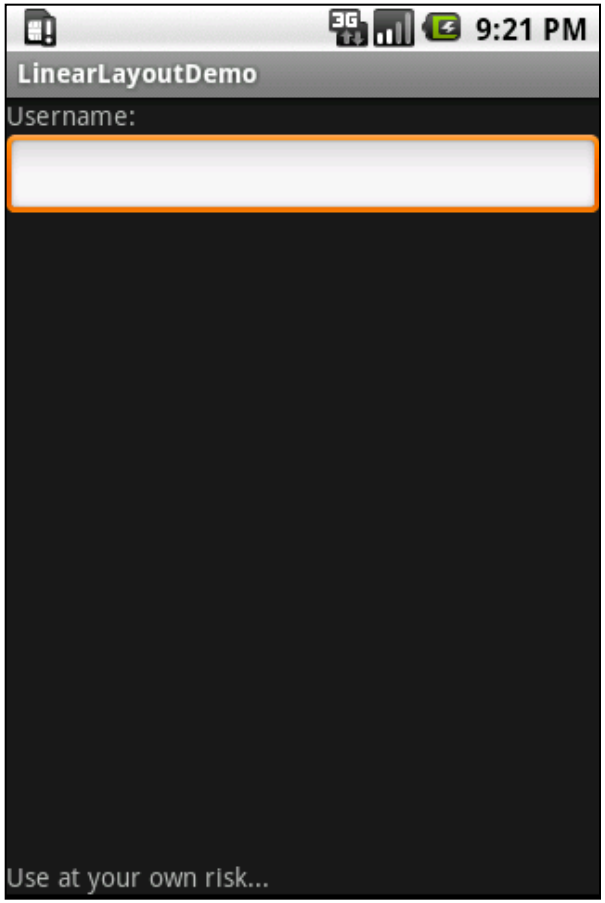
TableLayout



- Extension of LinearLayout
- Example:

```
<TableLayout android:layout_width="fill_parent"
   android:layout_height="fill_parent">
<TableRow>
   <TextView android:layout_width="wrap_content"
   android:layout_height="wrap_content" android:text="First Name:" />
   <EditText android:layout_width="wrap_content"
   android:layout_height="wrap_content" android:text="Barak" />
</TableRow>
<TableRow>
   <TextView android:layout_width="wrap_content"
   android:layout_height="wrap_content" android:text="Last Name:" />
   <EditText android:layout_width="wrap_content"
   android:layout_height="wrap_content" android:text="Obama" />
</TableRow>
</TableLayout>
```

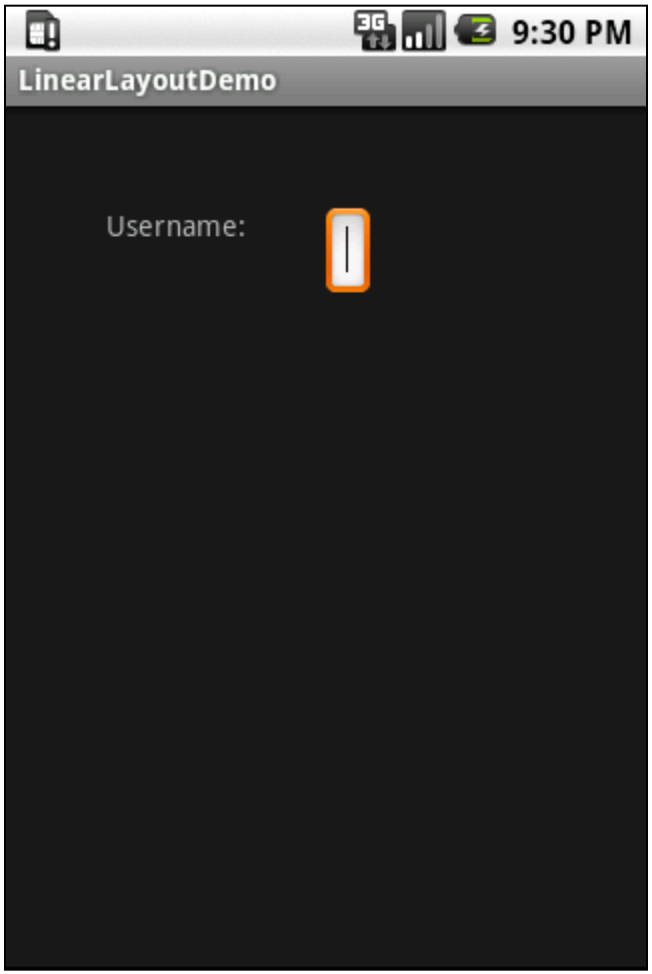
RelativeLayout



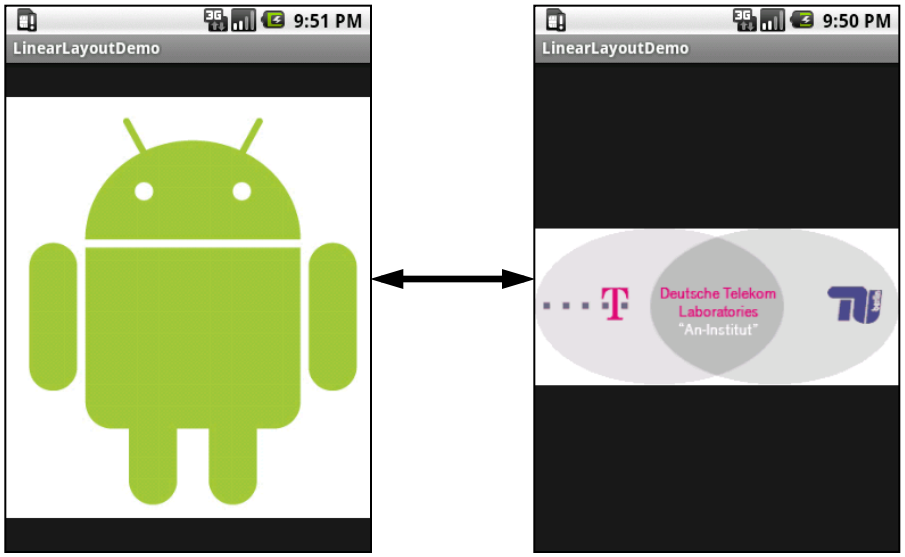

```
<RelativeLayout android:layout_width="fill_parent"
   android:layout_height="wrap_content">
   <TextView android:id="@+id/userNameLbl"
      android:text="Username: "
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true" />
   <EditText android:id="@+id/userNameText"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_below="@id/userNameLbl" />
   <TextView android:id="@+id/disclaimerLbl"
      android:text="Use at your own risk... "
      android:layout_width="fill_parent" android:layout_height="wrap_content"
      android:layout_alignParentBottom="true" />
</RelativeLayout>
```

AbsoluteLayout

```
<AbsoluteLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent" >
   <TextView android:text="Username:"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_x="50px"
      android:layout_y="50px" />
   <EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_x="160px"
      android:layout_y="50px" />
</AbsoluteLayout>
```



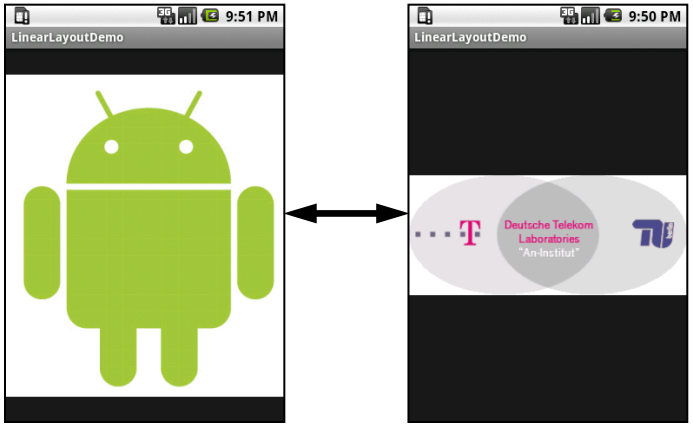
FrameLayout



- Displays one item at a time
- Stacks items if multiple visible
- XML

```
<FrameLayout... >
<ImageView
    android:id="@+id/imgView1"
    android:src="@drawable/one"
    android:scaleType="fitCenter"
    android:layout_width="fill_parent" android:layout_height="fill_parent" />
<ImageView android:id="@+id/imgView2"
    android:src="@drawable/two"
    android:scaleType="fitCenter"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:visibility="gone" />
</FrameLayout>
```

FrameLayout




```
public class FrameActivity extends Activity {
   protected void onCreate(Bundle state) {
      super.onCreate(state);
      setContentView(R.layout.frame);
      ImageView one = (ImageView) findViewById(R.id.oneImgView);
      ImageView two = (ImageView) findViewById(R.id.twoImgView);
      one.setOnClickListener(new OnClickListener() {
         public void onClick(View view) {
            ImageView two = (ImageView) findViewById(R.id.twoImgView);
            two.setVisibility(View.VISIBLE);
            view.setVisibility(View.GONE);
         }});
      two.setOnClickListener(new OnClickListener() {
         public void onClick(View view) {
            ImageView one = (ImageView) findViewById(R.id.oneImgView);
            one.setVisibility(View.VISIBLE);
            view.setVisibility(View.GONE);
         }});
}}
```

Dialogs

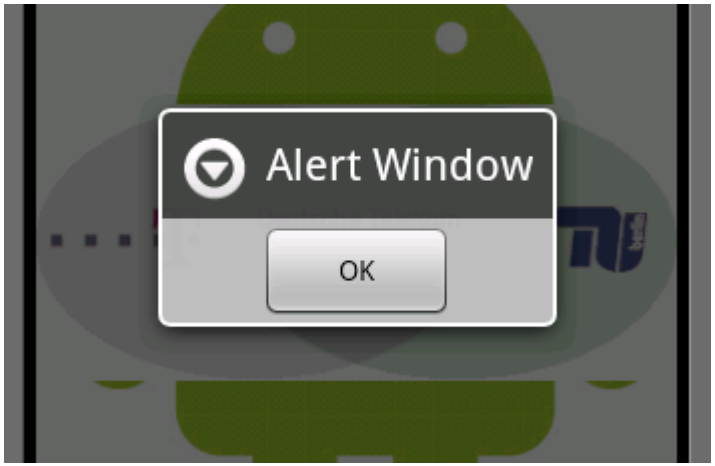
Alert Dialogs

- Alert dialog examples
  - Respond to prompt
  - Pick item or option from list
  - View progress
- Steps
  - Construct android.app.AlertDialog.Builder object
  - Set data (list of items) and parameters (e.g. number of buttons)
  - Set callback methods for buttons
  - Build and show the dialog

Example Alert Dialog

- Java

```
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.menu_testPick) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Alert Window");
        builder.setPositiveButton("OK", listener);
        AlertDialog ad = builder.create();
        ad.show();
        return true;
    }
    ...
}
```



Prompt Dialog

- Asks for text input and provides OK and Cancel buttons
- Steps
  - Create layout in XML
  - Load layout into view class
  - Construct Builder object
  - Set view in Builder object
  - Set buttons with their callbacks
  - Create and show the dialog

Example Prompt Dialog

- View definition

```
<LinearLayout android:orientation="vertical" ...>
    <TextView android:id="@+id/promptmessage"
              android:text="Your text goes here" ... />
    <EditText android:id="@+id/editText_prompt" ... />
</LinearLayout>
```

- On selection of menu item

```
LayoutInflater li = LayoutInflater.from(this);
View view = li.inflate(R.layout.prompt_dialog, null);
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Prompt");
builder.setView(view);
PromptListener pl = new PromptListener(view);
builder.setPositiveButton("OK", pl);
builder.setNegativeButton("Cancel", pl);
AlertDialog ad = builder.create();
ad.show();
```

Example Prompt Dialog

- Prompt dialog listener

```
public class PromptListener
implements android.content.DialogInterface.OnClickListener
{
    private View promptDialogView = null;
    private String promptReply = null;
    public PromptListener(View inDialogView) { promptDialogView = inDialogView; }
    public void onClick(DialogInterface v, int buttonId) {
        if (buttonId == DialogInterface.BUTTON_POSITIVE) {
            promptReply = getPromptText();
        }
    }
    private String getPromptText() {
        EditText et = (EditText) promptDialogView.findViewById(R.id.editText_prompt);
        return et.getText().toString();
    }
    public String getPromptReply() { return promptReply; }
}
```

Managed Dialogs

- Reusing previously created dialog instances
- Steps
  - Assign a unique ID (x) to the dialog
  - Tell the system to show the dialog with ID x
  - Android checks whether a dialog with ID x exists, if so call onPrepareDialog, else call onCreateDialog passing x
  - In onCreatDialog create appropriate dialog for ID
  - Android shows the dialog
  - Callbacks inform then button are clicked

Example Managed Dialog

```
public class MainActivity extends Activity {
    public final static int DIALOG_ID = 1;
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.menu_test) {
            showDialog(DIALOG_ID); return true;
        }
        return super.onOptionsItemSelected(item);
    }
    protected Dialog onCreateDialog(int id) {
        Log.d("MainActivity", "onCreateDialog " + id);
        if (id == DIALOG_ID) {
            // create dialog...
            return dialog;
        }
        return null;
    }
    protected void onPrepareDialog(int id, Dialog dialog) {
        Log.d("MainActivity", "onPrepareDialog " + id);
    }
```

ANIMATION

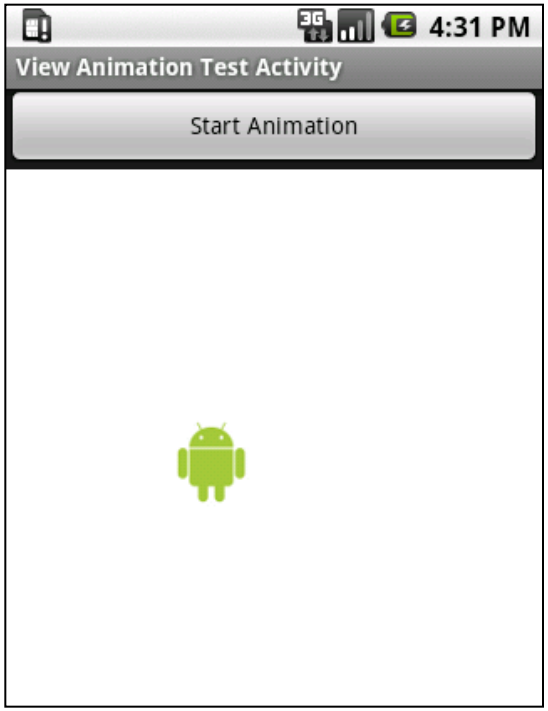
Animation

- Change color, position, size, orientation over time
- Types of animation
  - Frame-by-frame (aka draw) animation: play a series of frames
  - Layout animation: animate views inside a container view
  - View animation: animate general-purpose view
  - Property animation: introduced in Android 3.0, most flexible
- Tweening animation
  - Tweening = inbetweening
  - Generate intermediate frames between key frames
  - View 1 evolves smoothly into view 2
  - Periodically change parameters of a view
- Developer guide
  - http://developer.android.com/guide/topics/graphics/animation.html

Frame-by-Frame Animation Example

/res/layout/frame_animations_layout.xml

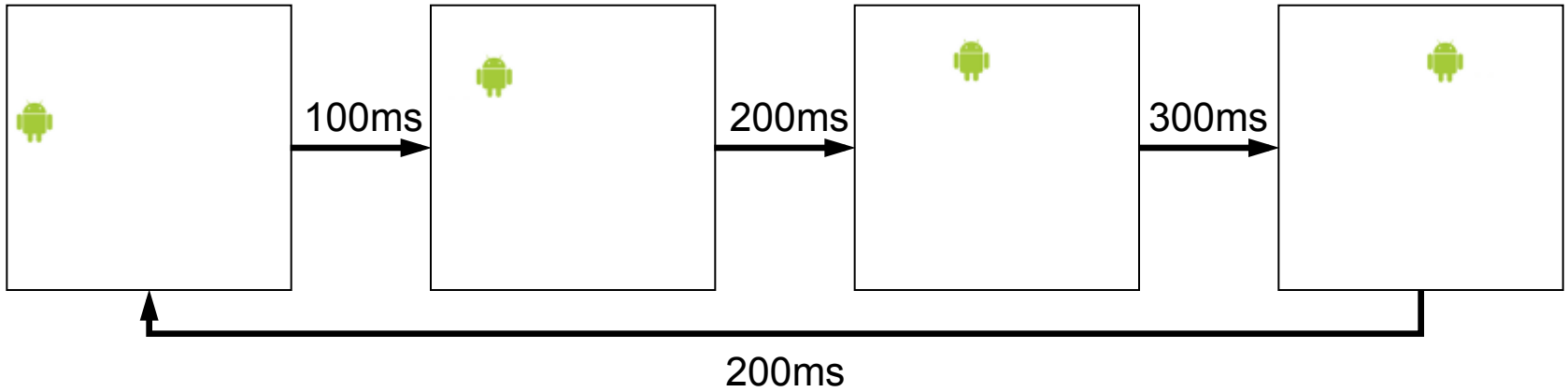
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
  android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent" >
   <Button android:id="@+id/startAnimationButton"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Start Animation" />
   <ImageView android:id="@+id/animView"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content" />
</LinearLayout>
```



Frame-by-Frame Animation Example

/drawable/animlist.xml

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/
   android"
android:oneshot="false">
   <item android:drawable="@drawable/img1" android:duration="100" />
   <item android:drawable="@drawable/img2" android:duration="200" />
   <item android:drawable="@drawable/img3" android:duration="300" />
   <item android:drawable="@drawable/img4" android:duration="200" />
</animation-list>
```



Frame-by-Frame Animation Example

FrameAnimationActivity.java

```
public class FrameAnimationActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.frame_animations_layout);
        Button b = (Button) findViewById(R.id.startAnimationButton);
        b.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) { animate(); }
        });
    }
    private void animate() {
        ImageView imgView =  (ImageView) findViewById(R.id.animView);
        imgView.setBackgroundResource(R.drawable.animlist);
        AnimationDrawable frameAnimation = (AnimationDrawable)
                imgView.getBackground();
        frameAnimation.start(); // stop(), setOneShot(), addFrame(...), ...
    }
}
```

Frame-by-Frame Animation Example

MainActivity.java

```
public class MainActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Intent intent = new Intent(this, FrameAnimationActivity.class);
        startActivity(intent);
    }
}
```

- Declare activity in AndroidManifest.xml

```
<activity android:name=".FrameAnimationActivity"
                android:label="View Animation Test Activity" />
```

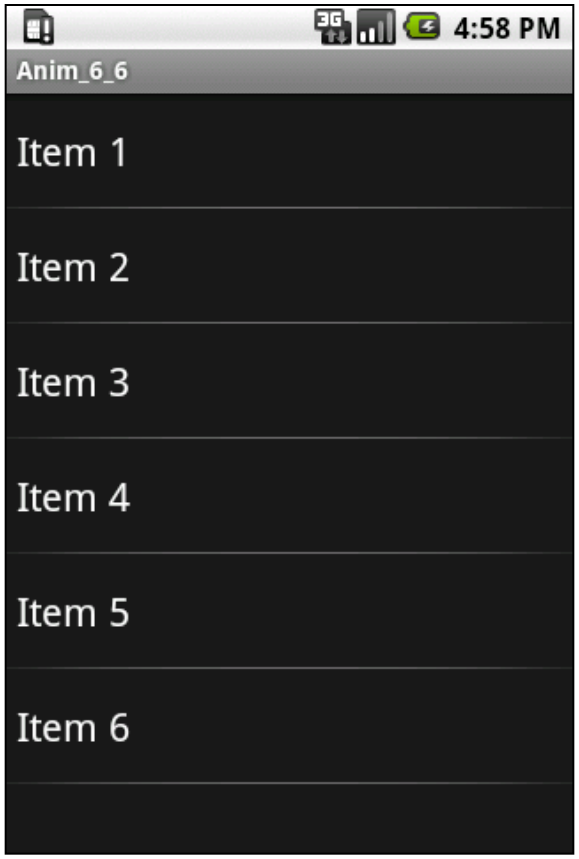
Layout Animation

- On all widgets derived from ViewGroup (ListView, GridView)
- Applies tweening to each component of the view group
- Tweening animation types
  - Scale animation
  - Rotate animation
  - Translate animation
  - Alpha animation (0 = fully transparent, 1 = fully opaque)
- Parameters for each tweening animation
  - Initial parameter value (from)
  - Final parameter value (to)
  - Duration (ms)
- Package android.view.animation

Layout Animation Example

/res/layout/list_layout.xml

```
<LinearLayout android:orientation="vertical" ...>
   <ListView android:id="@+id/list_view_id"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:layoutAnimation=
          "@anim/list_layout_controller" />
</LinearLayout>
```



Layout Animation Example

```
public class LayoutAnimationActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.list_layout);
        setupListView();
    }
    private void setupListView() {
        String[] listItems = new String[] { "Item 1", "Item 2", "Item 3", ... };
        ArrayAdapter<String> listItemAdapter =
            new ArrayAdapter<String>(this,
                                     android.R.layout.simple_list_item_1, listItems);
        ListView lv = (ListView) findViewById(R.id.list_view_id);
        lv.setAdapter(listItemAdapter);
    }
}
```

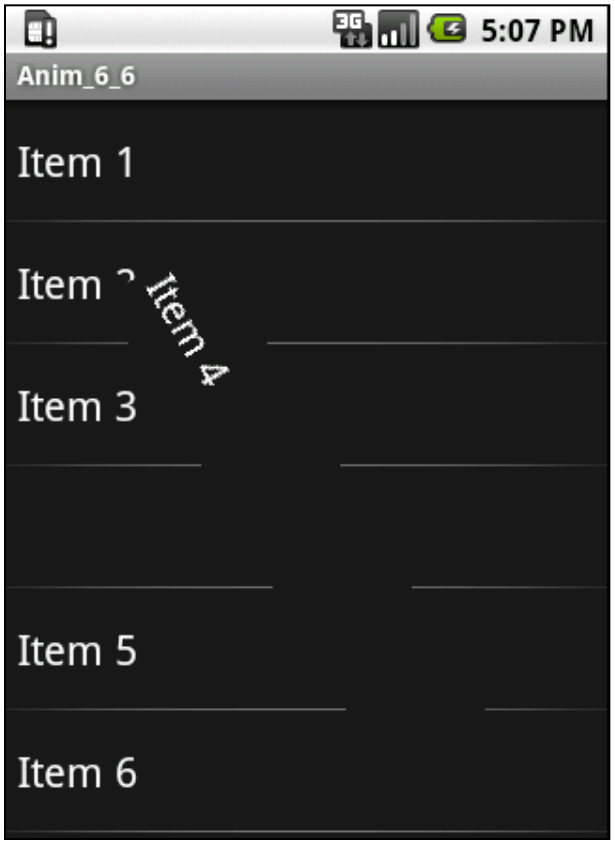
Layout Animation Example

/anim/rotate.xml

```
<rotate xmlns:android="http://schemas…"
  android:interpolator="@android:anim/accelerate_interpolator"
  android:fromDegrees="0.0"
  android:toDegrees="360.0"
  android:pivotX="50%"
  android:pivotY="50%"
  android:duration="1000" />
```

/anim/list_layout_controller.xml

```
<layoutAnimation xmlns:android="http://schem...
  android:delay="100%"
  android:animationOrder="reverse"
  android:animation="@anim/rotate" />
```



Layout Animations (Alpha, Translate)

- Alpha

```
<alpha xmlns:android="http://schemas…"
   android:interpolator="@android:anim/accelerate_interpolator"
   android:fromAlpha="0.0" android:toAlpha="1.0"
   android:duration="2000" />
```

- Alpha & translate combined

```
<set xmlns:android="http://schemas…"
      android:interpolator="@android:anim/accelerate_interpolator">
   <translate android:fromYDelta="-300%" android:toYDelta="0"
                  android:duration="1000" />
   <alpha android:fromAlpha="0.0" android:toAlpha="1.0"
                   android:duration="1000" />
</set>
```

View Animation

- Animate arbitrary views by modifying transformation matrix
  - Maps view to screen

View Animation Example

/res/layout/list_layout.xml

```
<LinearLayout xmlns:android="http://schemas..." ... >
    <Button android:id="@+id/btn_animate"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Start Animation" />
    <ImageView android:id="@+id/image_view_id"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:src="@drawable/androidlogo" />
</LinearLayout>
```

View Animation Example

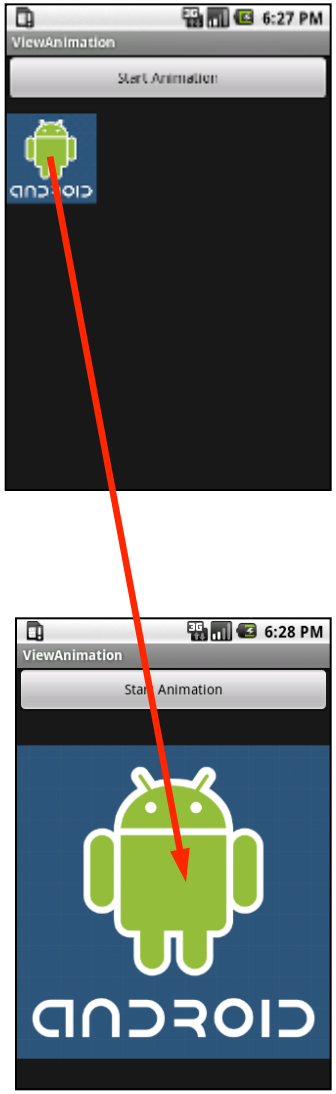
```
public class ViewAnimationActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.list_layout);

        Button b = (Button)this.findViewById(R.id.btn_animate);
        b.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                ImageView iv = (ImageView) findViewById(R.id.image_view_id);
                iv.startAnimation(new ViewAnimation());
            }
        });

    }
}
```

View Animation Example

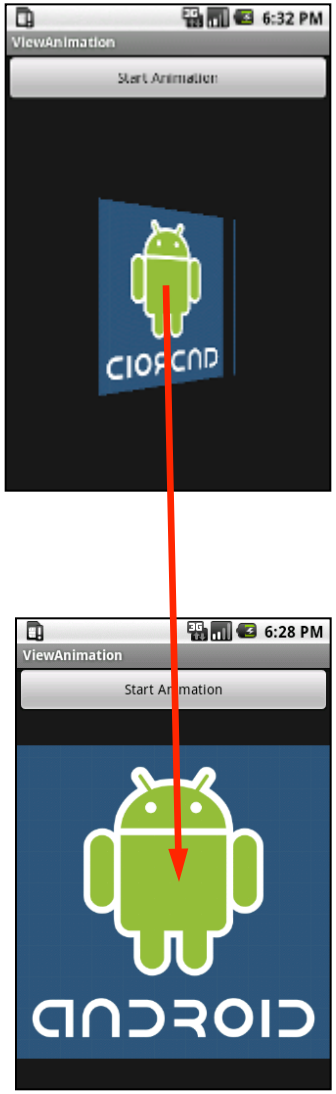
```
public class ViewAnimation extends Animation {
    public void initialize(int width, int height,
                           int parentW, int parentH) {
        super.initialize(width, height, parentW, parentH);
        setDuration(2500);
        setFillAfter(true);
        setInterpolator(new LinearInterpolator());
    }
    protected void applyTransformation(
            float interpolatedTime, Transformation t) {
        Matrix matrix = t.getMatrix();
        matrix.setScale(interpolatedTime, interpolatedTime);
    }
}
```



View Animation Example

```
Camera camera = new Camera(); // android.graphics.Camera

protected void applyTransformation(
        float interpolatedTime, Transformation t) {
    Matrix matrix = t.getMatrix();
    camera.save();
    camera.translate(0, 0, 1300 - 1300 * interpolatedTime);
    camera.rotateY(360.0f * interpolatedTime);
    camera.getMatrix(matrix);
    matrix.preTranslate(-w/2, -h/2);
    matrix.postTranslate(w/2, h/2);
    camera.restore();
}
```



Class android.graphics.Matrix

- 3x3 matrix
  - Linear transformations on R^3
  - Affine transformations and perspective projections on R^2 (homogeneous coordinates: (x,y) → (x,y,1))
- Translation
  - m.setTranslate(tx,ty)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Scaling
  - m.setScale(vx,vy)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} v_x & 0 & 0 \\ 0 & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Rotation
  - m.setRotate(theta)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Class android.graphics.Matrix

- Matrix multiplication = composition of transformations
  - M.preTranslate(tx,ty) means M' = M * T(tx, ty)
  - M.postTranslate(tx,ty) means M' = T(tx, ty) * M
  - M.preScale(sx, sy) means M' = M * S(tx, ty)
  - M.postScale(sx, sy) means M' = S(tx, ty) * M
  - ...
- Example: translate to origin, scale to half, translate back
  - M.setScale(0.5, 0.5);
  - M.preTranslate(-cx, -cy);
  - M.postTranslate(cx, cy);
  - Resulting matrix: M = T(cx, cy) * S(0.5, 0.5) * T(-cx, -cy)

Property Animation

- Animate object properties over time
- Characteristics of a property animation
  - Duration
  - Time interpolation (acceleration / deceleration)
  - Repeat count and behavior (number of times to repeat, reverse)
  - Animator sets (simultaneous animations)
  - Frame refresh delay (frame rate)

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

ValueAnimator

- Responsibility
  - Keeps track of animation state
    - Property start/end values, duration, repeat count, listeners, etc.
  - Computes interpolated values
  - Sets these values on animated object
- Elapsed fraction
  - Represents elapsed time (0% to 100%)
- Interpolated fraction
  - Maps time depending on behavior (linear, accelerated, etc.)
  - Uses TimeInterpolator
- Property value
  - Maps interpolated fraction to actual property value
  - Uses TypeEvaluator

ObjectInterpolator (extends ValueInterpolator)

- Modify object properties
  - Factory method: target object, property name, start/end values
  - Requires setter and getter methods
  - Example: “rotation” requires “setRotation” and “getRotation”
- Example

```
void animate() {
    ImageView iv = (ImageView) findViewById(R.id.image_view_id);
    ObjectAnimator anim = ObjectAnimator.ofFloat(iv, "rotation", 0, 90);
    anim.setDuration(3000);
    anim.setInterpolator(new BounceInterpolator());
    anim.start();
}
```

View Properties for ObjectInterpolator

- Example: ObjectAnimator.*ofFloat(iv, "rotation", 0f, 90f);*

- translationX, translationY (delta to left/top coordinates)
- rotation (2D, around z-axis)
- rotationX, rotationY (3D, around x-/y-axis)
- scaleX, scaleY (scaling around pivot point)
- pivotX, pivotY (center by default)
- x, y (location)
- alpha (transparency 0..1)
- and more: http://developer.android.com/reference/android/view/View.html

Predefined Time Interpolators

- LinearInterpolator
- AccelerateInterpolator
- DecelerateInterpolator
- AccelerateDecelerateInterpolator (default)
- BounceInterpolator
- AnticipateInterpolator
- OvershootInterpolator
- AnticipateOvershootInterpolator
- CycleInterpolator

Type Evaluators

- Allows creating animations on arbitrary object types
- “Linear interpolation” on arbitrary object types

- Interface definition

```
interface TypeEvaluator<T> {
        T evaluate(float fraction, T startValue, T endValue);
}
```

- Example: ArgbEvaluator knows how to evaluate between two RGB values (including alpha value)

Coordinating Multiple Animations

- Complex animations have multiple animated properties
  - Relative order of component animations
  - Relative starting times
  - Operations to perform on completion
- Example: Let ball fall, deform, bounce off ground, fade
- AnimatorSet; before, after, with

```
AnimatorSet bouncer = new AnimatorSet();
bouncer.play(bounceAnim).before(squashAnim1);
bouncer.play(squashAnim1).with(squashAnim2);
bouncer.play(squashAnim1).with(stretchAnim1);
bouncer.play(squashAnim1).with(stretchAnim2);
bouncer.play(bounceBackAnim).after(stretchAnim2);
animatorSet.start();
```

Animation Listeners

- Track state changes of animators
  - Animator.AnimatorListener

```
ValueAnimator fadeAnim = ObjectAnimator.ofFloat(target, "alpha", 1f, 0f);
fadeAnim.setDuration(250);
fadeAnim.addListener(new AnimatorListenerAdapter() {
    public void onAnimationEnd(Animator animation) {
        shapes.remove(((ObjectAnimator)animation).getTarget());
    }});
```

- Update view on every frame
  - ValueAnimator.AnimatorUpdateListener

```
public void onAnimationUpdate(ValueAnimator animation) {
        invalidate(); // invalidate view to trigger redraw
}
```

The End

Prof. Dr. Michael Rohs

michael.rohs@ifi.lmu.de

Mobile Interaction Lab, LMU München