htcDev™

New APIs (Camera and Animation in Jelly Bean and Ice Cream Sandwich)

GMIC-SV
San Jose
October 18, 2012

Lance Nanek, Developer Evangelist, HTC
@LanceNanek
lance_nanek@htc.com
htcdev.com

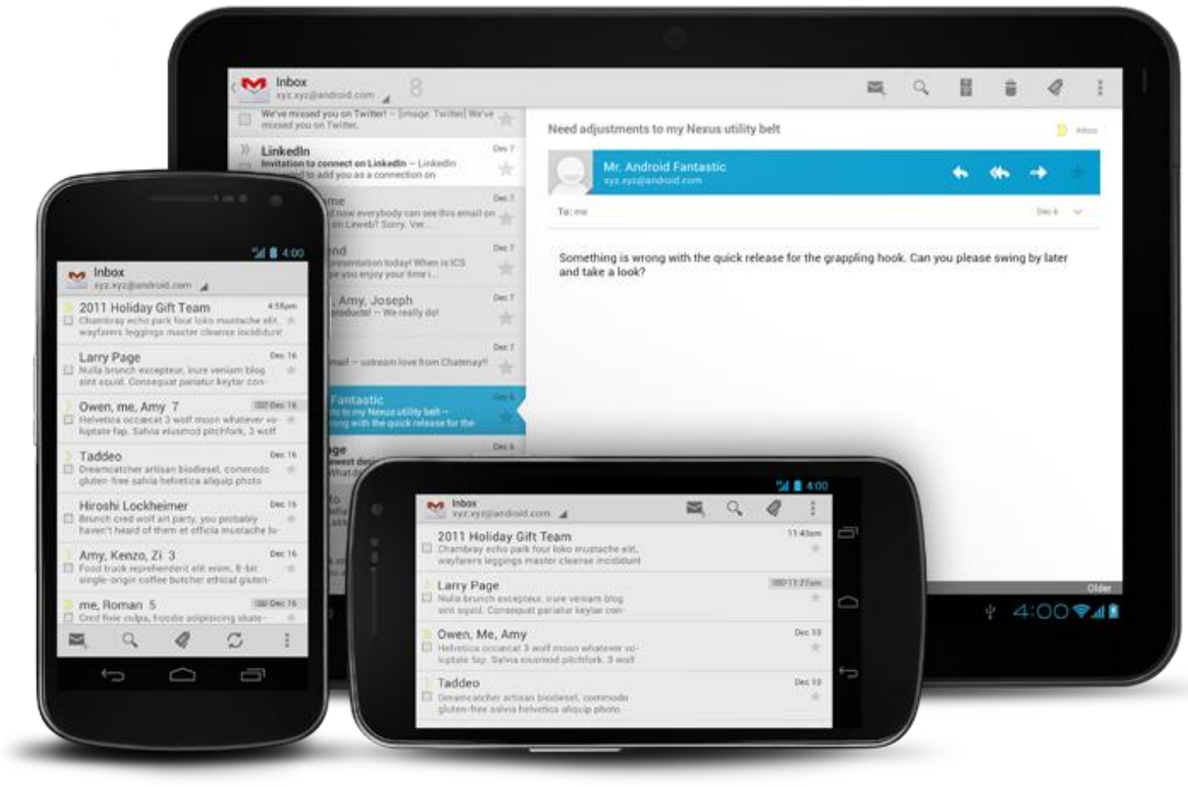
htc quietly brilliant™

Merging of Tablet and Phone OSes

- Android 4.X replaces HoneyComb for tablets and GingerBread for phones.
- Extensive new APIs for media, graphics, UI.
- Point releases have improved and enhanced upon these themes.

htcDev™

Android 4.0 Ice Cream Sandwich



ActionBar UX pattern reaches all new devices

htc quietly brilliant™

Android 4.1 Jelly Bean

Project Jackbusting (or Butter) brings low latency sound and graphics. Triple buffered graphics subsystem run according to vertical refresh rate of display.

htc quietly brilliant™

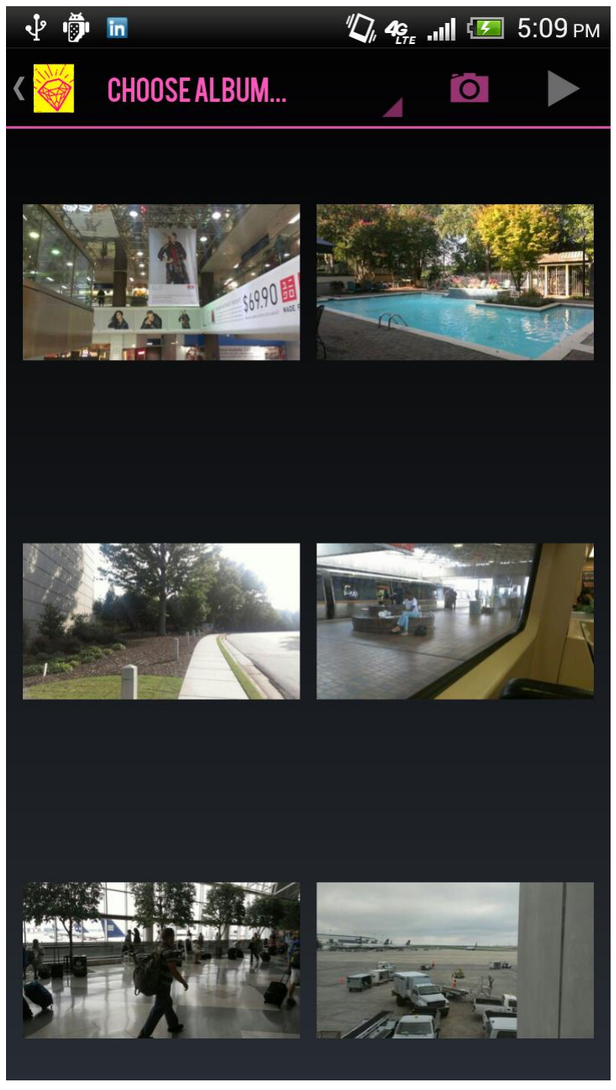
SDK Levels 14, 15, 16

Targeting these API levels requires an on screen menu such as provided by ActionBar, defaults to using the GPU for normal UI widgets, and other changes:

http://developer.android.com/reference/android/os/Build.VERSION_CODES.html

```
<manifest
 xmlns:a="http://schemas.android.com/apk/res/android"
package="com.htc.sample.android4"
a:versionCode="1“ a:versionName="0.0.1" >
<uses-sdk a:targetSdkVersion="16" />
```

Coding: Themed ActionBar App

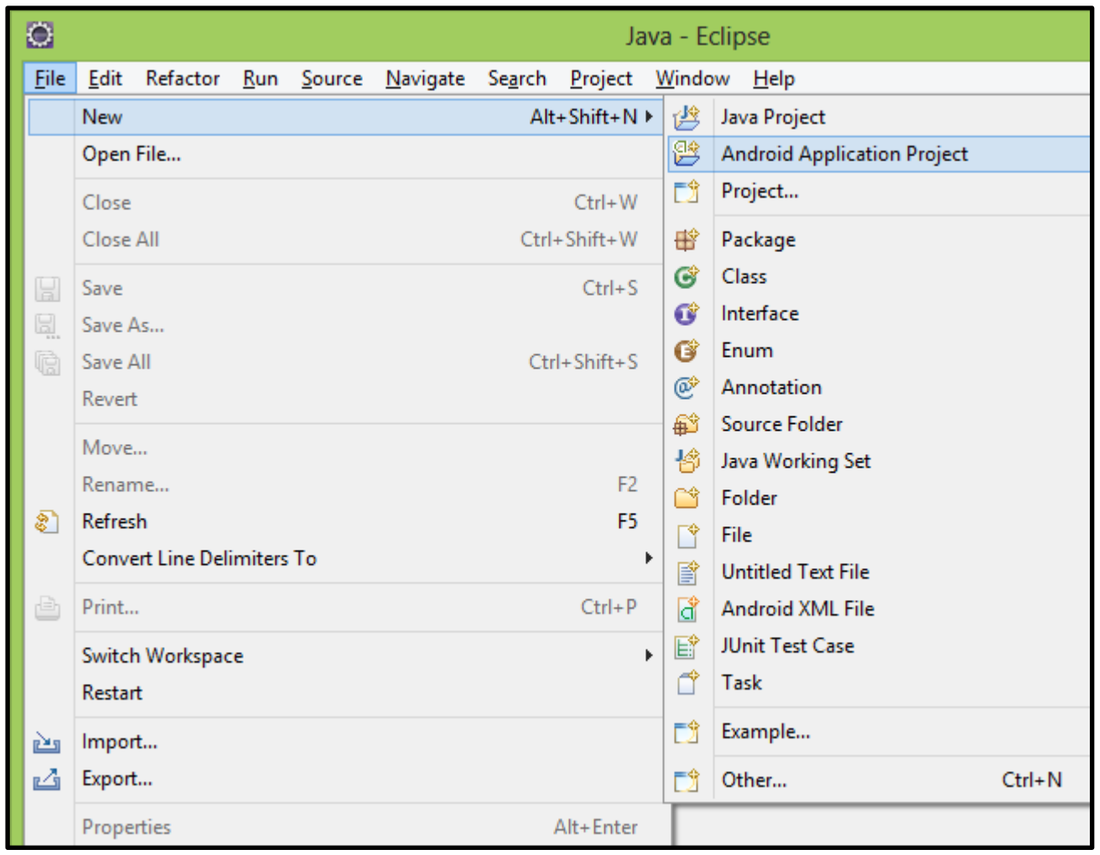


Coding project 1 will create a themed app that uses the ActionBar pattern now available on phones and tablets.

New apps are being written in this style and old apps are being retrofitted to use it as per the Android Design Site:

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

Creating an Android Project



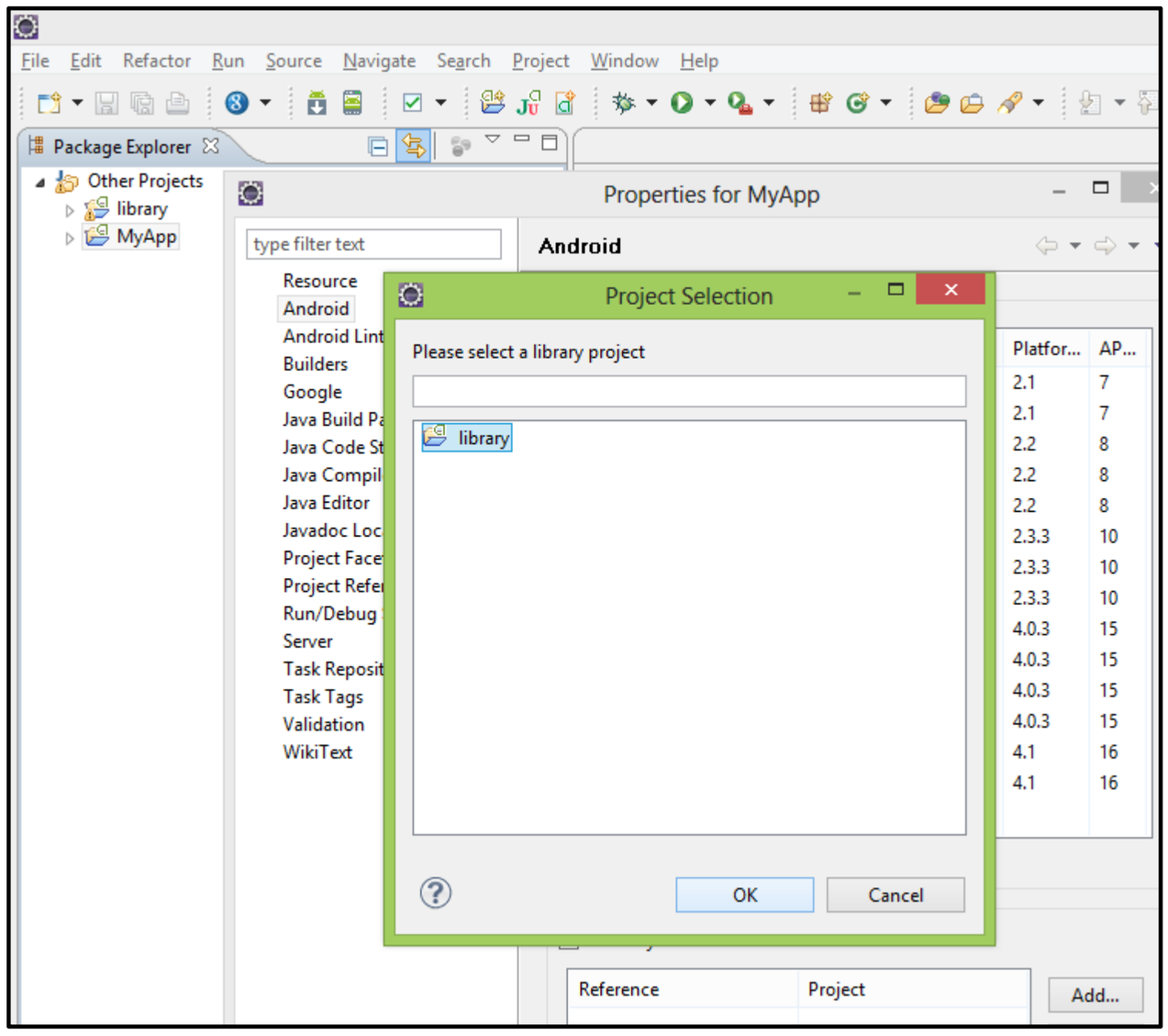
Setup the Android SDK and Eclipse IDE as per the Android developers site:

http://developer.android.com/sdk/installing/

Create a new Android Project via the menus.

htcDev™

Solving ActionBar Compatibility



Android 4.X is currently 25% of Android devices. To support older devices you can use the ActionBarSherlock library:

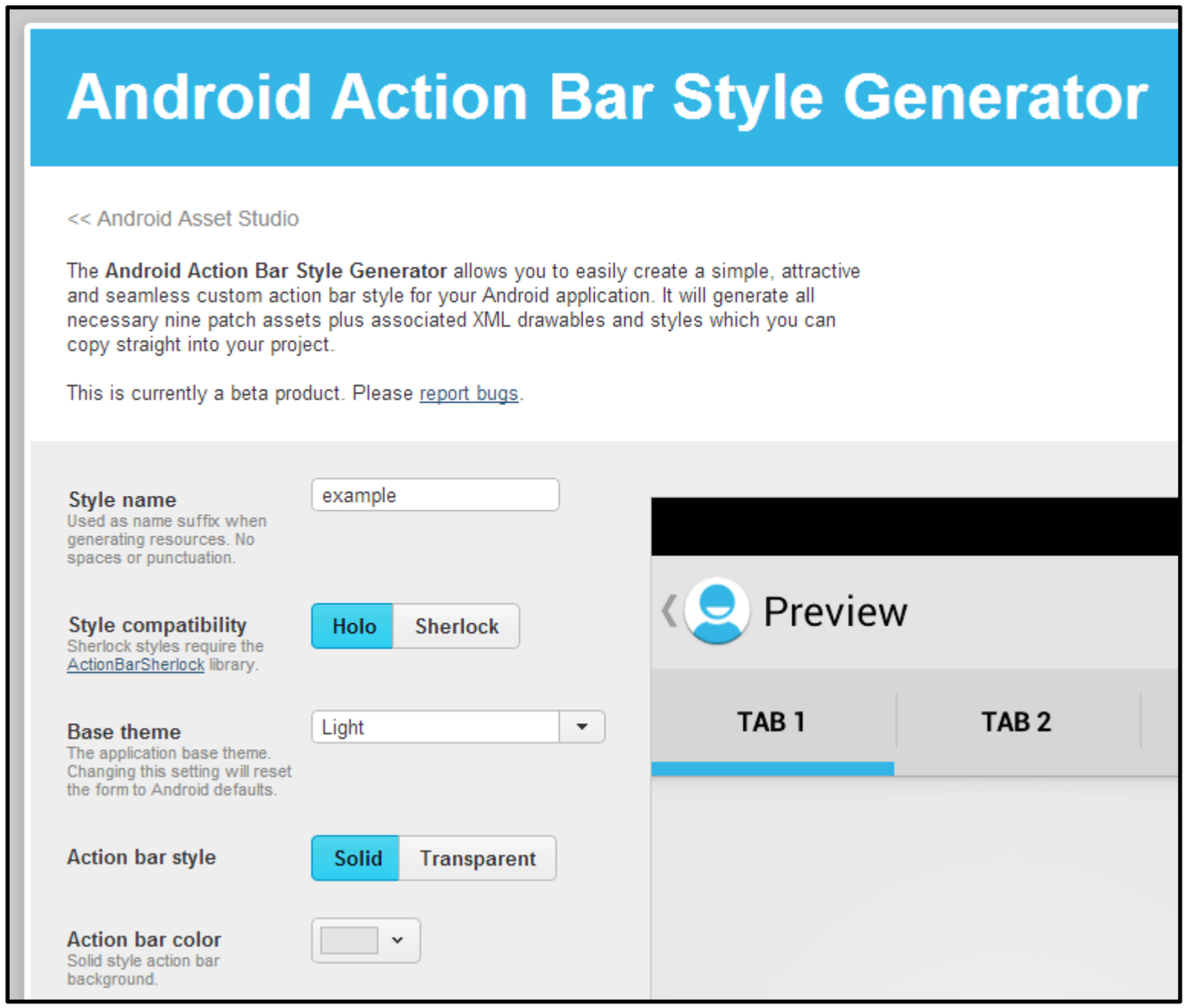
http://actionbarsherlock.com/

Download, unzip, create Android project from existing code in Eclipse, specify as library in the project properties of your app. Extend SherlockActivity instead of Activity in your classes and use support methods like getSupportMenuInflater().

htc quietly brilliant™

htcDev™

Theming ActionBar



You can color the ActionBar by using a style generator:

http://jgilfelt.github.com/android-actionbarstylegenerator/

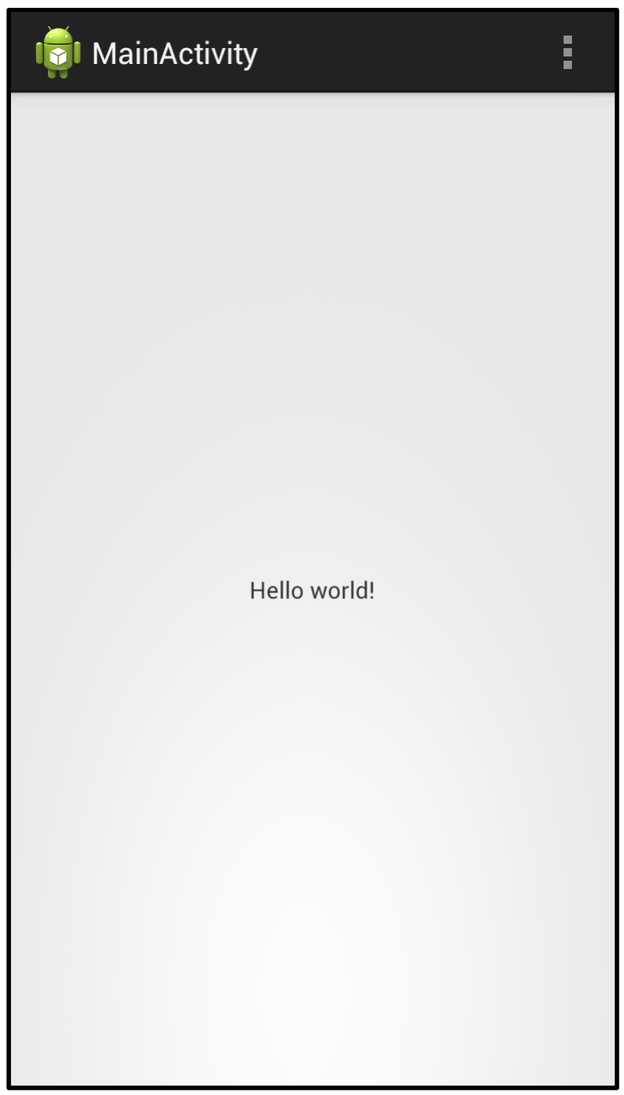
Download the generated ZIP and extract the resource contents into your app.

Android drawables can be downloaded for coloring here:

http://androiddrawables.com/menu.html

htc quietly brilliant™

Controlling System UI: Full Screen



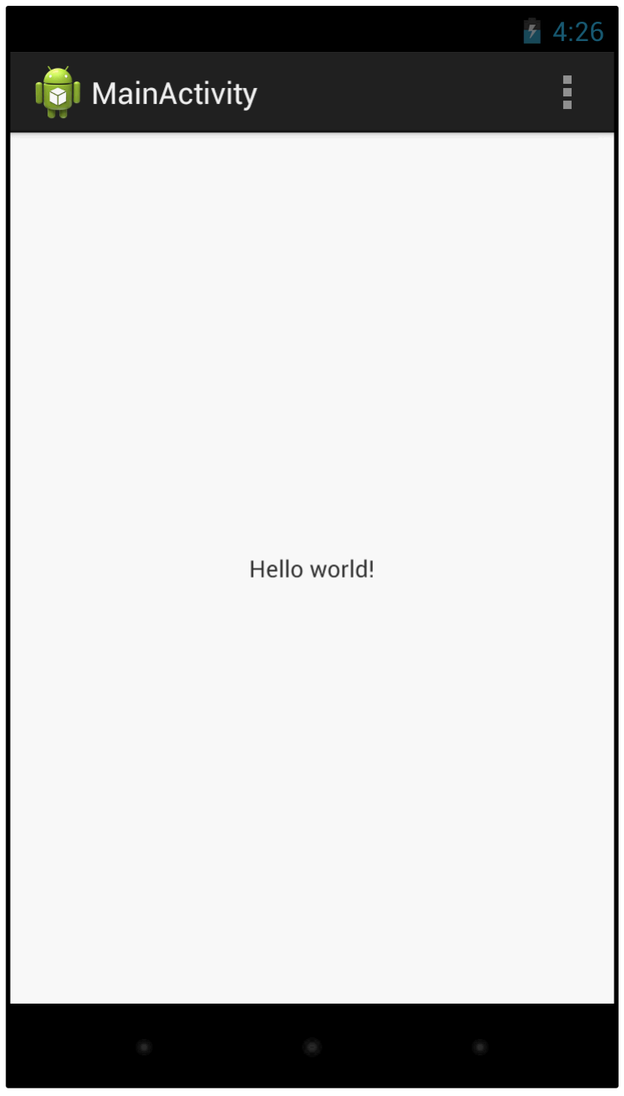
Full screen can be achieved on handsets by a window flag, set by code or by themes:

```
getWindow().addFlags(
  WindowManager.LayoutParams.FLAG_FULLSCREEN)
```

Jelly Bean also has a new constant for when you want full screen temporarily, until the user touches the screen, for example:

```
rootView.setSystemUiVisibility(
  View.SYSTEM_UI_FLAG_FLAG_FULLSCREEN);
```

Controlling System UI: Dimming



Navigation buttons like back and home can also be dimmed on tablets and phones without hardware buttons that show these things on the screen:

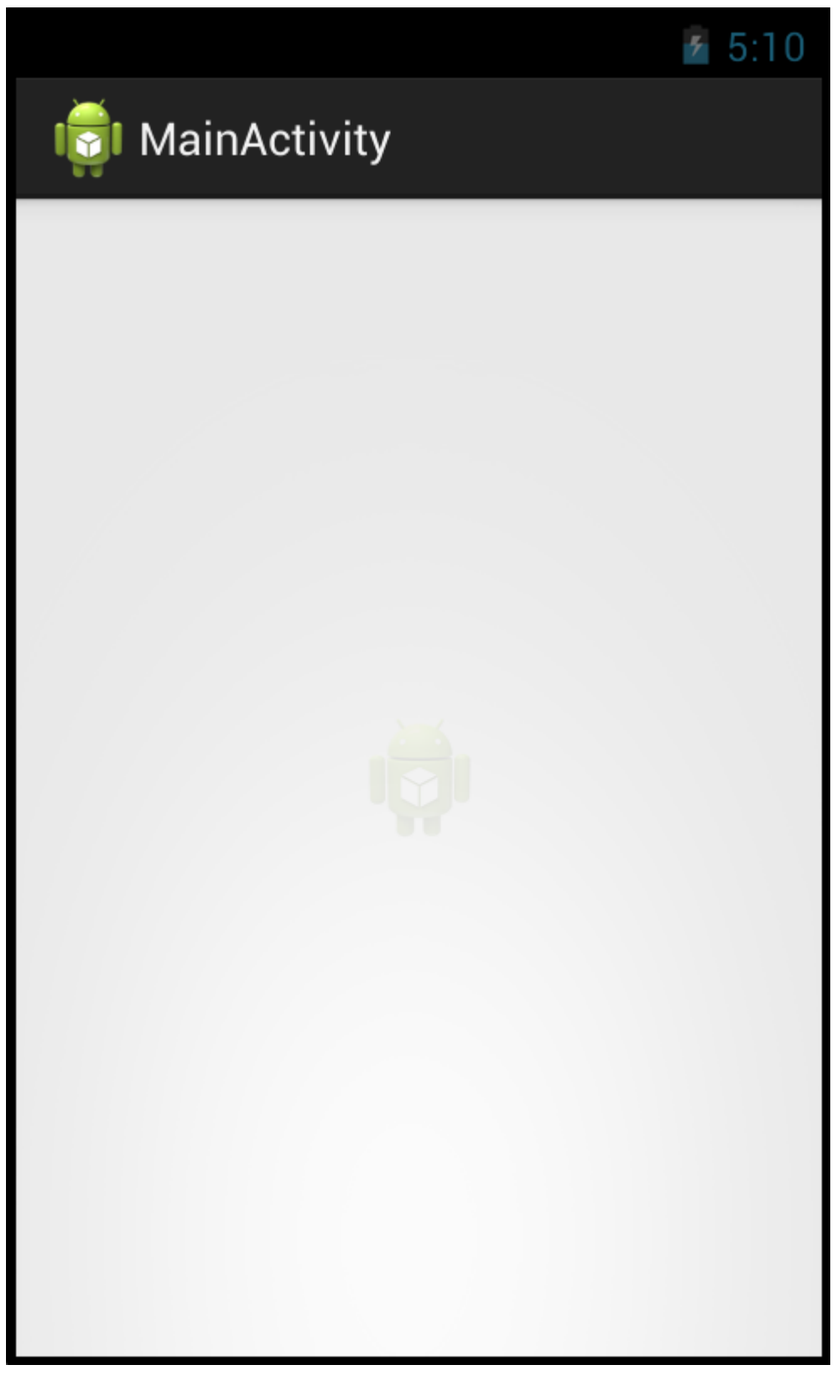
```
rootView.setSystemUiVisibility(
  View.SYSTEM_UI_FLAG_LOW_PROFILE );
```

And you can request it be hidden until user input:

```
rootView.setSystemUiVisibility(
  View.SYSTEM_UI_FLAG_HIDE_NAVIGATION);
```

SYSTEM_UI_FLAG_LAYOUT_STABLE and SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN can be used to prevent view moving and resizing on changes.

Animating Content In and Out



You can animate content being added and removed using the LayoutTransition class. First edit your layout/activity_main.xml layout file to add an ID to your container and center the content:

```
<RelativeLayout xmlns:a=
"http://schemas.android.com/apk/res/android"
    a:id="@+id/container"
    a:gravity="center"
```

Animating Content In and Out

Change content. In this example we'll add an image over 5 seconds:

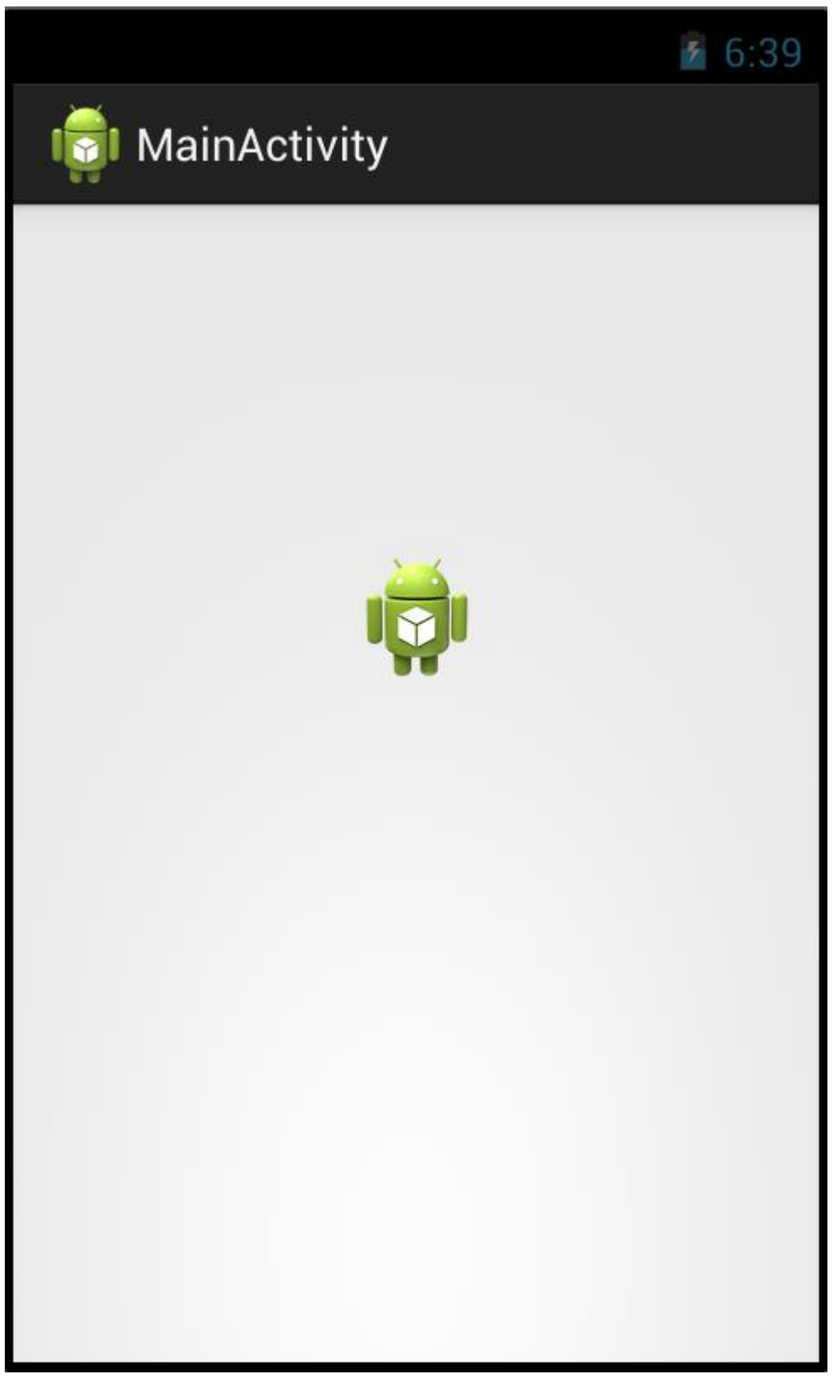
```
ViewGroup container = (ViewGroup) findViewById(R.id.container);
LayoutTransition transition = new LayoutTransition();
transition.setDuration(5000);
container.setLayoutTransition(transition);
container.removeAllViews();
ImageView image = new ImageView(this);
image.setImageResource(R.drawable.ic_launcher);
container.addView(image,
    LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
```

ICS brings to phones and adds setAnimateParentHierarchy(boolean).

Jelly Bean LayoutTransition.CHANGING animates more than show/hide.

Enable in XML via a:animateLayoutChanges="true".

Animating Object Properties



You can animate any object property as well. To animate alpha on an object to zero:

```
ObjectAnimator.ofFloat(object,
    "alpha", 0f).start();
```

To slide our image down using a constant for the property for better performance:

```
ObjectAnimator.ofFloat(image,
    View.TRANSLATION_Y, 100f).start();
```

More control via ValueAnimator, TypeEvaluator. More performance by ViewPropertyAnimator, PropertyValuesHolder. Jelly Bean also bundles multiple properties automatically.

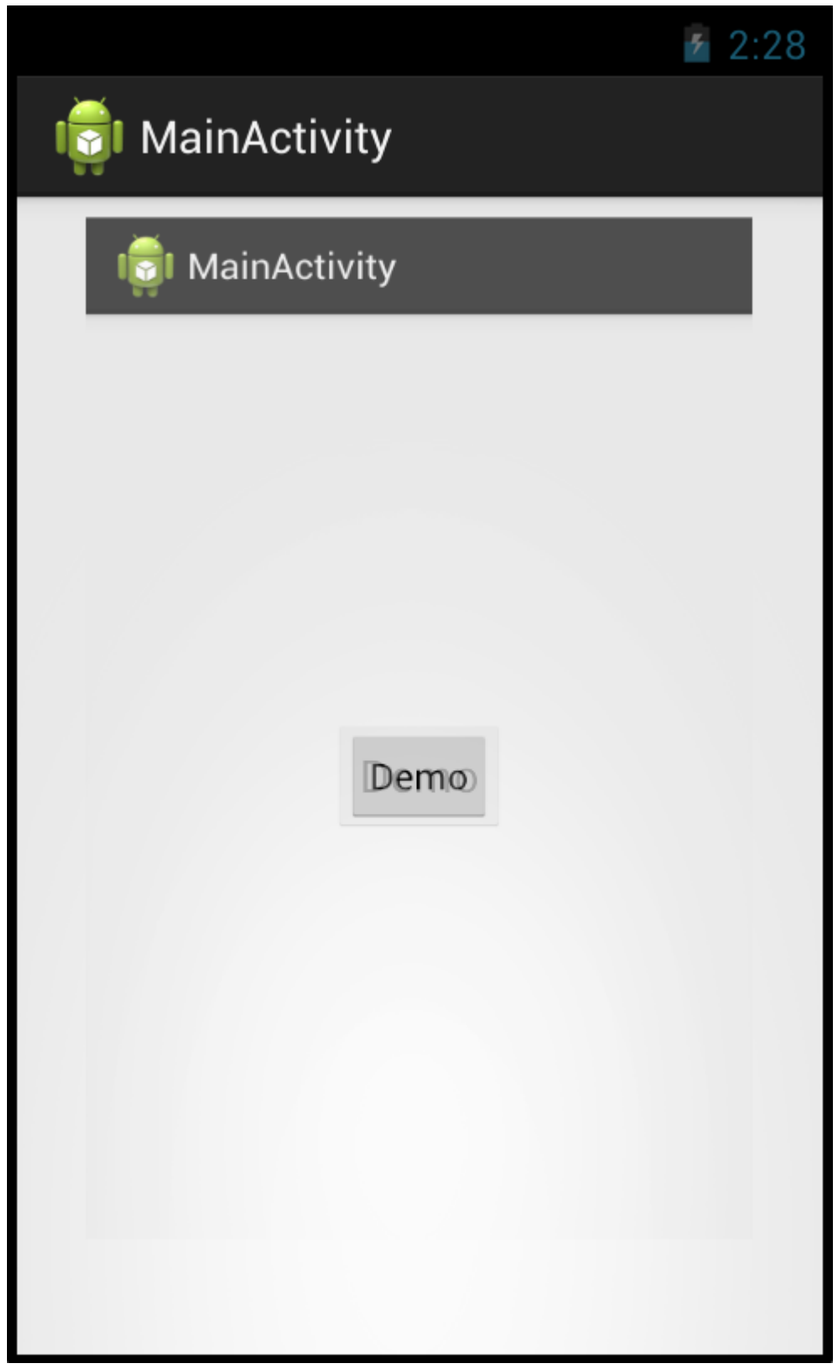
Animating Object Properties

To bounce our image up and down:

```
AnimatorSet bouncer = new AnimatorSet();
ObjectAnimator moveUp = ObjectAnimator.ofFloat(image,
View.TRANSLATION_Y, -100f);
ObjectAnimator moveDown = ObjectAnimator.ofFloat(image,
View.TRANSLATION_Y, 0f);
ObjectAnimator moveUpLess = ObjectAnimator.ofFloat(image,
View.TRANSLATION_Y, -50f);
ObjectAnimator moveDownAgain = ObjectAnimator.ofFloat(image,
View.TRANSLATION_Y, 0f);
bouncer.playSequentially(moveUp, moveDown, moveUpLess, moveDownAgain);
bouncer.setTarget(image);
bouncer.start();
```

ICS brings these to phones. Jelly Bean syncs with vertical screen refresh, offers methods to sync legacy code like TimeAnimator, postInvalidateOnAnimation(), postOnAnimation(Runnable action). XML and key frames also available.

Animating Start Activity



Animate moving between screens using the ActivityOptions class. This example opens a new Activity, scaling it up from the pressed View:

```
ActivityOptions opts =
    ActivityOptions.makeScaleUpAnimation(
        image, 0, 0,
        image.getWidth(),
        image.getHeight());
startActivity(new Intent(MainActivity.this,
    SecondActivity.class), opts.toBundle());
```

Also supported are makeThumbnailScaleUpAnimation, makeCustomAnimation.

ActionBar Home Button Setup

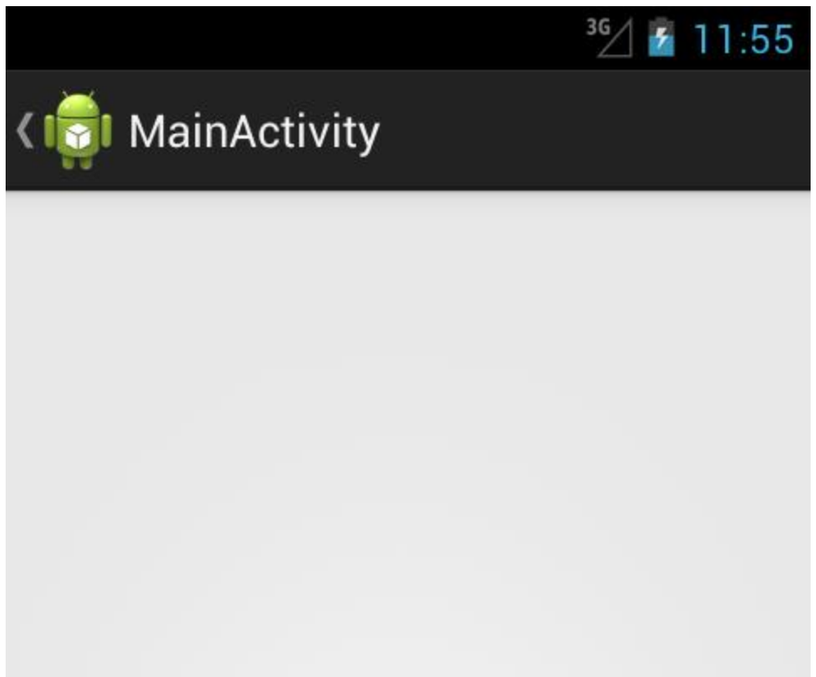
```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    final ActionBar bar = getSupportActionBar();
    bar.setDisplayOptions(ActionBar.DISPLAY_SHOW_HOME,
            ActionBar.DISPLAY_SHOW_HOME);
    bar.setHomeButtonEnabled(true);
    bar.setDisplayShowHomeEnabled(true);
    bar.setDisplayHomeAsUpEnabled(true);

    setContentView(R.layout.activity_second);
}
```

ActionBar Home Button Action

```
@Override
public boolean onOptionsItemSelected(final MenuItem aItem) {
    if (aItem.getItemId() == android.R.id.home) {
        finish();
        return true;
    }
    return super.onOptionsItemSelected(aItem);
}
```



New Media Intents

Camera.ACTION_NEW_PICTURE: This indicates that the user has captured a new photo. The built-in Camera app invokes this broadcast after a photo is captured and third-party camera apps should also broadcast this intent after capturing a photo.

Camera.ACTION_NEW_VIDEO: This indicates that the user has captured a new video. The built-in Camera app invokes this broadcast after a video is recorded and third-party camera apps should also broadcast this intent after capturing a video.

Receiving the Broadcast

Add a receiver in the application element in your AndroidManifest.xml:

```
<receiver a:name=".NewPictureReceiver">
    <intent-filter>
        <action a:name="android.hardware.action.NEW_PICTURE" />
        <data a:mimeType="image/*" />
    </intent-filter>
    <intent-filter>
        <action a:name="com.android.camera.NEW_PICTURE" />
        <data a:mimeType="image/*" />
    </intent-filter>
</receiver>
```

Create the Receiver Class

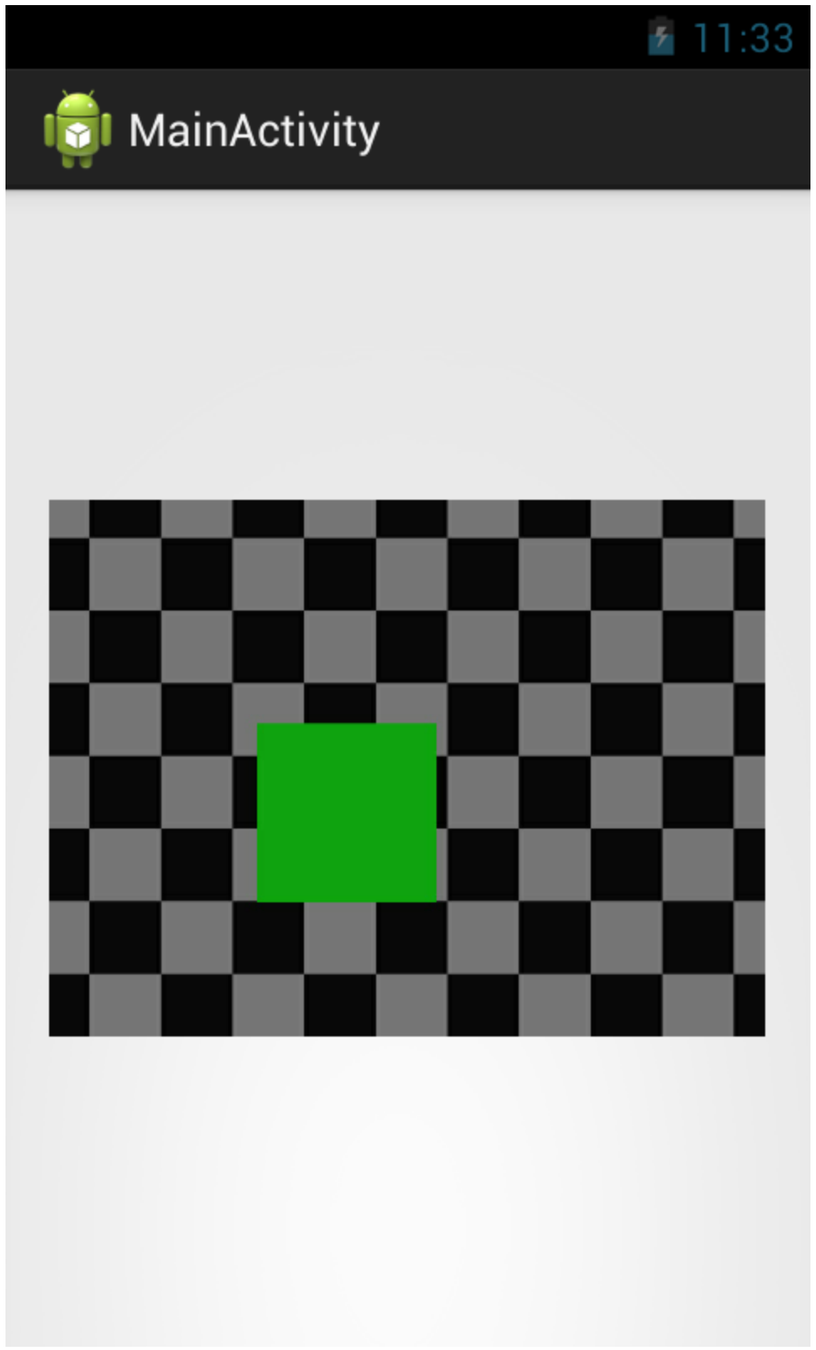
Implement the receiver to start the app:

```
public class NewPictureReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context aContext, Intent aIntent) {
        final Intent intent = new Intent(aContext, MainActivity.class);
        intent.setData(aIntent.getData());
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        aContext.startActivity(intent);
    }
}
```

Use the Image in Your App

```
public class MainActivity extends SherlockActivity {
    private Uri imageUri;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        imageUri = getIntent().getData();
final ImageView image = new ImageView(this);
if ( null == imageUri ) {
image.setImageResource(R.drawable.ic_launcher);
} else {
    image.setImageURI(imageUri);
}
```

Use the Image in Your App



Hardware Features

New feature to determine if you are on a TV:

```
<uses-feature
    a:name="android.hardware.type.television"
    a:required="true" />.
```

Nexus 7 does not have a front facing camera, for app to be available:

```
<uses-feature
    a:name="android.hardware.camera"
    a:required="false" />.
```

Images by capture intent are not automatically saved to disk, no camera shortcut.

New XXHDPI Density Displays

DisplayMetrics.XXHDPI constant added for 480dpi logical display density.

If you use compatible-screens element in AndroidManifest.xml, add these lines:

```
<compatible-screens>
        <!-- Nexus 7 -->
        <screen a:screenSize="large" a:screenDensity="213" />
        <!-- HTC Butteryfly J and other 5" 1080P phones. -->
        <screen a:screenSize="large" a:screenDensity="480" />
```

Hardware Acceleration of Views

Hardware acceleration of views is now on by default if your targetSdkVersion is 14 or higher. Make sure to test. There are some unsupported operations. Don't count on intersection views in the same dirt rectangle to get invalidated.

Levels it can be turned on/off if needed:

Application

```
<application a:hardwareAccelerated="false"
```

Activity

```
<activity a:hardwareAccelerated="false"
```

Window

```
getWindow().setFlags(
        LayoutParams.FLAG_HARDWARE_ACCELERATED,
        LayoutParams.FLAG_HARDWARE_ACCELERATED);
```

View

```
view.setLayerType(View.LAYER_TYPE_SOFTWARE, null);
```

htcDev™

New Camera APIs

Developers can take advantage of a variety of new camera features in Android 4.0. ZSL exposure, continuous focus, and image zoom let apps capture better still and video images, including during video capture. Apps can even capture full-resolution snapshots while shooting video. Apps can now set custom metering regions in a camera preview, then manage white balance and exposure dynamically for those regions. For easier focusing and image processing, a face-detection service identifies and tracks faces in a preview and returns their screen coordinates.

Camera.AutoFocusMoveCallback, setAutoFocusMoveCallback, FOCUS_MODE_CONTINUOUS_VIDEO, FOCUS_MODE_CONTINUOUS_PICTURE

Camera.FaceDetectionListener, setFaceDetectionListener(), startFaceDetection(), onFaceDetection(), Camera.Face, getMaxNumDetectedFaces()

htc quietly brilliant™

htcDev™

New Camera APIs

getMaxNumFocusAreas(), getMaxNumMeteringAreas(), setFocusAreas(), setMeteringAreas(), Camera.Area

FOCUS_MODE_CONTINUOUS_PICTURE, setFocusMode() , autoFocus(), cancelAutoFocus(), FOCUS_MODE_CONTINUOUS_VIDEO

takePicture() while recordingif isVideoSnapshotSupported()

setAutoExposureLock(), setAutoWhiteBalanceLock()

setDisplayOrientation() while the camera preview is running

htc quietly brilliant™

Camera Sounds

The MediaActionSound class provides a simple set of APIs to produce standard sounds made by the camera or other media actions. You should use these APIs to play the appropriate sound when building a custom still or video camera.

To play a sound, simply instantiate a MediaActionSound object, call load() to pre-load the desired sound, then at the appropriate time, call play().

FOCUS_COMPLETE
SHUTTER_CLICK
START_VIDEO_RECORDING
STOP_VIDEO_RECORDING

htcDev™

Easy Live Wallpaper Setup

New intent protocol to directly launch the live wallpaper preview activity so you can help users easily select your live wallpaper without forcing them to leave your app and navigate through the Home wallpaper picker.

To launch the live wallpaper picker, call startActivity() with an Intent using ACTION_CHANGE_LIVE_WALLPAPER and an extra that specifies your live wallpaper ComponentName as a string in EXTRA_LIVE_WALLPAPER_COMPONENT.

htc quietly brilliant™

More Media Functionality

More powerful media stream API. setSurface() to device video sink, setDataSource() allows custom HTTP headers to be sent with request.

More formats supported: HTTP/HTTPS live streaming protocol version 3, ADTS raw AAC audio encoding, WEBP images, Matroska video. HTTP Live streaming, Bluetooth A2DP and HSP devices, Support for RTP, MTP/PTP file transfer, DRM.

Input from keyboard, mouse, gamepad, joystick

Low level access to media codes via MediaCodec class. createEncoderByType(), configure(), getInputBuffers() , getOutputBuffers() , dequeueInputBuffer() , queueInputBuffer(), releaseOutputBuffer(). Encrypted media data via queueSecureInputBuffer() and MediaCrypto APIs.

Larger contact photos, 96x96 thumbnail and 256x256 display image.
MediaMetadataRetriever.METADATA_KEY_LOCATION location data.

More Media Functionality

setVideoStabilization(), setVideoStabilization(), and isVideoStabilizationSupported()

More CamcorderProfile constants, supporting time lapse and lower quality.

Media Effects fix red-eye, convert an image to grayscale, adjust brightness, adjust saturation, rotate an image, apply a fisheye effect, vignette, and much more. isEffectSupported(), EffectContext.createWithCurrentGlContext(), EffectContext.getFactory(), createEffect(), apply() on GL_TEXTURE_2D texture image.

Low level streaming API based on Khronos OpenMAX AL 1.0.1, usable with OpenSL ES API. Send processed data to the platform as a multiplexed stream of audio/video content in MPEG-2 transport stream format. The platform de-muxes, decodes, and renders the content. When render to SurfaceTexture instead of Surface, can apply effects.

More Powerful UI

Fragments and content loaders, Resizeable home screen widgets, Rich notifications, multi-selection, drag-drop, clipboard.

ViewPager, ActionBar navigation including tabs and drop downs.

View layers to determine hardware/software drawing, apply composition, and other effects. This API takes two parameters: the type of layer you want to use and an optional Paint object that describes how the layer should be composited. You can use the Paint parameter to apply color filters, special blending modes, or opacity to a layer.

The new Switch widget is a two-state toggle that users can drag to one side or the other (or simply tap) to toggle an option between two states. You can use the android:textOn and android:textOff attributes to specify the text to appear on the switch when in the on and off setting. The android:text attribute also allows you to place a label alongside the switch.

GridLayout

GridLayout is a new view group that places child views in a rectangular grid. Unlike TableLayout, GridLayout relies on a flat hierarchy and does not make use of intermediate views such as table rows for providing structure. Instead, children specify which row(s) and column(s) they should occupy (cells can span multiple rows and/or columns), and by default are laid out sequentially across the grid's rows and columns. The GridLayout orientation determines whether sequential children are by default laid out horizontally or vertically. Space between children may be specified either by using instances of the new Space view or by setting the relevant margin parameters on children.

htcDev™

TextureView

TextureView is a new view that allows you to display a content stream, such as a video or an OpenGL scene. Although similar to SurfaceView, TextureView is unique in that it behaves like a regular view, rather than creating a separate window, so you can treat it like any other View object. For example, you can apply transforms, animate it using ViewPropertyAnimator, or adjust its opacity with setAlpha(). Beware that TextureView works only within a hardware accelerated window.

htc quietly brilliant™

Memory Management and Performance

New ComponentCallbacks2 constants such as TRIM_MEMORY_RUNNING_LOW and TRIM_MEMORY_RUNNING_CRITICAL provide foreground processes more information about memory state before the system calls onLowMemory().

New getMyMemoryState(ActivityManager.RunningAppProcessInfo) method allows you to retrieve the general memory state.

New methods setUserVisibleHint() and getUserVisibleHint() allow a fragment to set a hint of whether or not it is currently user-visible. The system defers the start of fragments that are not user-visible until the loaders for visible fragments have run. The visibility hint is "true" by default.

Process isolation. Background processes, setForeground more important

RenderScript Performance

RenderScript graphics engine depracted. Still usable as a C99 to native cross-architecture compute engine, however, with even more performance:

Support for multiple kernels within one script.

Support for reading from allocation with filtered samplers from compute in a new script API rsSample.

Support for different levels of FP precision in #pragma.

Support for querying additional information from RS objects from a compute script.

Numerous performance improvements.

New pragmas are also available to define the floating point precision required by your compute Renderscripts. This lets you enable NEON like operations such as fast vector math operations on the CPU path that wouldn’t otherwise be possible with full IEEE 754-2008 standard.