CS371m - Mobile Computing

Nifty Stuff

We have touched on many of the basic features of the Android System and Apps

- Activities, Services, Content Resolvers, Broadcast Receivers
- Intents
- Activity Lifecycle
- UIs, widgets, ViewGroups, ActionBar
- Location Manager
- Sensor Manager
- 2D graphics
- Persistence, Shared Preferences
- Responsiveness
- Gestures
- SQLite
- Fragments
- Web APIs
- Navigation Strategies

Surface Scratched

- We have covered some of the basics
- Numerous features and capabilities we haven't covered
- Plus features added in every release
- Example: KitKat release, October 2013
- NFC via Host Card Emulation, printing framework, storage access framework, low power sensors, step detector, changes to SMS provider, full screen immersive mode, transition framework for animating scenes, Chromium WebView, screen recording, RenderScript Compute, IR Blaster, and more!
- http://tinyurl.com/nmfaxwu
- Lollipop: November 2014: http://tinyurl.com/prpuqgy
- Marshmallow: October 2015: http://tinyurl.com/okd5oul

AUDIO

Audio on Device

- Devices have multiple audio streams:
  - music, alarms, notifications, incoming call ringer, in call volume, system sounds, DTMF tones (dual tone multi frequency, the "traditional" phone sounds and tones)
- Most of the streams are restricted to system events, so we almost always use STREAM_MUSIC via a MediaPlayer or SoundPool

MEDIA PLAYER

Android Audio

- Using the MediaPlayer class
- Common Audio Formats supported:
  - MP3, MIDI (.mid and others), Vorbis (.ogg), WAVE (.wav) and others
- Sources of audio
  - local resources (part of app)
  - internal URIs (Content Provider for other audio available)
  - External URLs (streaming)

Audio Resources

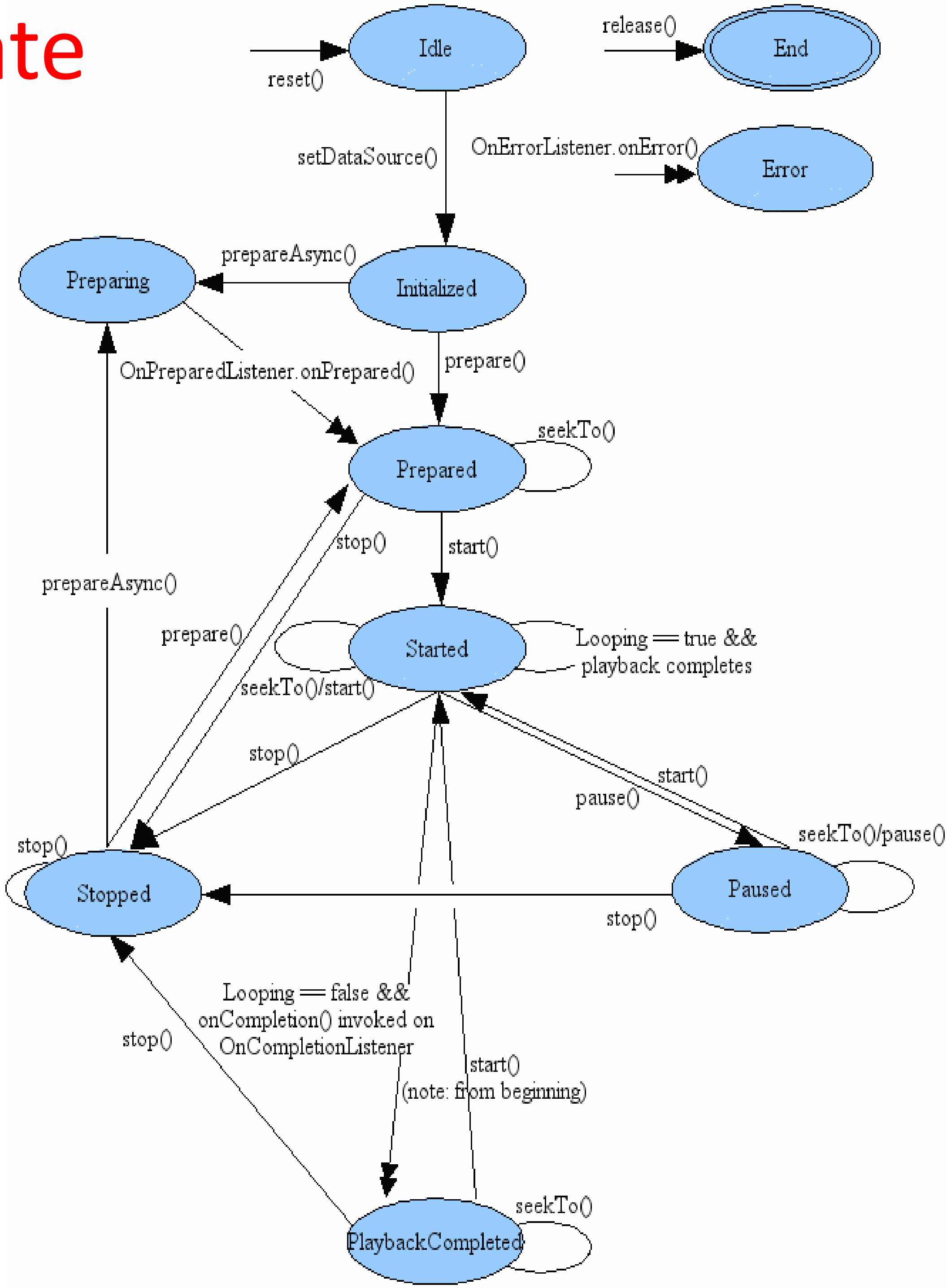
- res/raw directory
- assets/ directory
  - reference as file://andorid_asset in a URI
  - can share with other apps via URI
- Store media in application local directory, pull from network / web and store locally
- Store and use media on SD card
- Stream via the Internet

MediaPlayer

- Playback control of MediaPlayer managed as a state machine
- Idle
- Initialized
- Preparing
- Prepared
- Started
- Paused
- Playback Complete
- Stopped
- End
- Invalid state transitions result in errors

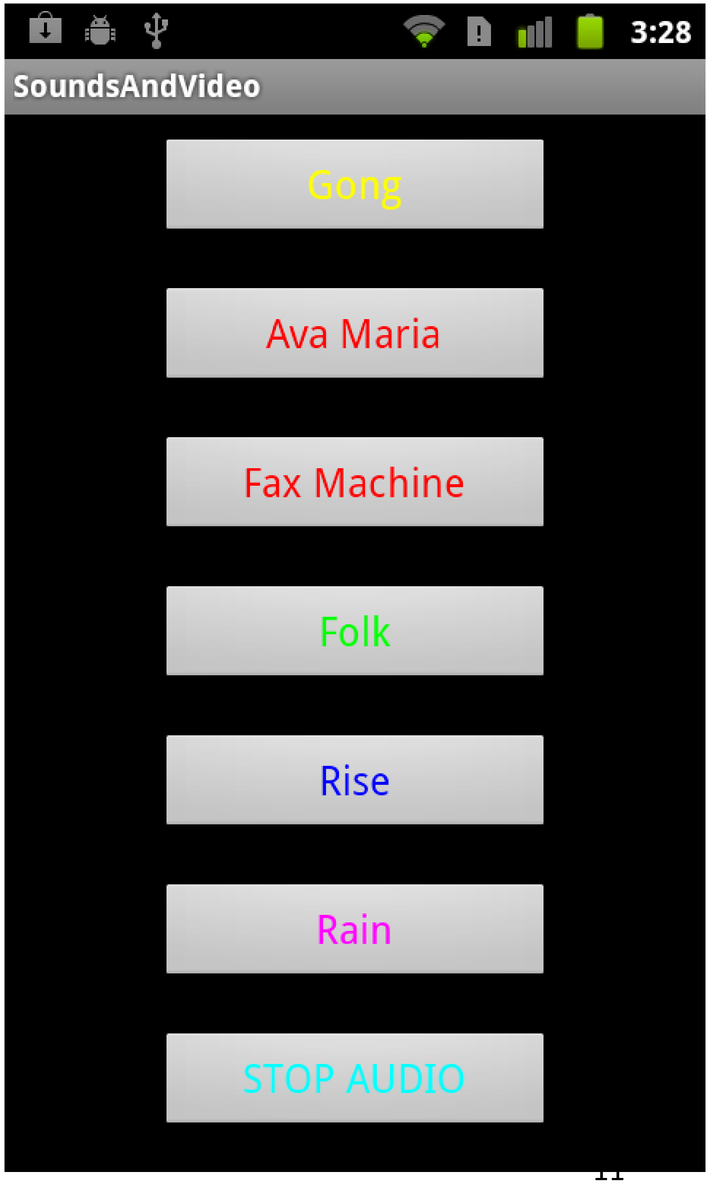
MediaPlayer State Diagram

- Single arrows are synchronous transitions
- Double arrows are asynchronous transitions



Simple Sound Demo App

- audio files local to app placed in res/raw
- CAUTION
  - large sound files difficult to install on emulator:
  - http://tinyurl.com/3pwljfj
  - better success with dev phones / actual devices



Using MediaPlayer

- MediaPlayer.create() method used for raw resources
- MediaPlayer.create() method for URIs
- various MediaPlayer.setDataSource() methods if not a raw resource and not a URI

Playing Local Audio

- To play audio local to the app
- use the MediaPlayer.create convenience method
  - when complete MediaPlayer in the **prepared** state
- start MediaPlayer
- approach:
  - build listeners for each button to call the playSound method with appropriate song id when clicked

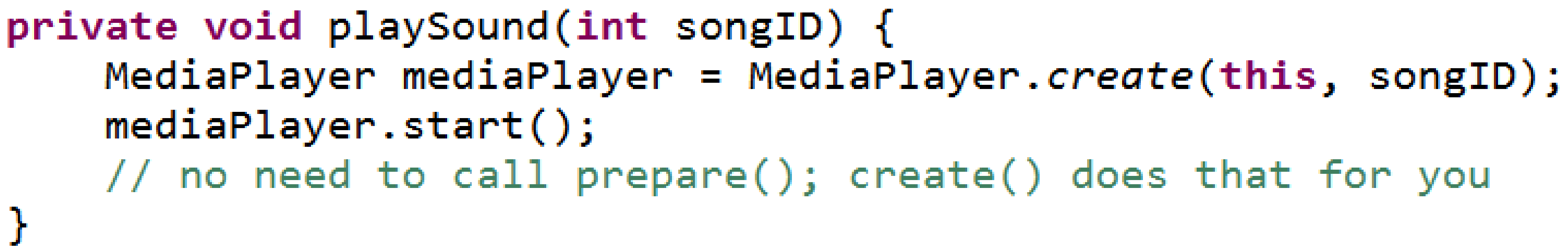
Simple Approach

button ids

ids for sound files

```
private void buildListeners() {
    int[] ids = {R.id.gong, R.id.ava, R.id.fax,
            R.id.folk, R.id.rise, R.id.rain};
    int[] songs = {R.raw.gong, R.raw.ava_maria,
            R.raw.fax, R.raw.music,
            R.raw.rise, R.raw.rain};

    for(int i = 0; i < ids.length; i++) {
        final Button button = (Button) findViewById(ids[i]);
        final int SONG_ID = songs[i];
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                playSound(SONG_ID);
            }
        });
    }
}
```

playSound method initial version

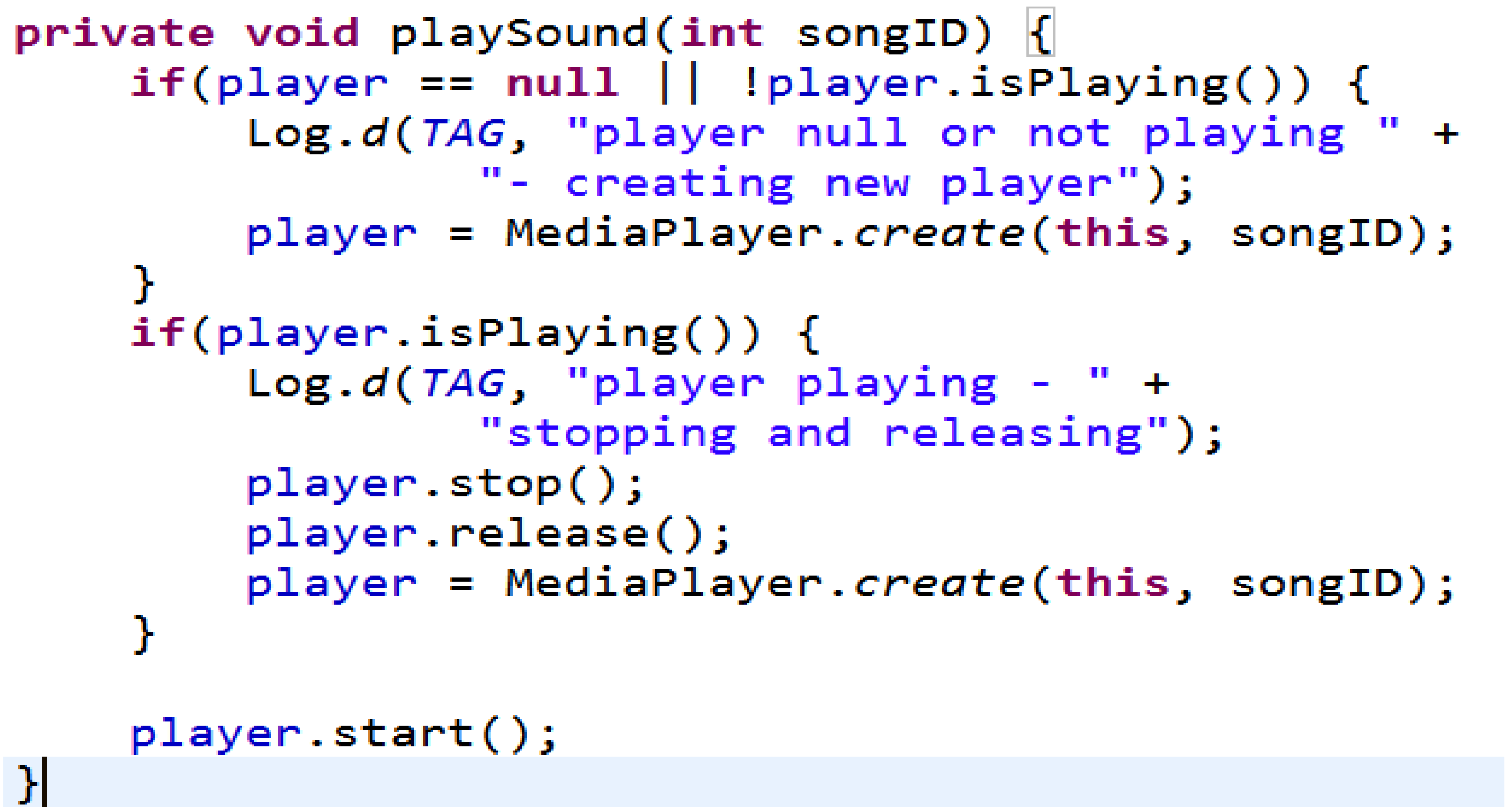


```
private void playSound(int songID) {
    MediaPlayer mediaPlayer = MediaPlayer.create(this, songID);
    mediaPlayer.start();
    // no need to call prepare(); create() does that for you
}
```

- downsides of first version of app
  - plays to completion
  - multiple sounds play at same time (desirable in some cases), code creates multiple media players
  - audio continues to play when app paused

Changing Behavior

- Add instance variable for MediaPlayer
- If playing stop and release before creating new Player



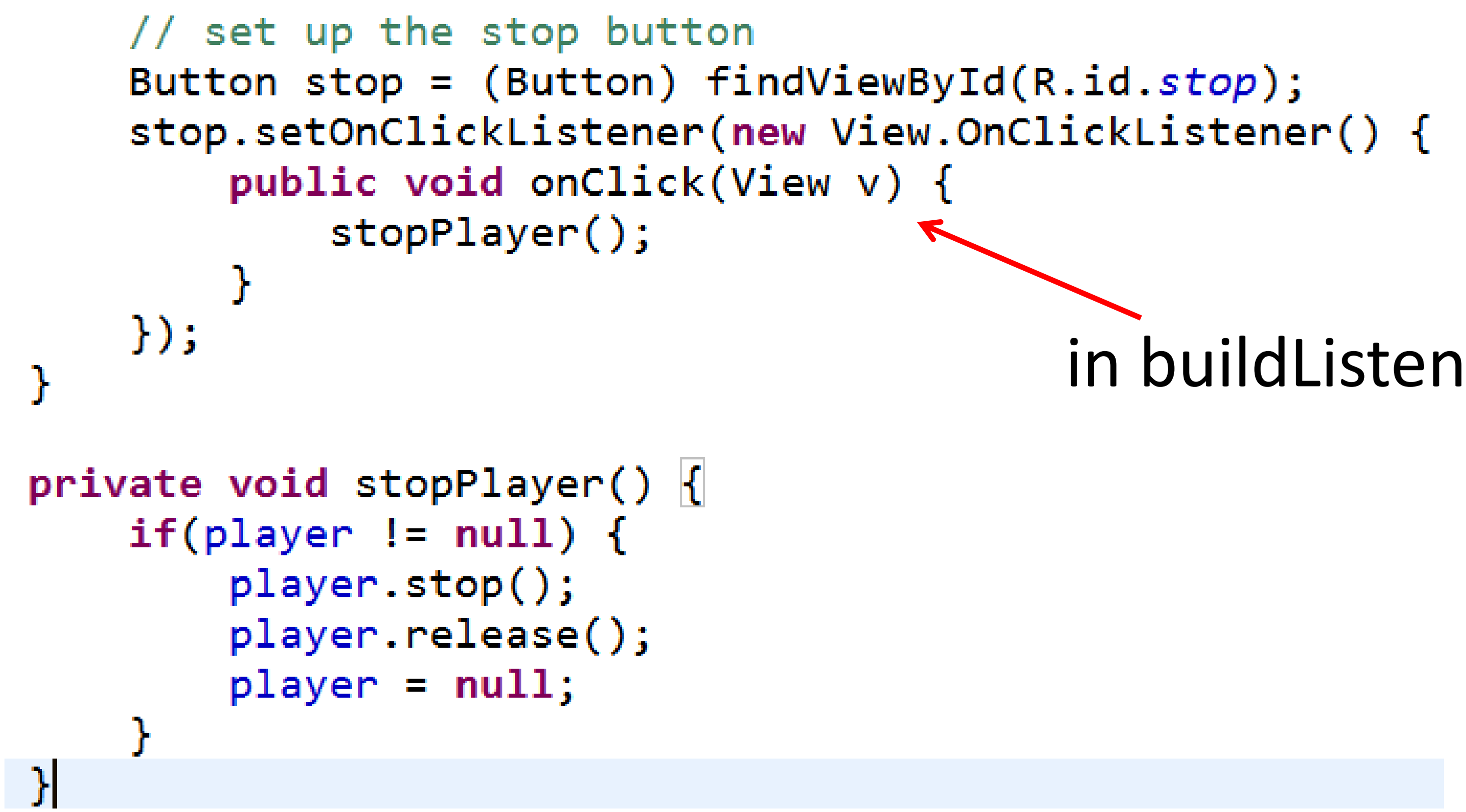
```
private void playSound(int songID) {
    if(player == null || !player.isPlaying()) {
        Log.d(TAG, "player null or not playing " +
                "- creating new player");
        player = MediaPlayer.create(this, songID);
    }
    if(player.isPlaying()) {
        Log.d(TAG, "player playing - " +
                "stopping and releasing");
        player.stop();
        player.release();
        player = MediaPlayer.create(this, songID);
    }

    player.start();
}
```

Cleaning Up

- Initial version did not end well
- Audio continued to play if back button pressed and even if home button pressed!
- Activity Life Cycle
- in onPause for app we should stop MediaPlayer and release

stopPlayer method

- Connect app stop button to stopPlayer
  - could use XML onClick and add View parameter or set up listener ourselves



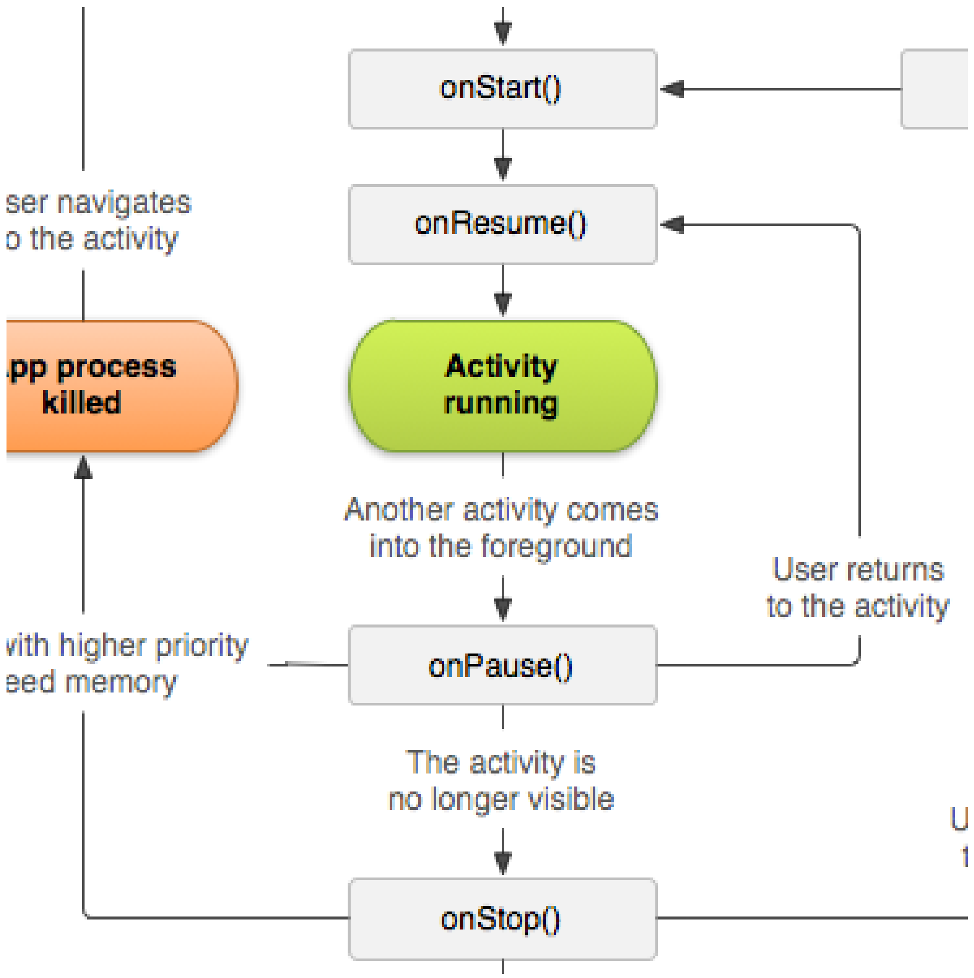
```
    // set up the stop button
    Button stop = (Button) findViewById(R.id.stop);
    stop.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            stopPlayer();
        }
    });
}

private void stopPlayer() {
    if(player != null) {
        player.stop();
        player.release();
        player = null;
    }
}
```

in buildListeners method

onPause

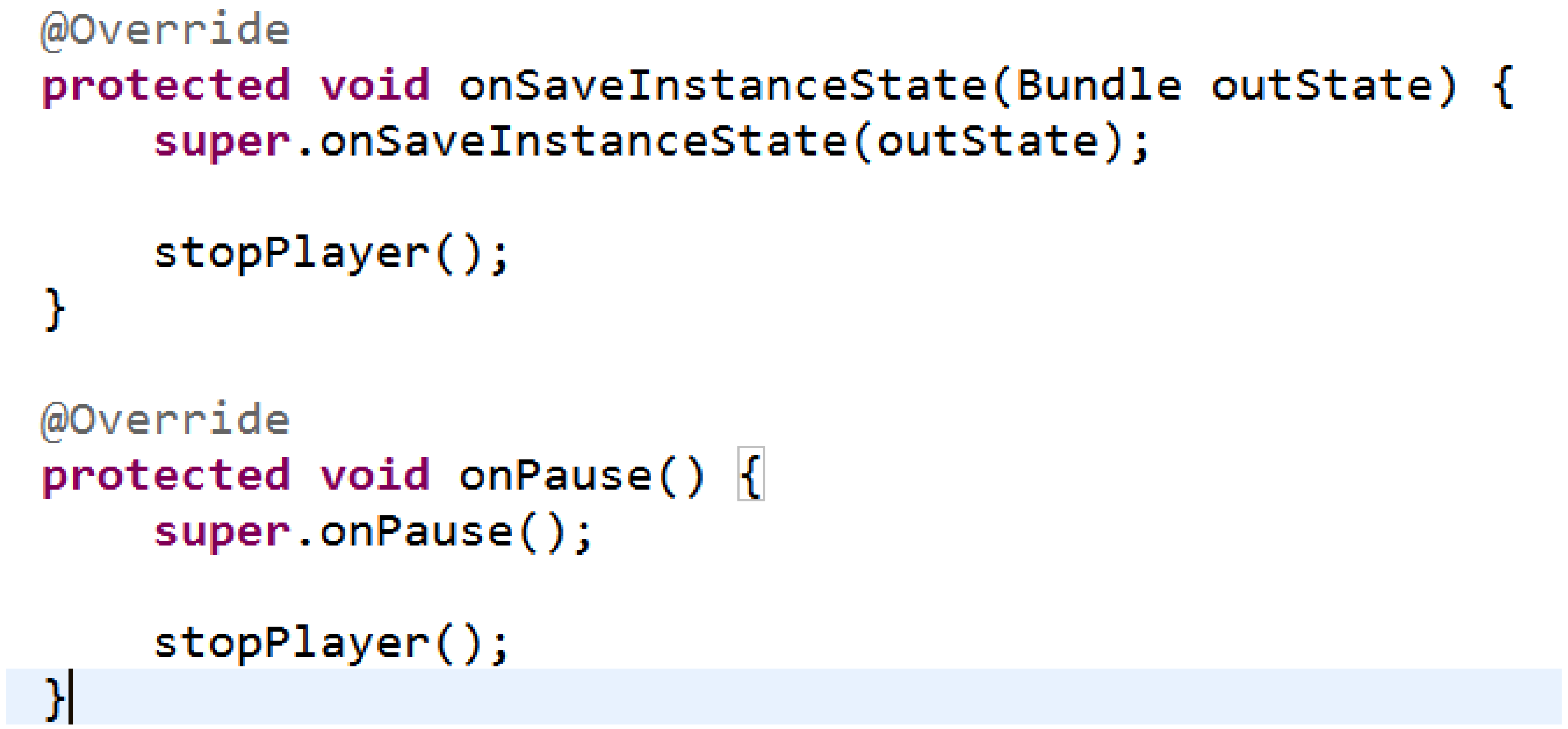
- onPause() should call the stopPlayer method
- what happens if activity resumed?

```
@Override
protected void onPause() {
    super.onPause();
    // stop the music!!
    stopPlayer();
}
```



Saving State

- Resume music where we left off if paused or activity destroyed due to orientation change



```
@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);

    stopPlayer();
}

@Override
protected void onPause() {
    super.onPause();

    stopPlayer();
}
```

Saving MediaPlayer State

```
if (player != null) {
    if (player.isPlaying()) {
        ed.putInt("songID", currentSongID);
        ed.putInt("audioLocation", player.getCurrentPosition());
        Log.d(TAG, "in stopPlayer - song id: " + currentSongID);
        Log.d(TAG,
                "in stopPlayer - audio Location: "
                        + player.getCurrentPosition());
    } else {
        ed.putInt("songID", -1);
    }
    player.stop();
    player.release();
    player = null;
} else {
    ed.putInt("songID", -1);
}
ed.apply();
     }
}
```

Save location in shared preferences

Restarting Audio

- In onResume check if audio was interrupted recreate player with same id and move to correct position
- Can write data to shared preferences or bundle (onSaveInstanceState) and pull out in onResume

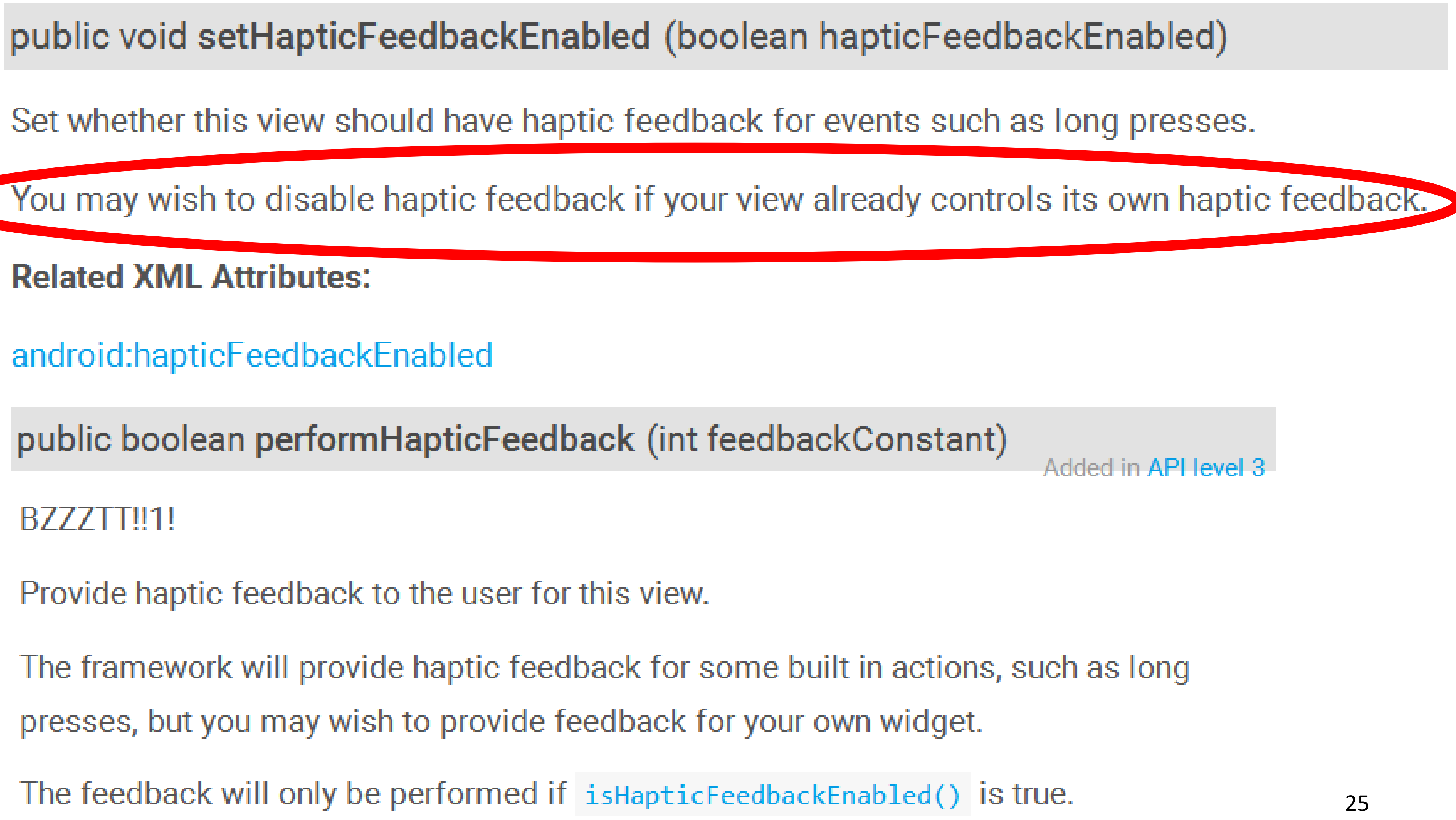
HAPTIC FEEDBACK

Haptic Feedback

- haptic: "of or relating to the sense of touch"
- Simple to add this kind of feedback to an app
- can enable haptic feedback for any view
- Can add complex haptic feedback of our own

Haptic Feedback on a View

- Methods from View class



```
public void setHapticFeedbackEnabled (boolean hapticFeedbackEnabled)
```

Set whether this view should have haptic feedback for events such as long presses.

You may wish to disable haptic feedback if your view already controls its own haptic feedback.

Related XML Attributes:

android:hapticFeedbackEnabled

```
public boolean performHapticFeedback (int feedbackConstant)
```

Added in API level 3

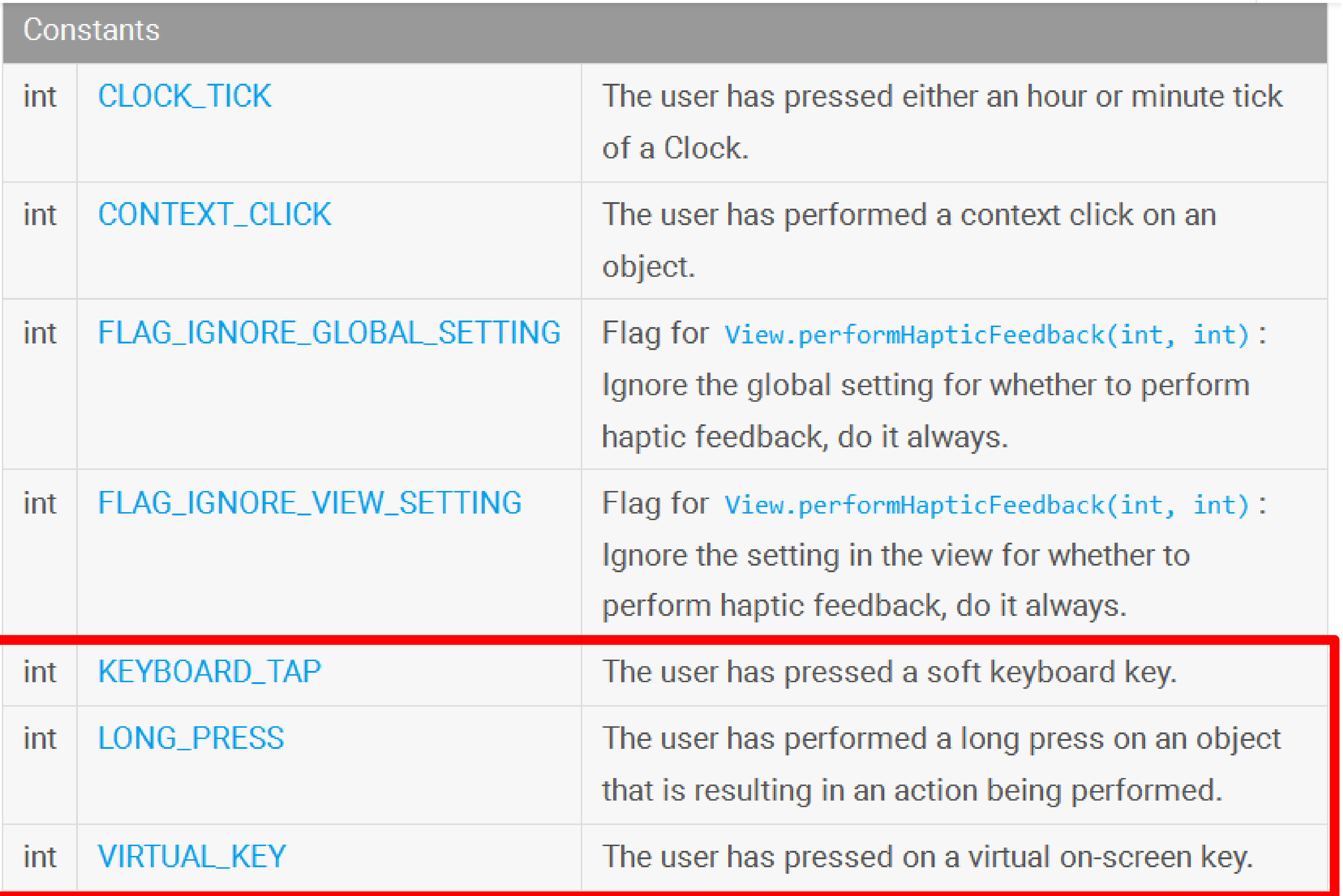
BZZZTT!!1!

Provide haptic feedback to the user for this view.

The framework will provide haptic feedback for some built in actions, such as long presses, but you may wish to provide feedback for your own widget.

The feedback will only be performed if `isHapticFeedbackEnabled()` is true.

HapticFeedbackConstants Class



Constants		
int	CLOCK_TICK	The user has pressed either an hour or minute tick of a Clock.
int	CONTEXT_CLICK	The user has performed a context click on an object.
int	FLAG_IGNORE_GLOBAL_SETTING	Flag for `View.performHapticFeedback(int, int)`: Ignore the global setting for whether to perform haptic feedback, do it always.
int	FLAG_IGNORE_VIEW_SETTING	Flag for `View.performHapticFeedback(int, int)`: Ignore the setting in the view for whether to perform haptic feedback, do it always.
int	KEYBOARD_TAP	The user has pressed a soft keyboard key.
int	LONG_PRESS	The user has performed a long press on an object that is resulting in an action being performed.
int	VIRTUAL_KEY	The user has pressed on a virtual on-screen key.

More Complex Feedback

- Also possible to create more complex haptic feedback for apps:
- Request permission
- Get the Vibrator object from the system
- call vibrate method

Haptic Feedback

- Request Permission

```
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

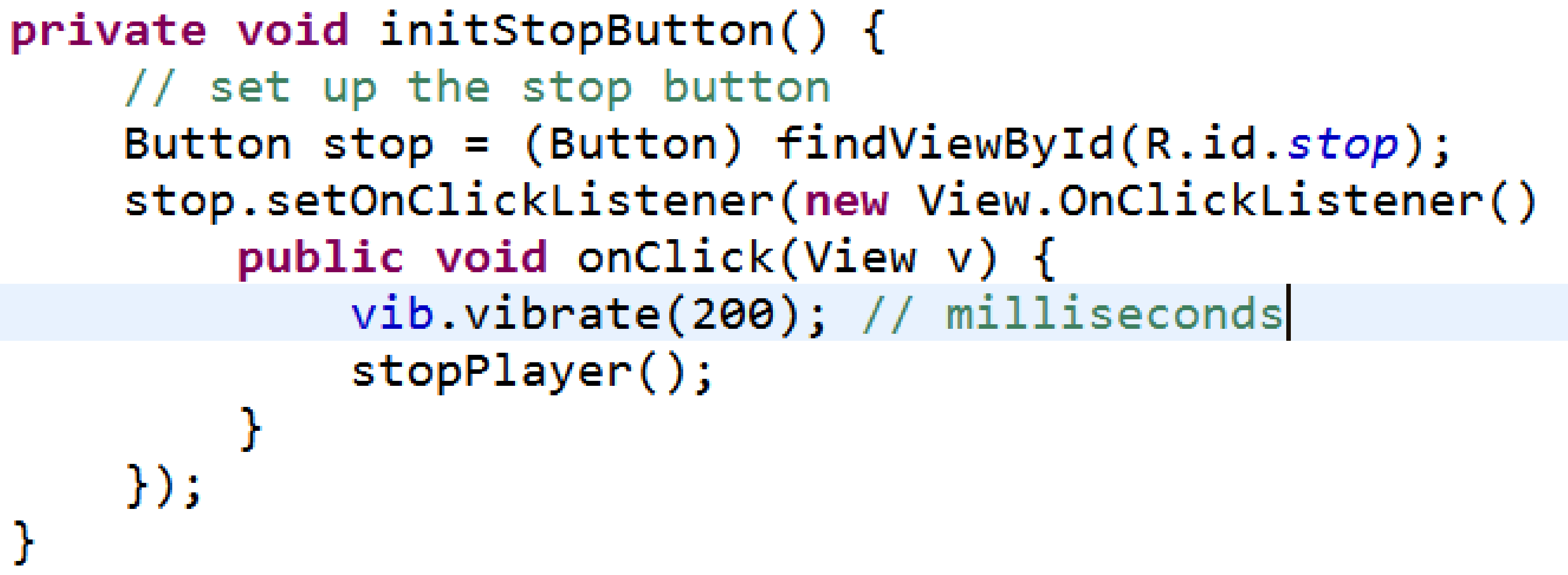
- Get Vibrator

```
private Vibrator vib;
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    vib = (Vibrator) this.getSystemService(VIBRATOR_SERVICE
```

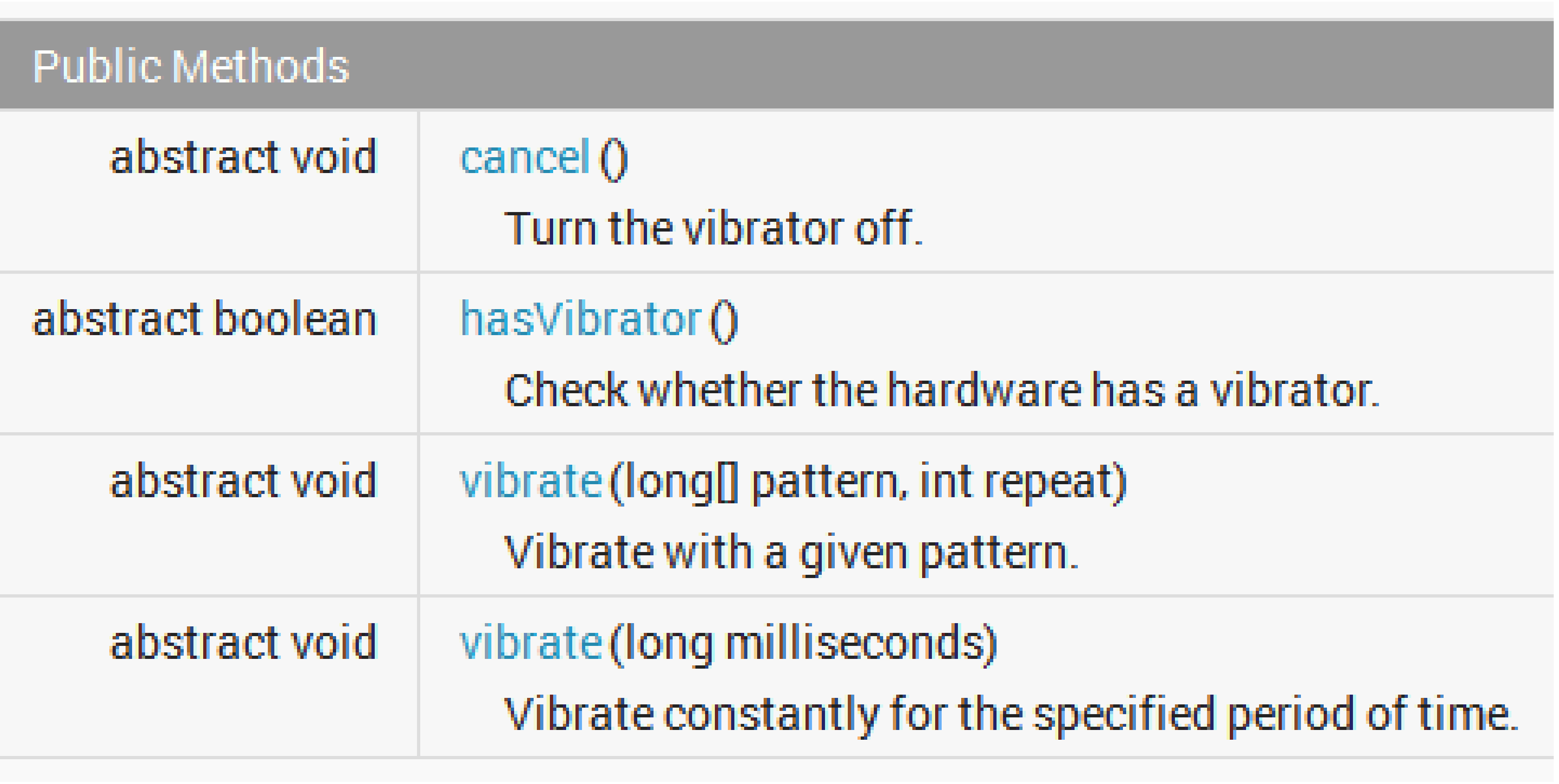
Haptic Feedback

- Create feedback



```
private void initStopButton() {
    // set up the stop button
    Button stop = (Button) findViewById(R.id.stop);
    stop.setOnClickListener(new View.OnClickListener()
        public void onClick(View v) {
            vib.vibrate(200); // milliseconds
            stopPlayer();
        }
    });
}
```

Vibrator Methods



Public Methods	
abstract void	cancel () Turn the vibrator off.
abstract boolean	hasVibrator () Check whether the hardware has a vibrator.
abstract void	vibrate (long[] pattern, int repeat) Vibrate with a given pattern.
abstract void	vibrate (long milliseconds) Vibrate constantly for the specified period of time.

RECORDING AUDIO

Simple and Complex

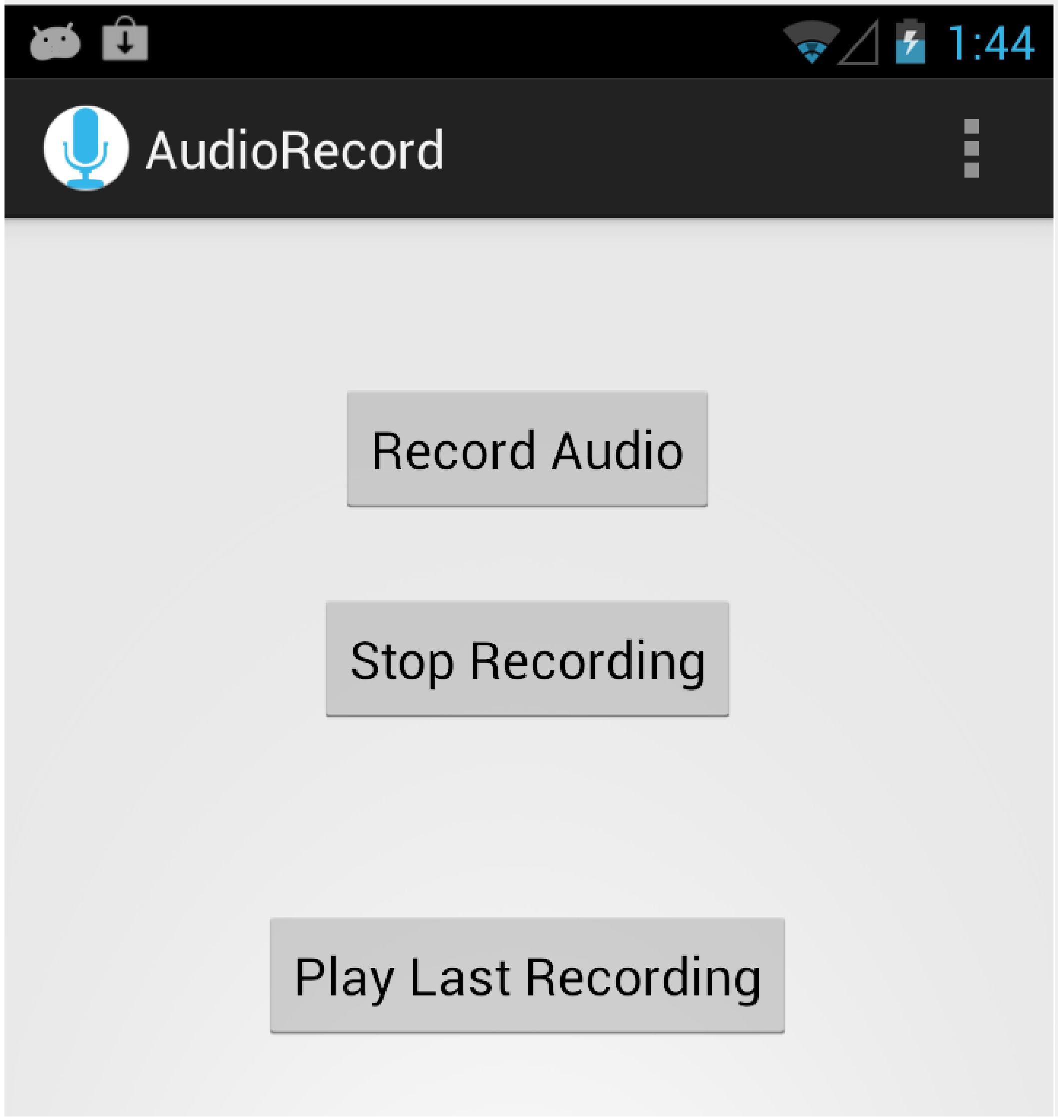
- Simple:
- There may already been an app to record sound on the device.
- Create an Intent and call startActivityForResult
- MediaStore.Audio.Media.RECORD_SOUND_ACTION
- Complex: read on

Recording Audio - MediaRecorder

1. create MediaRecorder object
2. set audio source
3. set audio format
4. set file format for audio
5. set file name to record to
6. prepare MediaRecorder for recording
7. start recording
8. stop and release MediaRecorder

Record Audio

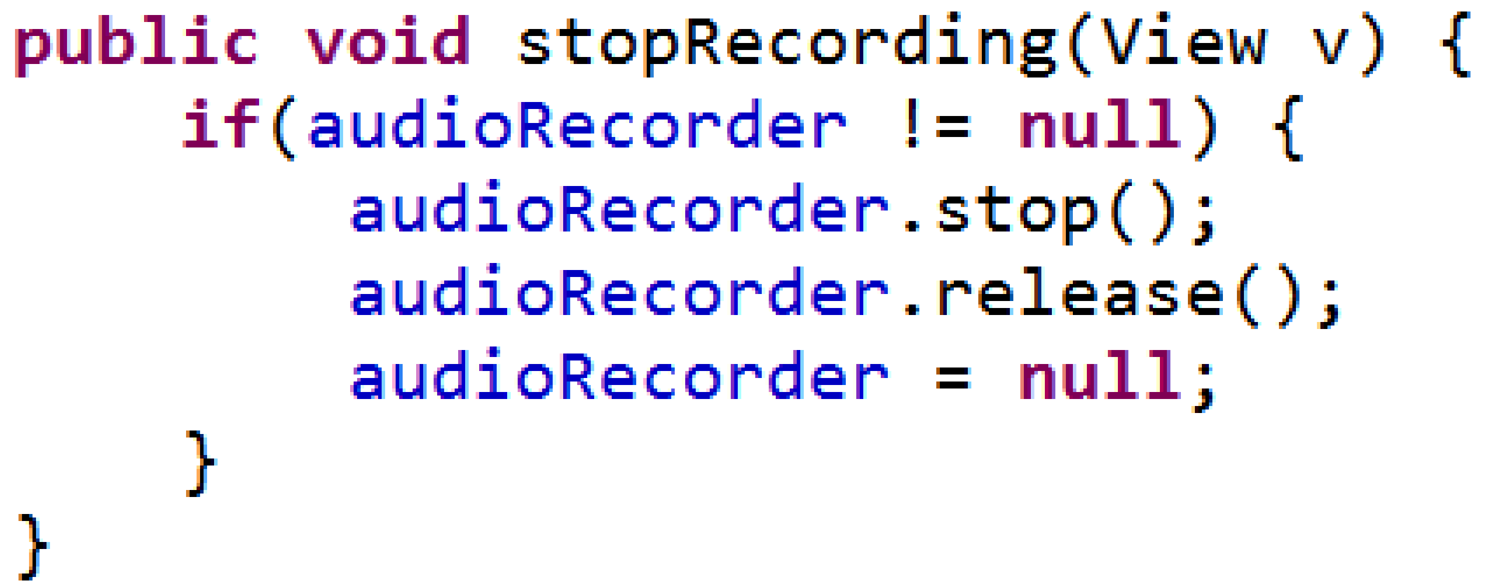
- Simple View with buttons to start and stop recording
- alternatively could change text on record audio button
  - single button or toggle switch



Record Audio

```
public void recordAudio(View v) {
    if(audioRecorder == null)
        audioRecorder = new MediaRecorder();
    String pathForAudioRecording = getFilesDir().getAbsolutePath();
    pathForAudioRecording += RECORD_FILE;

    audioRecorder.setAudioSource(
                MediaRecorder.AudioSource.MIC);
    audioRecorder.setOutputFormat(
                MediaRecorder.OutputFormat.DEFAULT);
    audioRecorder.setAudioEncoder(
                MediaRecorder.AudioEncoder.DEFAULT);

    audioRecorder.setOutputFile(pathForAudioRecording);

    try {
        audioRecorder.prepare();
        audioRecorder.start();
    }
    catch(IOException e) {
        Log.e(TAG, "unable to record. could not prepare or start Me
```

Stop Recording



```
public void stopRecording(View v) {
    if(audioRecorder != null) {
        audioRecorder.stop();
        audioRecorder.release();
        audioRecorder = null;
    }
}
```

- a few seconds of audio results in a file size of ~ 10 kb with default settings
- PERMISSIONS! -- must request RECORD_AUDIO and WRITE_EXTERNAL_STORAGE permissions in manifest file

SPEECH RECOGNITION

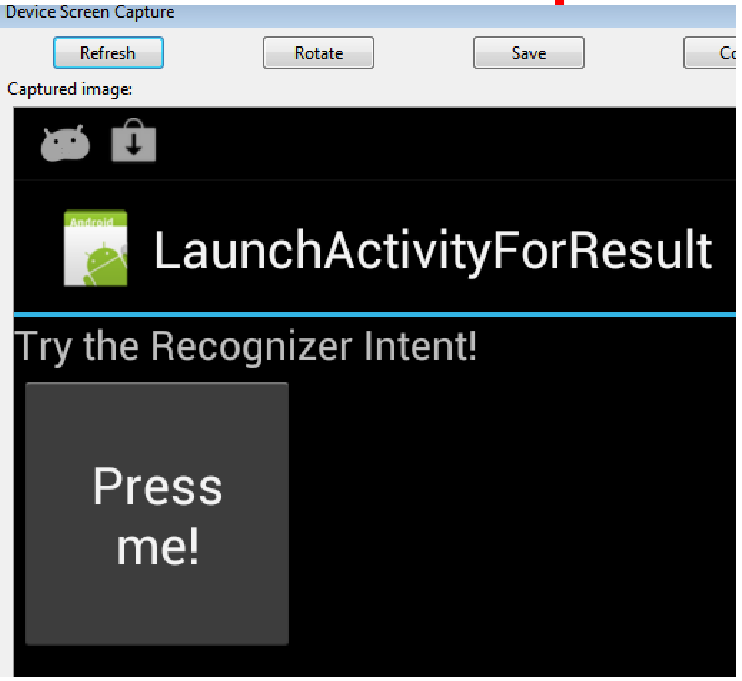
Speech Recognition

- android.speech package
- Simplest example - start an Intent for a result

  RecognizerIntent.ACTION_RECOGNIZE_SPEECH

- uses network
  - works on the dev phones
  - doesn't work on emulator

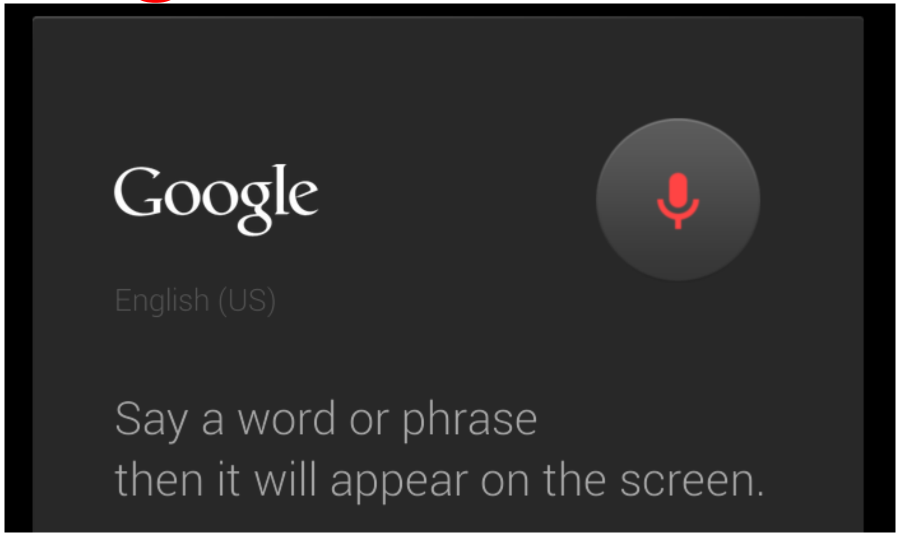
Speech Recognition


Device Screen Capture

Captured image:

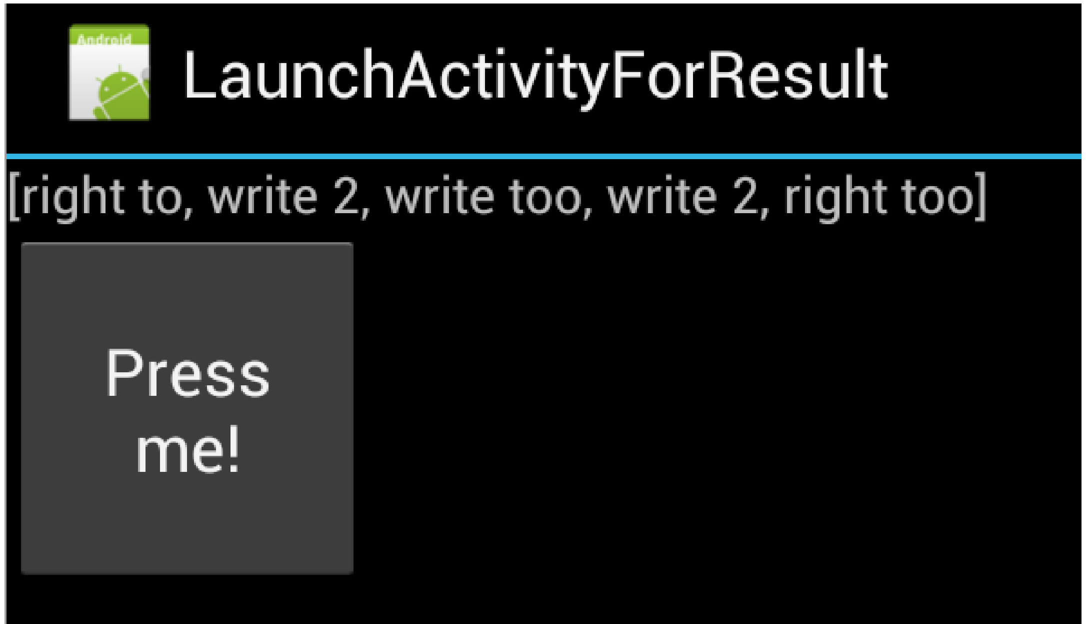
LaunchActivityForResult

Try the Recognizer Intent!



English (US)

Say a word or phrase
then it will appear on the screen.


LaunchActivityForResult

[right to, write 2, write too, write 2, right too]

Starting Intent

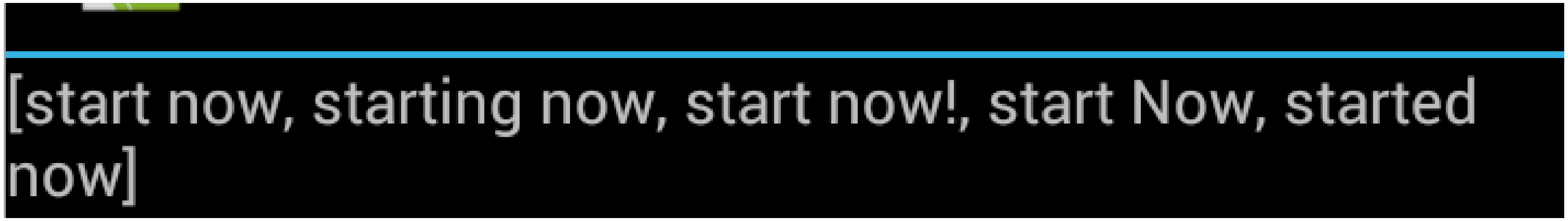
```
//setup button listener
Button startButton = (Button) findViewById(R.id.trigger);
startButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) {
        // RecognizerIntent prompts for speech and returns text
        Intent intent =
            new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);

        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
                RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
        intent.putExtra(RecognizerIntent.EXTRA_PROMPT,
        "Say a word or phrase\nthen it will appear on the screen.");
        startActivityForResult(intent, RECOGNIZER_EXAMPLE);
    }
});
```

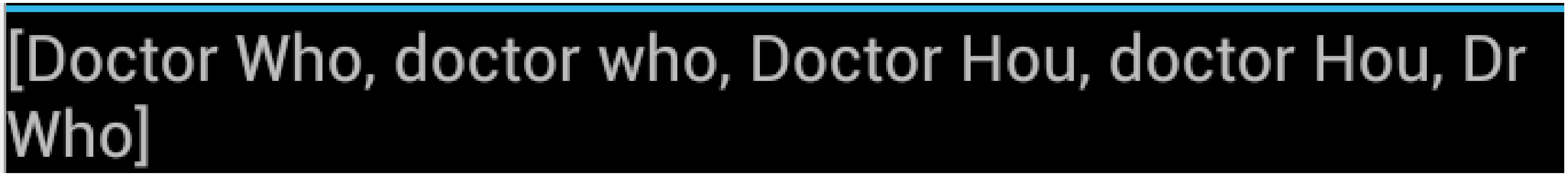
Responding to Result

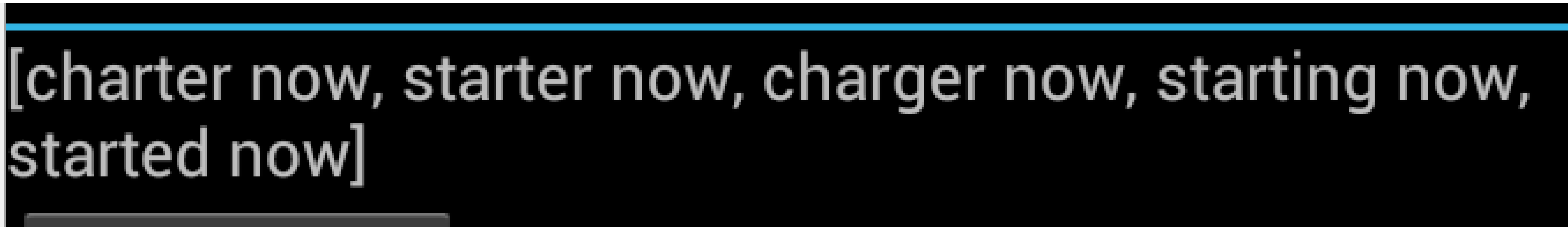
- Note: list of results, ordered by confidence

```
@Override
protected void onActivityResult(int requestCode,
            int resultCode, Intent data) {

    if (requestCode == RECOGNIZER_EXAMPLE
                && resultCode == RESULT_OK) {
        // returned data is a list of matches to the speech
        ArrayList<String> result =
            data.getStringArrayListExtra
            (RecognizerIntent.EXTRA_RESULTS);

        //display on screen
        tv.setText(result.toString());
    }

    super.onActivityResult(requestCode, resultCode, data);
}
```

Results


[start now, starting now, start now!, start Now, started now]


[Doctor Who, doctor who, Doctor Hou, doctor Hou, Dr Who]


[charter now, starter now, charger now, starting now, started now]

ACTIVITY RECOGNITION

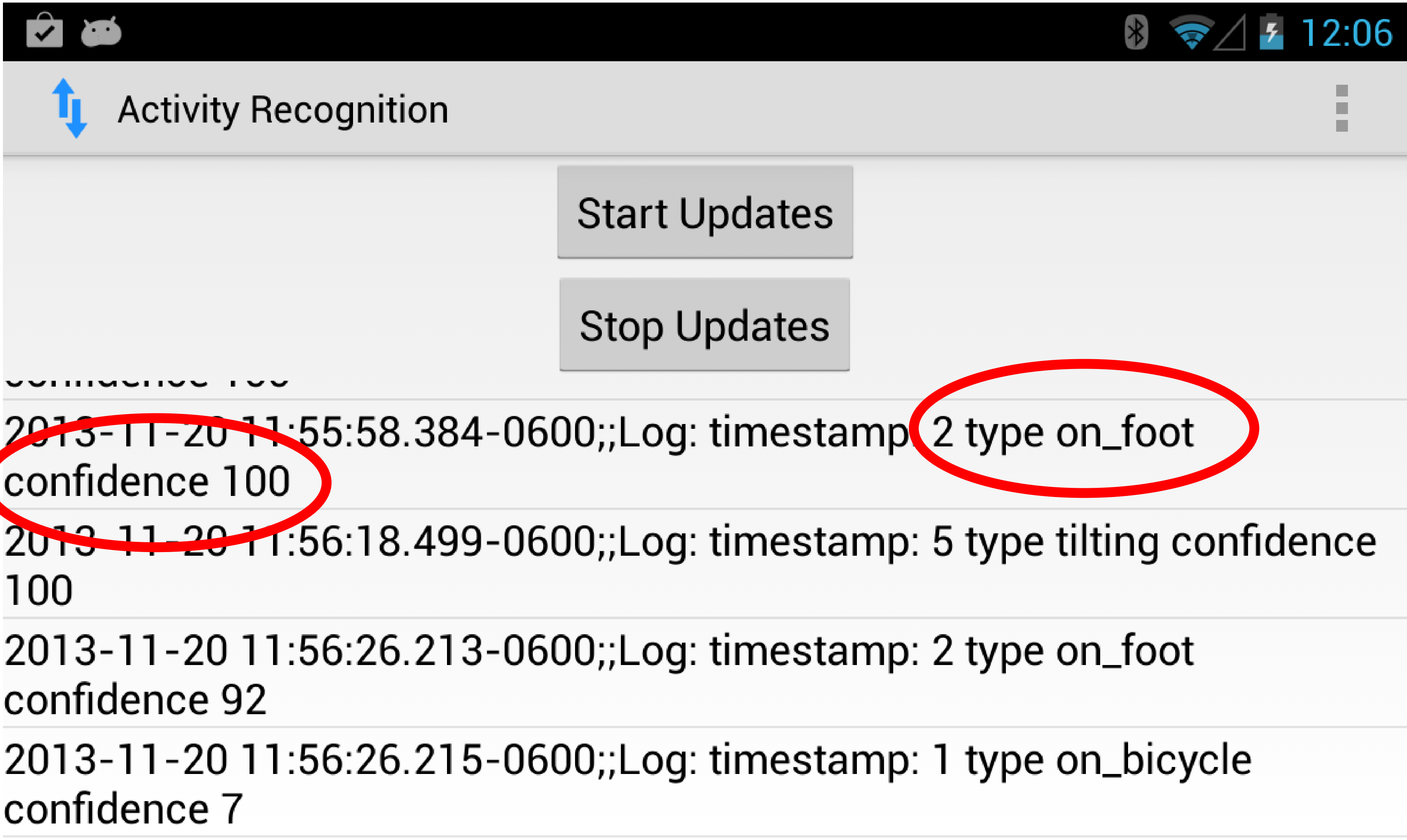
Activity Recognition

- Google service to try and determine user's current activity:
- Standing Still
- walking / on foot
- bicycle
- in vehicle
- unknown

Activity Recognition

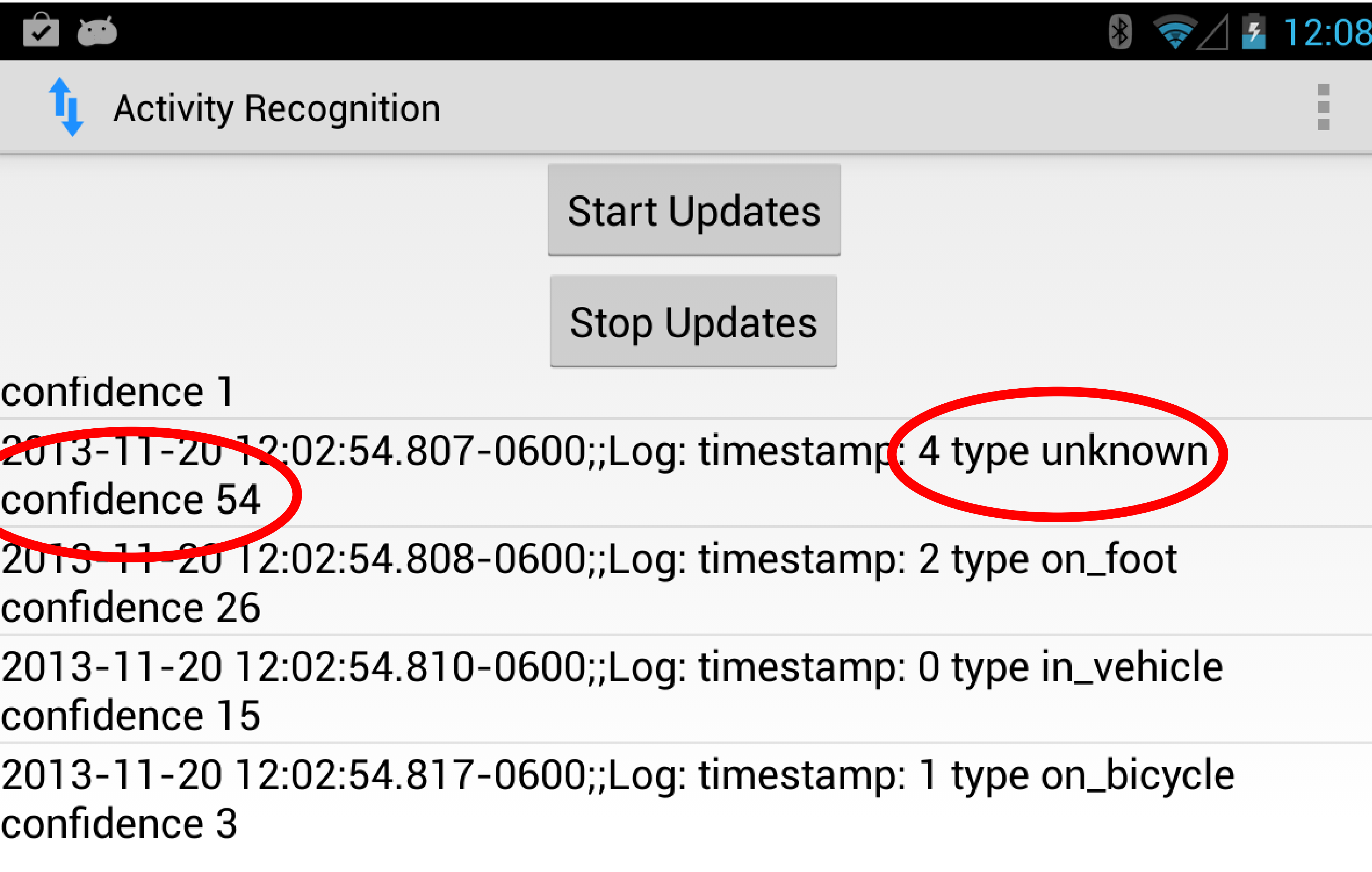
- Uses recognition client
- Related but separate from location client
- Request frequency of updates
- Get list of updates and confidence level of activity

Sample App



```
confidence 100
2013-11-20 11:55:58.384-0600;;Log: timestamp: 2 type on_foot
confidence 100
2013-11-20 11:56:18.499-0600;;Log: timestamp: 5 type tilting confidence
100
2013-11-20 11:56:26.213-0600;;Log: timestamp: 2 type on_foot
confidence 92
2013-11-20 11:56:26.215-0600;;Log: timestamp: 1 type on_bicycle
confidence 7
```

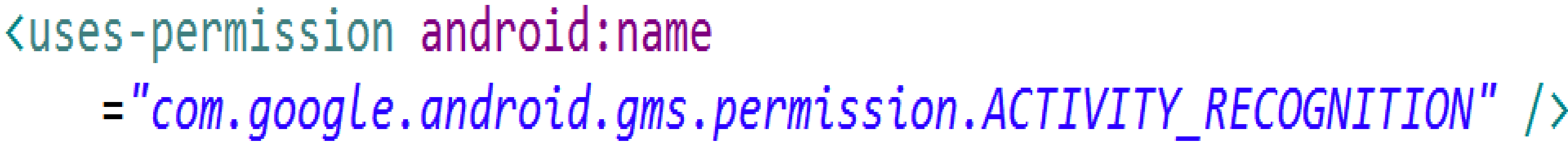
Sample App



```
confidence 1
2013-11-20 12:02:54.807-0600;;Log: timestamp: 4 type unknown
confidence 54
2013-11-20 12:02:54.808-0600;;Log: timestamp: 2 type on_foot
confidence 26
2013-11-20 12:02:54.810-0600;;Log: timestamp: 0 type in_vehicle
confidence 15
2013-11-20 12:02:54.817-0600;;Log: timestamp: 1 type on_bicycle
confidence 3
```

Recognition Activity

- Special Permission

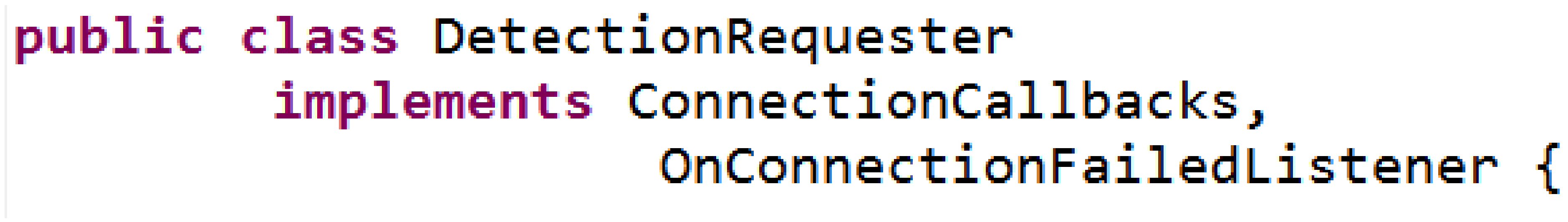


```
<uses-permission android:name
    ="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />
```

- Uses the ***Google Play services APK***
  - must ensure this is available on device

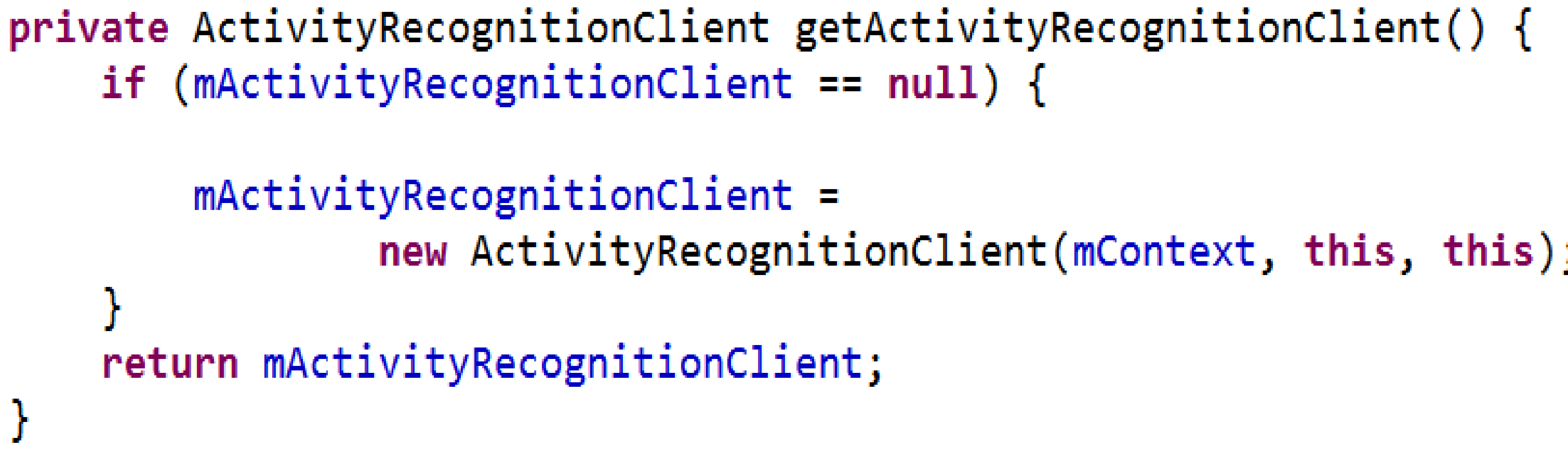
Recognition Activity

- Request updates on activities
- … and implement appropriate call back methods



```java
public class DetectionRequester
        implements ConnectionCallbacks,
                   OnConnectionFailedListener {
```

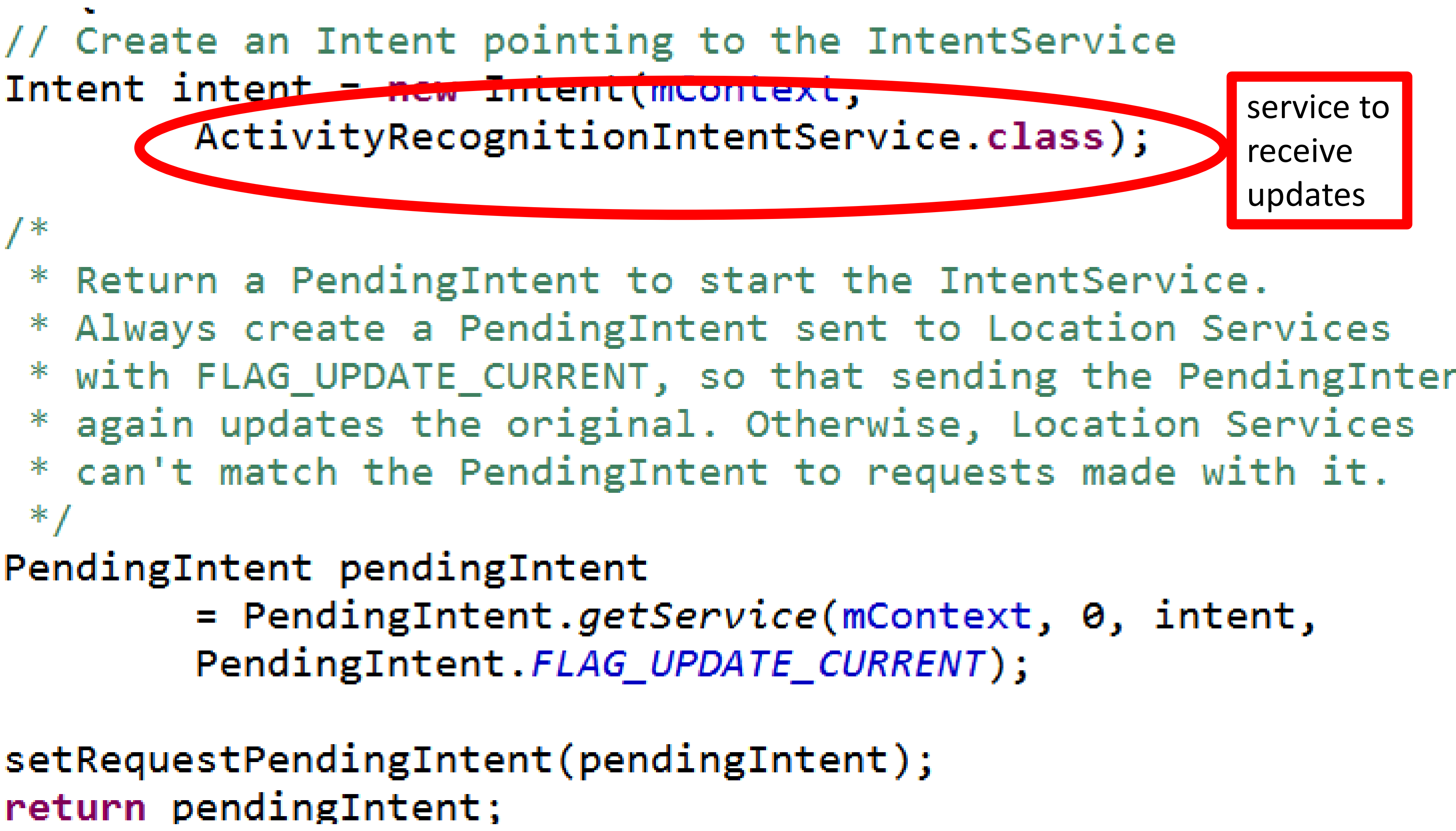
Get ActivityRecognitionClient



```java
private ActivityRecognitionClient getActivityRecognitionClient() {
    if (mActivityRecognitionClient == null) {

        mActivityRecognitionClient =
                new ActivityRecognitionClient(mContext, this, this);
    }
    return mActivityRecognitionClient;
}
```

- Create a *PendingIntent* - an Intent that will be handled at a later time

PendingIntent For ActvityRecognition

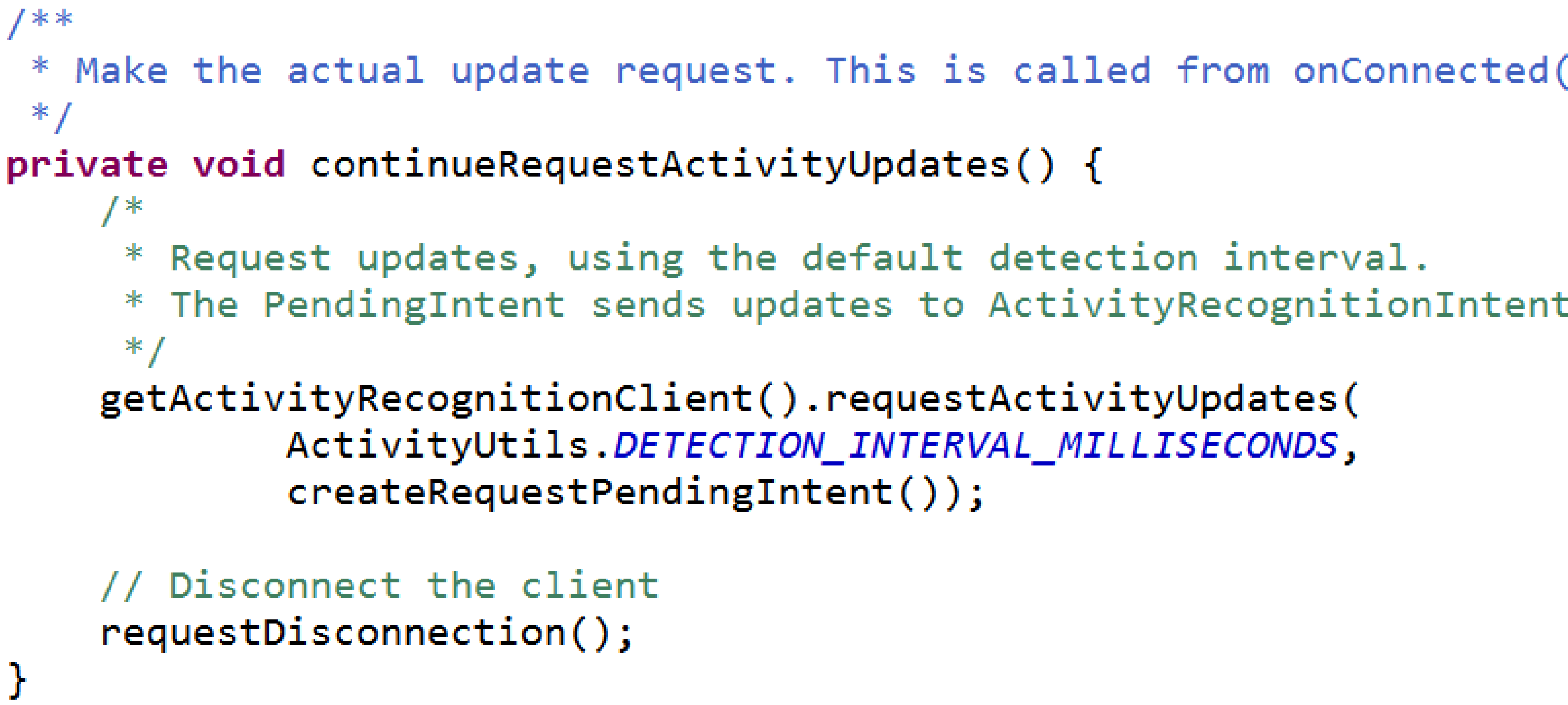


```
// Create an Intent pointing to the IntentService
Intent intent = new Intent(mContext,
        ActivityRecognitionIntentService.class);

/*
 * Return a PendingIntent to start the IntentService.
 * Always create a PendingIntent sent to Location Services
 * with FLAG_UPDATE_CURRENT, so that sending the PendingInter
 * again updates the original. Otherwise, Location Services
 * can't match the PendingIntent to requests made with it.
 */
PendingIntent pendingIntent
        = PendingIntent.getService(mContext, 0, intent,
        PendingIntent.FLAG_UPDATE_CURRENT);

setRequestPendingIntent(pendingIntent);
return pendingIntent;
```

service to receive updates

Request Activity Updates

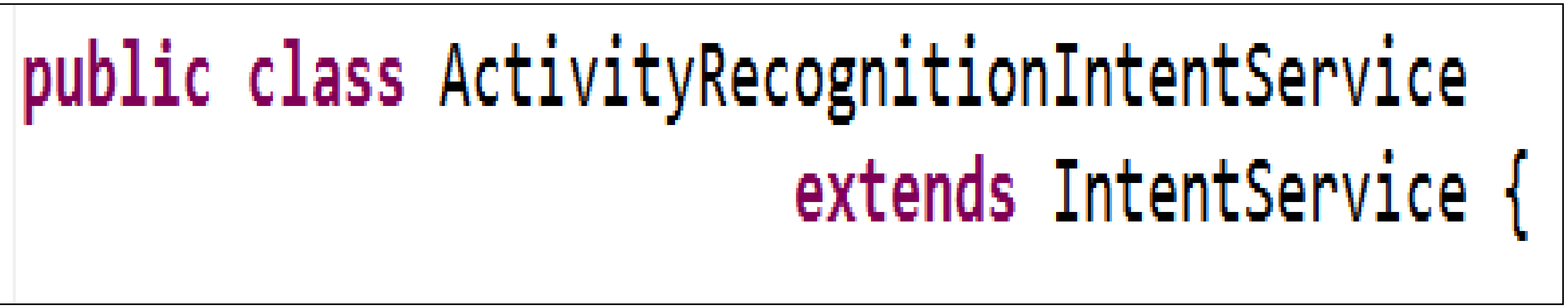
- Using the Pending Intent



```java
/**
 * Make the actual update request. This is called from onConnected(
 */
private void continueRequestActivityUpdates() {
    /*
     * Request updates, using the default detection interval.
     * The PendingIntent sends updates to ActivityRecognitionIntent
     */
    getActivityRecognitionClient().requestActivityUpdates(
            ActivityUtils.DETECTION_INTERVAL_MILLISECONDS,
            createRequestPendingIntent());

    // Disconnect the client
    requestDisconnection();
}
```

Getting Data Back

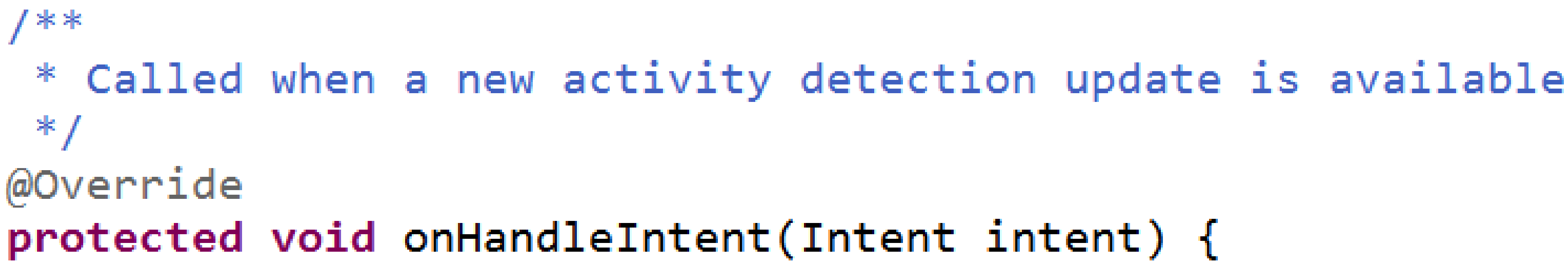
- The ActivityRecognition Service will send Intents with updates
- Create a subclass of IntentService to deal with these



```
public class ActivityRecognitionIntentService
                                extends IntentService {
```

IntentService

- Implement the onHandleIntent method to deal with incoming intents from ActivityRecognition



```
/**
 * Called when a new activity detection update is available
 */
@Override
protected void onHandleIntent(Intent intent) {
```

- Formats data from Intent, logs to shared preferences, notifies user if changed

NOTIFICATIONS

Notification

- Message to user from app OUTSIDE of the app's normal UI
- Appears first in *Notification Area*
  - *formerly status bar notifications*
- User looks at details of Notification in the *Notification Drawer*
- The Notification Area and Notification Drawer are both controlled by the system

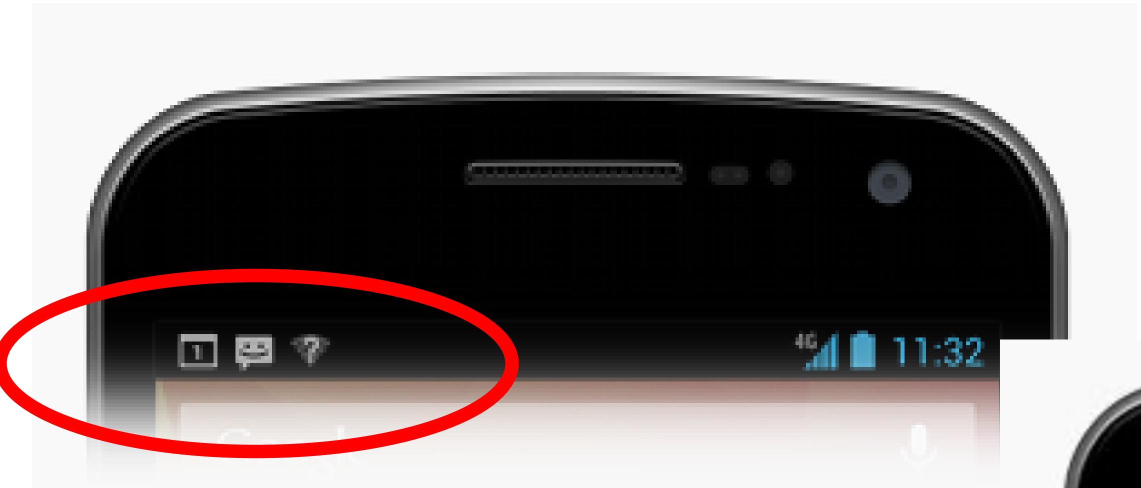


Figure 1. Notifications in the notification area

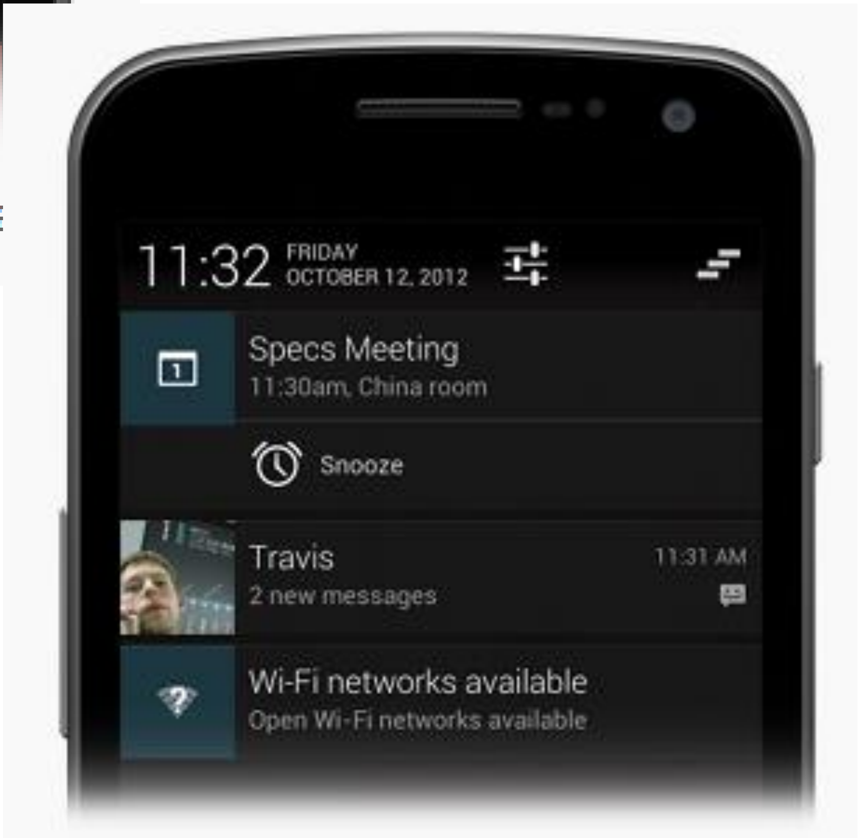
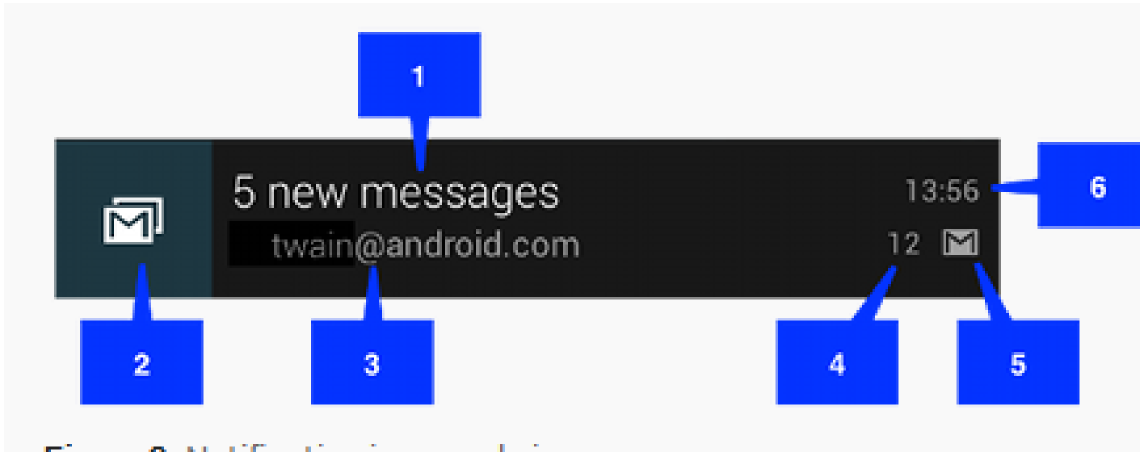


Figure 2. Notifications in the notification drawer.

http://developer.android.com/guide/topics/ui/notifiers/notifications.html

Notification Style - Normal

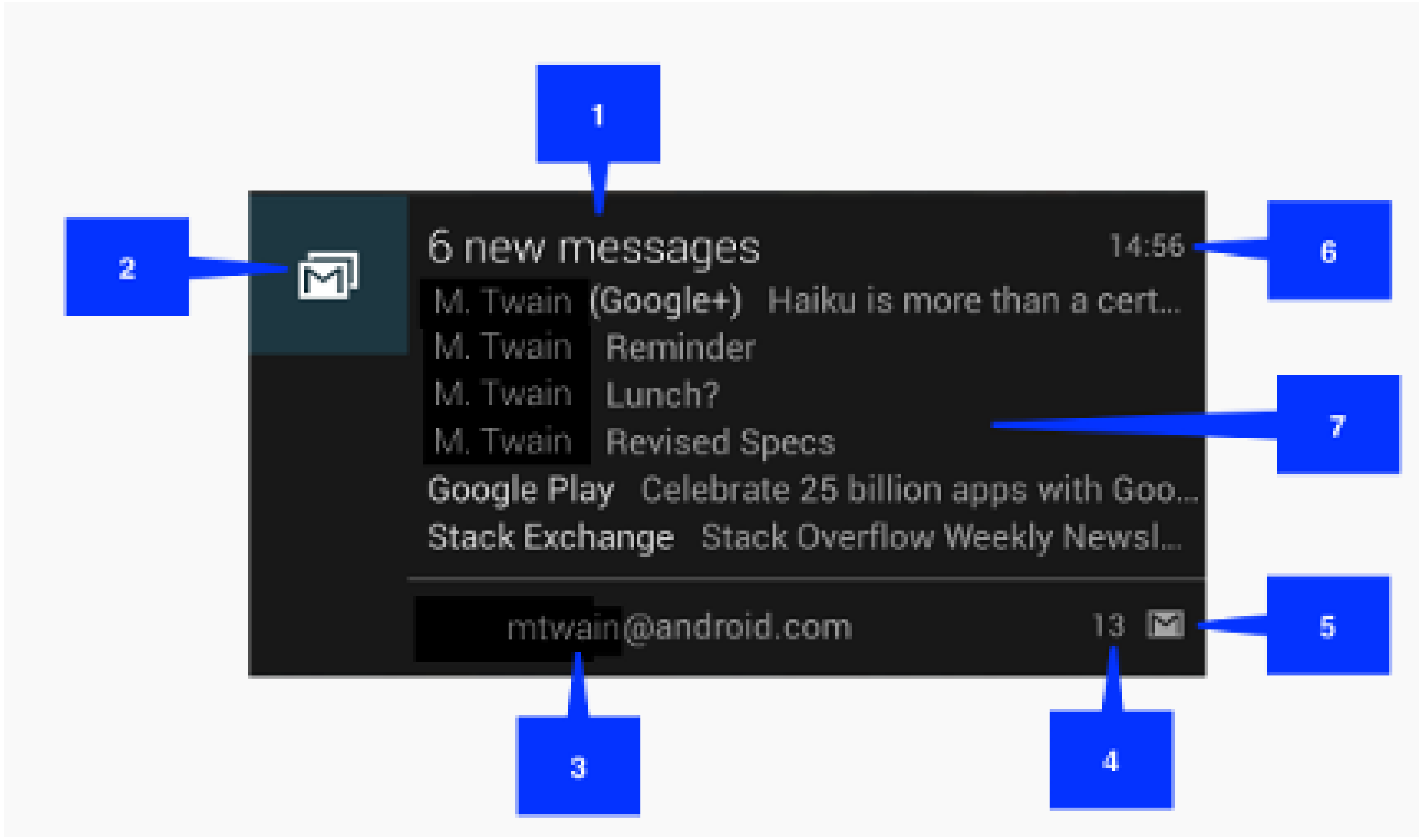
- In the drawer Notifications *Normal View* or *Big View*
- Normal View



1. content title
2. large icon
3. content text
4. content info
5. small icon
6. time issued

Notification Style - Big

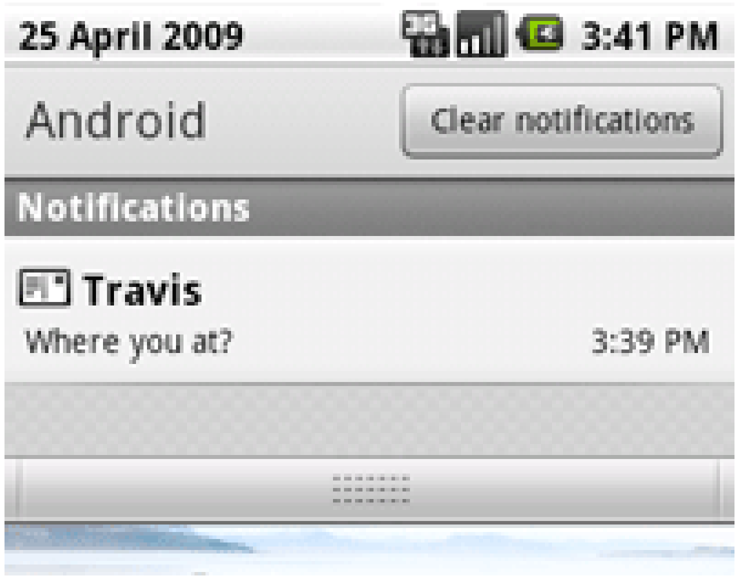
- Adds 7 -> details area



6 new messages 14:56
M. Twain (Google+) Haiku is more than a cert...
M. Twain Reminder
M. Twain Lunch?
M. Twain Revised Specs
Google Play Celebrate 25 billion apps with Goo...
Stack Exchange Stack Overflow Weekly Newsl...
mtwain@android.com 13

Notifications

- Advice for Notifications
  - use for time sensitive events
  - ... that involve other people
  - don't create notifications for events not directed at user
  - don't create notification for Activity that is active



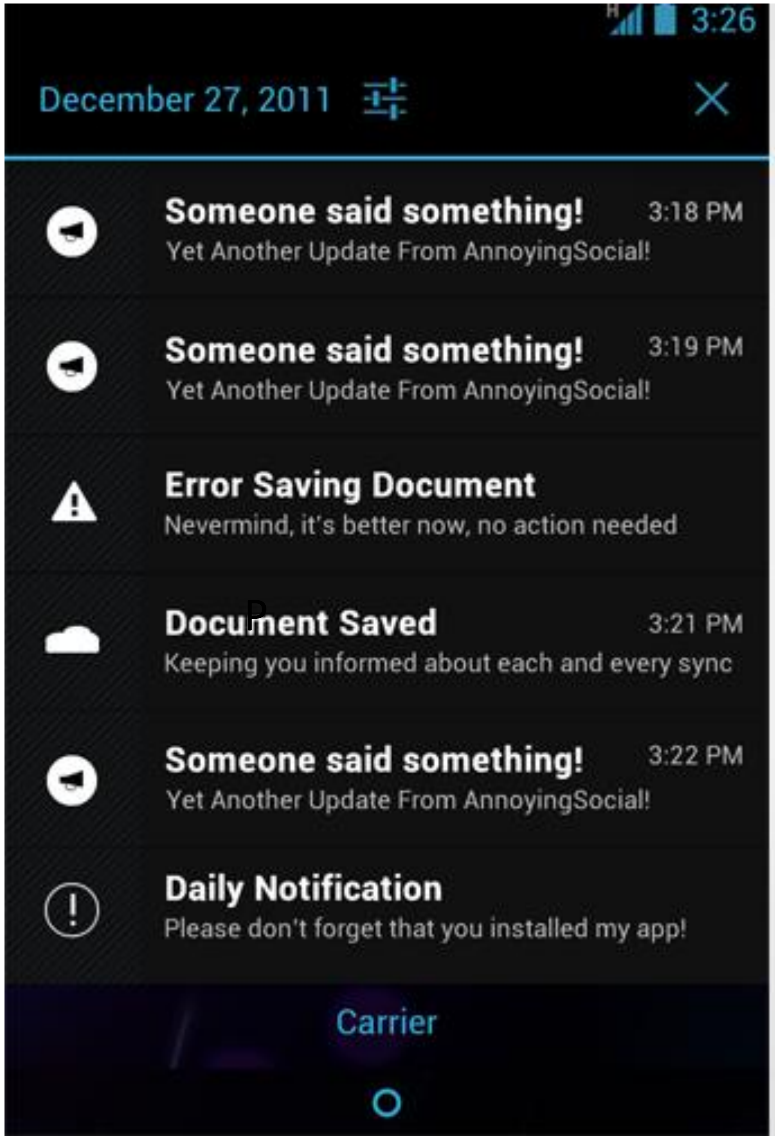
Travis
Where you at? 3:39 PM

Figure 2. The notifications window.

Notifications Window

- Don't create notifications for low level technical details
- Don't create notifications for errors that user can't fix or if app can recover on its own
- Don't create notifications for services user can't start and stop

Bad Notifications



3:26

December 27, 2011

Someone said something! 3:18 PM
Yet Another Update From AnnoyingSocial!

Someone said something! 3:19 PM
Yet Another Update From AnnoyingSocial!

Error Saving Document
Nevermind, it's better now, no action needed

Document Saved 3:21 PM
Keeping you informed about each and every sync

Someone said something! 3:22 PM
Yet Another Update From AnnoyingSocial!

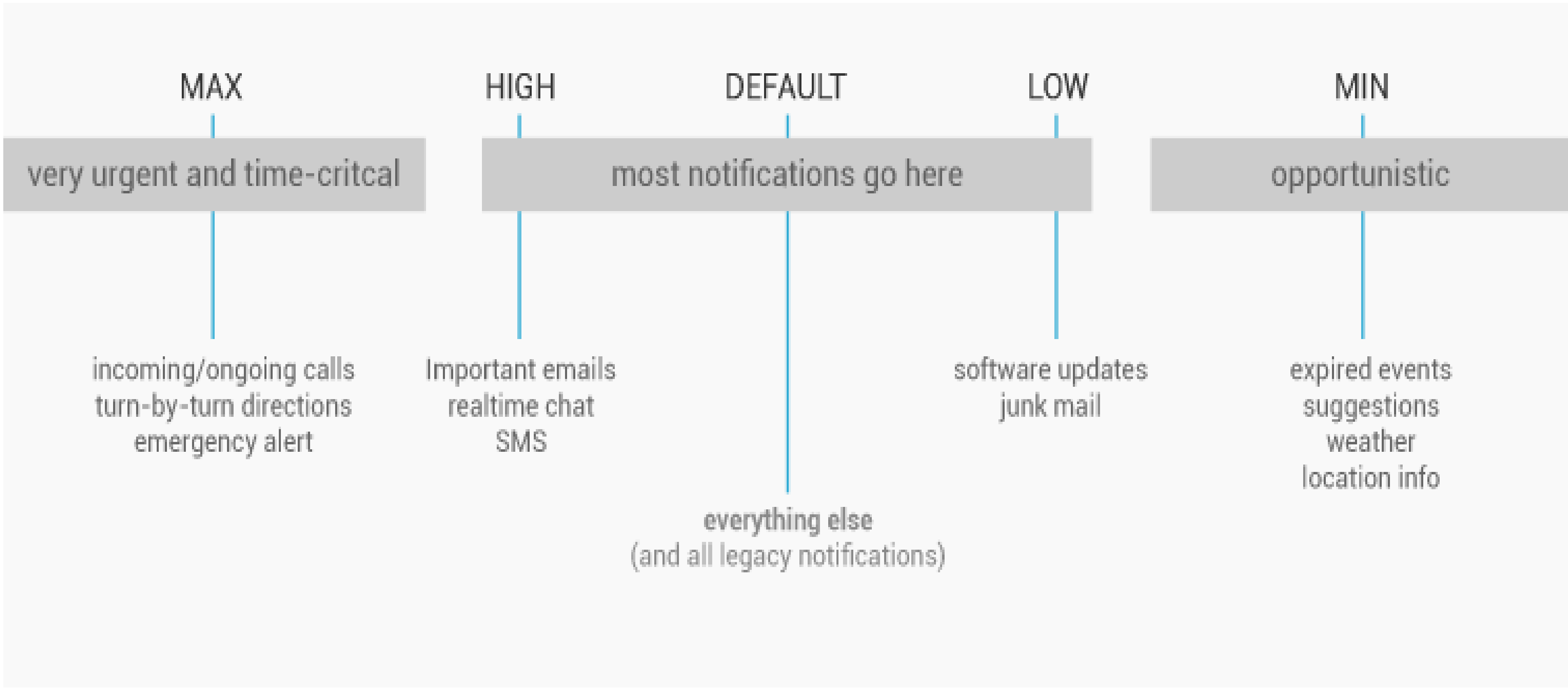
Daily Notification
Please don't forget that you installed my app!

Carrier

Notifications DOs

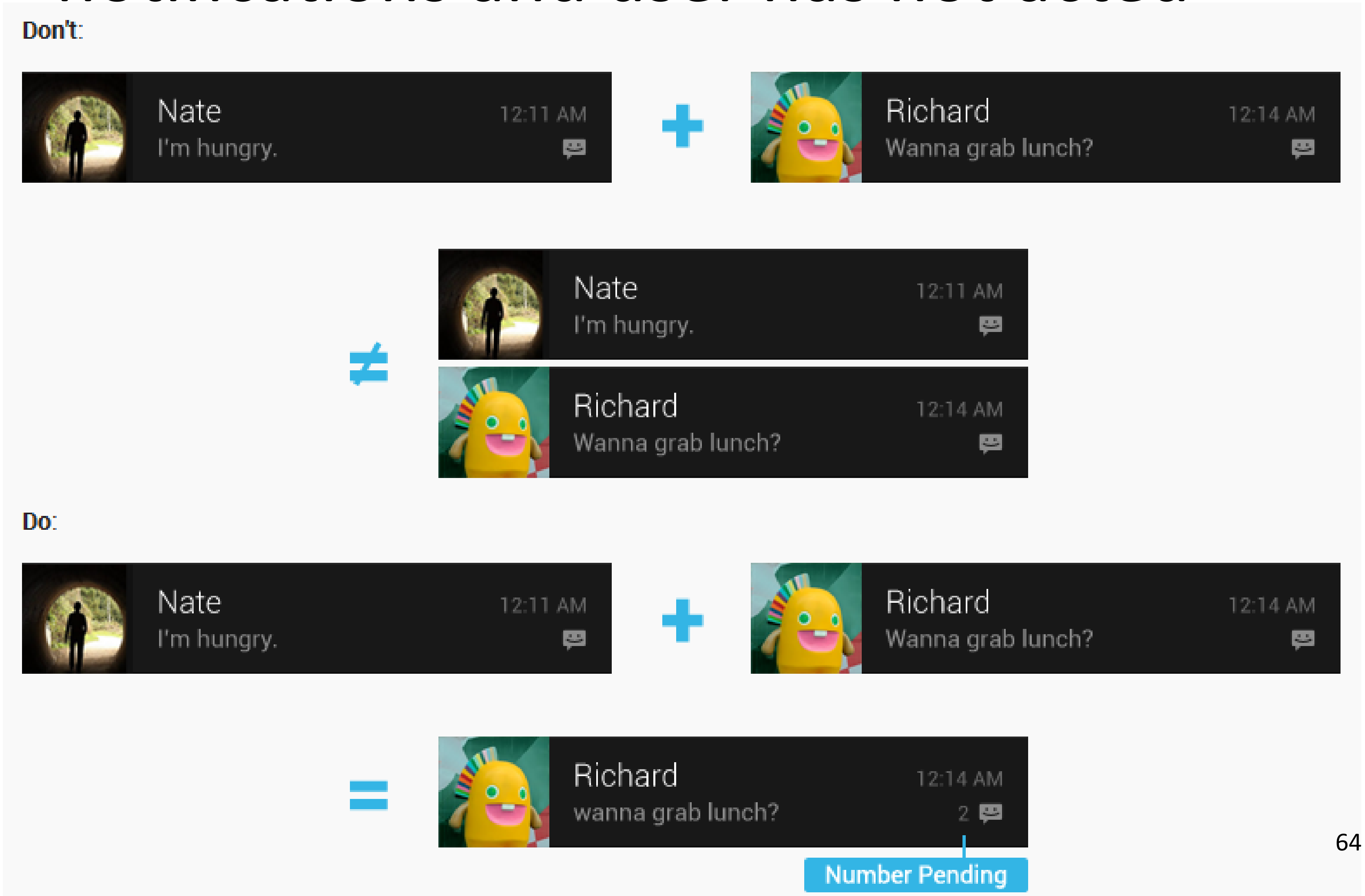
- Make it personal
  - if notification caused by action of a contact, include contact image or icon
- Navigate to correct place
  - may take user to deep in app, update the UP icon and back stack (UP vs. BACK)
- Manage priority
  - in Jelly Bean notifications have a priority

Notification Priority



More Notification DOs

- Stack notifications if app creates multiple notifications and user has not acted



More Notification Dos

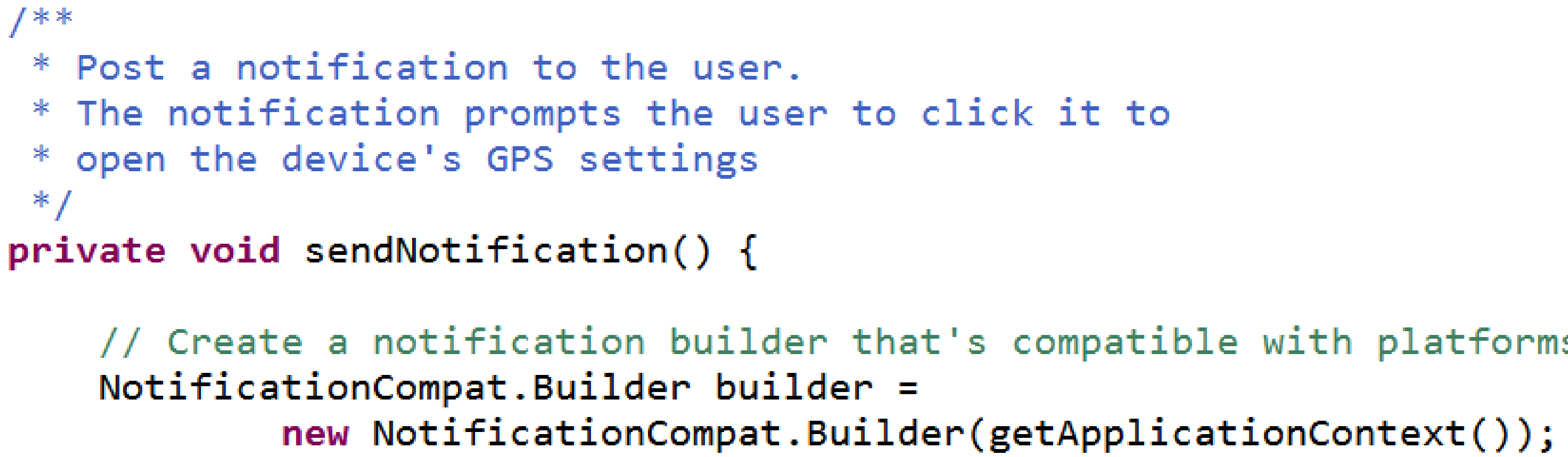
- Make them optional - PLEASE
- use distinct icon
  - but not a different color
- make LED glow based on priority

The Code of Notifications

- Notification Object
- Convenience NotificationBuilder
- Must have at a minimum
  - small icon
  - title
  - detail text

Simple Notification Building

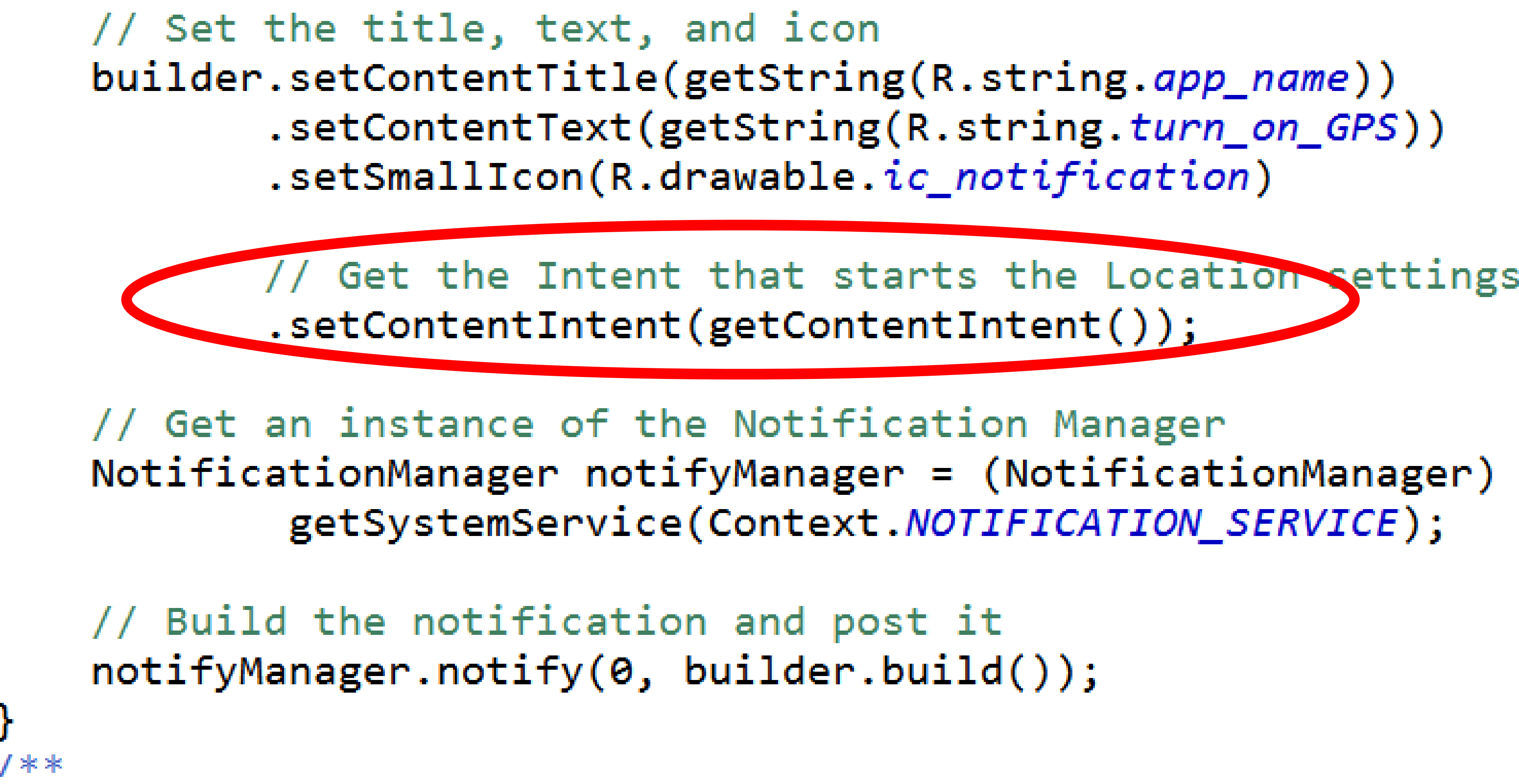
- Activity Recognition App
- Send notification to turn on GPS



```
/**
 * Post a notification to the user.
 * The notification prompts the user to click it to
 * open the device's GPS settings
 */
private void sendNotification() {

    // Create a notification builder that's compatible with platform
    NotificationCompat.Builder builder =
            new NotificationCompat.Builder(getApplicationContext());
```

Simple Notification Building

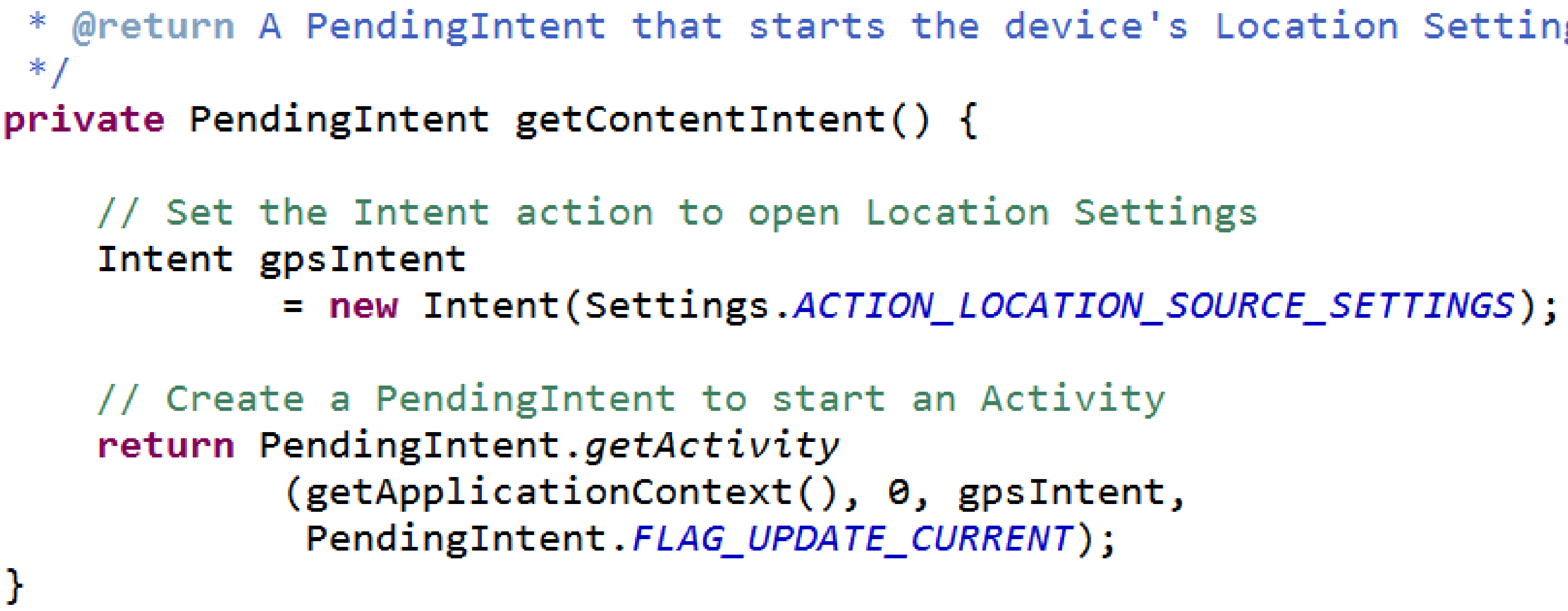
- Note setting title, text, and small icon



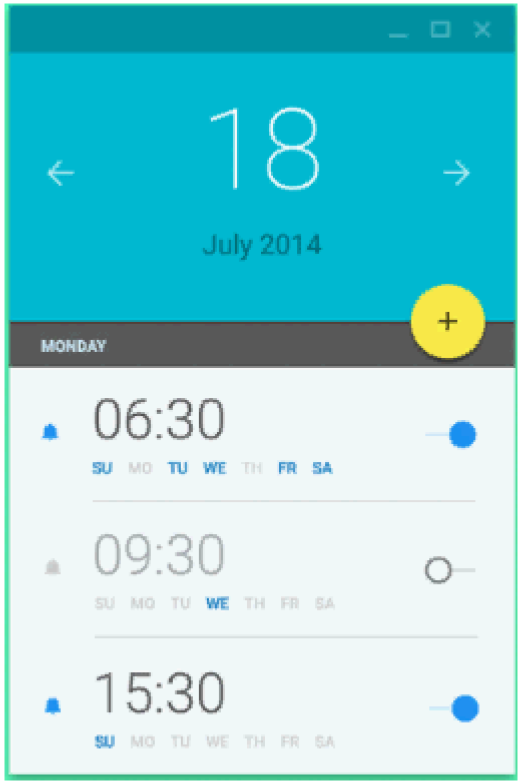
```
    // Set the title, text, and icon
    builder.setContentTitle(getString(R.string.app_name))
           .setContentText(getString(R.string.turn_on_GPS))
           .setSmallIcon(R.drawable.ic_notification)

           // Get the Intent that starts the Location settings
           .setContentIntent(getContentIntent());

    // Get an instance of the Notification Manager
    NotificationManager notifyManager = (NotificationManager)
            getSystemService(Context.NOTIFICATION_SERVICE);

    // Build the notification and post it
    notifyManager.notify(0, builder.build());
}
/**
```

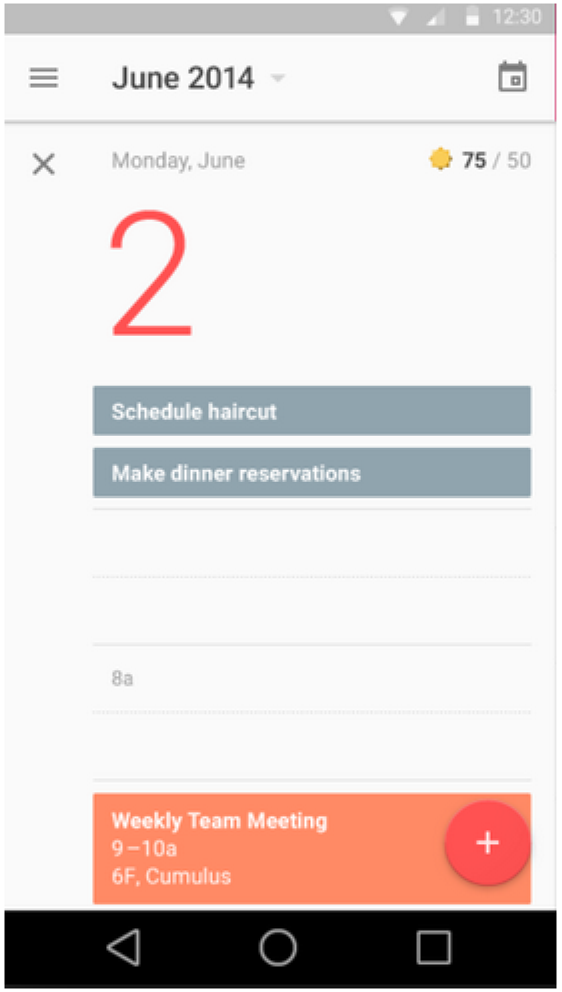
Notification Example

- Includes Intent so user can turn on the GPS
- Didn't just tell them to turn on GPS, make it easy for them to do so.



```
 * @return A PendingIntent that starts the device's Location Settin
 */
private PendingIntent getContentIntent() {

    // Set the Intent action to open Location Settings
    Intent gpsIntent
            = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);

    // Create a PendingIntent to start an Activity
    return PendingIntent.getActivity
            (getApplicationContext(), 0, gpsIntent,
             PendingIntent.FLAG_UPDATE_CURRENT);
}
```

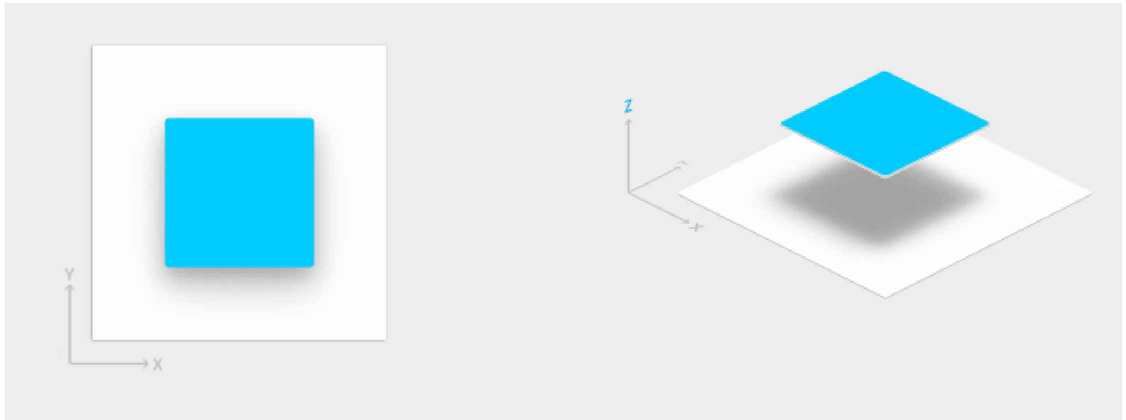




MATERIAL DESIGN

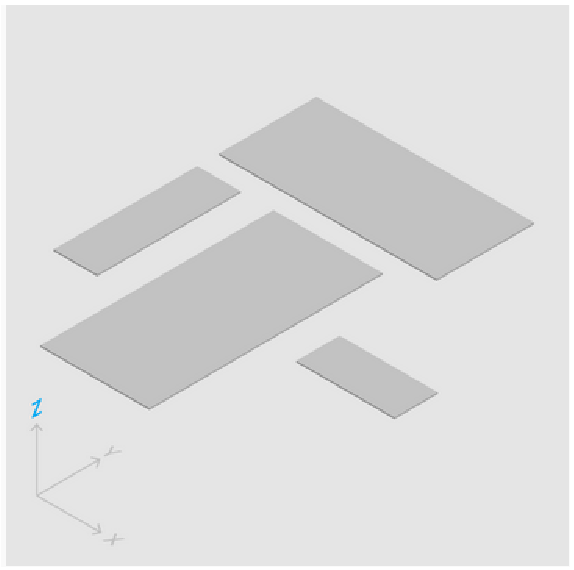
Material Design

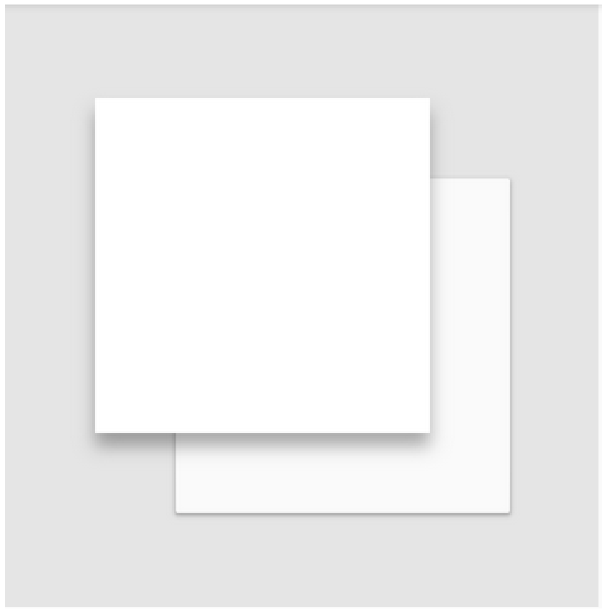
- Introduced in Lollipop, Fall 2014
- Material design is a system and set of guidelines to help users understand the user interface and functionality of apps
- "Took a step back" and build a design language based on classic design principles and real world materials, namely **paper and ink**.



Features of Material Design

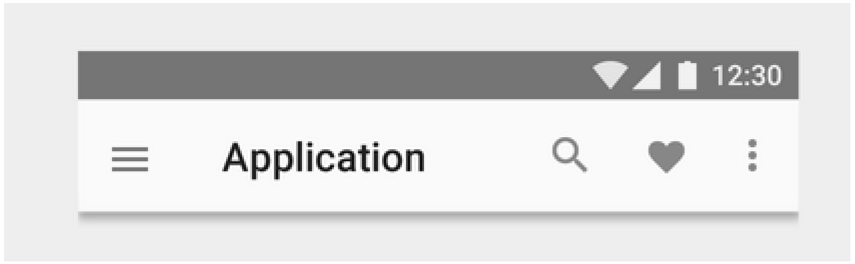
- Objects (visible widgets) in an app consist of paper and ink
- paper can de different sizes, but preferred shapes are rectangles and rounded rectangles
  - all elements are 1 dpi thick
- paper elements can be positioned next to and on top of each other in layers

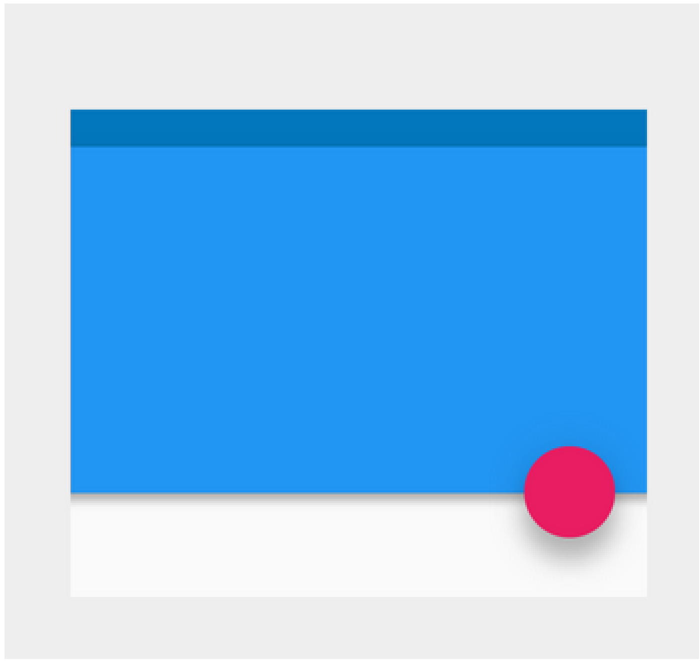


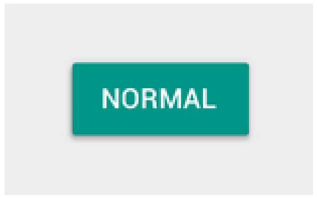


Features of Material Design

- Materials case shadows depending on their position



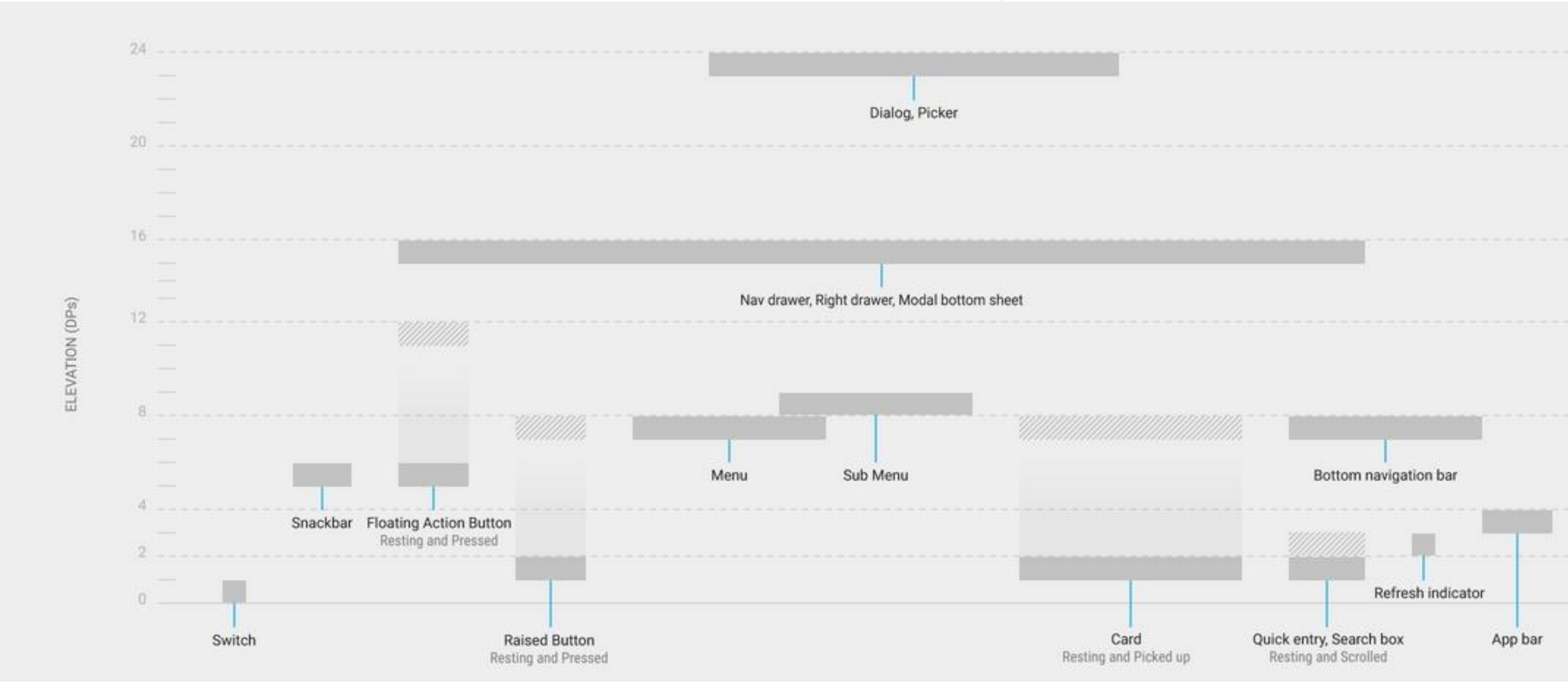




- Guidelines on elevations for various components, affects amount of shadow

Component Elevation Guide

Elevation (dp)	Component
24	Dialog
	Picker
16	Nav drawer
	Right drawer



Motion in Material Design

- Motion of interface elements used to convey meaning
- Motion should be natural without abrupt starts and stops
- User input from touch, voice, mouse, and keyboard (chrome on desktops)
- Motion (animation) of user interface elements in response to user actions
  - surface reaction, material response, radial reaction

Surface Reaction

- reaction of the ink to user input or action
- vary animation based on point of input
- Touch Ripple

Material Response

- Material responds to being touched or activated by the user
- indicates the paper component is made out of is active, being used
- Move or transform form paper based on interaction
- Move up when activated

Radial action

- ink should respond like water when touched
- ripples when touching a smooth surface of water

MATERIAL DESIGN FOR DEVELOPERS

How to Implement App with Material Design

- What tools are available to add material design features to our apps?
- Android 5.0 / Lollipop / API level 21 added:
- A new theme: material
- new widgets
- new view groups
- new apis for custom shadows and animations

Material Theme

- More complex than past themes such as theme and holo
- Includes:
- system widgets with option of picking color palette
- built in touch feedback animations for those system widgets
- activity transition animations

Using Material Theme

The material theme is defined as:

- `@android:style/Theme.Material` (dark version)
- `@android:style/Theme.Material.Light` (light version)
- `@android:style/Theme.Material.Light.DarkActionBar`

For a list of material styles that you can use, see the API reference for `R.style`.

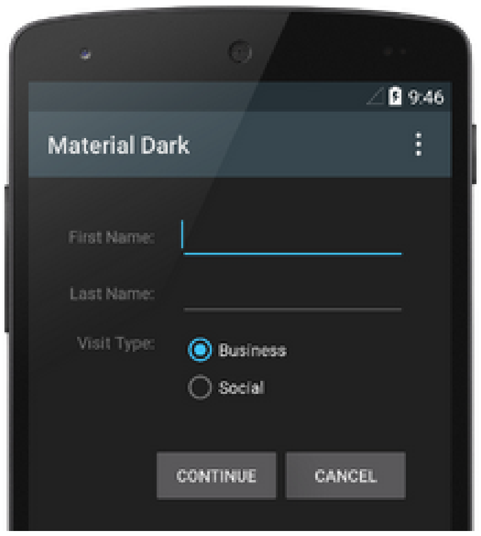


Figure 1. Dark material theme

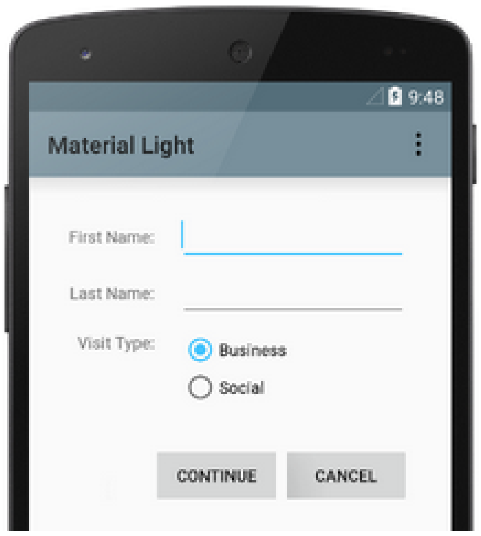


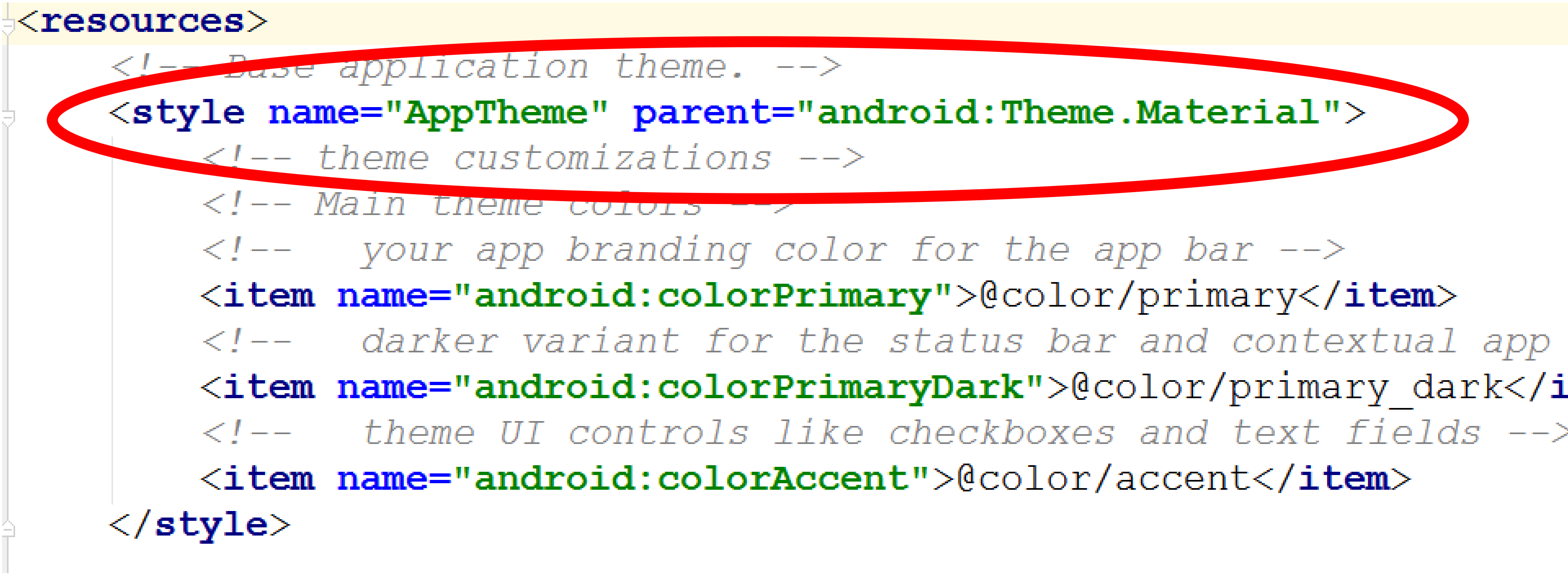
Figure 2. Light material theme

Creating Apps with Material Design

1. apply material theme to app
2. create layouts following material design guidelines
3. specify elevation of views to cast shadows
4. use system widgets for lists and cards
5. customize animations of app

Apply Material Theme

- Specify in styles.xml that your app style is using Material Design



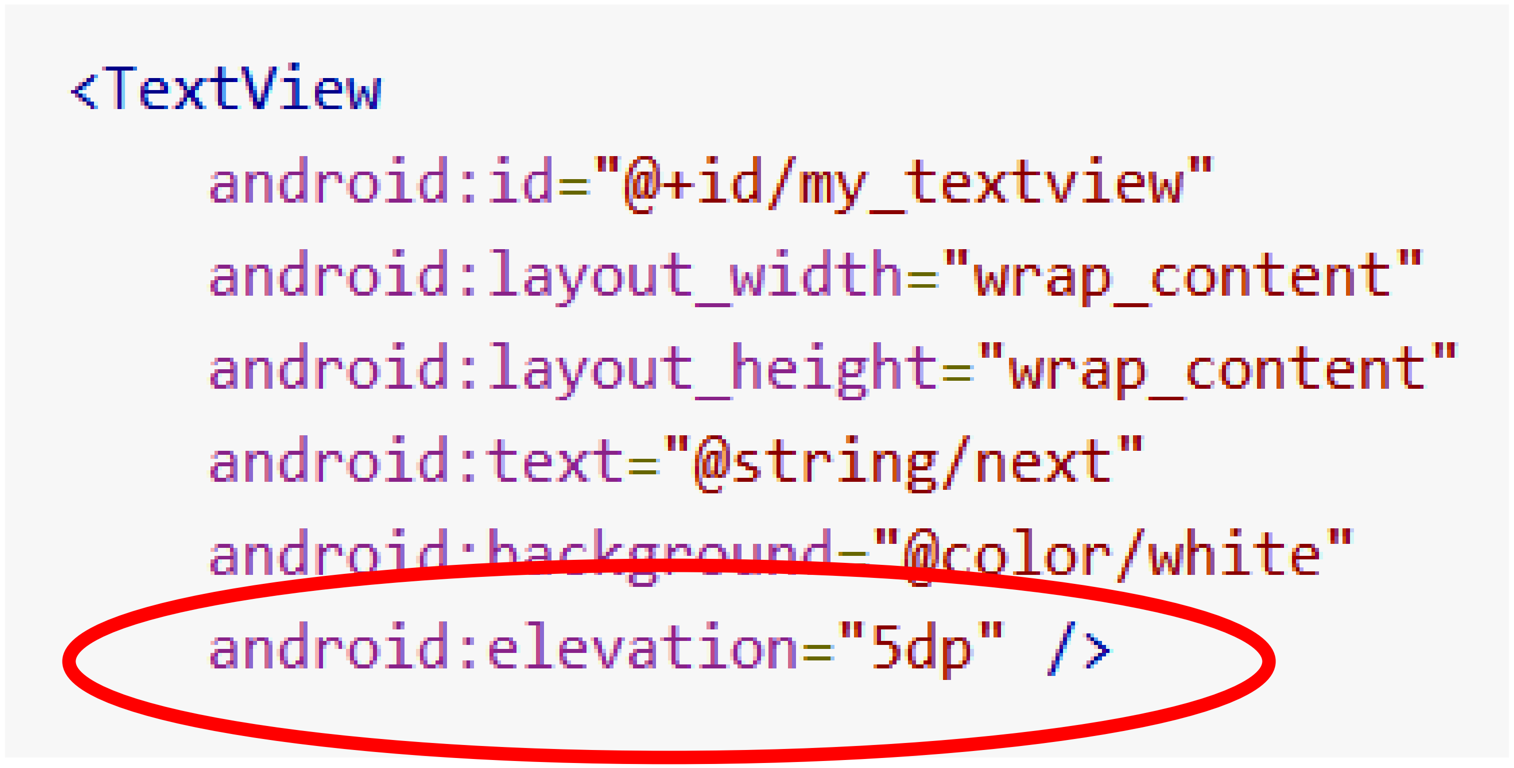
```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="android:Theme.Material">
        <!-- theme customizations -->
        <!-- Main theme colors -->
        <!--   your app branding color for the app bar -->
        <item name="android:colorPrimary">@color/primary</item>
        <!--   darker variant for the status bar and contextual app
        <item name="android:colorPrimaryDark">@color/primary_dark</i
        <!--   theme UI controls like checkboxes and text fields -->
        <item name="android:colorAccent">@color/accent</item>
    </style>
```

Create Lists and Cards

- Android 5.0 / Lollipop / API level 21 added view groups to make apps based around material design possible
- RecycleView
- a more advanced and flexible version of ListView
- inlcudes
- layout managers for positioning items
- built in animations

Create Layouts

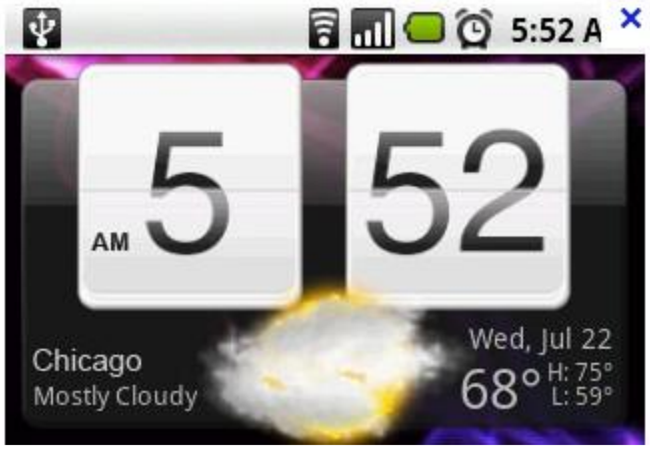
- When using material design your layouts have a elevation property
- affects shadows created by widget



```
<TextView
    android:id="@+id/my_textview"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/next"
    android:background="@color/white"
    android:elevation="5dp" />
```

WIDGETS

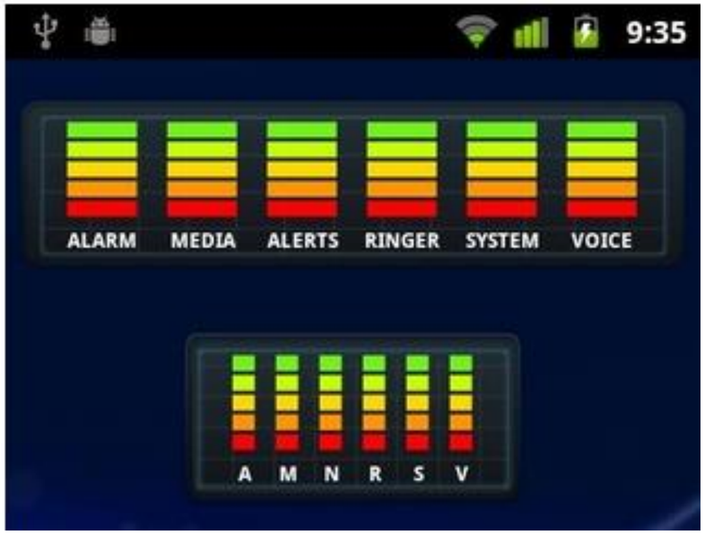
Widgets



- Referred to as an App Widget
- widgets are miniature application views than can be added to other applications
  - normally the Home Screen View
  - other "App Widget Hosts"
- Widget sent periodic updates
- Widgets essentially a BroadcastRecevier with XML layout

Widgets

- To create App Widget:
- Create a AppWidgetProviderInfo
- object that contains metadata for the App Widget, layout, update frequency
  - normally defined in XML
- Implement AppWidgetProvider class that defines basic methods to update Widget
- create layout: not all layouts and UI widgets supported



NFC - NEAR FIELD COMMUNICATION

NFC

- Another short range wireless technology
- VERY short range: 4 cm or less to initiate a connection
- Allows transmission of a small amount ("payload") of data between NFC tag or another Android device
- NFC Tags offer read and write capability
- More complex tags can perform mathematical ops, use encryption, and even have an operating enviroment

NFC Modes of Operation

1. Reader / Writer mode
   - allows NFC device to read and / or write passive NFC tags and stickers
2. P2P mode
   - allows NFC device to exhange data with other NFC peers
   - Used by Android Beam
3. Card emulation
   - allow NFC device to act as an NFC card

NFC Basics

- Reading NDEF data from an NFC tag
- NDEF: NFC Data Exchange Format
- *tag dispatch system*: analyzes discovered tags, categories data, and starts applications (via intent filters) interested in data
- tag dispatch system scanning for NFC tags if screen unlocked (unless turned off in settings)

Android Beam

- Android Beam allows simple peer to peer data exchange between two Android powered devices
- enable Android Beam for an app by calling:
  - setNdefPushMessage(), accepts a NdefMessage to beam
  - OR setNdefPushMessageCallback() create NDEF message when close enough to beam

BLUETOOTH

Bluetooth

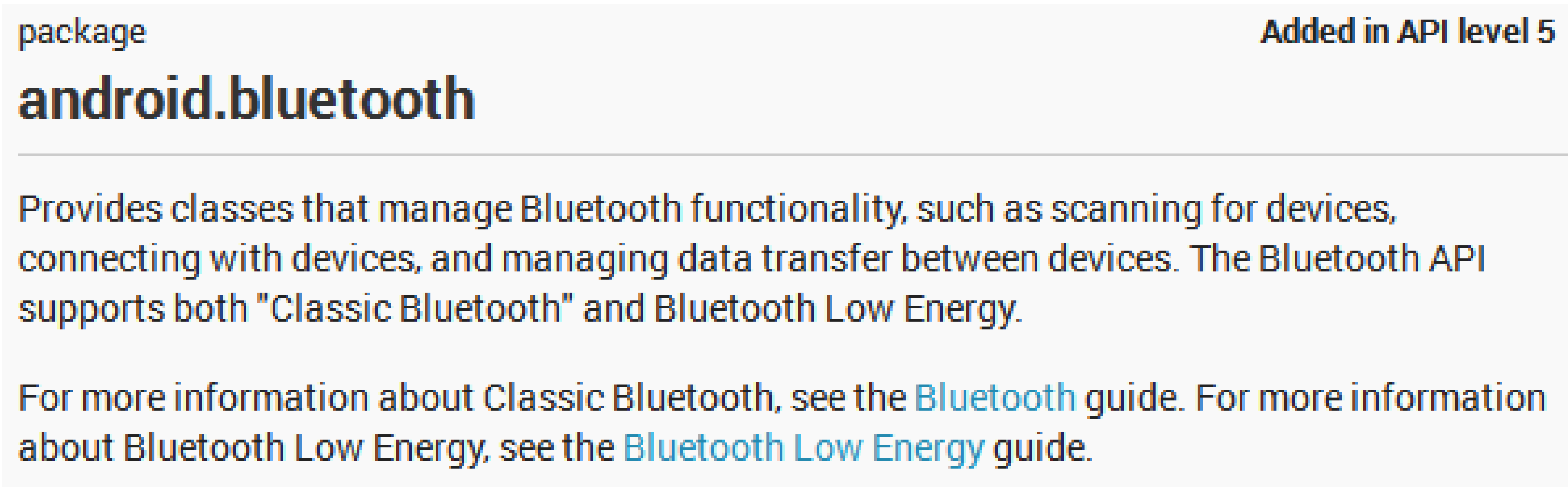
- wireless technology for exchanging data between devices over short distances
- three different classes of Bluetooth devices with ranges of 1, 10, and 100 meters
- Android devices provide access to the Bluetooth network stack

Android and Bluetooth

- Android Bluetooth APIs
- Android devices can:

1. Scan for other Bluetooth Devices
2. Check local Bluetooth adapter for paired devices
3. Connect to other Bluetooth devices
4. transfer data to and from connected devices

Bluetooth API



package

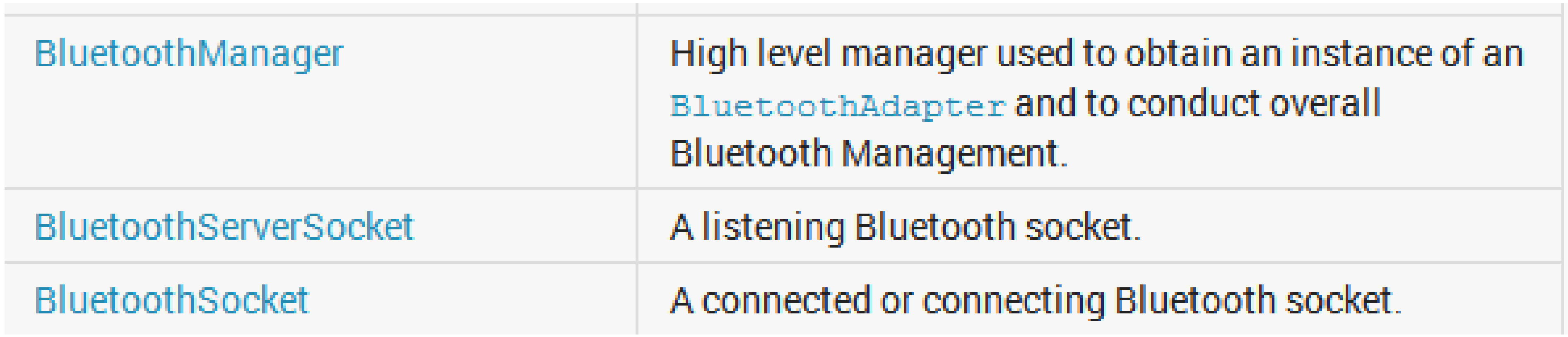
Added in API level 5

android.bluetooth

Provides classes that manage Bluetooth functionality, such as scanning for devices, connecting with devices, and managing data transfer between devices. The Bluetooth API supports both "Classic Bluetooth" and Bluetooth Low Energy.

For more information about Classic Bluetooth, see the Bluetooth guide. For more information about Bluetooth Low Energy, see the Bluetooth Low Energy guide.

A few classes:



BluetoothManager	High level manager used to obtain an instance of an `BluetoothAdapter` and to conduct overall Bluetooth Management.
BluetoothServerSocket	A listening Bluetooth socket.
BluetoothSocket	A connected or connecting Bluetooth socket.

Android - Bluetooth Basics

- Four major steps:

1. Setting up Bluetooth
   - permissions, BluetoothAdapter (like a manager), enable Bluetooth
2. finding paired or available devices
3. connecting devices
4. transferring data

http://developer.android.com/guide/topics/connectivity/bluetooth.html

COPY AND PASTE

Copy and Paste

- clipboard based framework
- simple and complex data types can be copied and pasted
  - text strings, complex data structures, text and binary stream data
- Simple text stored on clipboard
- complex data handled via content providers

Copy and Paste

- to copy and paste:
- data placed in clip object, clip object placed on system-wide clipboard
- clip object can be:
  - text, a simple String
  - URI for copying complex data from a content provider
  - Intents to copy application shortcuts
- only one clip on clipboard at a time

Copy and Paste

- An app can support some or all of the data types
- Examine data on clipboard and decide if user should have option to paste it
  - may not make sense to allow pasting of URI / content provider data or Intents

WI-FI DIRECT

Wi-Fi Direct

- Added in ICS, Android 4.0 API level 14
- allows devices with appropriate hardware to connect directly via Wi-Fi with no intermediate access point
- discover and connect to other devices
- much larger range than Bluetooth
- Useful for applications that share data among users
  - multi player game, photo sharing

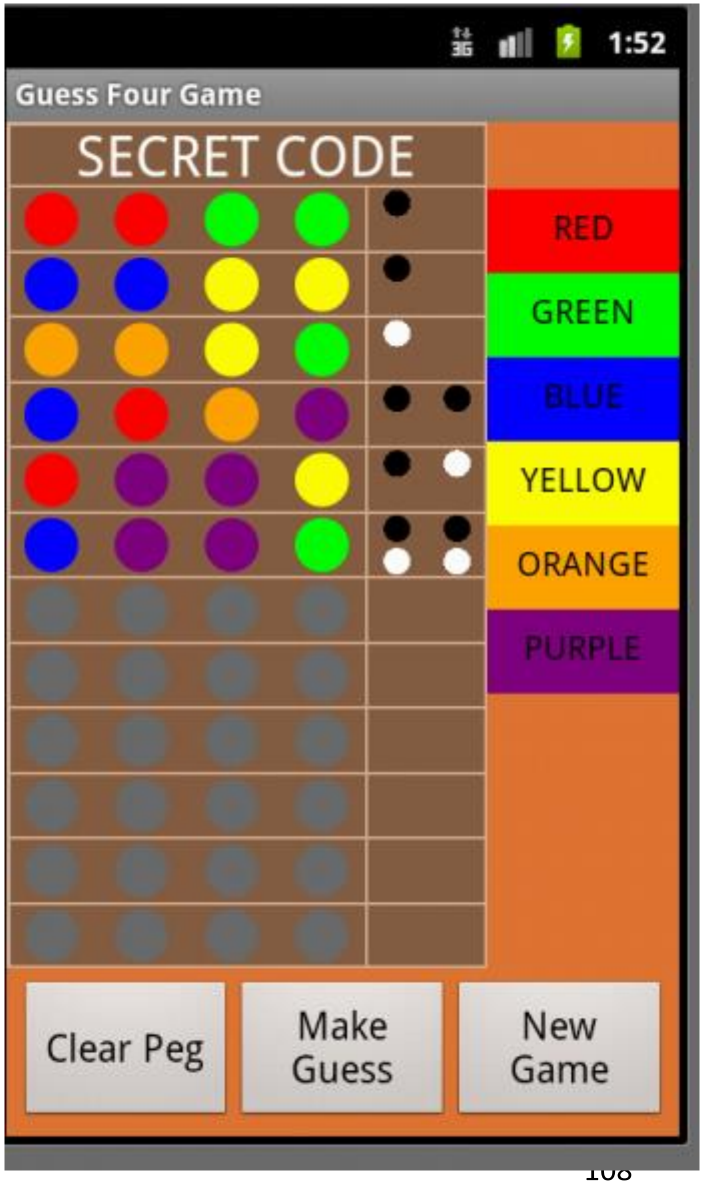
Wi-Fi Direct

- WifiP2pManager class provides methods to discover, request, and connect to peers
- Various Listeners that provide information on success or failure of method calls from WifiP2pManager
- Intents notify application of events detected by Wi-Fi direct framework such as newly discovered peer
  - implement broadcast receiver for intents from Android system about Wifi Direct

3D GRAPHICS

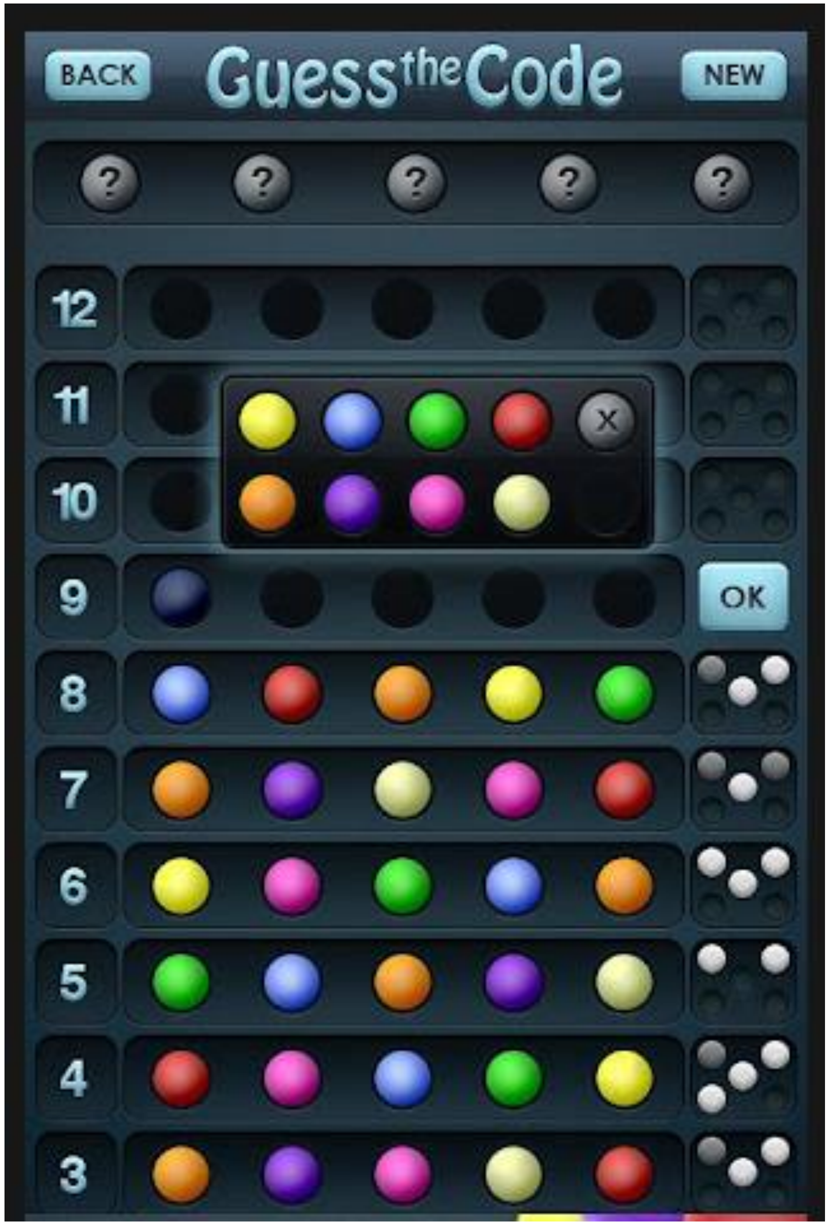
2D Graphics

- android.graphics library for 2D graphics (not Java AWT and Swing)
- classes such as Canvas, Drawable, Bitmap, and others to create 2D graphics
- Various attempts to make two d graphics appear more "lifelike" and 3 dimensional

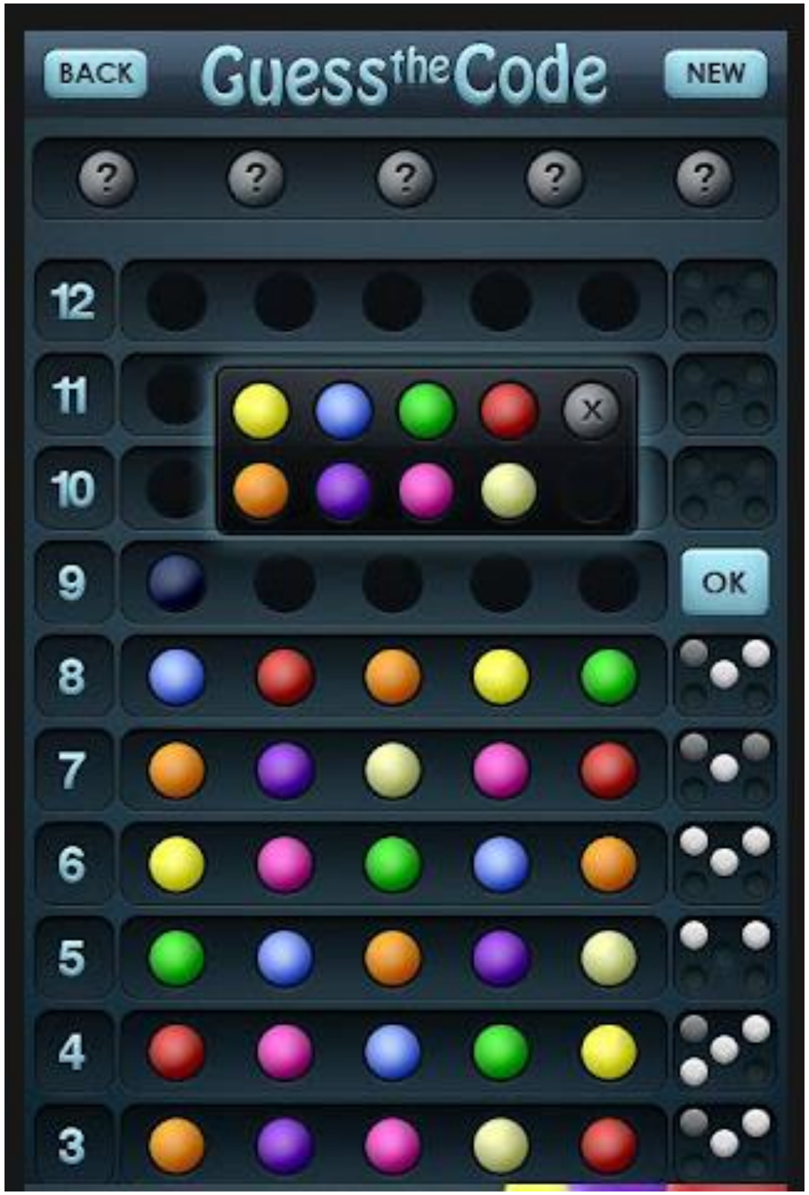


Gradients

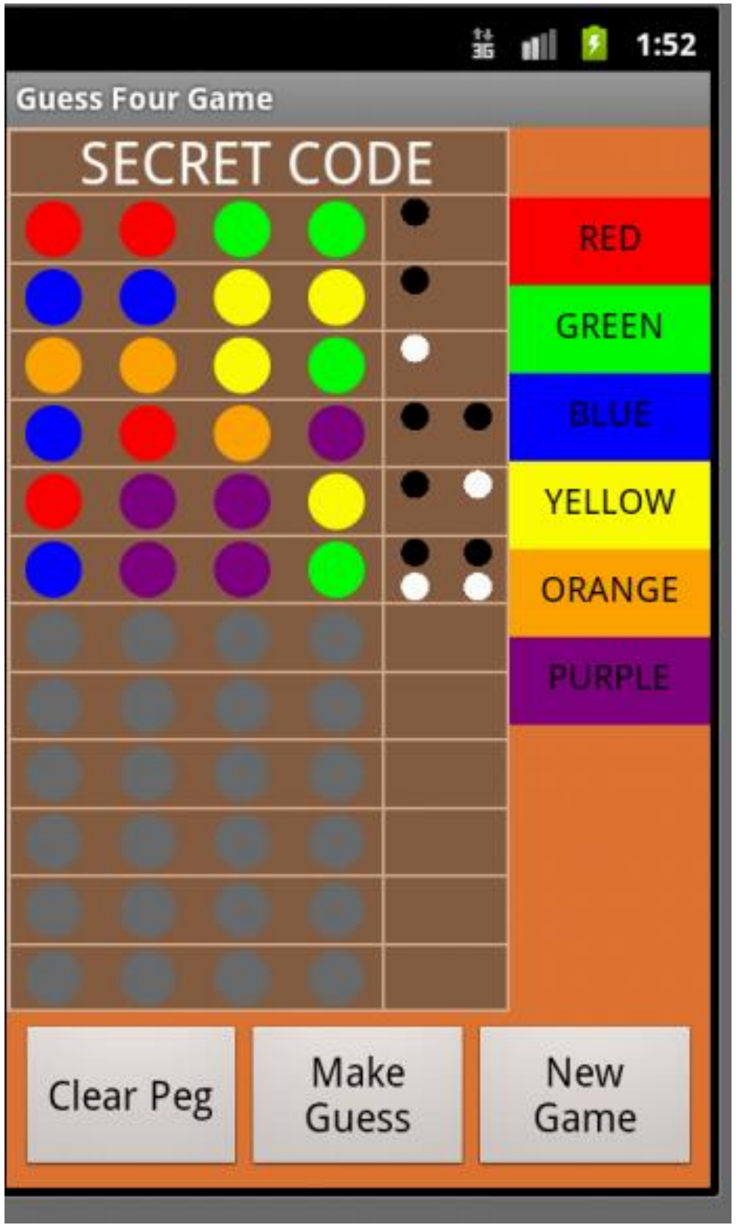
- Gradient Paints can add depth to 2d primitives
- Notice the gradient paint on the pegs and shading on numbers



2D Graphics



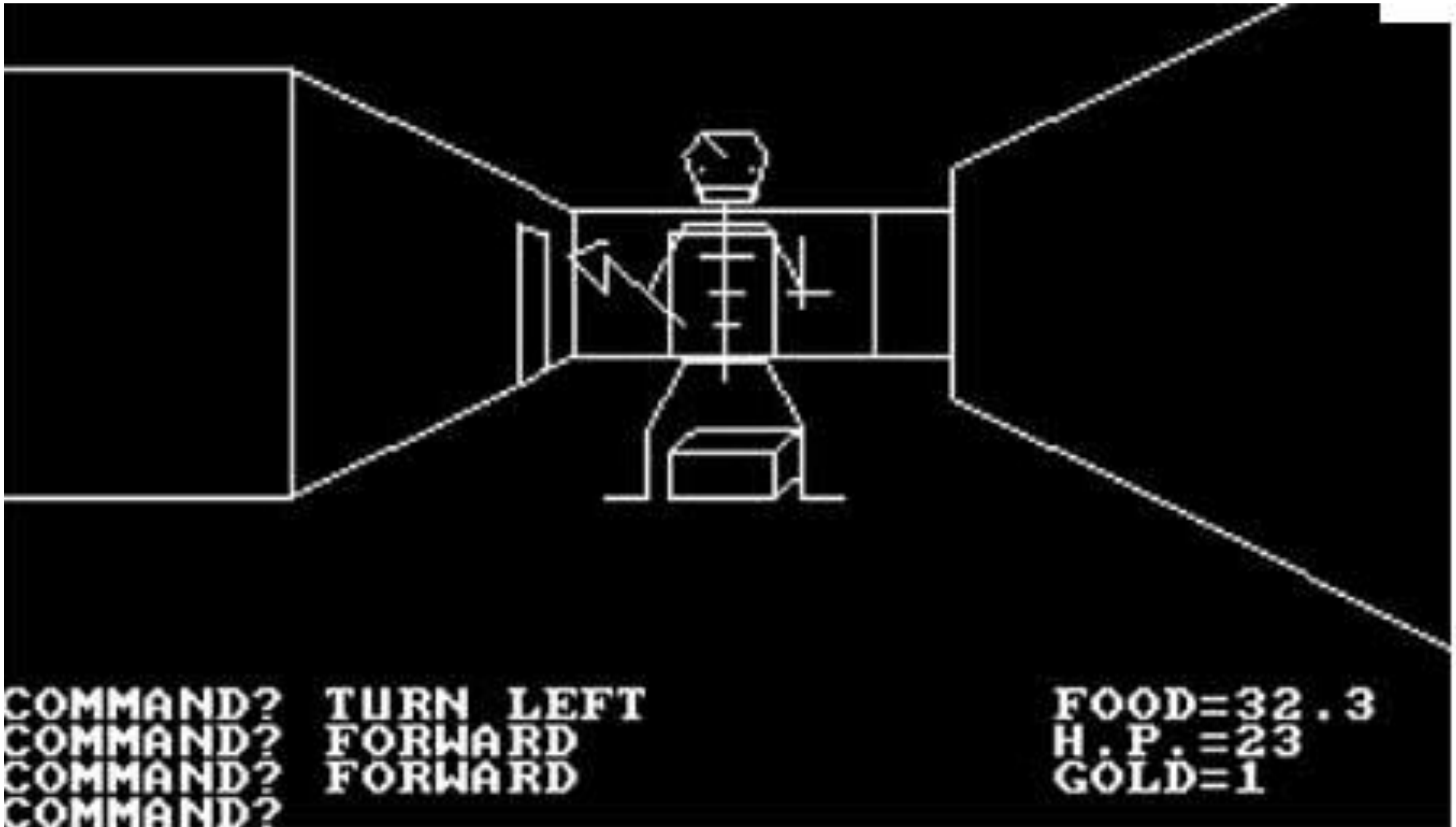
BACK
Guess the Code
NEW
OK



1:52
Guess Four Game
SECRET CODE
RED
GREEN
BLUE
YELLOW
ORANGE
PURPLE
Clear Peg
Make Guess
New Game

Wireframe Graphics

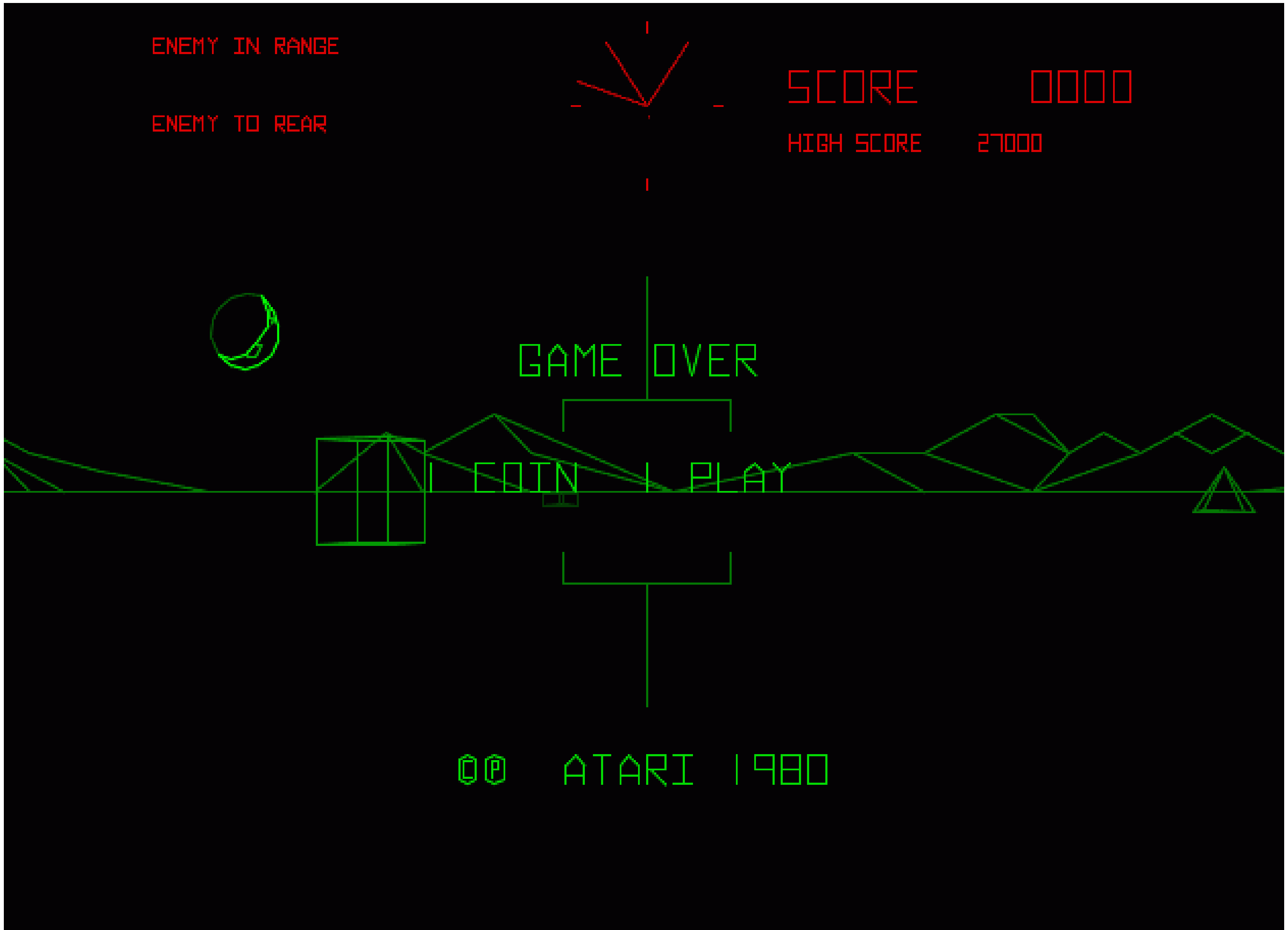
- 1979, Richard Garriott, Akalabeth
- Ultima series, Ultima Online



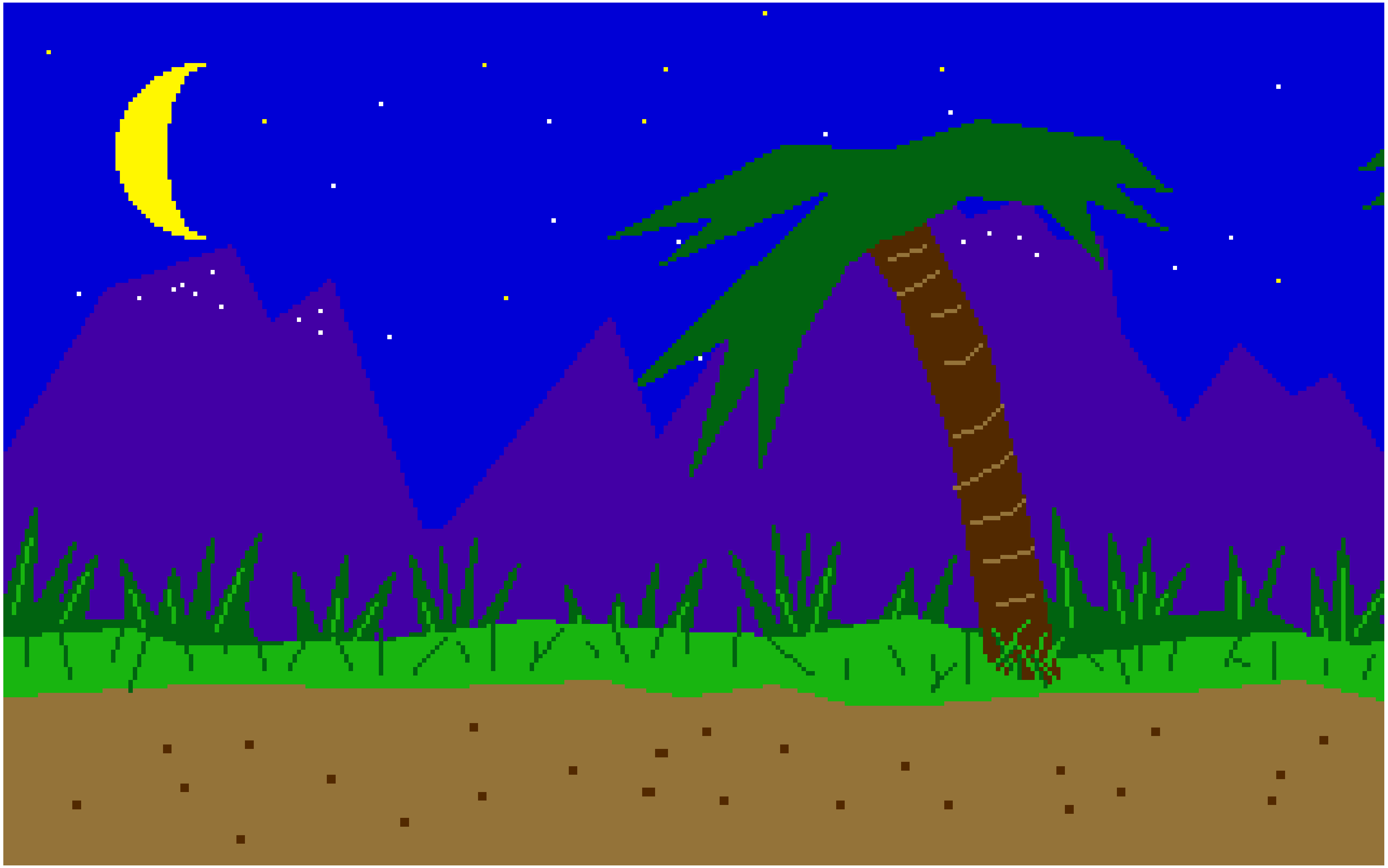
```
COMMAND? TURN LEFT                FOOD=32.3
COMMAND? FORWARD                  H.P.=23
COMMAND? FORWARD                  GOLD=1
COMMAND?
```

Wireframe Vector Graphics

- BattleZone - 1980

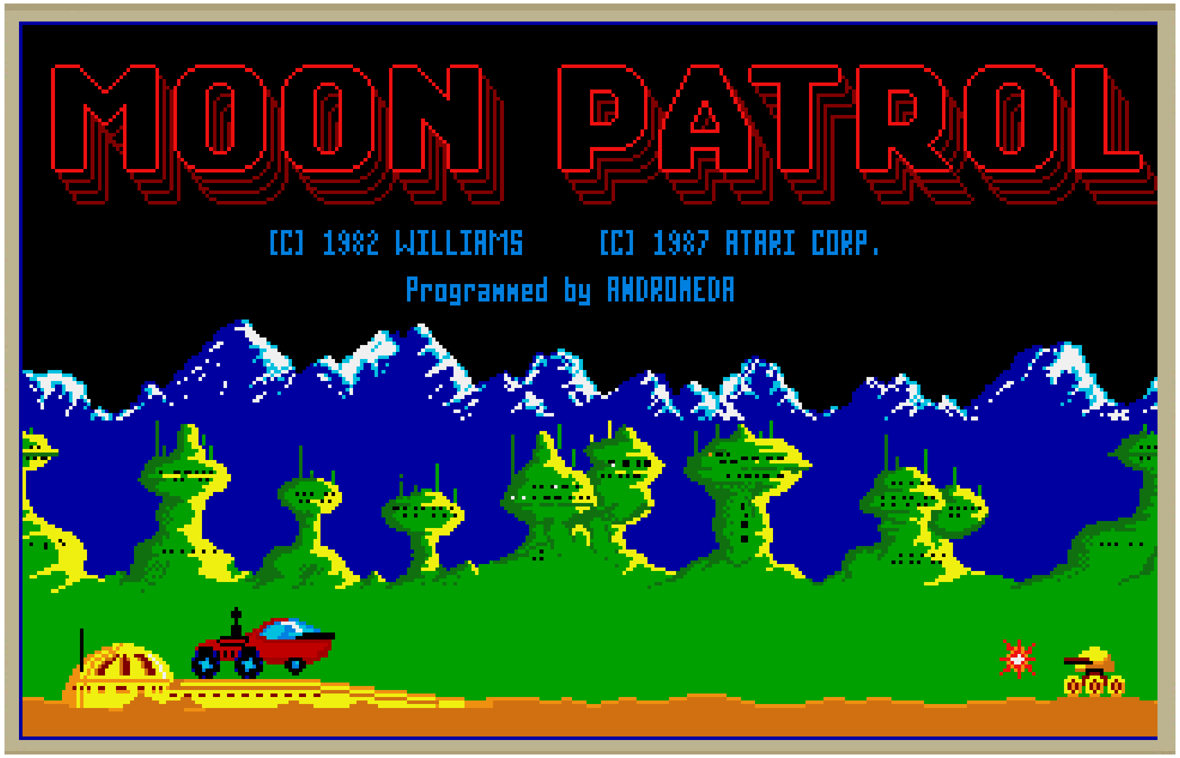


Parallax Scrolling Example



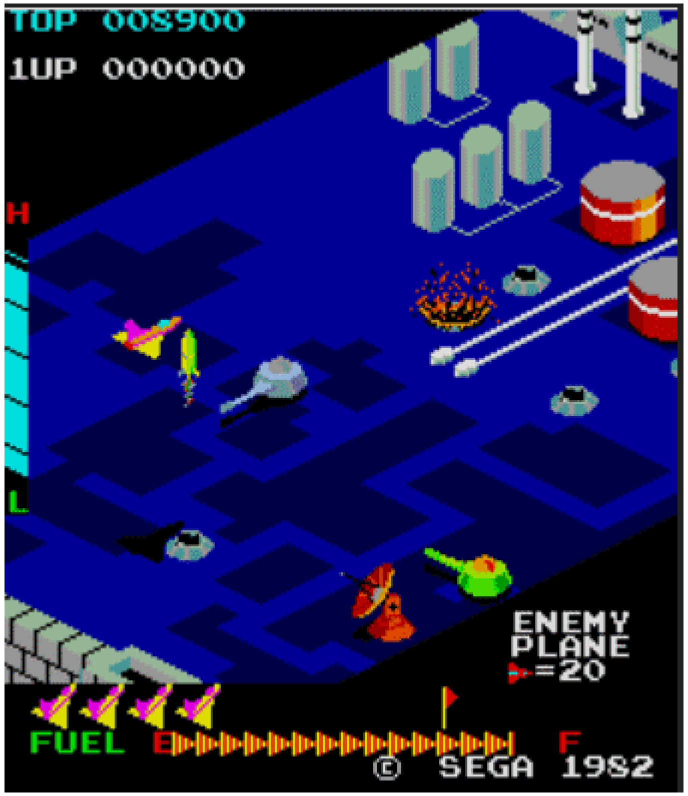
Parallax Scrolling

- multiple backgrounds
- backgrounds closer to view move at a faster speed than backgrounds farther away

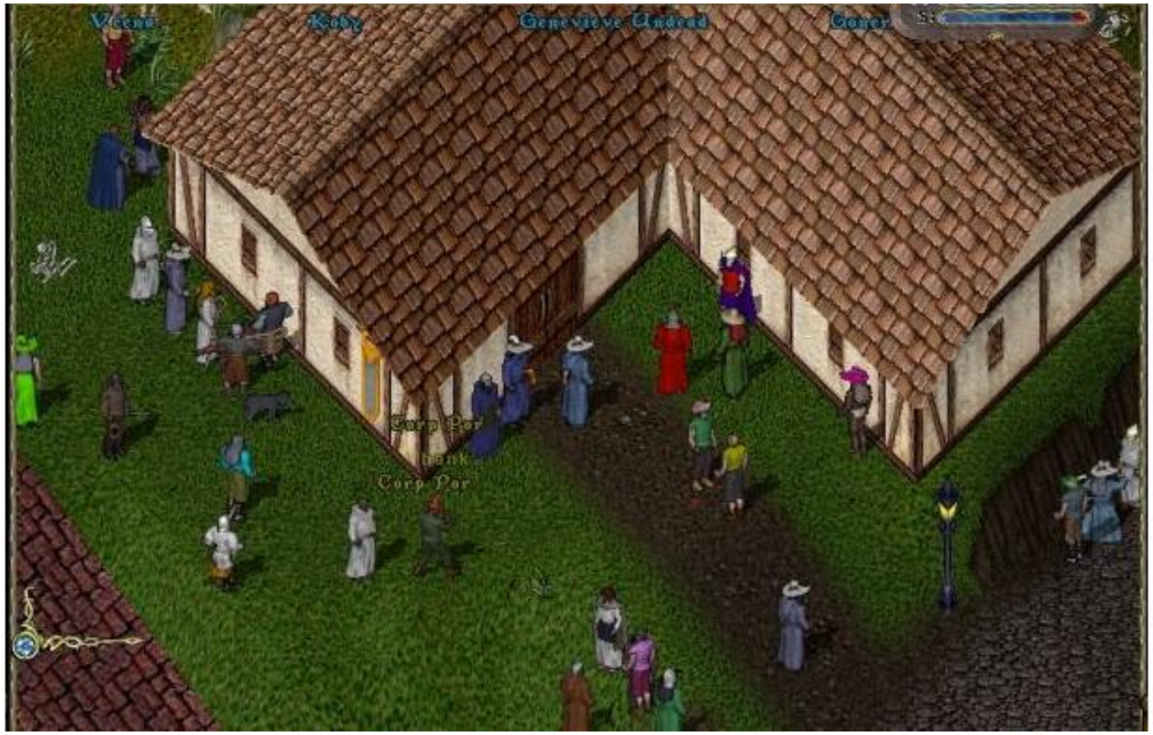


2.5D

- Isometric Graphics
- "rotate" object to reveal details on the side



Zaxxon



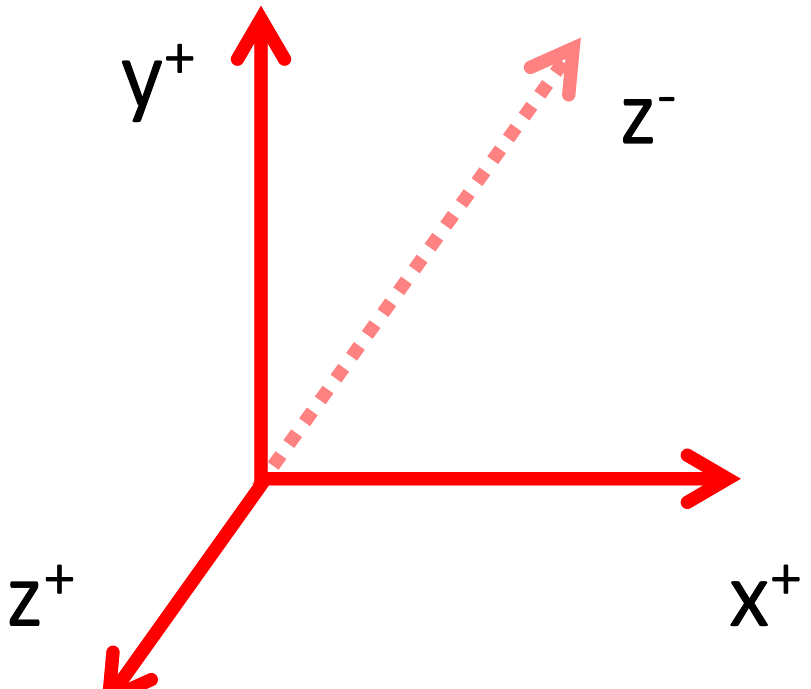
Ultima Online

3D Graphics

- Create 3D model
  - a small scene or a large world
- Model rendered into a 2D projection
- model includes
  - objects (boxes, cones, cylinders, sphere, user defined models)
  - lighting
  - cameras
  - textures
  - dynamic behaviors

3D Coordinate System

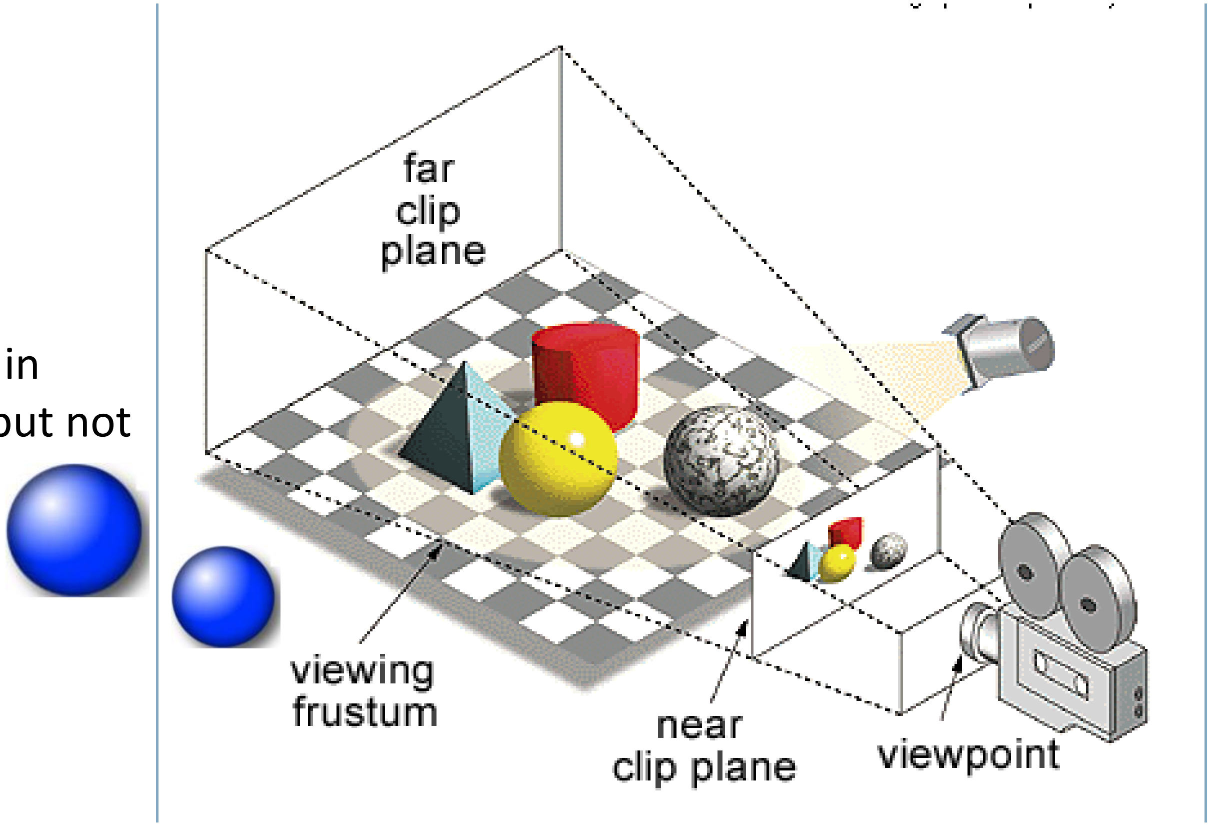
- x and y as expected (positive y is up, not down as in 2d graphics
- z axis - positive z is out of screen, negative z is into screen



Visual Portion

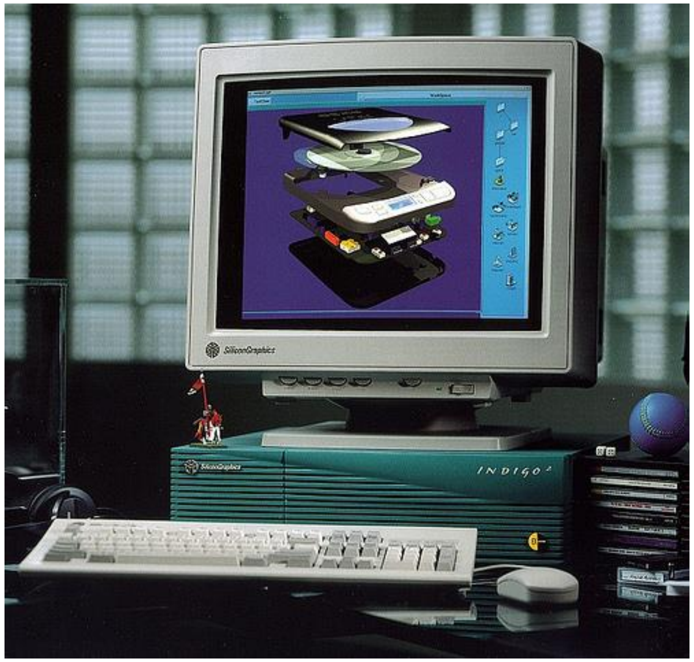
- Portion of 3D Scene that is rendered is contained in a *frustum (pro: frəstəm)*
  - a pyramid or cone with its top cut off

objects in scene, but not visible



OpenGL

- Developed by Silicon Graphics Inc.
  - developer of high end graphics systems and machines in 80s and 90s
- Integrated Raster Imaging System Graphics Library
  - 1992 OpenGL
  - maintained by non profit Khronos Group



OpenGL

- low level, procedural API
  - programmer responsible for defining steps to create and render (show) a scene
- alternatives use a scene graph where programmer describes scene and actions (behaviors) and library manages the details of rendering it
  - Example of Graphics libraries that use Scene Graphs: Java3D, Acrobat 3D, AutoCAD, CorelDRAW, RenderMan (Pixar)

OpenGL ES

- ES = Embedded Systems
- Used in a wide variety of devices, not just Android
  - iPad, iPhone, Blackberry, symbian, Nintendo3DS, Playstation 3, Web GL
- OpenGL version ES 2.0 API supported in Android 2.2 and higher (API levels 8 and higher)
  - prior versions of Android support ES 1.1
- emulator DOES NOT support ES 2.0

Android and OpenGL ES

- two ways of working with GL:
  - through the framework APIandroid.opengl package
  - via the Android Native Development Kit (NDK)
    - companion tool to Android SDK to build portions of apps in native code in C or C++
- Required Android classes for first approach:
  - GLSurfaceView and GLSurfaceView.Renderer

GLSurfaceView

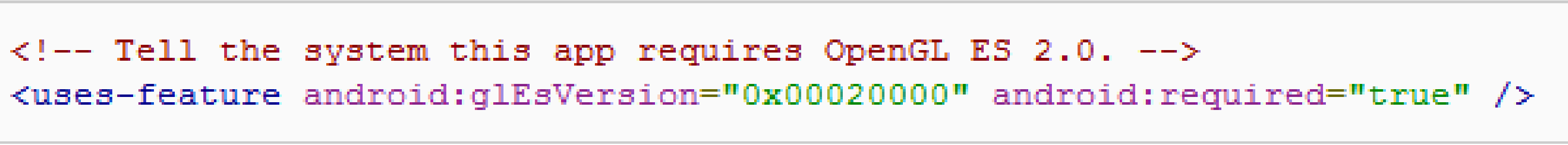
- Similar to SurfaceView
- draw and manipulate objects using Open GL API calls
- to respond to touch screen events subclass GLSurfaceView and implement touch listeners

GLSurfaceView.Renderer

- An interface
- Must implement these methods:
  - onSurfaceCreated for actions that only happen once such as initializing GL graphics objects
  - onDrawFrame() work horse method to create movement and animation
  - onSurfacechanged() called when size of view changes or orientation

Manifest Requirements

- To use OpenGL ES 2.0 (Android 2.0 and later)



```
<!-- Tell the system this app requires OpenGL ES 2.0. -->
<uses-feature android:glEsVersion="0x00020000" android:required="true" />
```

- if app uses texture compression formats must declare which formats application supports
  - <support-gl-texture>

Steps to Use OpenGL

- Create activity using GLSurfaceView and GLSurfaceView.Renderer
- Create and draw graphics objects
- define projection for screen geometry to correct for non square pixels
- define a camera view
- perform actions to animate objects
- make view touch interactive if desired

Sample Program

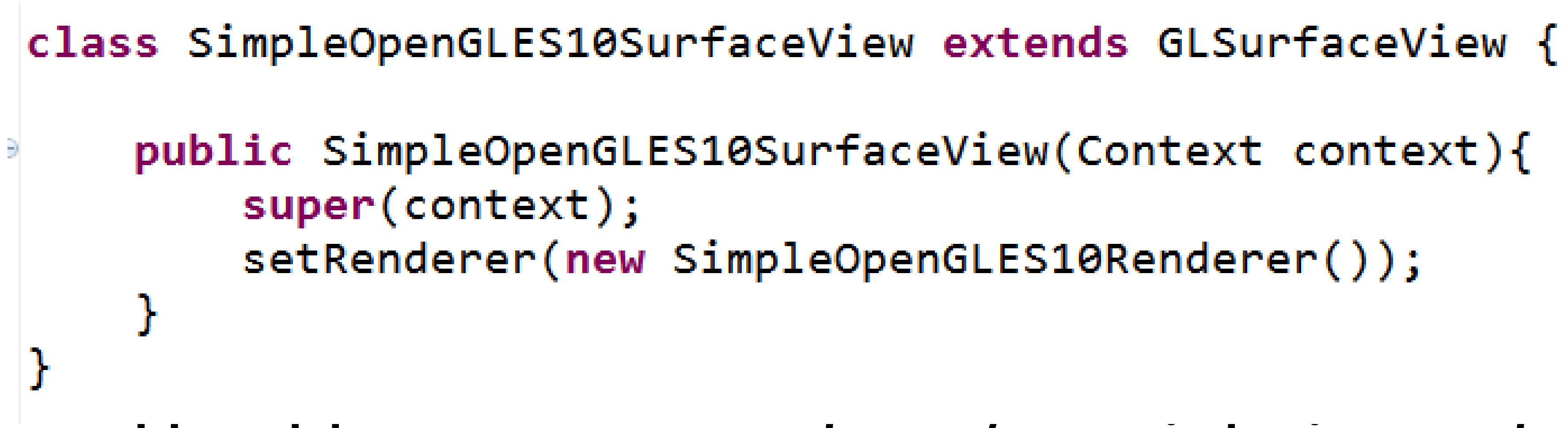
- Demonstrate set up of required elements
- draw and rotate a 3d object (a pyramid)
- Create Simple Activity that has a GLSurfaceView as its content view
- To draw objects must implement GLSurfaceView.Renderer

Activity

```
public class ShowOpenGLSurfaceView  extends Activity {

    private GLSurfaceView mGLView;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mGLView = new SimpleOpenGLES10SurfaceView(this);
        setContentView(mGLView);
    }

    protected void onPause() {
        super.onPause();
        mGLView.onPause();
    }

    protected void onResume() {
        super.onResume();
        mGLView.onResume();
    }
}
```

GLSurfaceView

- Shell of class



```
class SimpleOpenGLES10SurfaceView extends GLSurfaceView {

    public SimpleOpenGLES10SurfaceView(Context context){
        super(context);
        setRenderer(new SimpleOpenGLES10Renderer());
    }
}
```

- Used to manage surface (special piece of memory), manage EGL display (embedded graphics library, renders on thread decoupled from I thread, and more

Skeleton Renderer

```
class SimpleOpenGLES10Renderer implements Renderer {

    public void onDrawFrame(GL10 gl) {
        // Redraw background color
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
    }

    public void onSurfaceChanged(GL10 gl, int width, int height) {
        // specifies the affine transformation of
        // x and y from
        // normalized device coordinates to window coordinates
        gl.glViewport(0, 0, width, height);
    }

    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        // Set the background frame color
        gl.glClearColor(0.9f, 0.6f, 0.3f, 1.0f); // rgba
    }

}
```

OpenGL Documentation

- Android Documentation for GL10 list constants and methods but have no other useful information
- Check the OpenGL ES documentation
- http://www.khronos.org/opengles/sdk/1.1/docs/man/

Low Level Graphics Libraries

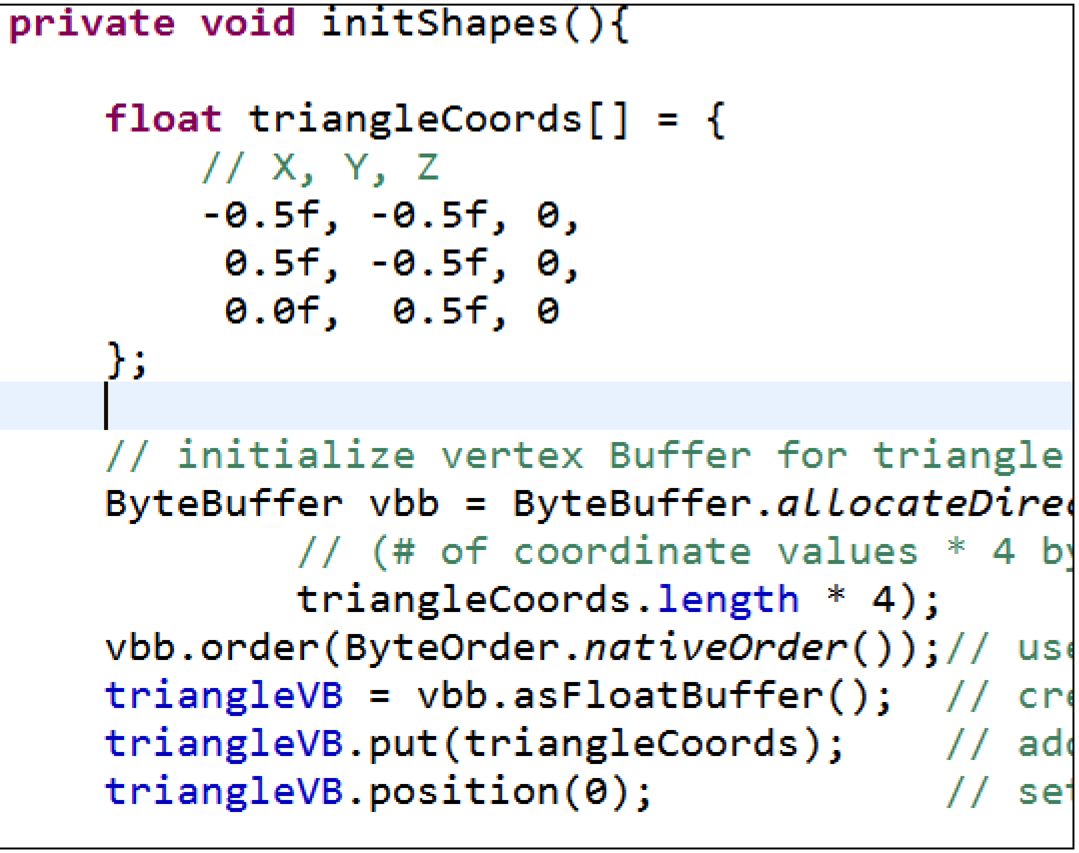
- "What makes the situation worse is that the highest level CS course I've ever taken is cs4, and quotes from the graphics group startup readme like '*these paths are abstracted as being the result of a topological sort on the graph of ordering dependencies for the entries*' make me lose consciousness in my chair and bleed from the nose."

  -mgrimes, Graphics problem report 134

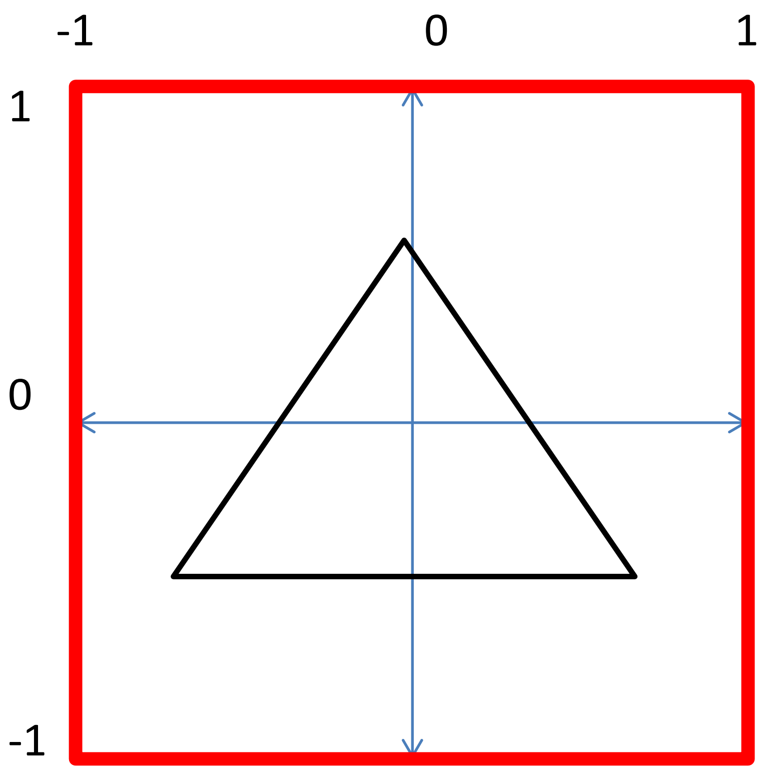
Draw a Shape

- Draw a simple, flat Triangle using OpenGL
- (X,Y,Z) coordinate system
- (0, 0, 0) center of frame
- (1, 1, 0) is top right corner of frame
- (-1, -1, 0) is bottom left corner of frame
- must define vertices of our triangle

Define Triangle



```
private void initShapes(){

    float triangleCoords[] = {
        // X, Y, Z
        -0.5f, -0.5f, 0,
         0.5f, -0.5f, 0,
         0.0f,  0.5f, 0
    };

    // initialize vertex Buffer for triangle
    ByteBuffer vbb = ByteBuffer.allocateDire
            // (# of coordinate values * 4 by
            triangleCoords.length * 4);
    vbb.order(ByteOrder.nativeOrder());// use
    triangleVB = vbb.asFloatBuffer();  // cre
    triangleVB.put(triangleCoords);    // add
    triangleVB.position(0);            // se
```



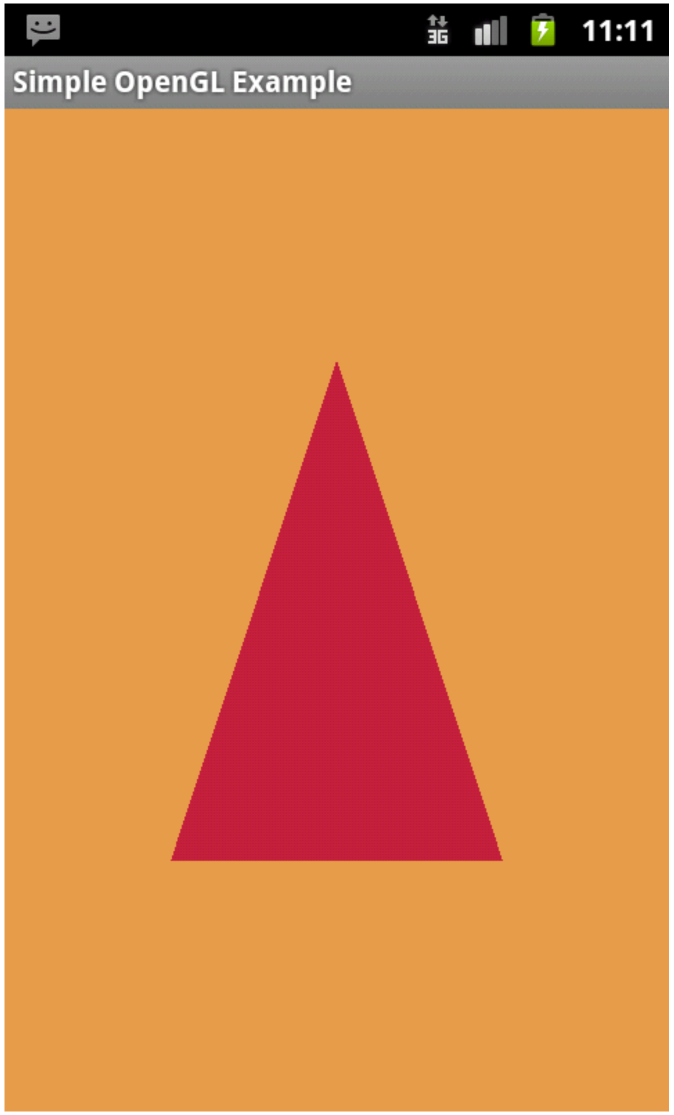
Draw Triangle

- init OpenGL to use vertex arrays
- call drawing API to draw triangle

```
class SimpleOpenGLES10Renderer implements Renderer {

    private FloatBuffer triangleVB;

    public void onDrawFrame(GL10 gl) {
        // Redraw background color
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);

        // Draw the triangle
        gl.glColor4f(0.63671875f, 0.768f, 0.227f, 0.0f);
        // coordinates per vertex, type, stride (offset between vertices)
        gl.glVertexPointer(3, GL10.GL_FLOAT, 0, triangleVB);
        // mode, first, count of vertices
        gl.glDrawArrays(GL10.GL_TRIANGLES, 0, 3);
    }

    public void onSurfaceChanged(GL10 gl, int width, int height) {
        // specifies the affine transformation of
        // x and y from
        // normalized device coordinates to window coordinates
        gl.glViewport(0, 0, width, height);

        initShapes();
    }
```

Result

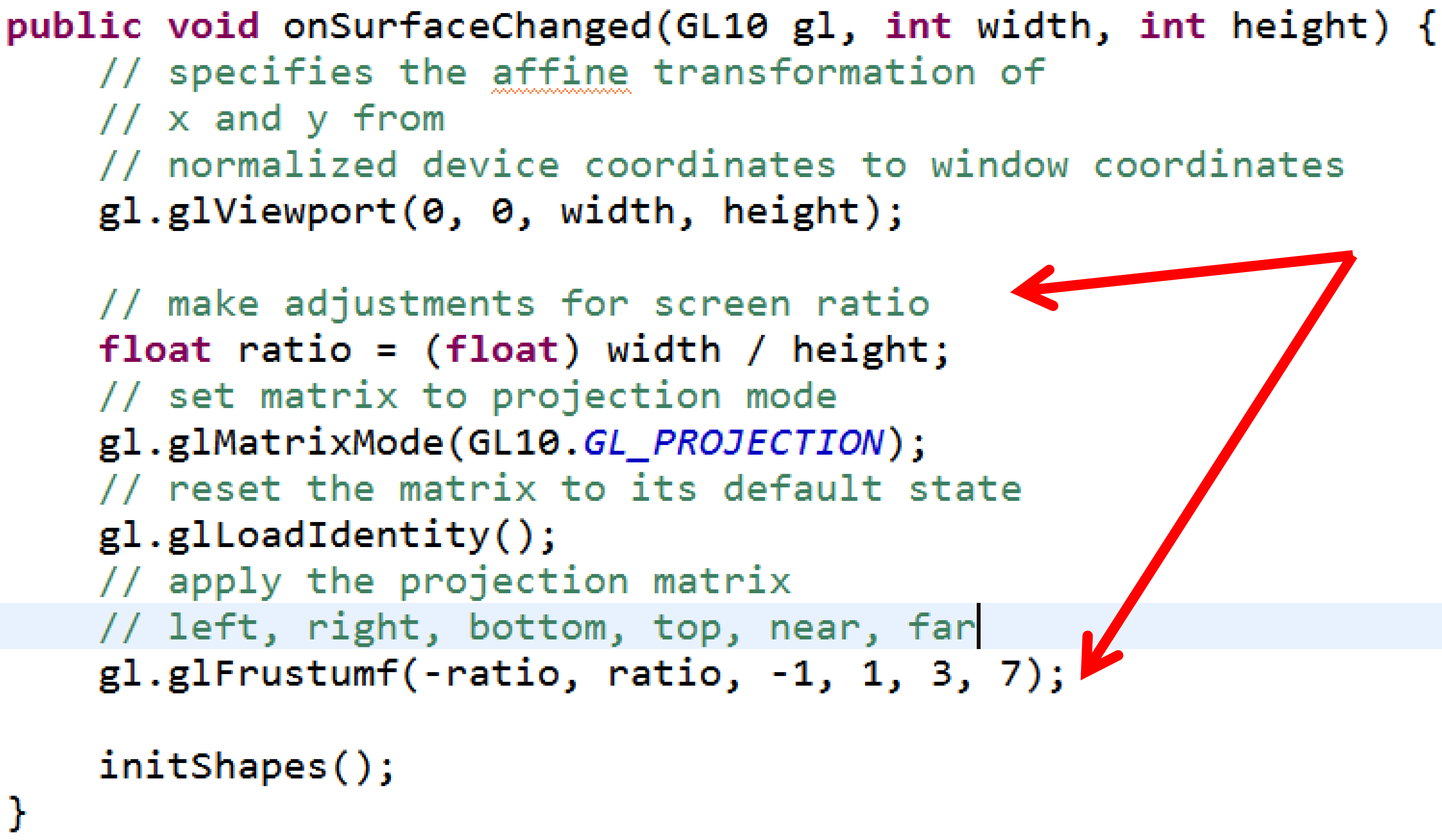
- oooo, ahhhh
- Graphics coordinate system assumes a square but mapped to a rectangular frame



Correcting Projection

- Apply an OpenGL projection view and camera (eye point) to transform coordinates of the triangle
  - "correct" the position onSurfaceChanged and onDrawframe()

onSurfaceChanged

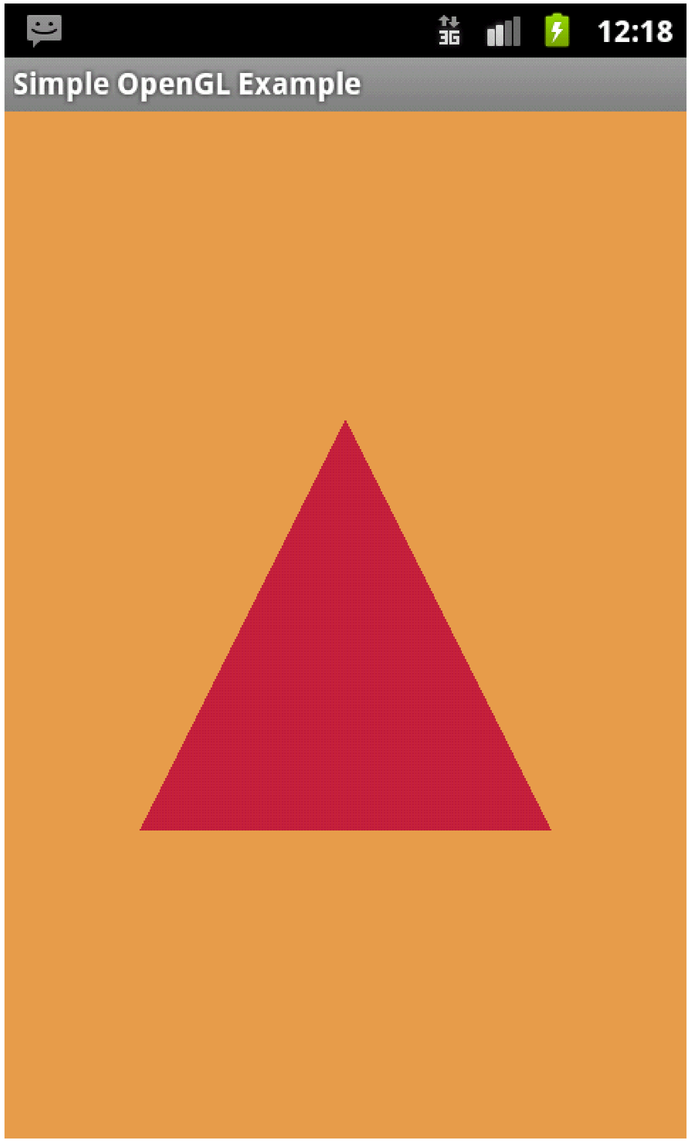


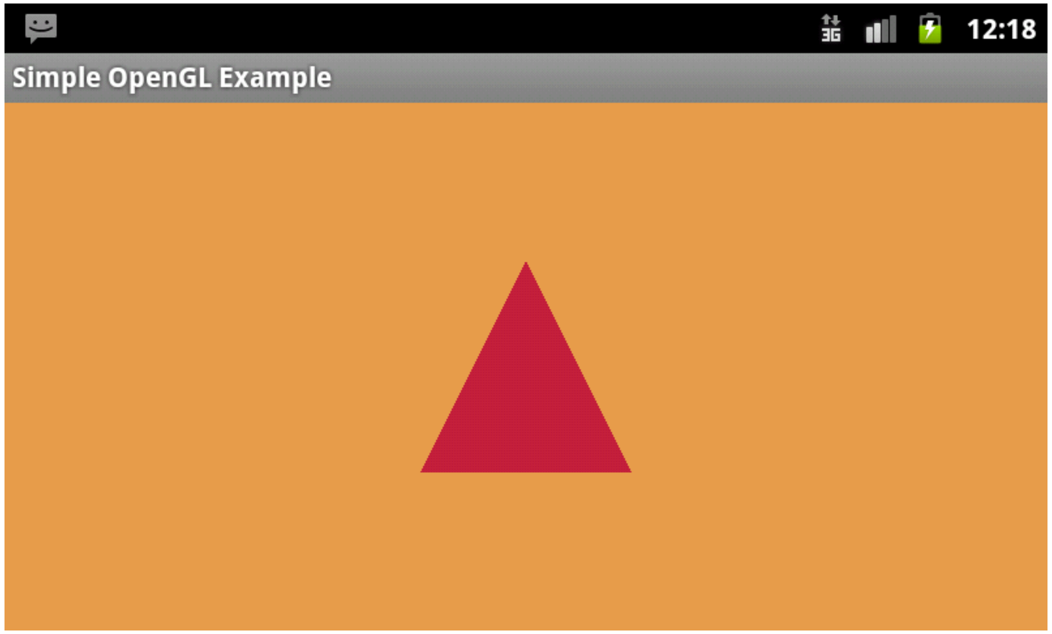
```java
public void onSurfaceChanged(GL10 gl, int width, int height) {
    // specifies the affine transformation of
    // x and y from
    // normalized device coordinates to window coordinates
    gl.glViewport(0, 0, width, height);

    // make adjustments for screen ratio
    float ratio = (float) width / height;
    // set matrix to projection mode
    gl.glMatrixMode(GL10.GL_PROJECTION);
    // reset the matrix to its default state
    gl.glLoadIdentity();
    // apply the projection matrix
    // left, right, bottom, top, near, far
    gl.glFrustumf(-ratio, ratio, -1, 1, 3, 7);

    initShapes();
}
```

onDrawFrame

```java
public void onDrawFrame(GL10 gl) {
    // Redraw background color
    gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);


     // Set GL_MODELVIEW transformation mode
    gl.glMatrixMode(GL10.GL_MODELVIEW);
    gl.glLoadIdentity();   // reset the matrix to its default state

    // When using GL_MODELVIEW, you must set the view point
    // GLU = Graphics Library Utilities

    GLU.gluLookAt(gl, 0, 0, -5, // gl10, eyeX, eyeY, eyeZ
            0f, 0f, 0f, // centerX, centerY, centeZ
            0f, 1.0f, 0.0f);  // upX, upY, upZ

    // Draw the triangle
    gl.glColor4f(0.77f, 0.12f, 0.23f, 1);
    // coordinates per vertex, type, stride (offset between vertices)
    gl.glVertexPointer(3, GL10.GL_FLOAT, 0, triangleVB);
    // mode, first, count of vertices
    gl.glDrawArrays(GL10.GL_TRIANGLES, 0, 3);
}
```

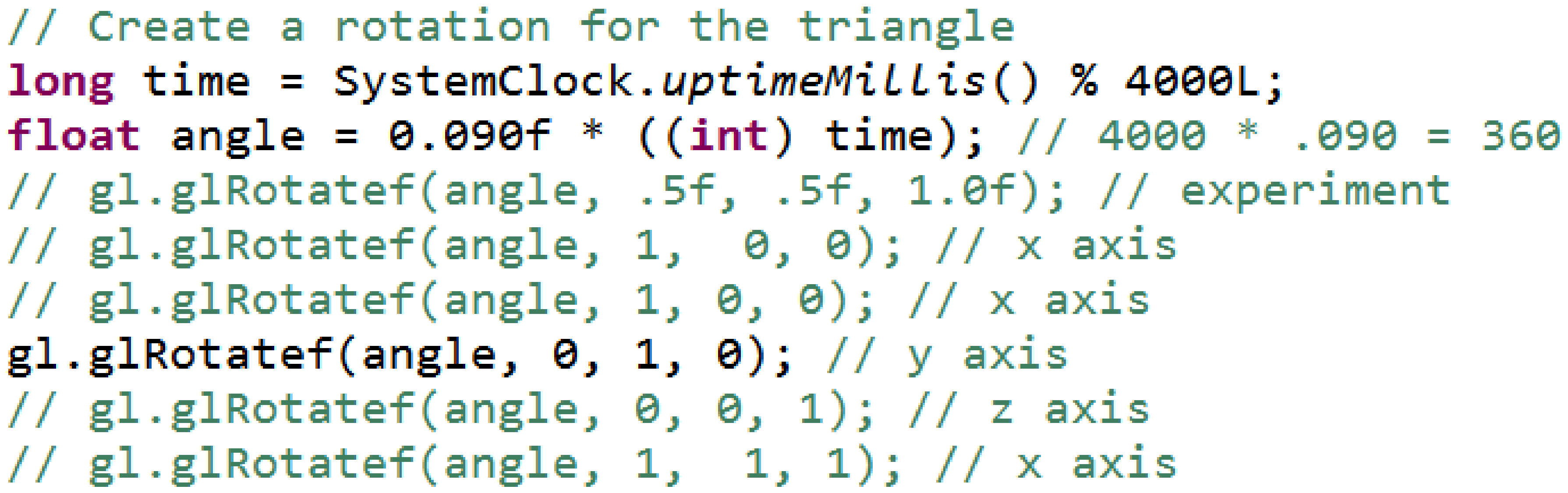
Result of Correcting Projection





Adding Motion

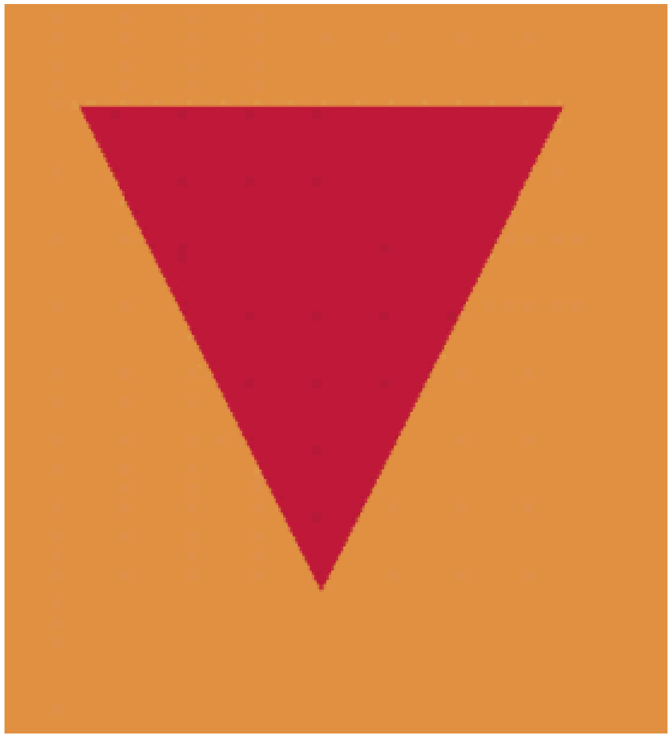
- in onDrawFrame
- define vector of rotation

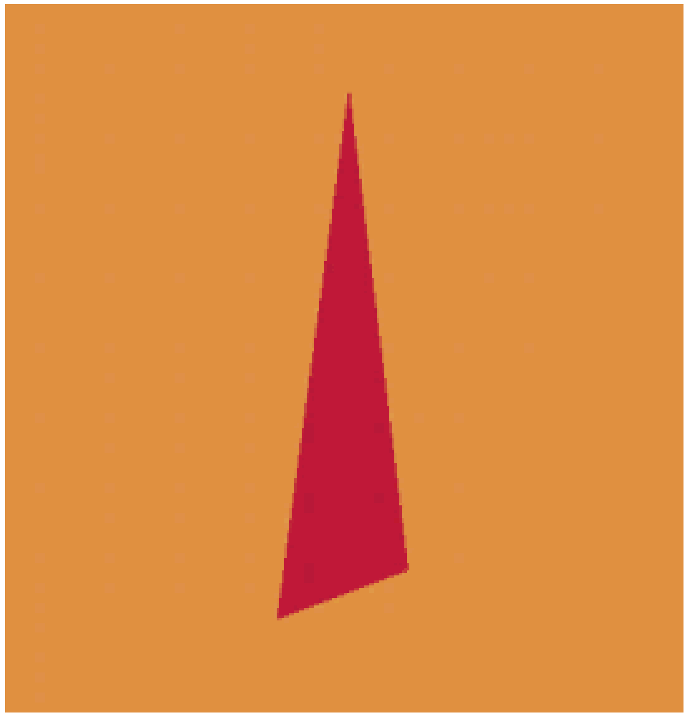


```
// Create a rotation for the triangle
long time = SystemClock.uptimeMillis() % 4000L;
float angle = 0.090f * ((int) time); // 4000 * .090 = 360
// gl.glRotatef(angle, .5f, .5f, 1.0f); // experiment
// gl.glRotatef(angle, 1,  0, 0); // x axis
// gl.glRotatef(angle, 1, 0, 0); // x axis
gl.glRotatef(angle, 0, 1, 0); // y axis
// gl.glRotatef(angle, 0, 0, 1); // z axis
// gl.glRotatef(angle, 1,  1, 1); // x axis
```

Results

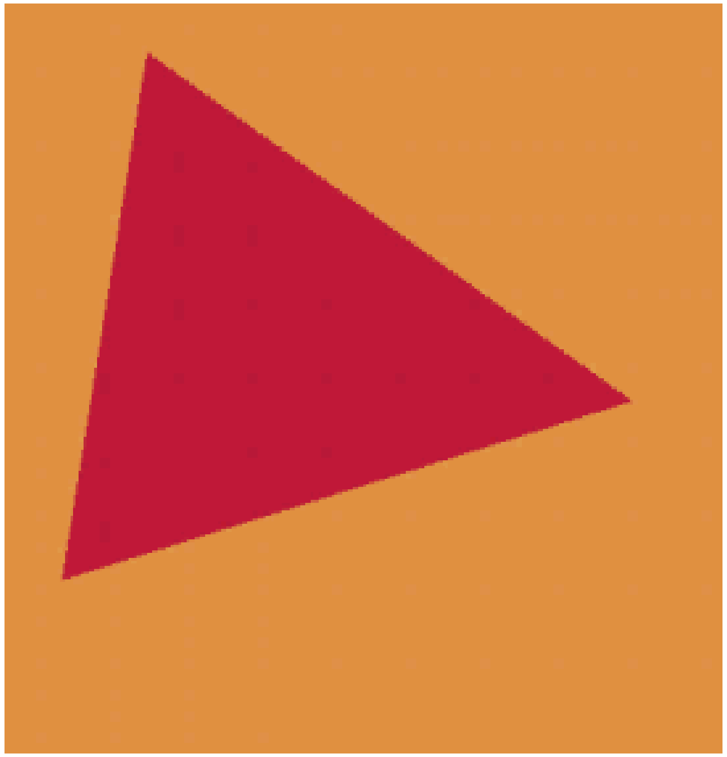
X Axis (angle, 1, 0, 0)   Y Axis (angle, 0, 1, 0)

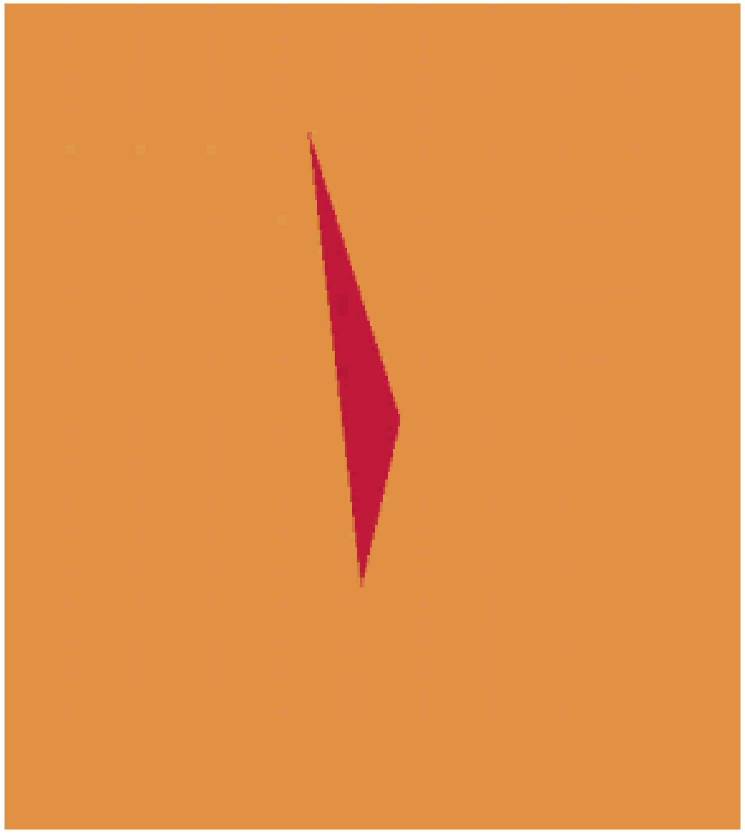




Results

Z Axis (angle, 0, 0, 1)   Y Axis (angle, -1, 1, -1)



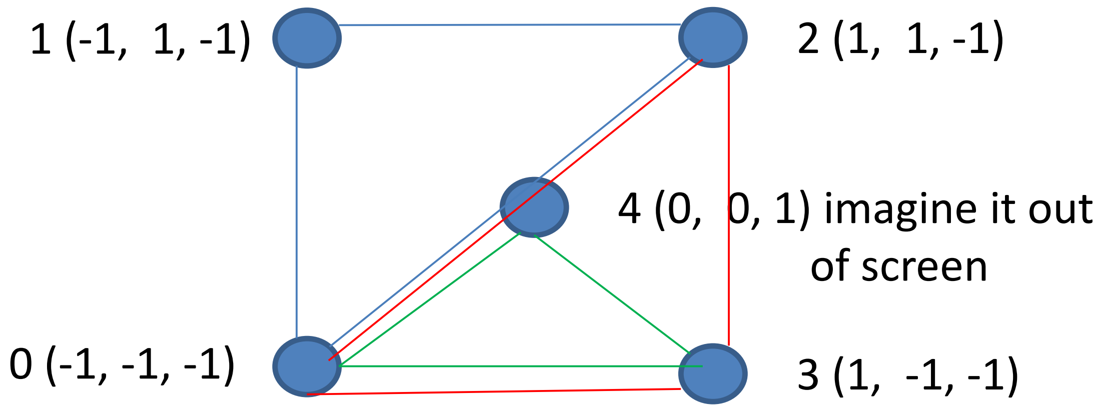


Another Example

- Draw a pyramid that bounces around the screen
- Same basic steps as previous apps
- Activity with GLSurfaceView
- Implementation of GLSurfaceView.Renderer
- Pyramid class that defines the geometry and appearance of 3d pyramid object

Constructing Pyramid

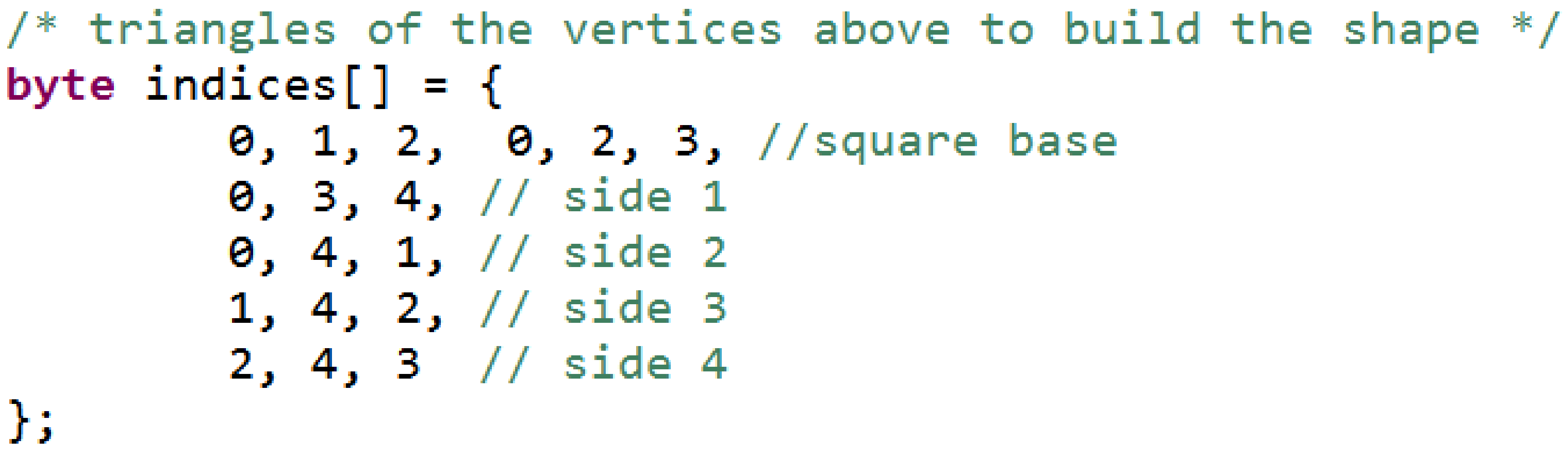
- specify vertices for 6 triangles
- 4 sides, 2 triangles for the base



```
int one = 0x10000;
/* square base and point top to make a pyramid */
int vertices[] = {
        -one, -one, -one,
        -one,  one, -one,
        one,  one,  -one,
        one,  -one,  -one,
        0, 0, one
};
```

Constructing Pyramid

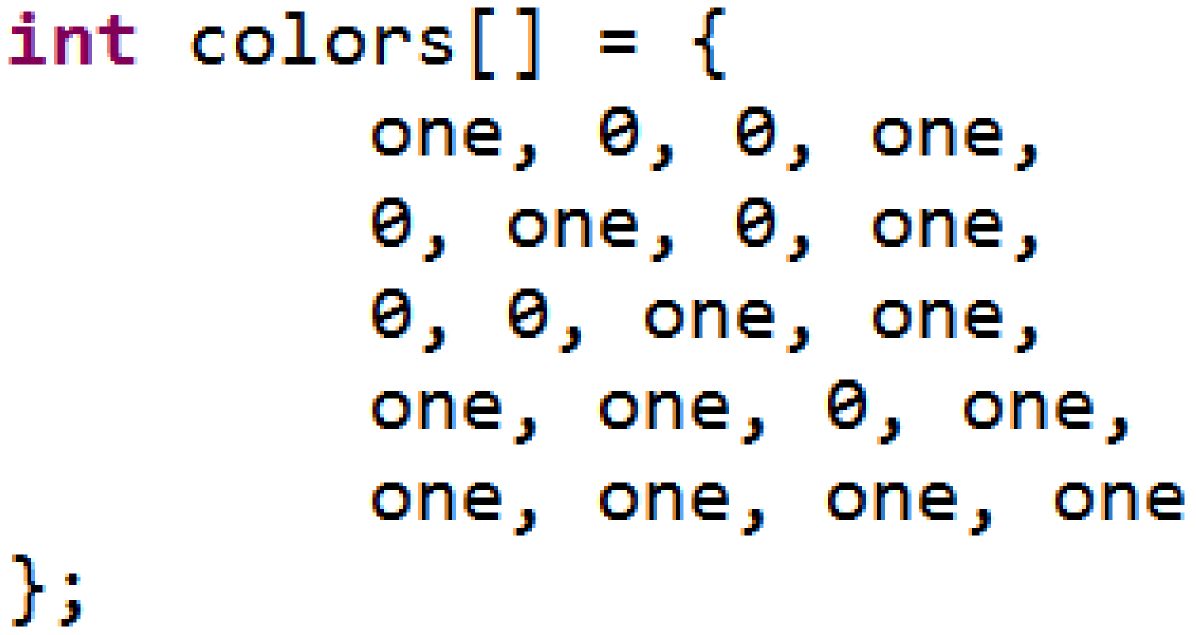
- Indices refers to set or coordinate (x, y, z)



```
/* triangles of the vertices above to build the shape */
byte indices[] = {
        0, 1, 2,  0, 2, 3, //square base
        0, 3, 4, // side 1
        0, 4, 1, // side 2
        1, 4, 2, // side 3
        2, 4, 3  // side 4
};
```

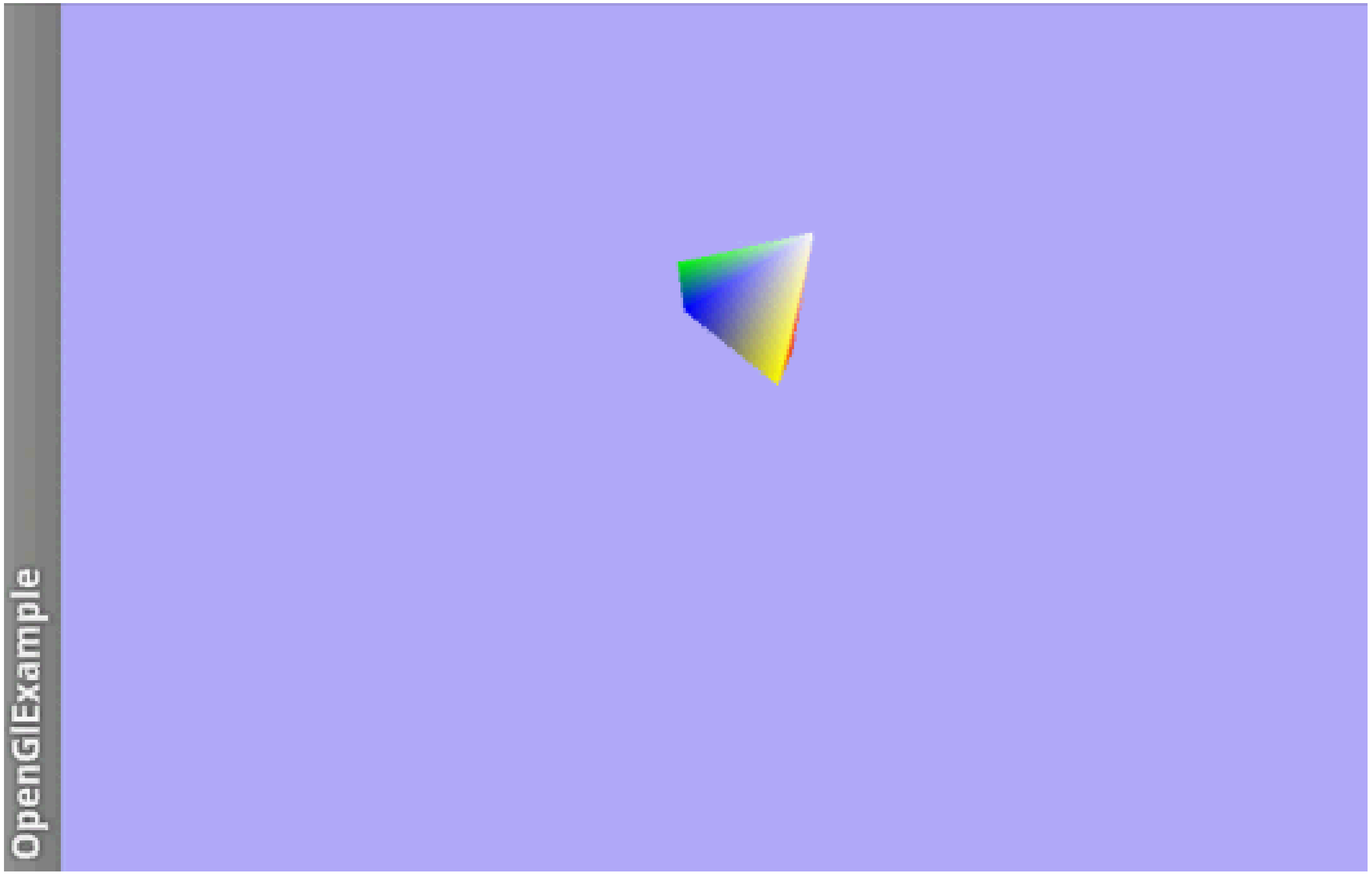
Coloring Pyramid

- Define colors for each of the 5 vertices
- Colors blend from one vertex to another
- recall, rgba



```
int colors[] = {
        one, 0, 0, one,
        0, one, 0, one,
        0, 0, one, one,
        one, one, 0, one,
        one, one, one, one
};
```

Result



OpenGL Options

- Renderscript
  - high performance, but low level
  - scripts written in C
- OpenGLUT, OpenGL Utility Toolkit
  - not officially part of Android, Android GLUT Wrapper
  - include more geometric primitives

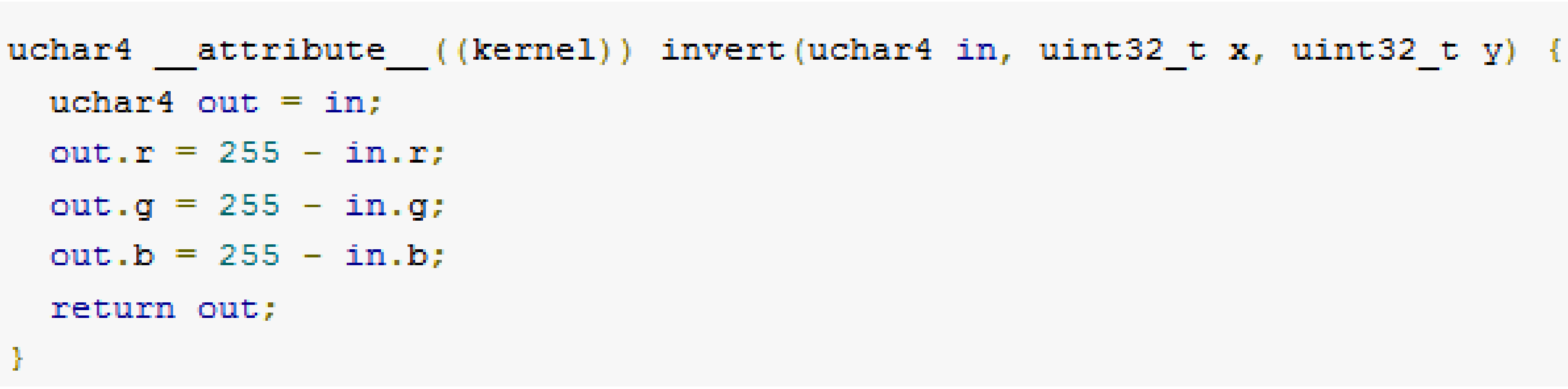
RENDERSCRIPT

RenderScript

- Component of Android OS
- Purpose: run computationally intensive tasks at high performance
- data-parallel computation
  - RandomArt!
- Use RenderScript to parallelize work across all processors on a device
  - multi core CPUs

RenderScript

- Write a "RenderScript kernel" in "C99 like" language
  - ISO/IEC 9899:1999 version of C
- scripts contains sub kernels, functions, and variables



```
uchar4 __attribute__((kernel)) invert(uchar4 in, uint32_t x, uint32_t y) {
  uchar4 out = in;
  out.r = 255 - in.r;
  out.g = 255 - in.g;
  out.b = 255 - in.b;
  return out;
}
```

RenderScript

- Plus a Java API to interact with the scripts and control execution
  - conduit between your application code and the RenderScript