Action Bars & Menus

Content partially based on material by Victor Matos
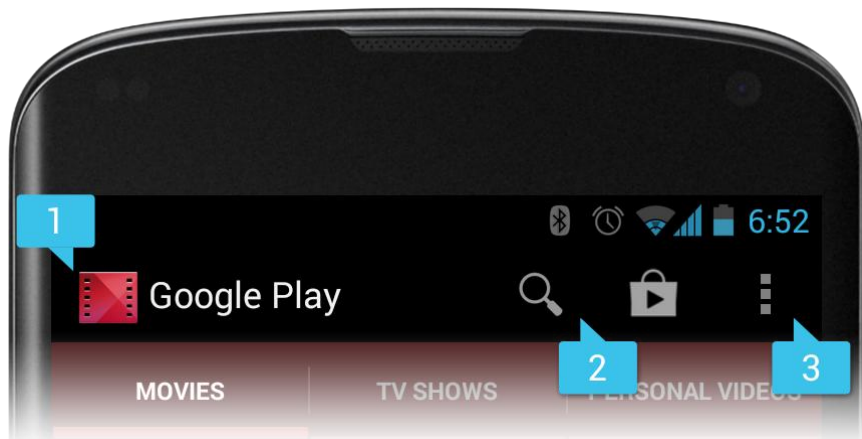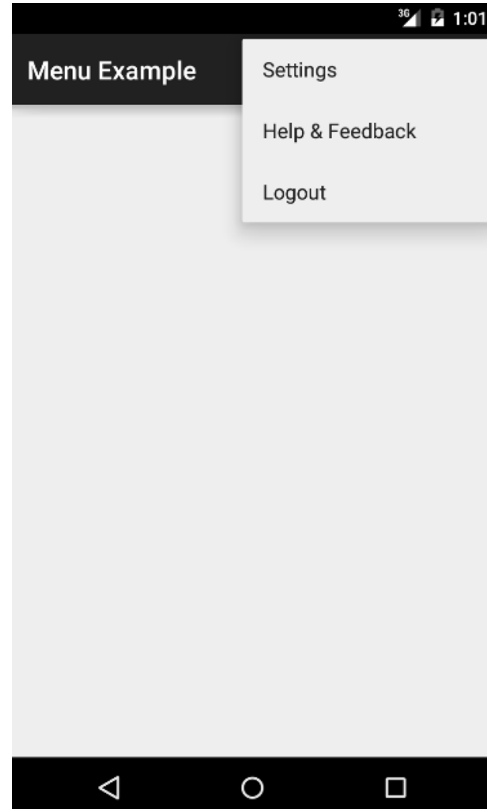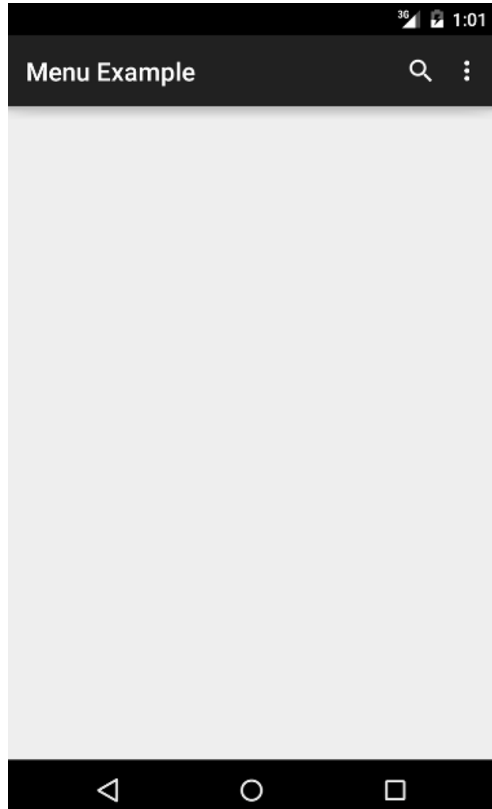
The Action Bar

Figure 1. An action bar that includes the [1] app icon, [2] two action items, and [3] action overflow.

- Provides a dedicated space for giving your app an identity and indicating the user's location in the app.
- Makes important actions prominent and accessible in a predictable way (such as Search).
- Supports consistent navigation and view switching within apps (with tabs or drop-down lists).

Simple ActionBar Example

MainActivity.java

```java
public class MenuExampleActivity extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            Log.d("COEN268", "Settings clicked");
            // TODO: Launch Settings Panel
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

menu_main.xml

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
tools:context=".MainActivity">
    <item android:id="@+id/action_search" android:title="Search"
        android:icon="@drawable/abc_ic_search_api_mtrl_alpha"
        app:showAsAction="ifRoom"
        android:orderInCategory="100"/>
    <item android:id="@+id/action_settings"
android:title="@string/action_settings"
        android:orderInCategory="101" />
    <item android:id="@+id/action_help" android:title="Help &amp; Feedback"
        android:orderInCategory="102"/>
    <item android:id="@+id/action_logout" android:title="Logout"
        android:orderInCategory="103" app:showAsAction="never" />
</menu>
```
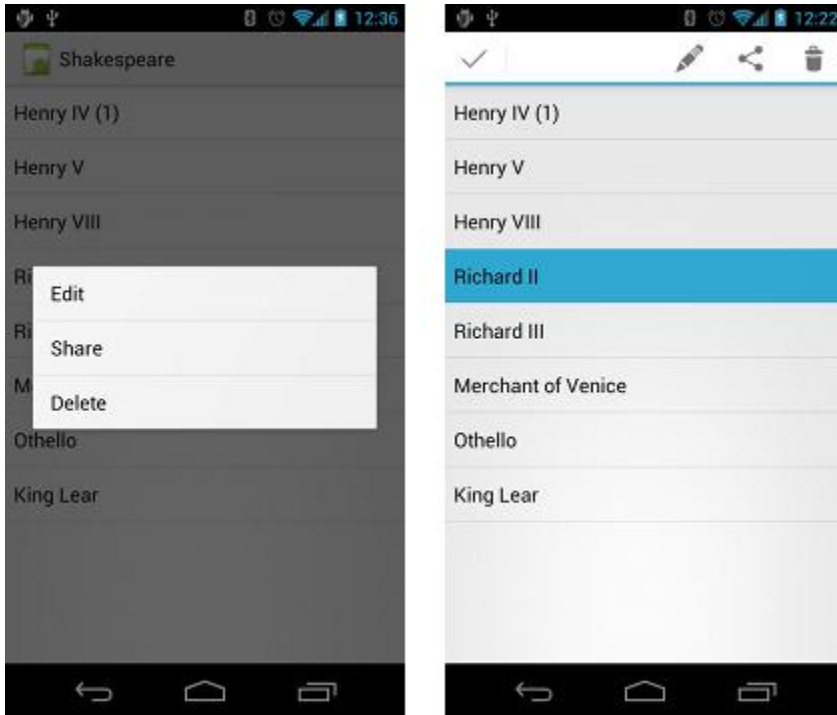
Contextual Menus

Figure 3. Screenshots of a floating context menu (left) and the contextual action bar (right).

- Allow users to take actions on specific items.
- Typically triggered with a long-press.
- Types of Context Menus
  - **Floating context menu.** A menu appears as a floating list of menu items. Conceptually similar to a right click in a desktop application.
  - **Contextual action mode.** Displays a contextual action bar at the top of the screen with action items that affect the selected item(s).

Floating Context Menu Example

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    // Register the View that has the context menu
    registerForContextMenu(yourView);
}
```

context_menu.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">
    <item android:id="@+id/action_edit" android:title="Edit"
        android:orderInCategory="100" app:showAsAction="never" />
    <item android:id="@+id/action_delete" android:title="Delete"
        android:orderInCategory="101" app:showAsAction="never" />
</menu>
```

Override Methods in Activity

```java
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenu.ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.context_menu, menu);
}


@Override
public boolean onContextItemSelected(MenuItem item) {
    AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo)
item.getMenuInfo();
    switch (item.getItemId()) {
        case R.id.action_edit:
            // TODO: Implement Edit
            return true;
        case R.id.action_delete:
            // TODO: Implement Delete
            return true;
        default:
            return super.onContextItemSelected(item);
    }
}
```

Up Navigation

All screens in your app that are not the main entrance to your app (the "home" screen) should offer the user a way to navigate to the logical parent screen in the app's hierarchy by pressing the Up button in the action bar.
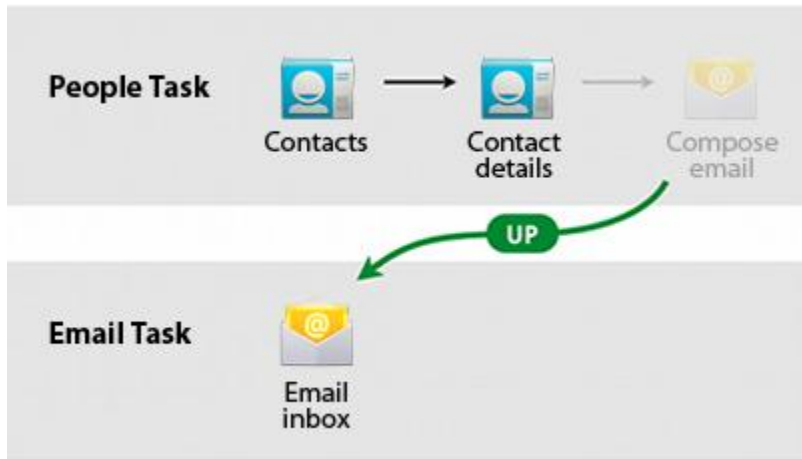

Figure 1: The *Up* button in Gmail.

Up Navigation vs. Back Navigation

Up Navigation: Based on the hierarchical relationships between screens.

Back Navigation: reverse chronological order through the history of screens the user has recently worked with. It is generally based on the temporal relationships between screens.

Up Navigation

Step 1. Specify the parent Activity

```xml
            <activity
                android:name=".ChildActivity"
                android:label="@string/title_activity_child"
                android:parentActivityName=".MainActivity">
                <meta-data
                    android:name="android.support.PARENT_ACTIVITY"
                    android:value=".MainActivity" />
            </activity>
```
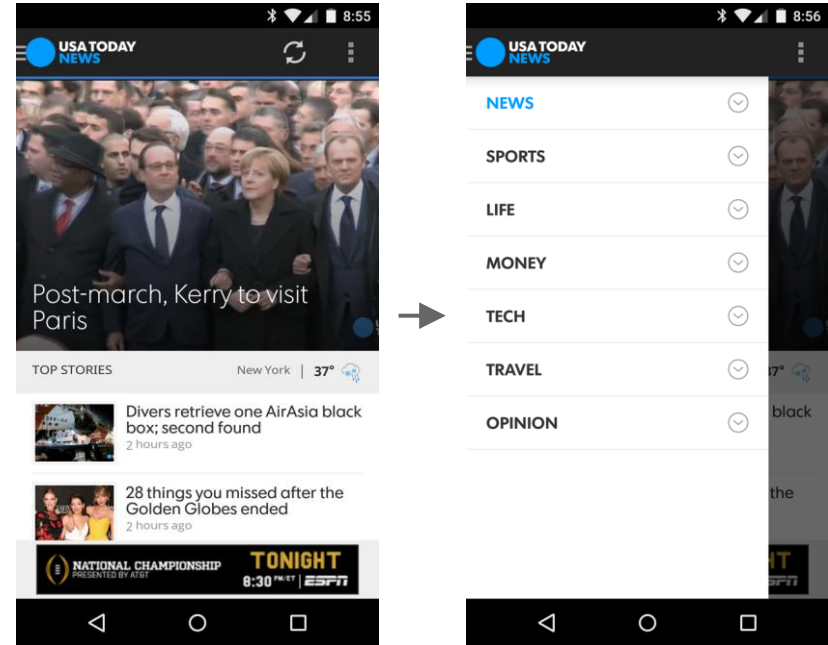
Step 2. Add Up Action in onCreate()

```java
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
```
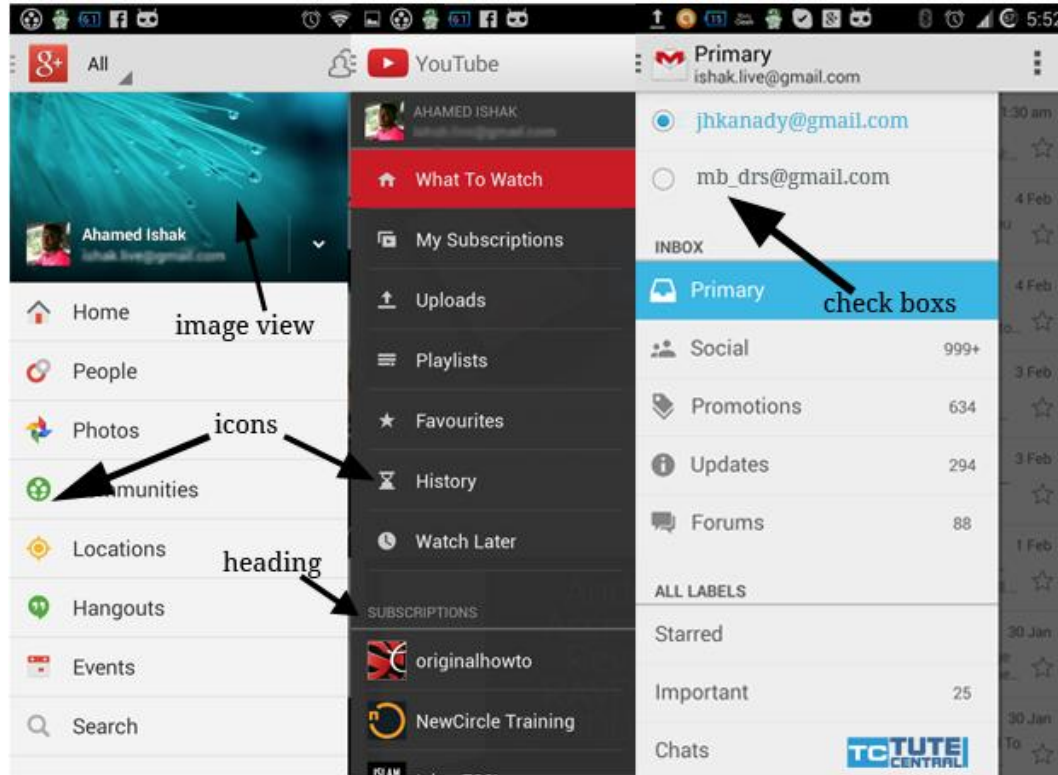
Navigation Drawer

- Displays the app's main navigation options
- Panel that displays the app's main navigation options on the left edge of the screen.
- Hidden most of the time, but is revealed when:
  - the user swipes a finger from the left edge of the screen
  - the user touches the app icon in the action bar while at the top level of the app

Navigation Drawer Customization

When to use Navigation Drawer

- More than 3 top-level views - (e.g., news app with different top-level categories)
- Deep navigation branches - enable efficient navigation from lower-level screens to other important places in your app
- Cross-navigation from lower levels

Deprecated: Legacy 6-Item Menu

- Pre-Android 3.0 (API level 11): Android-powered devices required to have a dedicated Menu button.
- Highly discouraged.

Questions?