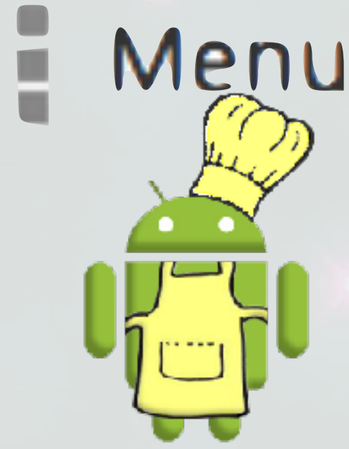
SCIENCES SORBONNE UNIVERSITÉ

Menus

Renaultⓐlrde.epita.fr

Usage



Better User Experience

- All action on the GUI are grouped

Since Android 3.0 the menu button is no longer mandatory

- Before it was used to display 6 elements (maximum)

A menu has a GUI so it is described through an XML

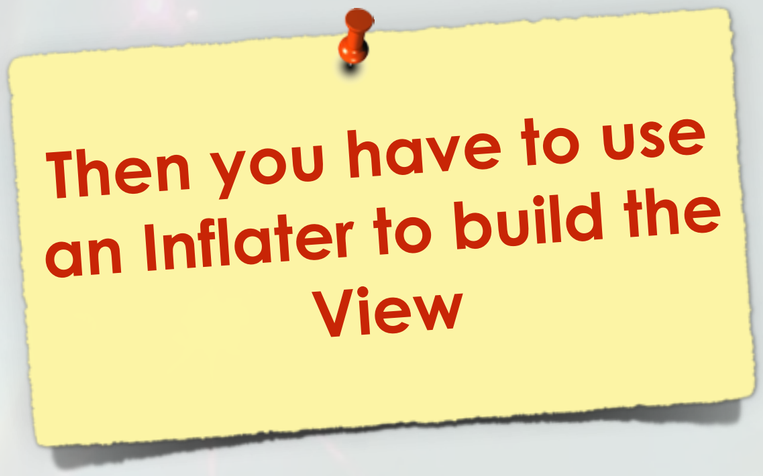
A lot of menu exists

- ContextMenu
- PopupMenu
- NavigatinDrawer
- OptionMenu
- ActionBar
- ...

Menu's Structure

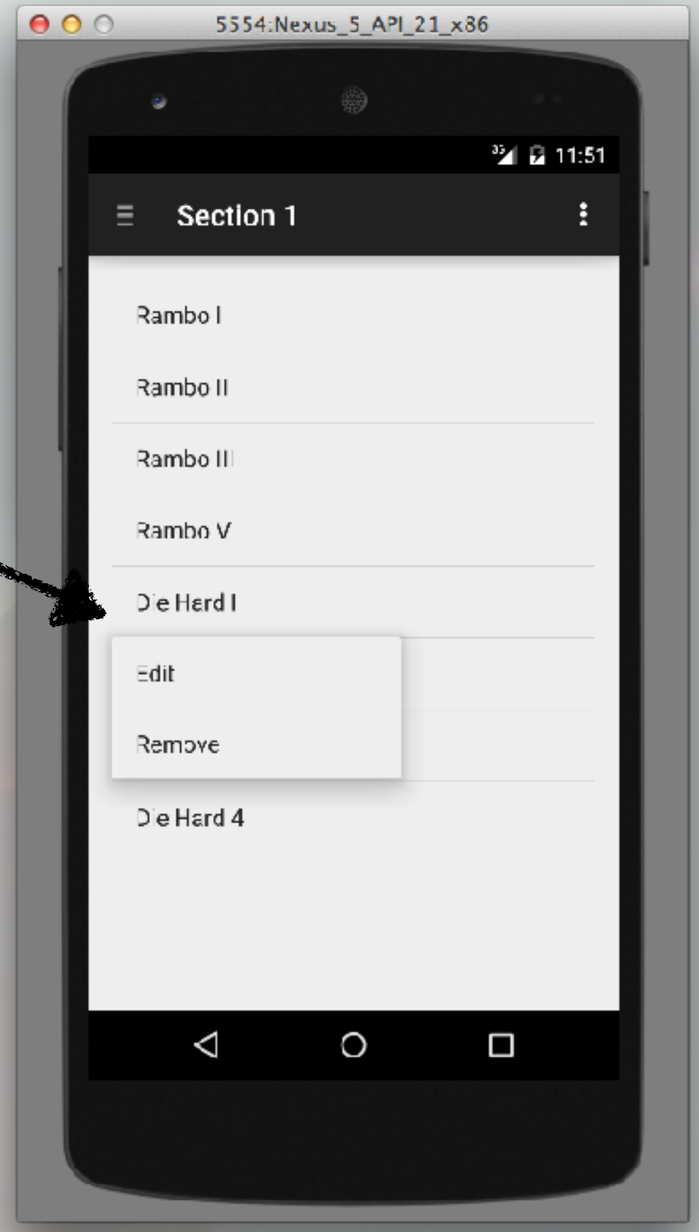
Define an XML ressource file to describe the structure

```
<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/edit"   android:title="Edit" />
    <item android:id="@+id/remove" android:title="Remove" />
</menu>
```



Popup Menu

- The menu appears above the specified View
- The simplest menu to build
- Trigger an action to display the menu
  - Long click for instance
- Register then for clicks
  - Implements MenuItem.OnMenuItemClickListener



Popup menu: implementation


```
ListView listView = (ListView) rootView.findViewById(R.id.listView);
adapter = new ArrayAdapter<String>(getActivity(),
                                   android.R.layout.simple_list_item_1, values);
listView.setAdapter(adapter);
listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> parent,
                                   View view, int position, long id) {
      PopupMenu popupMenu = new PopupMenu(getActivity(), view);
      MenuInflater menuInflater = popupMenu.getMenuInflater();
      menuInflater.inflate(R.menu.mymenu, popupMenu.getMenu());
      popupMenu.getMenu().findItem(R.id.edit)
              .setOnMenuItemClickListener(
                  new MenuItem.OnMenuItemClickListener() {
                    @Override
                    public boolean onMenuItemClick(MenuItem item) {
                        Toast.makeText(getActivity(),
                          item.getTitle(), Toast.LENGTH_LONG).show();
                        return false;
                    }
                }
              );
      popupMenu.show();
      return true;
}});
```

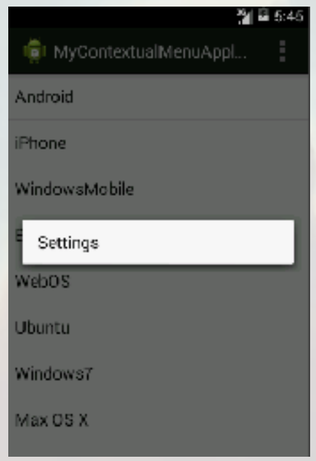
Contextual Menus

Dedicated actions for specific views

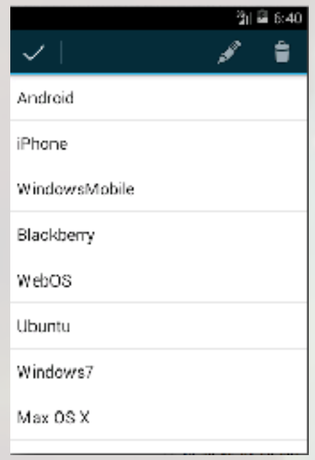
- Can be associated to any kind of Views

Two kind of contextual menus

- Floatting menus:
  - close to popup menu
  - the menu is displayed by a long press



- ActionMode:
  - one action bar is displayed with all possible actions



Floatting Menu

Register the view as a receiver for context menus

```
registerForContextMenu(listView);
```

Overload onCreateContextMenu to trigger the contextual menu

- onContextItemSelected is useful to know which item has been clicked

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenu.ContextMenuInfo menuInfo)
{
    super.onCreateContextMenu(menu, v, menuInfo);
    getActivity().getMenuInflater()
                 .inflate(R.menu.mymenu, menu);
}
```

Contextual Menu with ActionBar

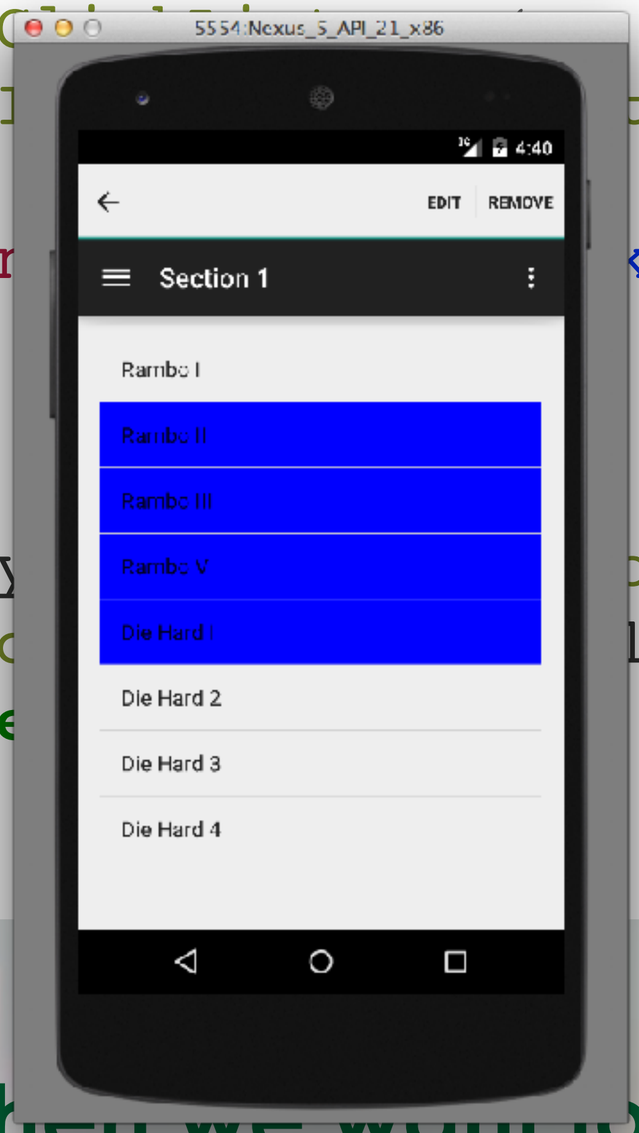
Implement ActionMode.Callback

```
private ActionMode.Callback modeCallback = new ActionMode.Callback()
{
  @Override
  public boolean onCreateActionMode(ActionMode mode, Menu menu) {
      MenuInflater inflater = getActivity().getMenuInflater();
      inflater.inflate(R.menu.mymenu, menu);
      return true;
  }
  @Override
  public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
      return false;
  }
  @Override
  public boolean onActionItemClicked(ActionMode mode, MenuItem item)
{
    return false;
  }
  @Override
  public void onDestroyActionMode(ActionMode mode) {
 }
};
```

Use the Callback

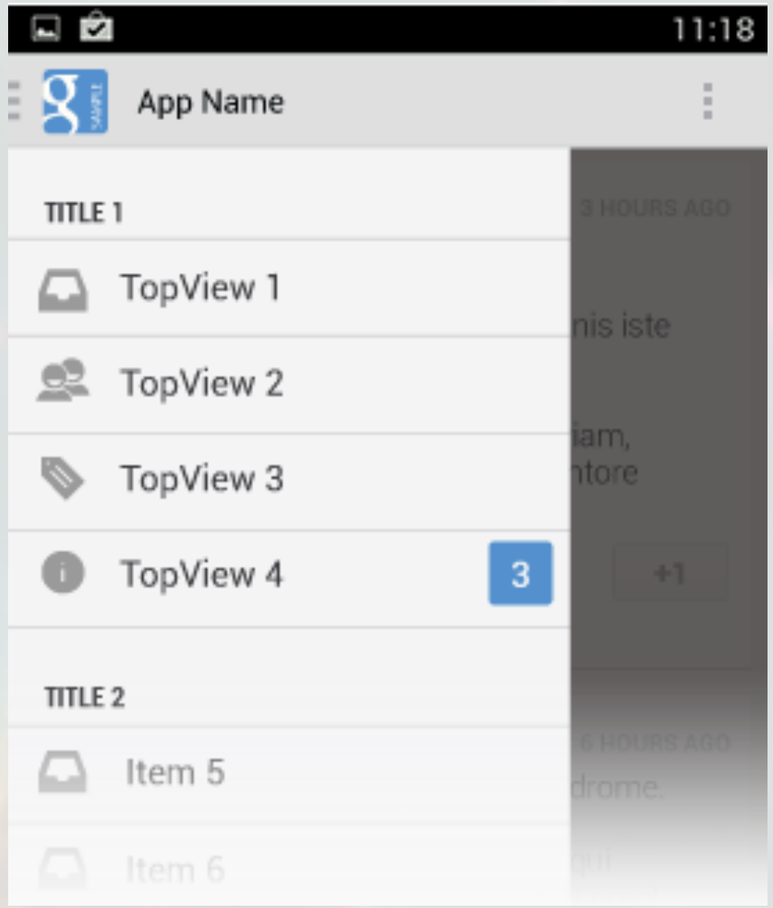
Run the callback with a long press

```
listView.setOnItemLongC
    new AdapterView.OnI                    tener() {
        @Override
        public boolean                     (AdapterView<?> parent,
                                             View view,
                                             int position, long id)
    {
            getActivity                    ode(modeCallback);
            view.setBac                    lor.BLUE);
            return true
    }
});
```



Very interesting when we want to perform an action on multiple views

Navigation Drawer



- **Hidden panel**

- **Can be revealed**
  - By a click
  - By a left-to-right swipe

- **Group options, actions, informations**
  - A full fragment

- **<u>Guideline:</u> the drawer must be opened during the first opening of the application**

The main View


```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <FrameLayout android:id="@+id/container"
                 android:layout_width="match_parent"
                 android:layout_height="match_parent" />

    <!-- The drawer is given a fixed width in dp and extends the full
      height  of the container. -->
     <fragment android:id="@+id/navigation_drawer"
         android:layout_width="@dimen/navigation_drawer_width"
         android:layout_height="match_parent"
         android:layout_gravity="start"
         android:name="com.example.admin.mynavigationdrawerapplication
                       .NavigationDrawerFragment"
        tools:layout="@layout/fragment_navigation_drawer" />
</android.support.v4.widget.DrawerLayout>
```

Navigation Drawer's View

Use a List for instance as the main View of the drawer

```
<ListView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:choiceMode="singleChoice"
    android:divider="@android:color/transparent"
    android:dividerHeight="0dp"
    android:background="#cccc"
    tools:context=".NavigationDrawerFragment" />
```

Manipulate the NavigationDrawer

Use openDrawer from the main activity

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mNavigationDrawerFragment =
       (NavigationDrawerFragment) getSupportFragmentManager()
       .findFragmentById(R.id.navigation_drawer);
    DrawerLayout mDrawerLayout =
            (DrawerLayout) findViewById(R.id.drawer_layout);
     mDrawerLayout.openDrawer(Gravity.START);
}
```

Dont forget to add dependencies in build.graddle

```
dependencies {
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support:design:28.0.0'
}
```

Add Header to the NavigationDrawer

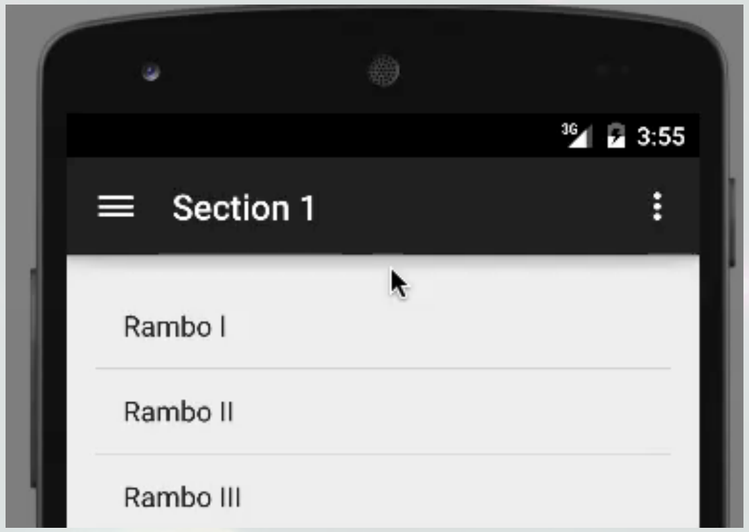
Specify the headerLayout attribute

```
<android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:menu="@menu/drawer_view"
        app:headerLayout="@layout/nav_header" />
```

And the just define layout/nav_header

Integration with the ActionBar

A fine grained integration with the ActionBar is possible



Use BarDrawerToggle

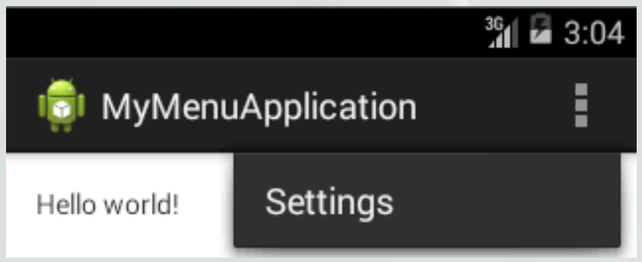
- Some effects are available only from android.support.v7.app.ActionBarDrawerToggle
- Objectives: ease the interaction with the action bar
- Ease the manipulation from the main activity
- Android Studio is providing canvas for these projects

Setup DrawerToggle into NavigationDrawer


```
mDrawerToggle = new ActionBarDrawerToggle(
        getActivity(),                    /* host Activity         */
        mDrawerLayout,                    /* DrawerLayout object */
      //R.drawable.ic_drawer,             /* nav drawer image to
                                             replace 'Up' caret  */
        R.string.navigation_drawer_open,  /* "open drawer"
                                          description for accessibility */
        R.string.navigation_drawer_close  /* "close drawer"
                                          description for accessibility */
    ) {
        @Override
        public void onDrawerClosed(View drawerView) {
            super.onDrawerClosed(drawerView);
            // do something...
        }

        @Override
        public void onDrawerOpened(View drawerView) {
            super.onDrawerOpened(drawerView);
           // do something
        }
```

Summary



- **The action Bar offers another menu which is highly configurable**

- **There are a lot of menus**
  - that can be mixed
  - almost fully configurable
    - **layout, highlight some elements**

- **Do not overload your application with menus**

- **Some menus are useful**
  - for navigation (NavigationDrawer)
  - for multiple editing (ActionMode.Callback)