Android
Using Menus

Victor Matos
Cleveland State University

Notes are based on:

The Busy Coder's Guide to Android Development
by Mark L. Murphy

ISBN: 978-0-9816780-0-9
&
Android Developers
http://developer.android.com/index.html

Using Menus

- Menus are used to add functionality without cluttering the UI.
- A menu is displayed as an overlapping layer on top of the current UI.
- In principle an *unlimited* number of additional operations could be added in a single menu
- An application could have any number of menus.

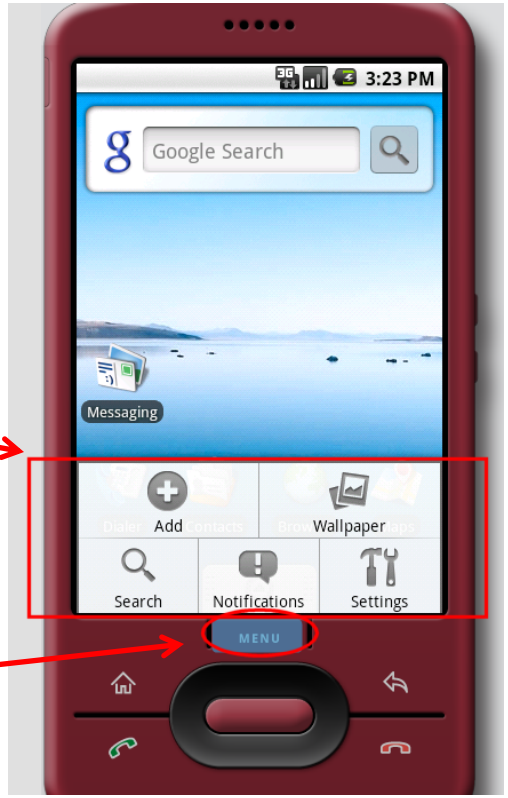
Android supports two types of menus: **options menu** and **context menu**.

1. The ***options menu*** is triggered by pressing the hardware/virtual **Menu** button on the device, while

2. the ***context menu*** is raised by a *tap-and-hold interaction* on the widget associated to the menu.

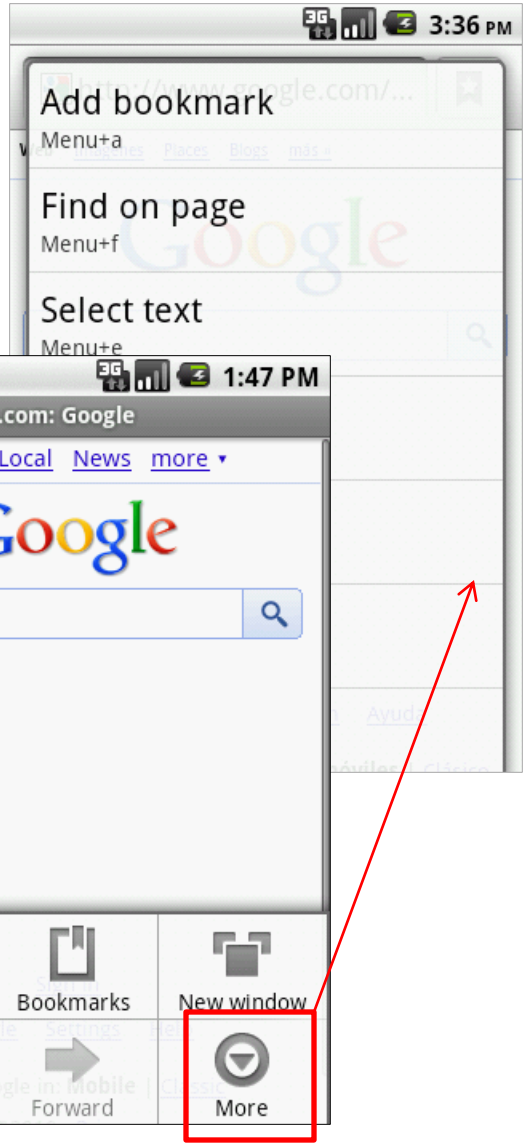
Using Menus

Figure 1.
Using an ***option menu & physical menu button***



Options available in this context

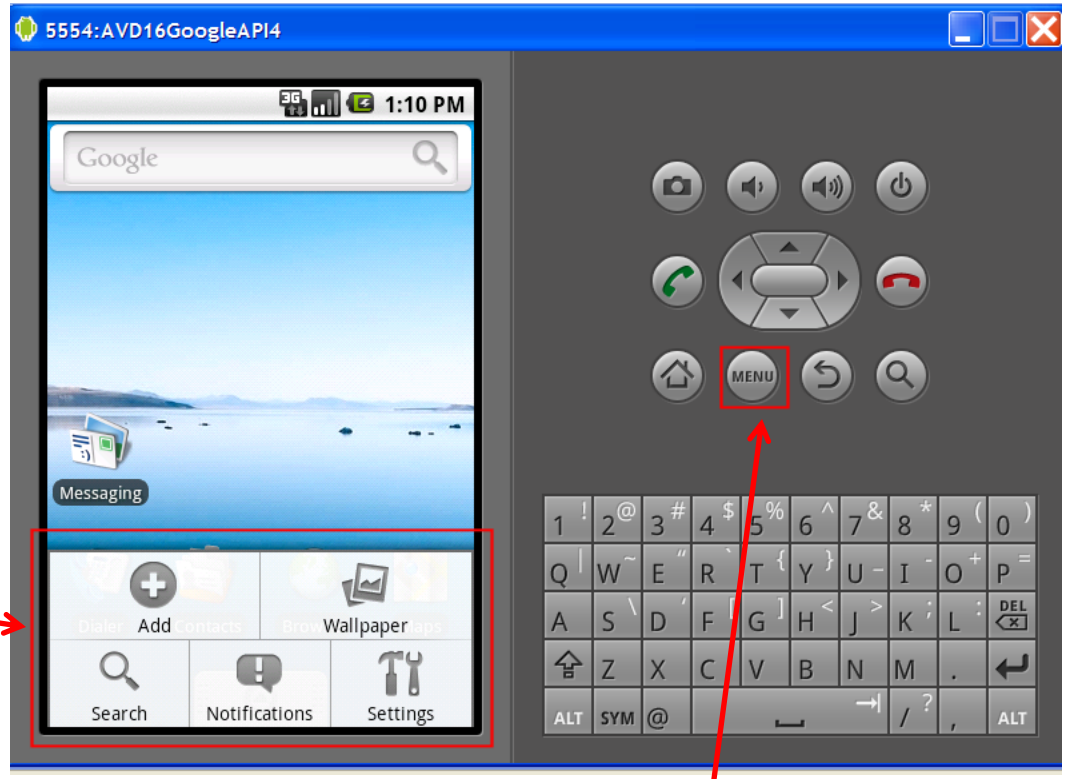
Press the physical **Menu** button



A maximum of six entries per menu. Excess will be displayed as part of the **More** option

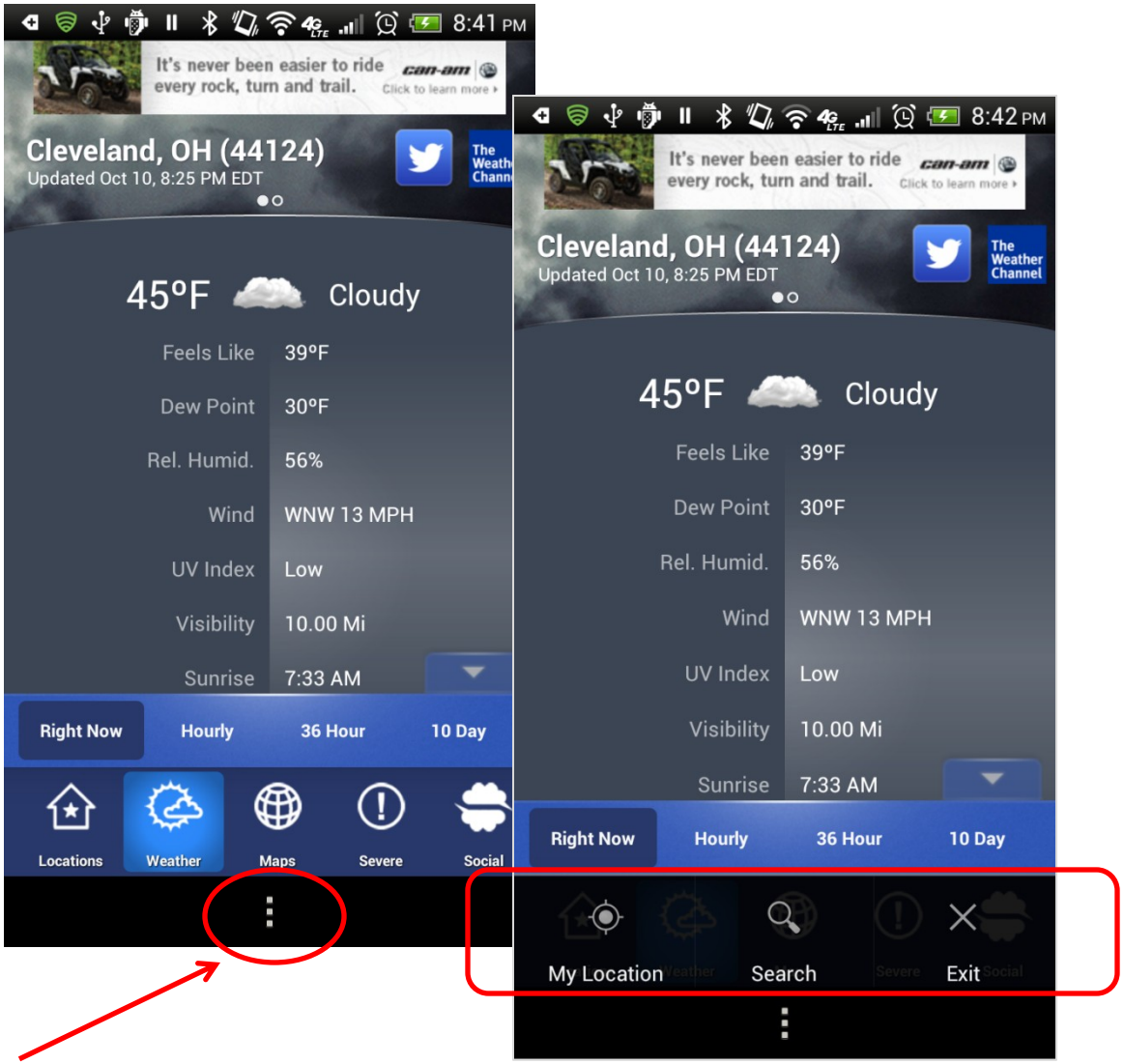
Using Menus

Figure 2.
Using an ***option menu & emulator's hardware menu button***



Using Menus

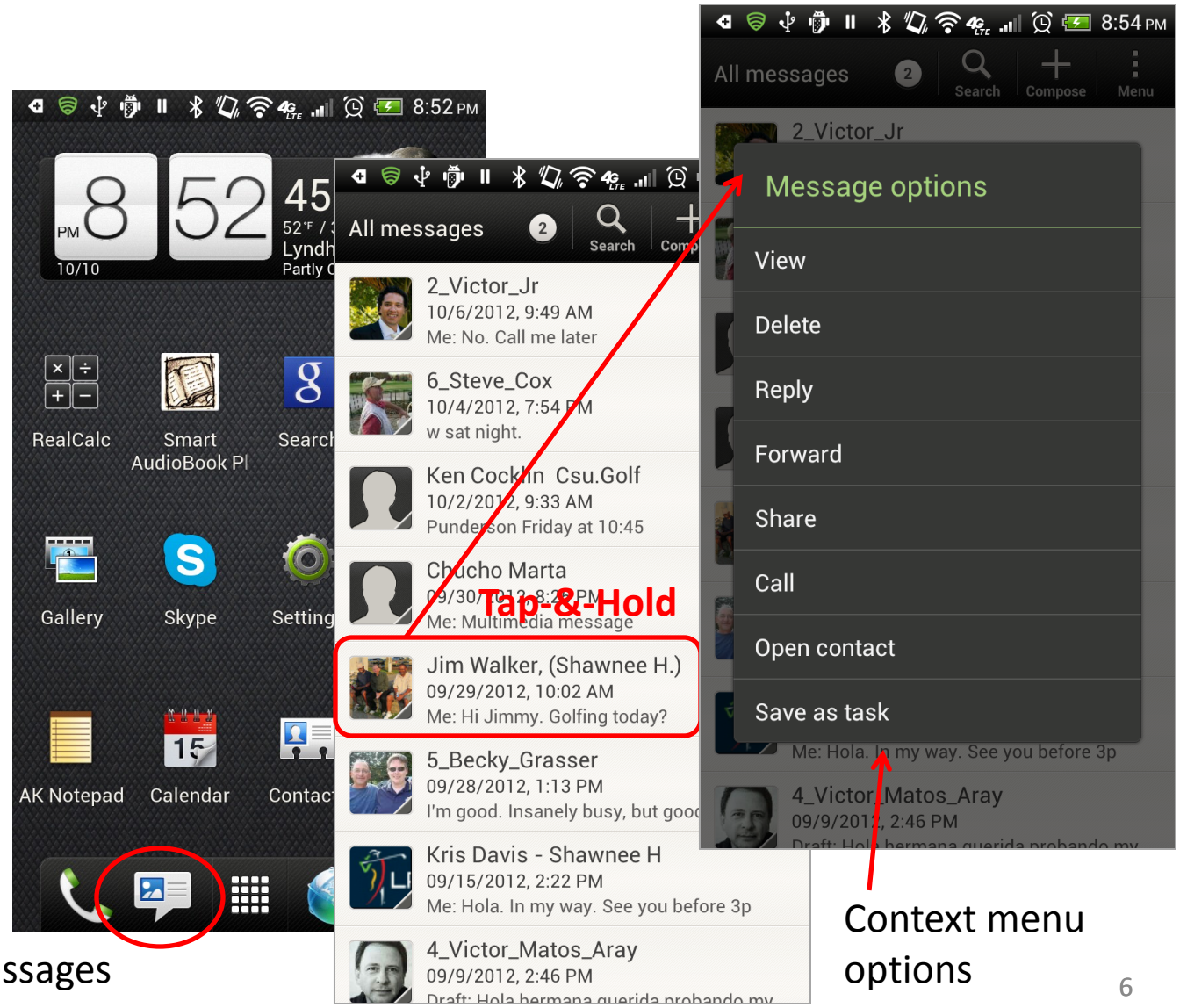
Figure 3.
Using an ***option menu & device's virtual menu button***
(HTC One phone)



Press **Menu** button

Using Menus

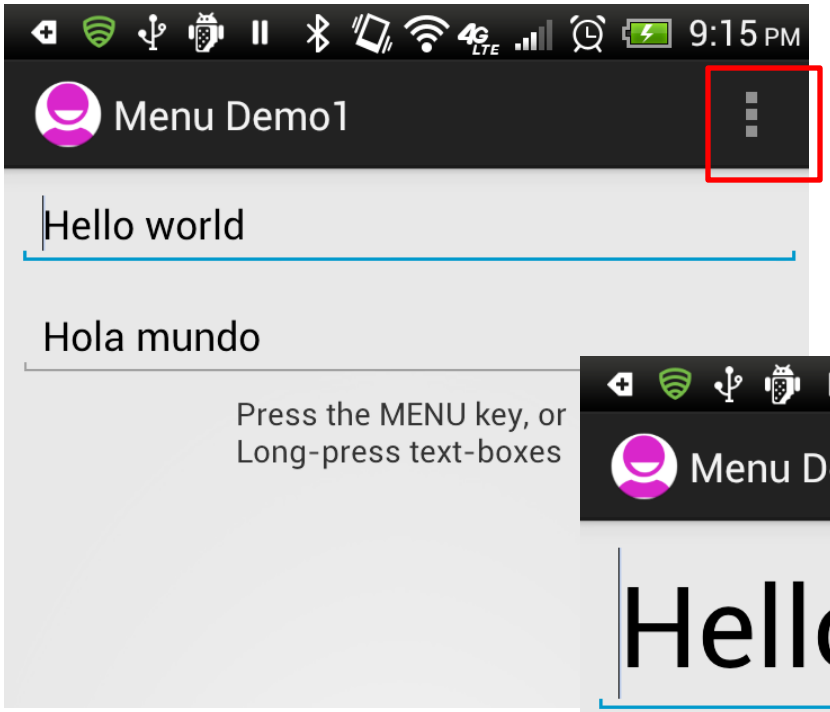
Figure 4.
Dealing with text-messages by using the built-in *Messaging* app's ***context menu***



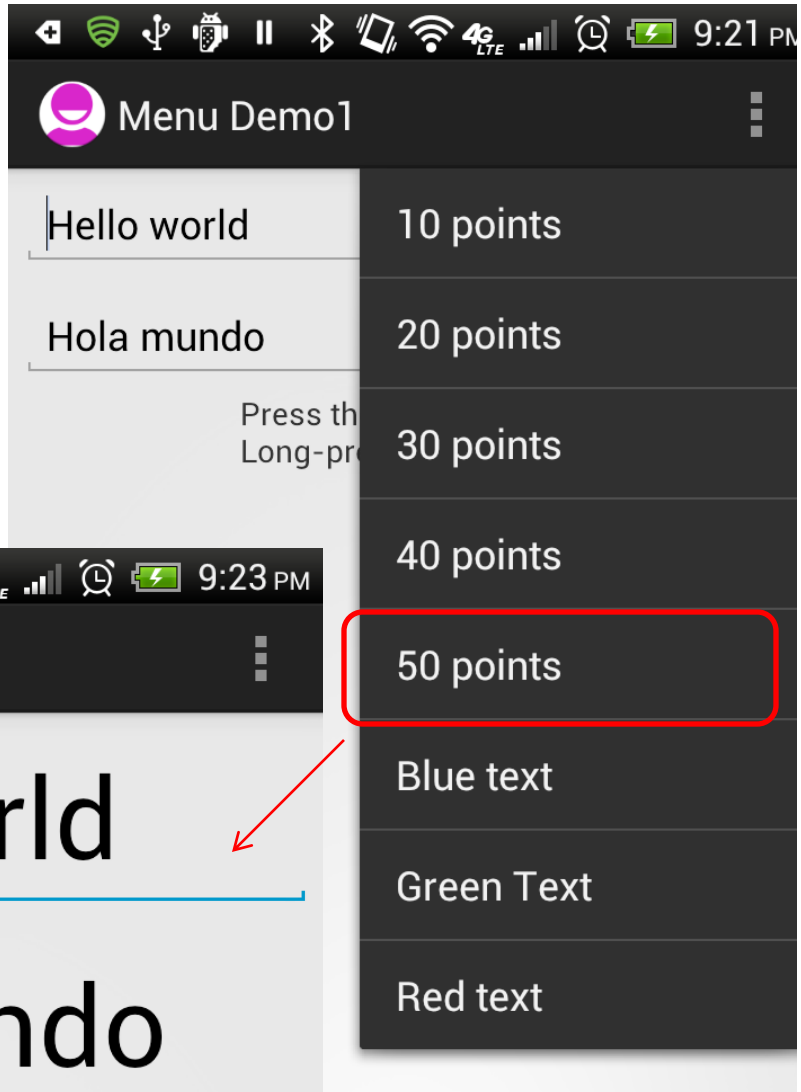
Using Menus

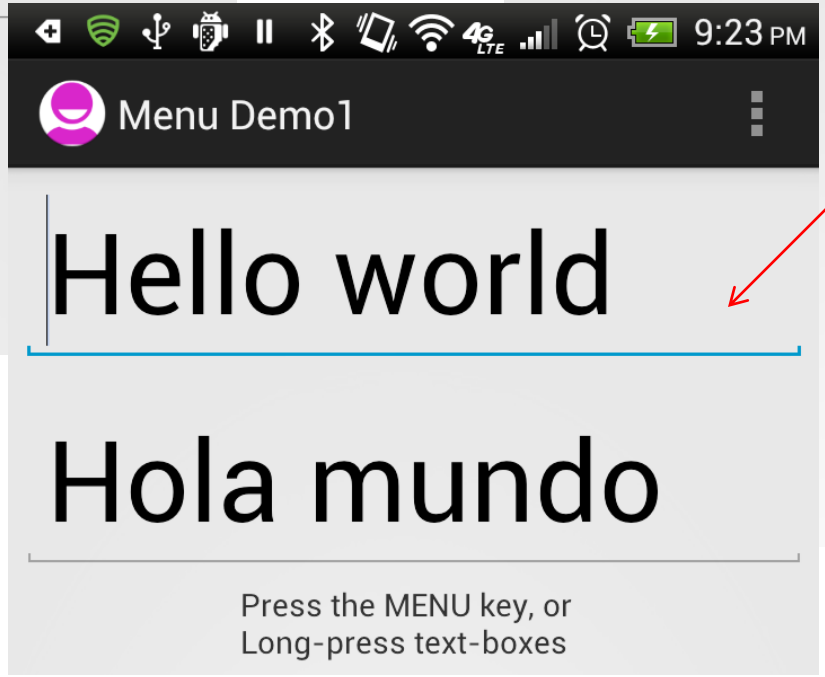
Example 1

This **Option Menu** offers a way of changing the text size & color of the EditText boxes.



Option Menu button





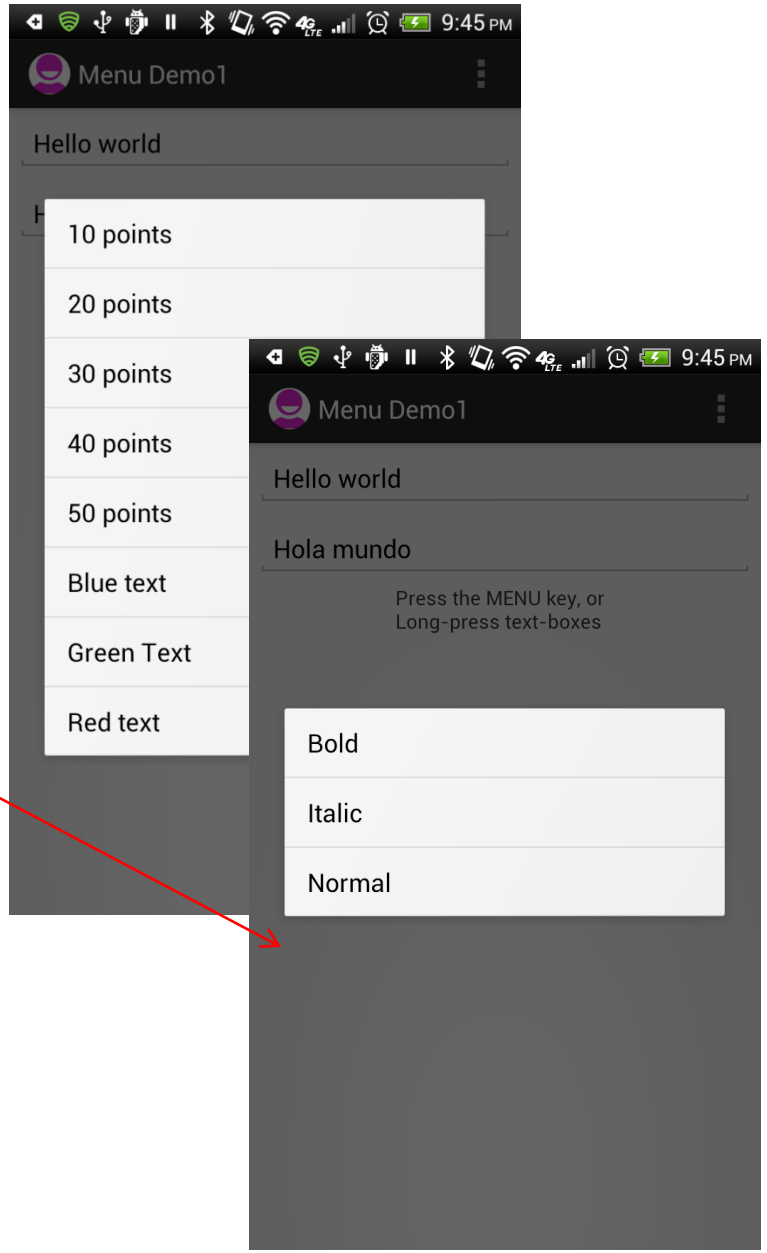
After choosing option:
50 points

Using Menus

Example 1: Using a **Context Menu**

Each view could have an associated Context Menu

Long-press a textbox to invoke its Context Menu

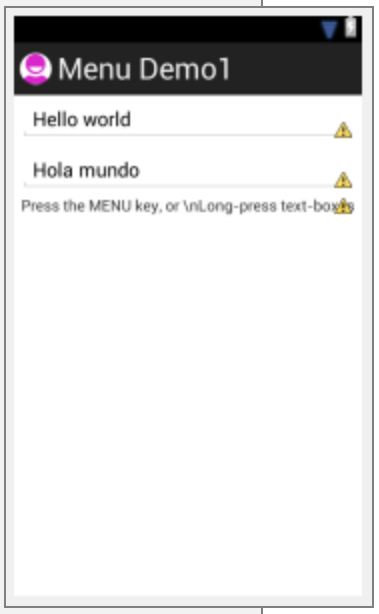


Using Menus

Example 1: Using **Option and Context Menu**

The app shows two text boxes. Menus are used to change text's size, color, and style.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <EditText
        android:id="@+id/txtBox1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:text="Hello world" />

    <EditText
        android:id="@+id/txtBox2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:text="Hola mundo" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Press the MENU key, or \nLong-press text-boxes" />

</LinearLayout>
```



Using Menus

Example 1: Using **Option and Context Menu**

The app shows two text boxes. Menus are used to change text's size, color, and style.

```
public class MainActivity extends Activity {
    EditText txtBox1;
    EditText txtBox2;
    Integer[] arrayPointSize = {10, 20, 30, 40, 50};

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        txtBox1 = (EditText)findViewById(R.id.txtBox1);
        txtBox2 = (EditText)findViewById(R.id.txtBox2);

        // you may register an individual context menu for each view
        registerForContextMenu(txtBox1);
        registerForContextMenu(txtBox2);

    } //onCreate
```

Using Menus

Example 1: Using **Option and Context Menu**

The app shows two text boxes. Menus are used to change text's size, color, and style.

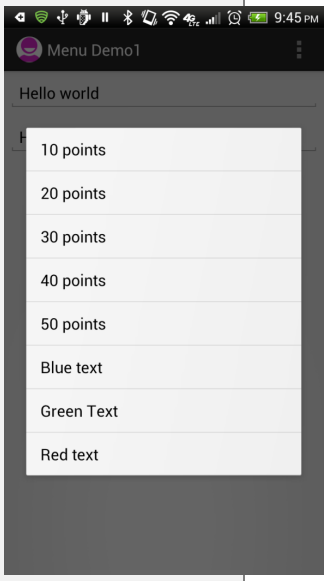
```
// set the option menu for the current activity
      @Override
      public boolean onCreateOptionsMenu(Menu menu) {
                // only one Option menu per activity
                populateMyFirstMenu(menu);
                return true;
      }

// detect what view is calling and create its context menu
      @Override
      public void onCreateContextMenu(ContextMenu menu, View v,
                              ContextMenuInfo menuInfo) {
                // decide what context menu needs to be made
                if (v.getId() == txtBox1.getId())
                          // create a menu for txtBox1 box
                          populateMyFirstMenu(menu);
                if (v.getId() == txtBox2.getId()){
                          // create a menu for txtBox2 box
                          populateMySecondMenu(menu);
                }
      } //onCreateContextMenu
```

Using Menus

Example 1: Using **Option and Context Menu**

The app shows two text boxes. Menus are used to change text's size, color, and style.

```
private void populateMyFirstMenu(Menu menu){
     int groupId = 0; int order= 0;
     //arguments: groupId, optionId, order, title
     menu.add(groupId, 1, 1, "10 points");
     menu.add(groupId, 2, 2, "20 points");
     menu.add(groupId, 3, 3, "30 points");
     menu.add(groupId, 4, 4, "40 points");
     menu.add(groupId, 5, 5, "50 points");

     menu.add(groupId, 6, 8, "Red text");
     menu.add(groupId, 7, 7, "Green Text");
     menu.add(groupId, 8, 6, "Blue text");
} //populateMyFirstMenu

private void populateMySecondMenu(Menu menu){
     int groupId = 0; int order= 0;
     //arguments: groupId, optionId, order, title
     menu.add(groupId,  9, 1, "Bold");
     menu.add(groupId, 10, 2, "Italic");
     menu.add(groupId, 11, 3, "Normal");
}//populateMySecondMenu
```



Using Menus

Example 1: Using **Option and Context Menu**

The app shows two text boxes. Menus are used to change text's size, color, and style.

```
// called whenever an item in your context menu is selected
@Override
public boolean onContextItemSelected(MenuItem item) {
     return applyMenuOption( item );
}

// called whenever an item in your options menu is selected.
@Override
public boolean onOptionsItemSelected(MenuItem item) {
     return applyMenuOption( item );
}
```

Note:

`boolean` event observers of the type ***onEvent(. . .)*** by convention return:

true to indicate the triggered event has been consumed by the method, and

false to signal that the event is still alive and could be consumed by others.

Using Menus

Example 1: Using **Option and Context Menu**

```
// apply the action associated to selected item
private boolean applyMenuOption(MenuItem item){

   int menuItemId = item.getItemId(); //1, 2, 3, ...11
   String strMsg2 = txtBox2.getText().toString();

   if (menuItemId <= 5) {
      // first five option are for setting text size (10pt, 20pt, ... )
      int newPointSize = arrayPointSize[menuItemId - 1];
      txtBox1.setTextSize(newPointSize);
      txtBox2.setTextSize(newPointSize);
   }
   else {
      // either change color on txtBox1 or style on txtBox2
      if (menuItemId == 6)
         txtBox1.setTextColor(color.background_dark | Color.RED); // red
      else if (menuItemId == 7)
         txtBox1.setTextColor(0xff00ff00); // green
      else if (menuItemId == 8)
         txtBox1.setTextColor(0xff0000ff); // blue
      else if (menuItemId == 9)
         txtBox2.setText(beautify(strMsg2, "BOLD")); //bold
      else if (menuItemId == 10)
         txtBox2.setText(beautify(strMsg2, "ITALIC")); //italic
      else if (menuItemId == 11)
         txtBox2.setText(beautify(strMsg2, "NORMAL")); //normal
   }
   return false;
} //applyMenuOption
```

Using Menus

Example 1: Using **Option and Context Menu**

The app shows two text boxes. Menus are used to change text's size, color, and style.

```
    // changing text style using HTML formatting
    // Spanned is text to which you could add formatting features

    private Spanned beautify (String originalText, String selectedStyle){

        Spanned answer = null;

        if (selectedStyle.equals("BOLD"))
            answer = Html.fromHtml("<b>" + originalText +"</b>");

        else if (selectedStyle.equals("ITALIC"))
            answer = Html.fromHtml("<i>" + originalText +"</i>");

        else if (selectedStyle.equals("NORMAL"))
            answer = Html.fromHtml("<normal>" + originalText +"</normal");

        return answer;
    } //beautify

} //class
```

Using Menus

Comments on Creating an Option & Context Menu

Step1.
Indicate which widget(s) on your activity have context menus. To do this, call

registerForContextMenu(*theWidget***)**

Step2.
Implement **onCreateContextMenu(…)**, populate your menu adding text, icons, etc. to the different options. Use input **menu** parameter to determine which menu to build (assuming your activity has more than one).

The ***onCreateContextMenu()*** method gets the *ContextMenu* itself, the *View* the context menu is associated with, and a *ContextMenu.ContextMenuInfo*, which tells you which item in the list the user did the tap-and-hold over, in case you want to customize the context menu based on that information

Using Menus

Comments on Creating an Option & Context Menu

- *onCreateContextMenu()* is called each time the context menu is requested.
- Unlike the *options menu* (which is only built once per activity), *context menus* are discarded once they are used.
- To find out which context menu choice was made, implement *onContextItemSelected()* on the activity.

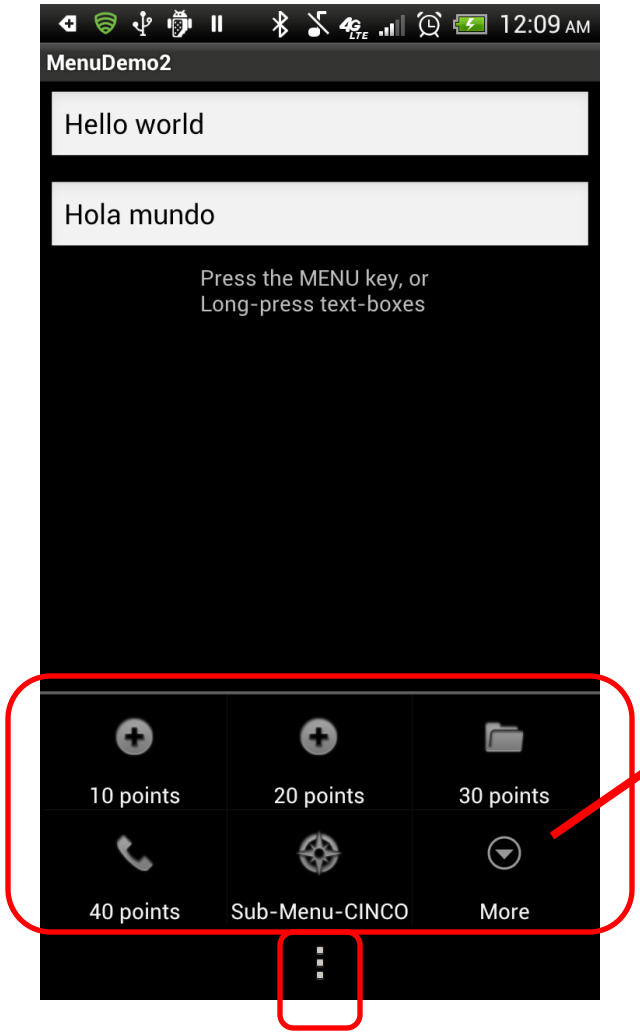
Using Menus

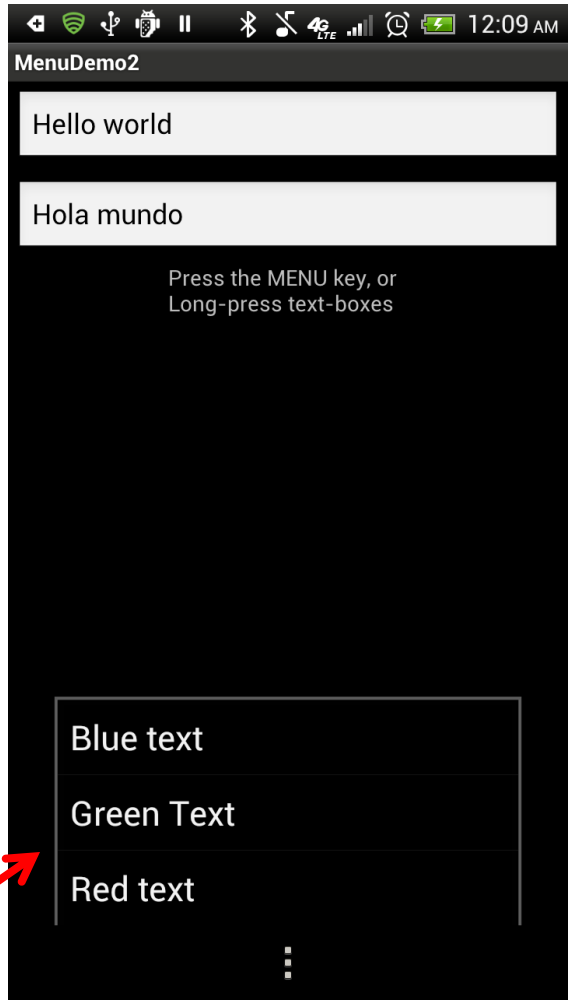
Extending Example1.

Example 2: Enhancing **Option/Context Menu**

A maximum of six options are displayed on the Option Menu.

If you have more than six selections the button **More** will display the remaining entries





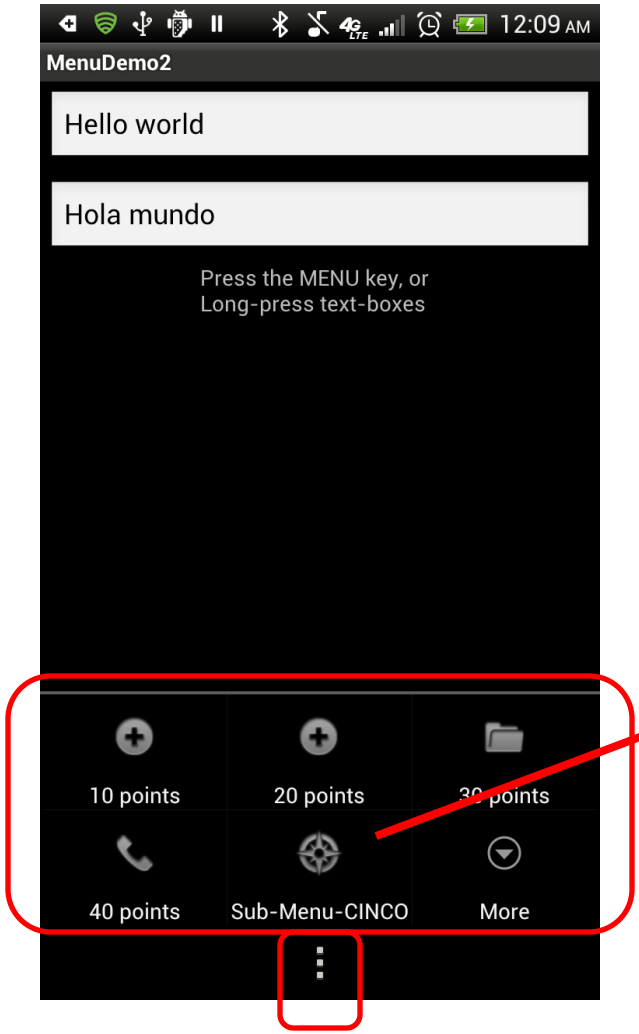
Using Menus

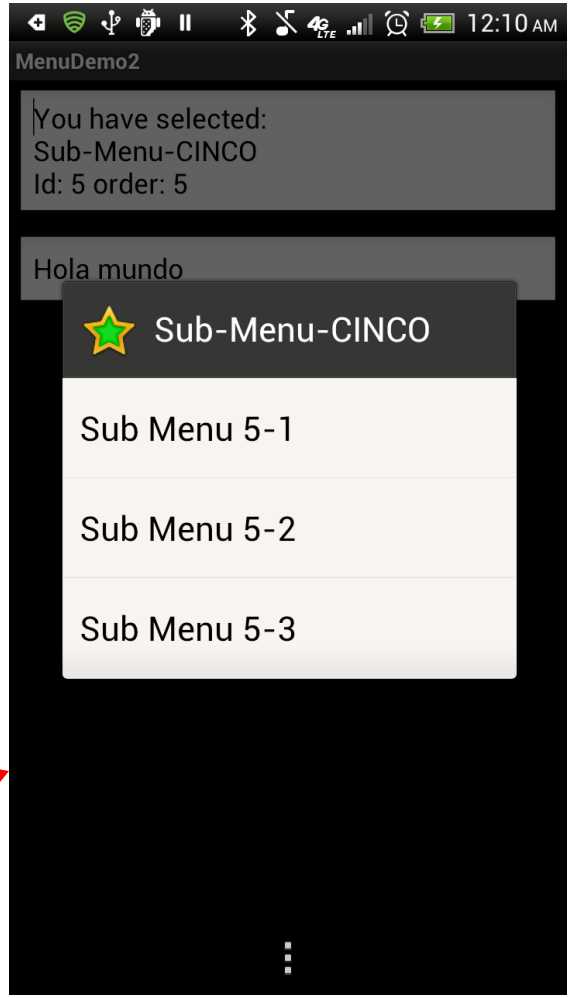
Extending Example1.

Example 2: Enhancing **Option/Context Menu**

A **Sub-Menu** item shows a DialogBox displaying the associated sub-options.

In this example item-5 is a SubMenu type.





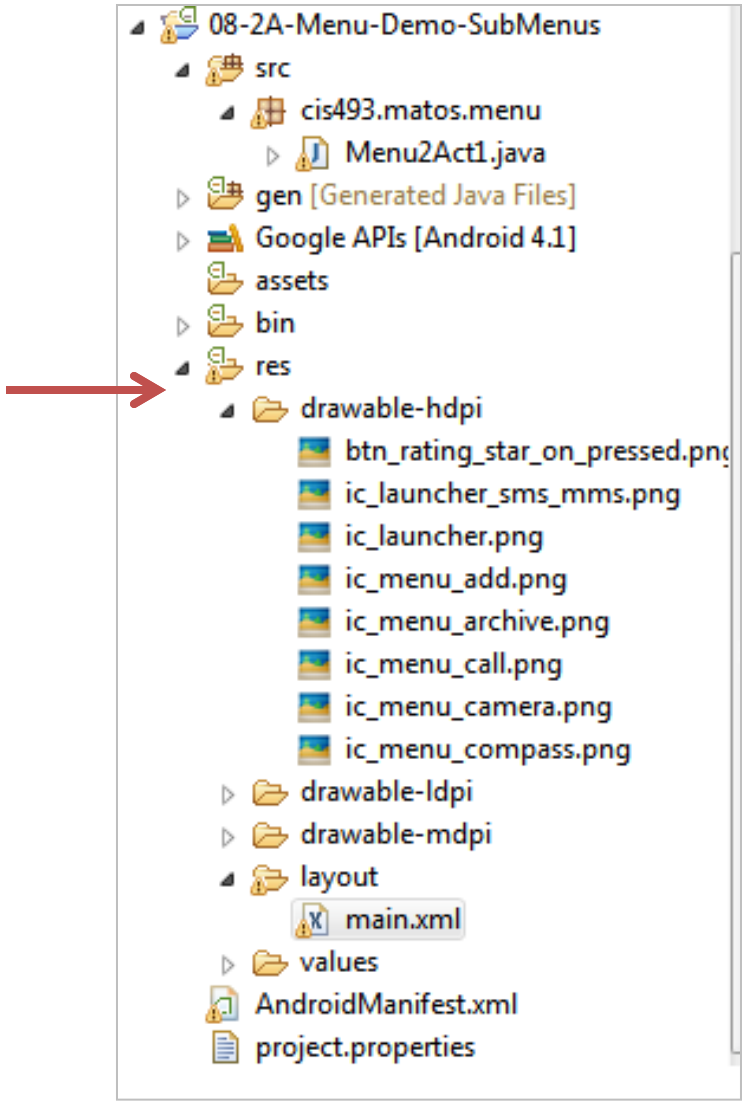
Using Menus

Extending Example1.

Example 2: Enhancing
Option/Context Menu

Add icons to the app's res/drawable folder.

A handfull of icons are available in the folder:
…Android-sdk/docs/images/icon-design



Using Menus

Extending Example1.

Example 2: Enhancing **Option/Context Menu**

Replace the method **populateMyFirstMenu** with the following code

```
private void populateMyFirstMenu(Menu menu){
   int groupId = 0;
   //arguments: groupId, optionId, order, title
   MenuItem item1 = menu.add(groupId, 1, 1, "10 points");
   MenuItem item2 = menu.add(groupId, 2, 2, "20 points");
   MenuItem item3 = menu.add(groupId, 3, 3, "30 points");
   MenuItem item4 = menu.add(groupId, 4, 4, "40 points");
   //MenuItem item5 = menu.add(groupId, 5, 5, "50 points");

   MenuItem item6 = menu.add(groupId, 6, 8, "Red text");
   MenuItem item7 = menu.add(groupId, 7, 7, "Green Text");
   MenuItem item8 = menu.add(groupId, 8, 6, "Blue text");

   //set icons
    item1.setIcon(R.drawable.ic_menu_add);
    item2.setIcon(R.drawable.ic_menu_add);
    item3.setIcon(R.drawable.ic_menu_archive);
    item4.setIcon(R.drawable.ic_menu_call);
```

Remove this line from previous version

Icons are added to first five entries of the Option Menu

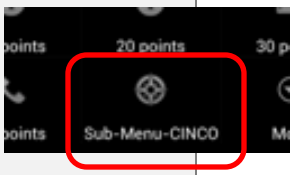
Using Menus

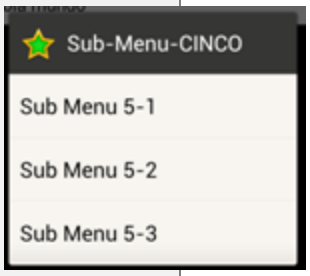
Extending Example1.

Example 2: Enhancing **Option/Context Menu**

Replace the method **populateMyFirstMenu** with the following code

```
        // adding a sub-menu as fifth entry of this menu
        // .addSubMenu(int groupId, int itemId, int order, CharSequence title)
        int smGroupId = 0; // don't care, same as Menu.NONE
        int smItemId = 5;  // fifth element
        int smOrder = 5;   // don't care, same as Menu.NONE

        SubMenu mySubMenu5 =  menu.addSubMenu(smGroupId, smItemId, smOrder,
                                              "Sub-Menu-CINCO");
        mySubMenu5.setHeaderIcon(R.drawable.btn_rating_star_on_pressed);
        mySubMenu5.setIcon(R.drawable.ic_menu_compass);

        // .add(int groupId, int itemId, int order, CharSequence title)

        MenuItem sub51 = mySubMenu5.add(smGroupId,5,1,"Sub Menu 5-1");
        MenuItem sub52 = mySubMenu5.add(smGroupId,5,2,"Sub Menu 5-2");
        MenuItem sub53 = mySubMenu5.add(smGroupId,5,3,"Sub Menu 5-3");


    } //populateMyFirstMenu
```





Using Menus

Extending Example1.

Example 2: Enhancing **Option/Context Menu**

Continuation…
Replace the method
applyMenuOption
with the following code

```
private boolean applyMenuOption(MenuItem item){

    int menuItemId = item.getItemId(); //1, 2, 3, ...11

    String strMsg2 = txtBox2.getText().toString();

    if (menuItemId < 5) {
        // first four options are for setting text size
        int newPointSize = arrayPointSize[menuItemId - 1];
        txtBox1.setTextSize(newPointSize);
        txtBox2.setTextSize(newPointSize);
    }
    else if (menuItemId == 5) {
        // the sub-menu (attached to 5th item) is processed here
        txtBox1.setText (
                    "You have selected: \n" +item.getTitle()
                  + "\nId: " + menuItemId
                  + " order: " + item.getOrder() );
}
    // either change color on text1 or style on text2
    else if (menuItemId == 6)
        txtBox1.setTextColor(0xffff0000); // red
```

Same as before

Take care of sub-menu here

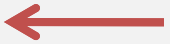
Using Menus

Extending Example1.

Example 2: Enhancing **Option/Context Menu**

Continuation…
Replace the method **applyMenuOption** with the following code

```
    else if (menuItemId == 7)
        txtBox1.setTextColor(0xff00ff00); // green
    else if (menuItemId == 8)
        txtBox1.setTextColor(0xff0000ff); // blue

    else if (menuItemId == 9)
        txtBox2.setText(beautify(strMsg2, "BOLD")); //bold
    else if (menuItemId == 10)
        txtBox2.setText(beautify(strMsg2, "ITALIC")); //italic
    else if (menuItemId == 11)
        txtBox2.setText(beautify(strMsg2, "NORMAL")); //normal

    return false;

} //applyMenuOption
```



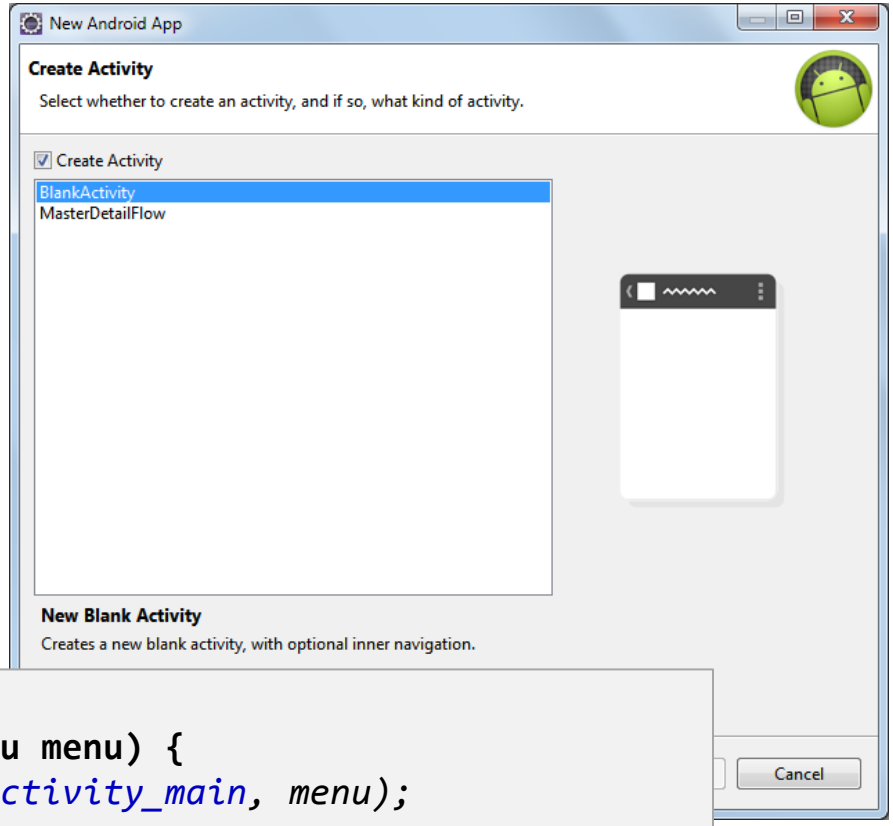
Same as in Example1

Using Menus

Example 3: Using the Menu-Snippet Created by Eclipse+ADK

Assume you are using SDK 4.x.

Applications created using the Eclipse Wizard allows you to choose a new “Blank Activity”.



Those activities include the code fragment:

```
@Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
```

Using Menus

Example 3: Using the Menu-Snippet Created by Eclipse+ADK

The call to **onCreateOptionsMenu** inflates the skeleton of an XML menu file stored in your app's **/res/menu/** .

The following example is an extension of the basic xml file

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/menu_settings1"
        android:orderInCategory="100"
        android:title="Menu-Option-1"/>
    <item
        android:id="@+id/menu_settings2"
        android:orderInCategory="110"
        android:title="Menu-Option-2"/>
    <item
        android:id="@+id/menu_settings3"
        android:orderInCategory="120"
        android:title="Menu-Option-3"/>

</menu>
```

Using Menus

Example 3: Using the Menu-Snippet Created by Eclipse+ADK

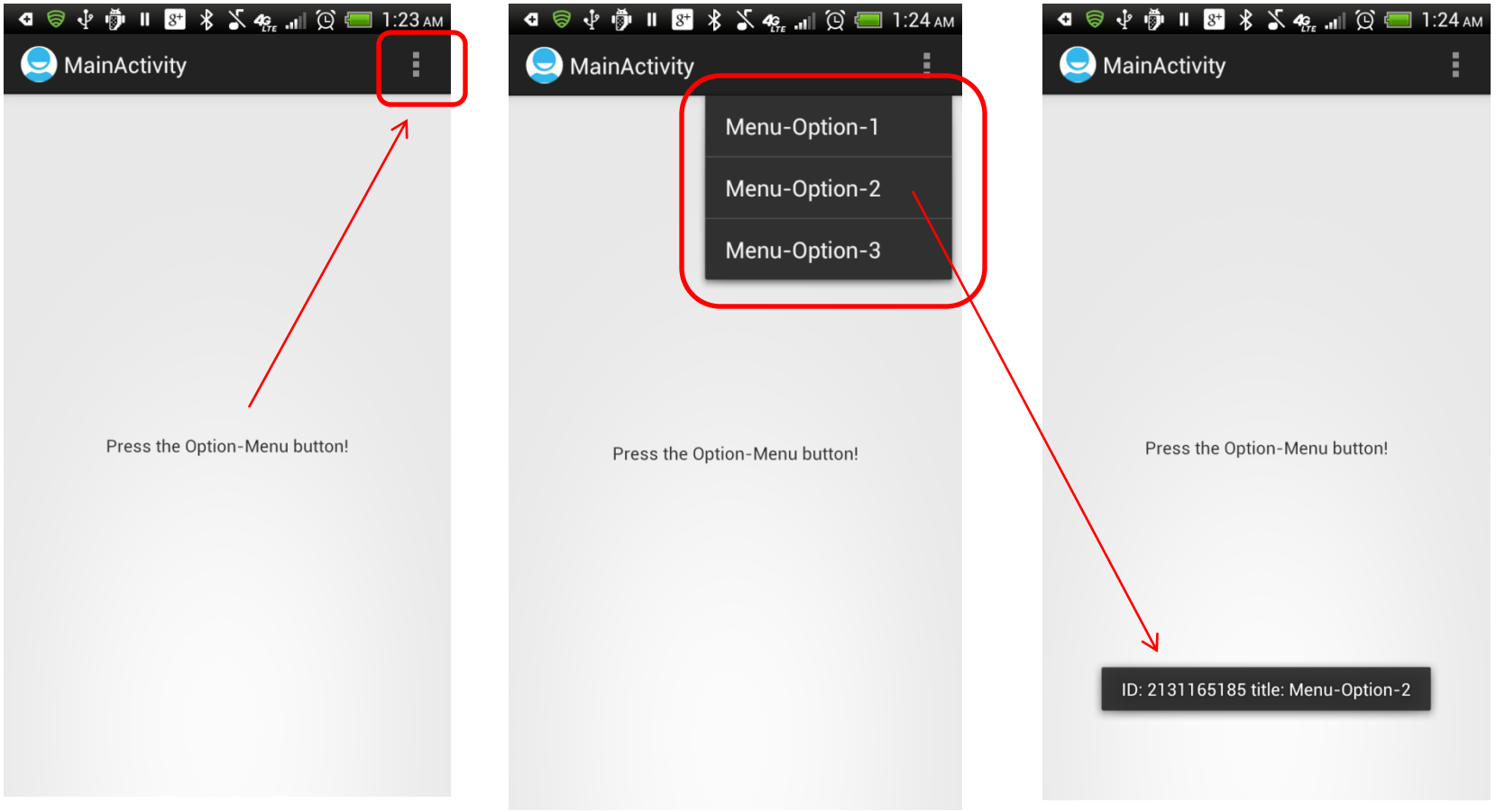
To add functionality to the menu you need to implement the **onOptionsItemSelected** method. For instance

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {

      String text = "ID: " + item.getItemId()
                  + " title: " + item.getTitle();

      Toast.makeText(this, text, 1).show();
      return true;
}
```

Using Menus

Example 3: Using the Menu-Snippet Created by Eclipse+ADK

Our example produces the following images:



Using Menus

Questions ?