Android Dialog Boxes

AlertDialog - Toast

Victor Matos

Cleveland State University

Notes are based on:

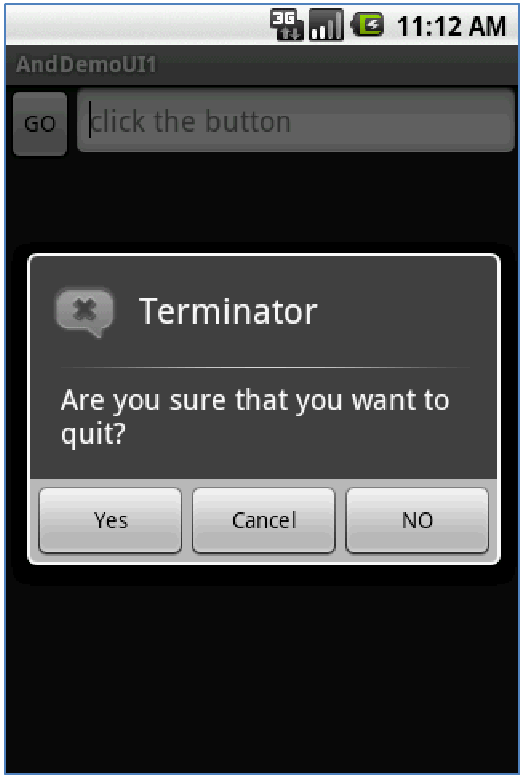
Android Developers

http://developer.android.com/index.html

The DialogBox

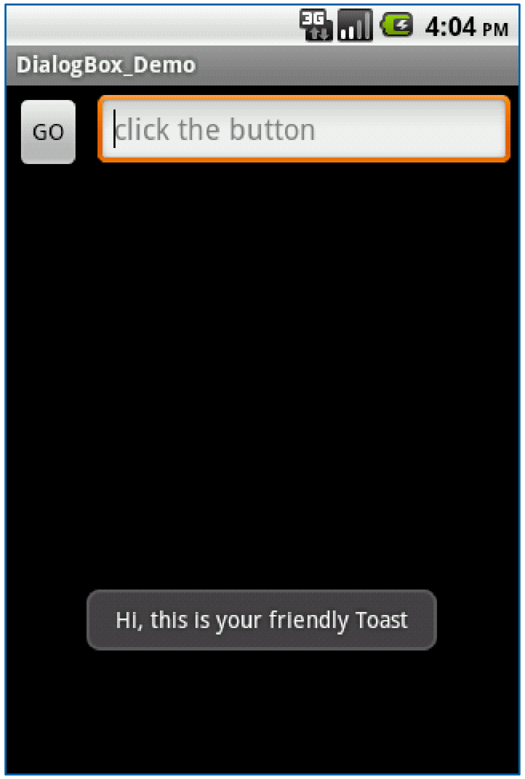
Android provides two primitive forms of dialog boxes:

1. **AlertDialog** boxes, and
2. **Toast** controls



Terminator

Are you sure that you want to quit?



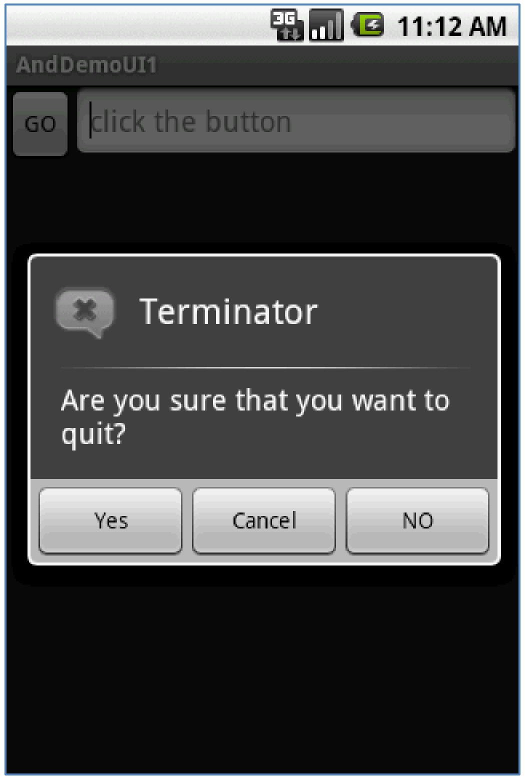
Hi, this is your friendly Toast

The AlertDialog

The *AlertDialog* is an *almost modal* screen that

(1) presents a brief message to the user typically shown as a small floating window that partially obscures the underlying view, and

(2) collects a simple answer (usually by clicking an option button) .



Are you sure that you want to quit?

Note:
A *modal* view remains on the screen waiting for user's input. The rest of the application is on hold. It has to be dismissed by an explicit user's action.

The AlertDialog

Warning !!!

An *AlertDialog* is **NOT** a typical *inputBox* (as in .NET)

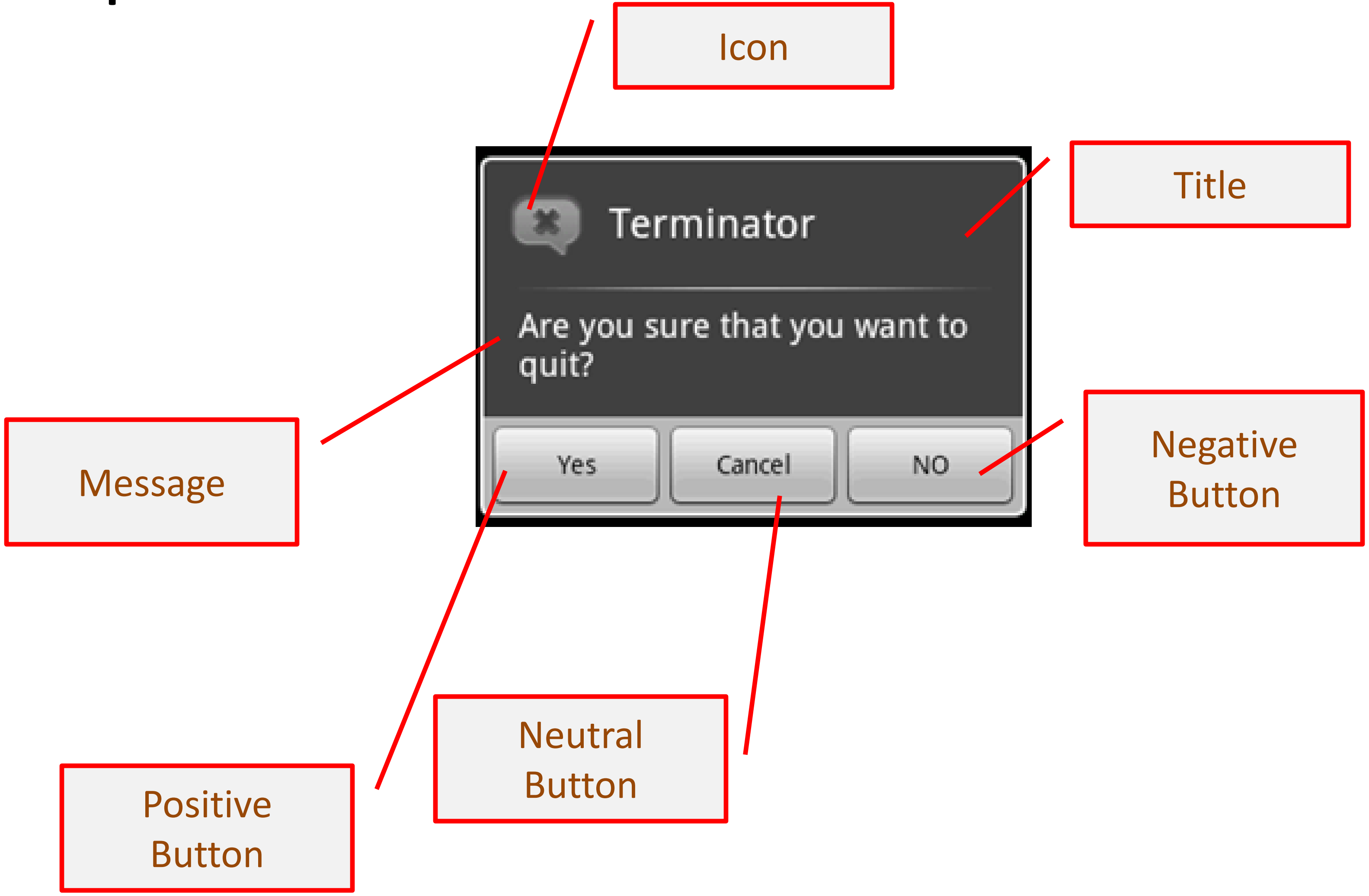
Why?

An *AlertDialog* box is modal as it needs user intervention to be terminated

However
it *does not stop the main thread* (code following the call to show the *DialogAlert* box is executed without waiting for the user's input)

The AlertDialog

Example:



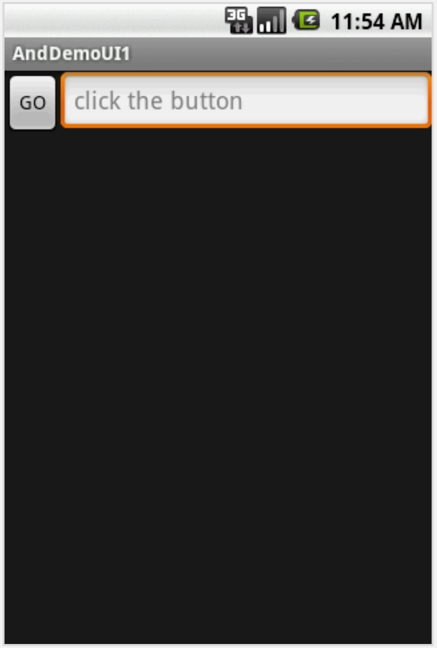
Terminator

Are you sure that you want to quit?

The AlertDialog

Example: A simple Dialog Box

```
<LinearLayout
    android:id="@+id/LinearLayout01"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal">
<Button
   android:text="GO"
   android:id="@+id/btnGo"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content">
 </Button>
 <EditText
   android:hint="click the button"
   android:id="@+id/txtMsg"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content">
 </EditText>

</LinearLayout>
```



The AlertDialog

Example: A simple dialog box

```
package cis493.selectionwidgets;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class AndDemoUI1 extends Activity {

  Button btnGo;
  EditText txtMsg;
  String msg;
```

The AlertDialog

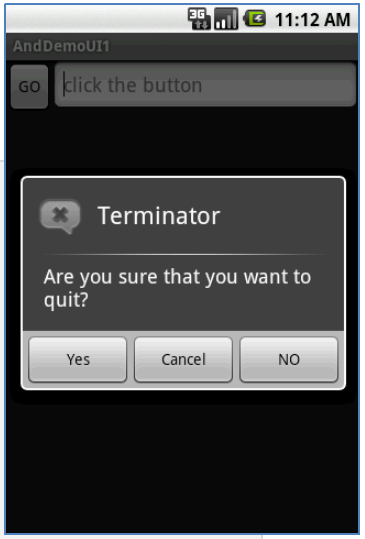
Example: A simple dialog box

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        txtMsg =  (EditText)findViewById(R.id.txtMsg);
        btnGo = (Button) findViewById(R.id.btnGo);
        btnGo.setOnClickListener(new OnClickListener() {
          @Override
          public void onClick(View arg0) {

              AlertDialog dialBox = createDialogBox();
              dialBox.show();

              // WARNING: (in general...)
              // after showing a dialog you should have NO more code. Let the buttons of
              // the dialog box handle the rest of the logic. For instance, in this
              // example a modal dialog box is displayed (once shown you can not do
              // anything to the parent until the child is closed) however the code in
              // the parent continues to execute after the show() method is
              // called.
              txtMsg.setText("I am here!");
          }
    });

    }//onCreate
```

The AlertDialog

Example: A simple dialog box

```
private AlertDialog createDialogBox(){

    AlertDialog myQuittingDialogBox =

       new AlertDialog.Builder(this)

       //set message, title, and icon
       .setTitle("Terminator")
       .setMessage("Are you sure that you want to quit?")
       .setIcon(R.drawable.ic_menu_end_conversation)

       //set three option buttons
       .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton) {
                   //whatever should be done when answering "YES" goes here
                   msg = "YES " + Integer.toString(whichButton);
                   txtMsg.setText(msg);
            }
       })//setPositiveButton
```



Are you sure that you want to quit?

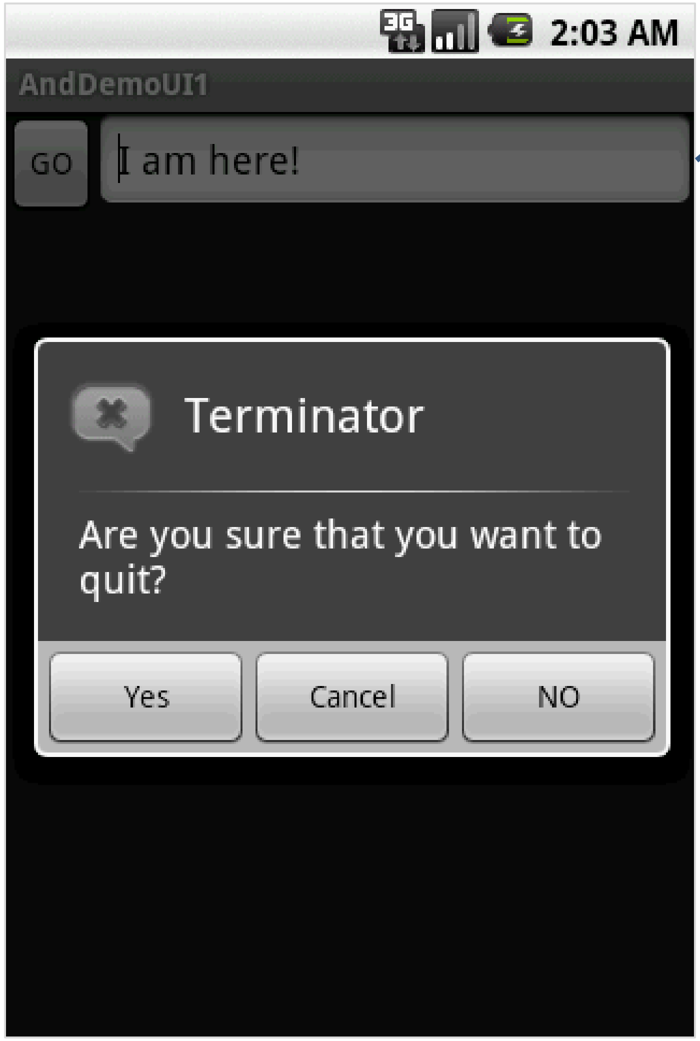
The AlertDialog

Example: A simple dialog box

```
        .setNeutralButton("Cancel",new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton) {
                  //whatever should be done when answering "CANCEL" goes here
                  msg = "CANCEL " + Integer.toString(whichButton);
                  txtMsg.setText(msg);
            }//OnClick
        })//setNeutralButton

        .setNegativeButton("NO", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton) {
                  //whatever should be done when answering "NO" goes here
                  msg = "NO " + Integer.toString(whichButton);
                  txtMsg.setText(msg);
             }
        })//setNegativeButton

        .create();
        .return myQuittingDialogBox;

    }// createDialogBox

}// class
```

The AlertDialog

Example: A simple AlertDialog box



2:03 AM

AndDemoUI1

GO

I am here!

Terminator

Are you sure that you want to quit?

Yes

Cancel

NO

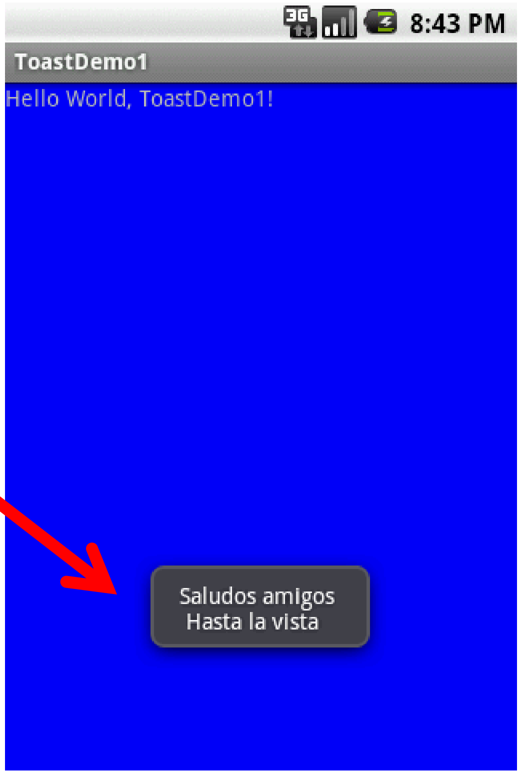
This text is set right after showing the dialog box

The Toast View

A **Toast** is a transient view containing a quick little message for the user.

They appear as a floating view over the application.

They never receive focus.



Hello World, ToastDemo1!

Saludos amigos
Hasta la vista

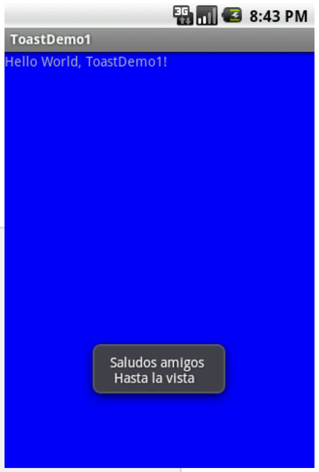
The Toast View

Example: A simple Toast

```
Toast.makeText ( context,  message,  duration ).show();
```

Context:	A reference to the view's environment (what is around me…)
Message:	The thing you want to say
Duration:	SHORT or LONG exposure

The Toast View



Example: A simple Toast

```
package cis493.dialogboxes;

import android.app.Activity;
import android.os.Bundle;
import android.widget.Toast;

public class ToastDemo1 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Toast.makeText(
               getApplicationContext(),
               "Saludos amigos \n Hasta la vista",
               Toast.LENGTH_LONG).show();
    }
}
```

The Toast View

As an aside

Context:

On Android a Context is mostly used to load and access resources.

All widgets receive a Context parameter in their constructor.

In a regular Android application, you usually have two kinds of Context, *Activity* and *Application*. The first one is typically passed to classes and methods that need a Context.

Views have a reference to the entire activity and therefore to anything your activity is holding onto; usually the entire View hierarchy and all its resources.

The Toast View

Customizing a Toast View

By default Toast views are displayed at the center-bottom of the screen.

However the user may change the placement of a Toast view by using either of the following methods:

```
void setGravity (int gravity, int xOffset, int yOffset)
```

Set the location at which the notification should appear on the screen.

```
void setMargin (float horizontalMargin, float verticalMargin)
```

Set the margins of the view.

The Toast View

Customizing a Toast View

The following method uses offset values based on the pixel resolution of the actual device. For instance, the G1 phone screen contains 360x480 pixels.

```
void setGravity (int gravity, int xOffset, int yOffset)
```

Gravity: Overall placement. Typical values include: *Gravity.CENTER, Gravity.TOP, Gravity.BOTTOM, …*

xOffset: Assume Gravity.CENTER placement on a G1 phone. The *xOffset* range is -160,…,0,…160 (left, center, right)

yOffset: The range on a G1 is: -240,…,0,…240 (top, center, bottom)

The Toast View

Customizing the Toast View

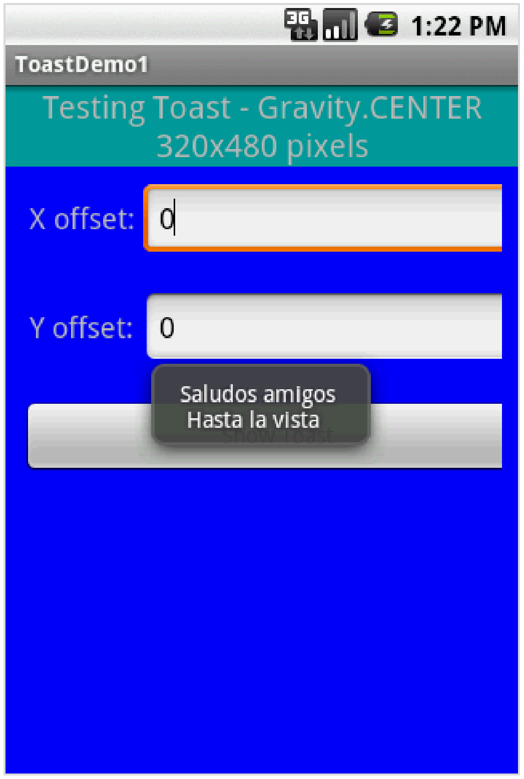
A second method to place a Toast is ***setMargin***. The screen is considered to have a center point where horizontal and vertical center lines meet. There is 50% of the screen to each side of that center point (top, botton, left, right). Margins are expressed as a value between: -50,…, 0, …, 50.

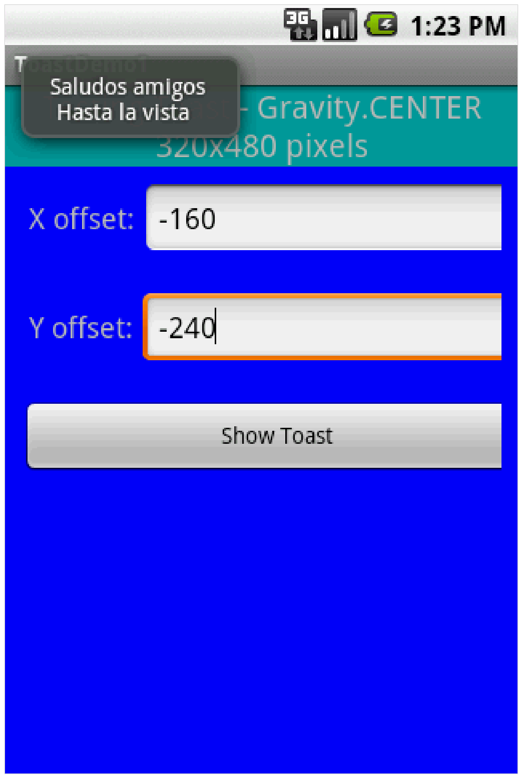
```
void setMargin (float horizontalMargin, float verticalMargin)
```

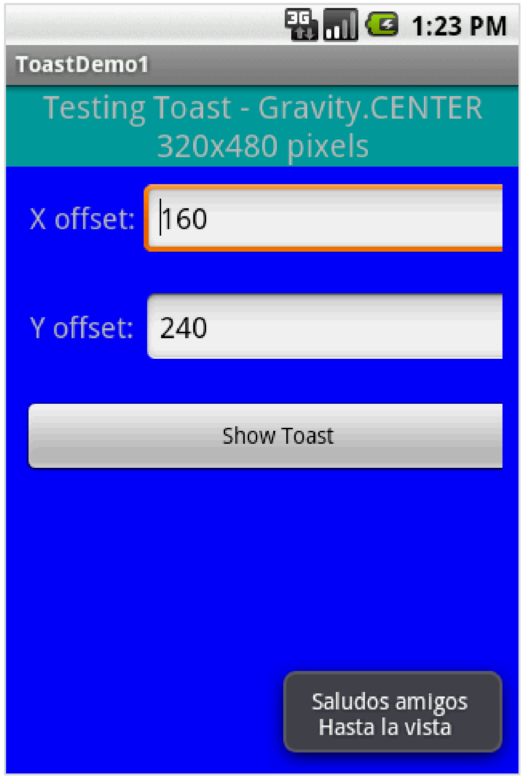
Note:
The pair of margins (-50, -50) represent the upper-left corner of the screen, (0, 0) is the center, and (50, 50) the lower-right corner.

The Toast View

Example: Changing the placement of a Toast view.







Using the **`setGravity(…)`** method with Gravity.CENTER, and x and y offsets of (resp.):

0, 0	(center)
-160, -240	(top-left)
160, 240	(right-bottom)

The Toast View

Example: Changing the placement of a Toast view.

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
android:id="@+id/myTableLayout"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#ff0000ff"
android:orientation="vertical"
android:stretchColumns="1,2"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<TableRow
android:id="@+id/myRow1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:orientation="horizontal"
>
<TextView
android:id="@+id/myCaption"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff009999"
android:text="Testing Toast - Gravity.CENTER  320x480 pixels"
android:textSize="20sp"
android:gravity="center"
android:layout_span="2"
>
</TextView>
</TableRow>
<TableRow
android:id="@+id/myRow1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff0000ff"
android:padding="10px"
android:orientation="horizontal"
>
<TextView
android:id="@+id/xLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text=" X offset:  "
android:textSize="18sp"
>
</TextView>
<EditText
android:id="@+id/xBox"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="0"
android:textSize="18sp"
android:inputType="numberSigned"
>
</EditText>
</TableRow>
<TableRow
android:id="@+id/myRow2"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff0000ff"
android:padding="10px"
android:orientation="horizontal"
>
<TextView
android:id="@+id/yLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text=" Y offset:  "
android:textSize="18sp"
>
</TextView>
<EditText
android:id="@+id/yBox"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="0"
android:textSize="18sp"
android:inputType="numberSigned"
>
</EditText>
</TableRow>
<TableRow
android:id="@+id/myRow3"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff0000ff"
android:padding="10px"
android:orientation="horizontal"
>
<Button
android:id="@+id/btn1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text=" Show Toast  "
android:layout_span="2"
>
</Button>
</TableRow>
</TableLayout>
```

The Toast View

Example: Changing the placement of a Toast view.

```
package cis493.dialogboxes;

import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class ToastDemo1 extends Activity {
    EditText xBox;
    EditText yBox;
    Button   btn1;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main2);

        xBox = (EditText)findViewById(R.id.xBox);
        yBox = (EditText)findViewById(R.id.yBox);
```

The Toast View

Example: Changing the placement of a Toast view.

```
btn1 = (Button)findViewById(R.id.btn1);
btn1.setOnClickListener(new OnClickListener() {
     @Override
     public void onClick(View v) {
         try {
              Toast myToast =  Toast.makeText(
                   getApplicationContext(),
                   "Saludos amigos \n Hasta la vista",
                   Toast.LENGTH_LONG);
              myToast.setGravity(Gravity.CENTER,
                        Integer.valueOf(xBox.getText().toString()),
                        Integer.valueOf(yBox.getText().toString()) );
              myToast.show();

              } catch (NumberFormatException e) {
                   Toast.makeText(getApplicationContext(),
                   e.getMessage(),
                   Toast.LENGTH_LONG).show();
              }
       }// onClick
     }); // listener
 }// onCreate
}// class
```

The Toast View

Example: **Showing Fancy Toast views.**

Toasts could be modified to display a custom combination of color/shape/text/background.

You need to follow the next steps:

1. Define the XML layout of the new custom view
2. Make sure there is a *TextView* named: **`text`**
3. Additionally you could attach an android: background to the TextView.
4. The background could be a figure (such as a *.png* file) or an XML defined shape (see next example).

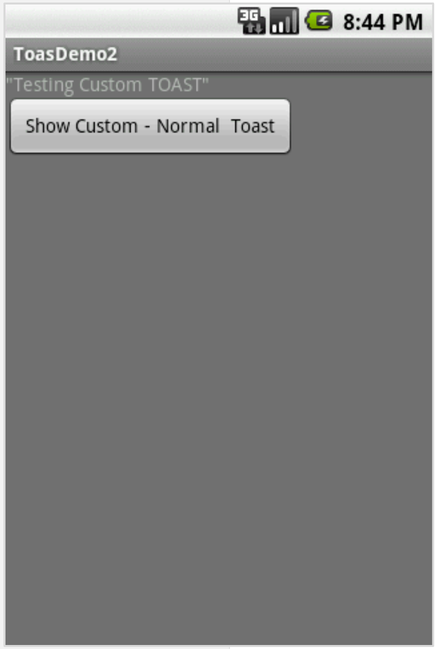
Example taken from:
http://hustleplay.wordpress.com/2009/07/23/replicating-default-android-toast/

The Toast View

Example: **Showing Fancy Toast views.**

Let's begin with the application's main layout.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#777"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text='"Testing Custom TOAST"'/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/btnShowToast"
    android:text=" Show Custom - Normal  Toast ">
</Button>
</LinearLayout>
```



The Toast View

Example: Showing Fancy Toast views.

Now we create our custom Toast layout (called: *my_toast_layout.xml*. It must contain a TextView called '*text*')

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/my_toast_layout_root"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dp"
    >
 <TextView
    android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="20dp"
    android:background="@drawable/my_border"
    >
  </TextView>

</LinearLayout>
```

Required TextView

Optional background

The Toast View

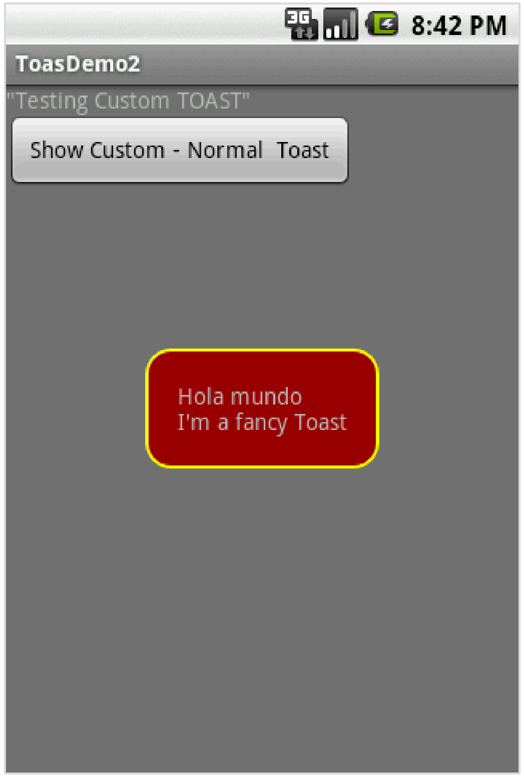
Example: Showing Fancy Toast views.
Finally we take care of the optional background element (*my_border.xml*). In this example we define a <shape> (but it could be any .png image). This XML (or image) is saved in the folder: /res/drawable

```
<?xml version="1.0" encoding="UTF-8" ?>
<shape
       xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="rectangle">
  <stroke  android:width="2dp" android:color="#ffffff00" />
  <solid   android:color="#ff990000" />
  <padding android:left="10dp"  android:top="4dp"
           android:right="10dp" android:bottom="4dp" />
  <corners android:radius="15dp" />
</shape>
```

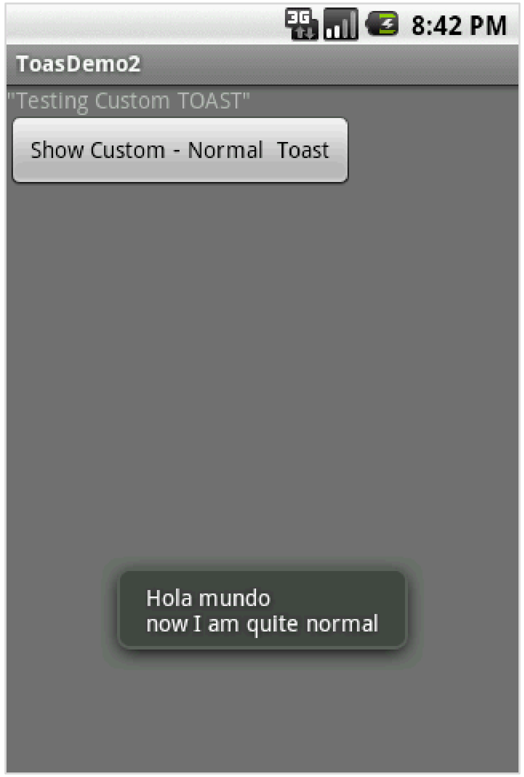
The Toast View

Example: **Showing Fancy Toast views.**

Testing the application



A Toast displayed with our custom layout



A Toast displayed using standard formatting

The Toast View

Example: Showing Fancy Toast views.

```
package cis493.dialogboxes;

import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class ToastDemo2 extends Activity {

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

The Toast View

Example: Showing Fancy Toast views.

```
Button btnShowToast = (Button) findViewById(R.id.btnShowToast);
btnShowToast.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        //custom made TOAST
        LayoutInflater inflater = getLayoutInflater();
        View layout = inflater.inflate(
                R.layout.my_toast_layout,
                (ViewGroup) findViewById(R.id.my_toast_layout_root));
        TextView text = (TextView) layout.findViewById(R.id.text);
        Toast toast = new Toast(getApplicationContext());
            text.setText("Hola mundo \nI'm a fancy Toast");
            toast.setGravity(Gravity.CENTER, 0, 0);
            toast.setDuration(Toast.LENGTH_SHORT);
            toast.setView(layout);
            toast.show();
        // normal TOAST
        Toast.makeText(getApplicationContext(),
                "Hola mundo \nnow I am quite normal",
                Toast.LENGTH_SHORT).show();
        }
      });
    }
}
```

The Toast View

Example: Showing Fancy Toast views.

As an aside:

Inflating a View

You may want occasionally to modify the way Android renders a particular view (perhaps a different color, style, or shape).

Once the Hierarchy View has been displayed, you can take any terminal node and **extend it** by inflating a custom '*view sub-tree*'. Also, by using layout inflation we may draw a new Hierarchy on top of the existing screen.

In our example, our customized rendition of a Toast box (including a colorful background) is defined in an XML file. Depicting the image of the custom Toast is accomplished by inflating the XML layout spec.

The Toast View

Example: Showing Fancy Toast views.

As an aside:

Inflating a View

Syntax

`public View inflate (int resource, ViewGroup root)`

Inflate a new view hierarchy from the specified xml resource.

Parameters

resource ID for an XML layout resource to load, root: optional view to be the parent of the generated hierarchy.

Returns

The root View of the inflated hierarchy. If root was supplied, this is the root View; otherwise it is the root of the inflated XML file.

```
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(
                 R.layout.my_toast_layout,
                 (ViewGroup) findViewById(R.id.my_toast_layout_root));
TextView text = (TextView) layout.findViewById(R.id.text);
```

Dialog Boxes