Lesson 9

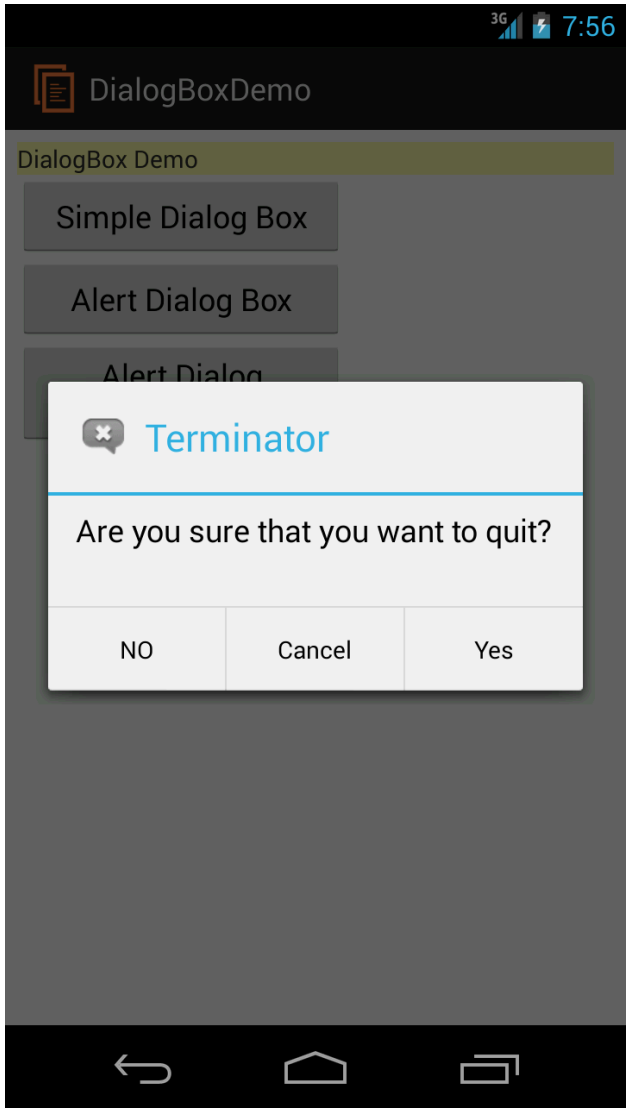
Dialog Boxes & Toast Widgets

Victor Matos
Cleveland State University

Android DialogBoxes

Android provides two primitive forms of dialog boxes:

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



Toasts are transitory boxes that –for a few seconds- flash a message on the screen, and then vanish without user intervention.

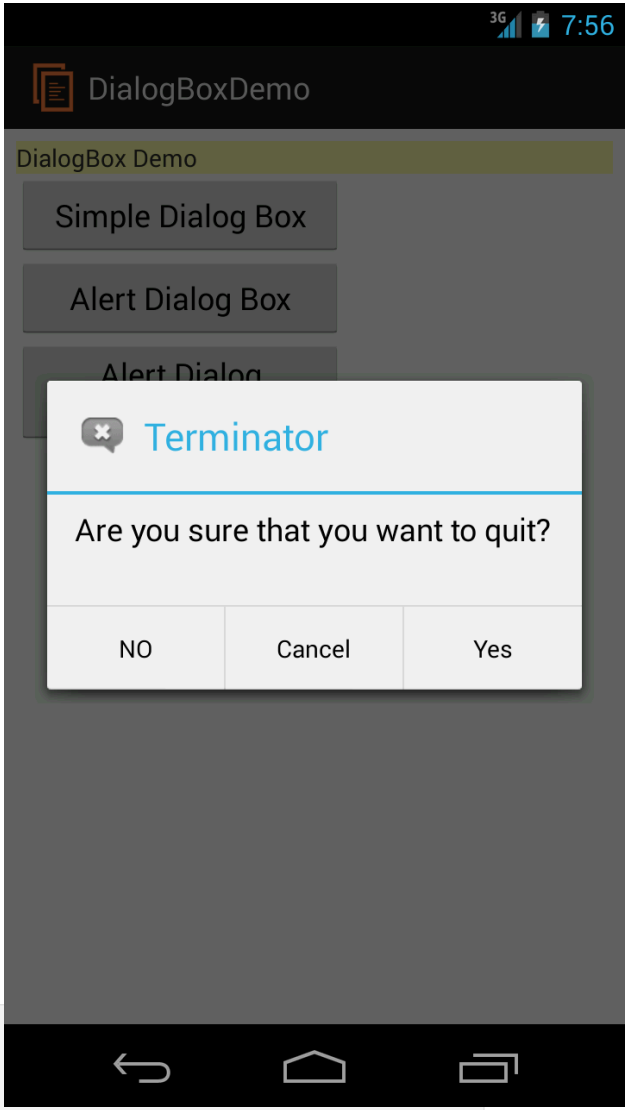
The AlertDialog Box

The **AlertDialog** is a message box that:

(1) Displays as a small floating window on top of the (obscured) current UI.

(2) The dialog window presents a message to the user as well as three optional buttons.

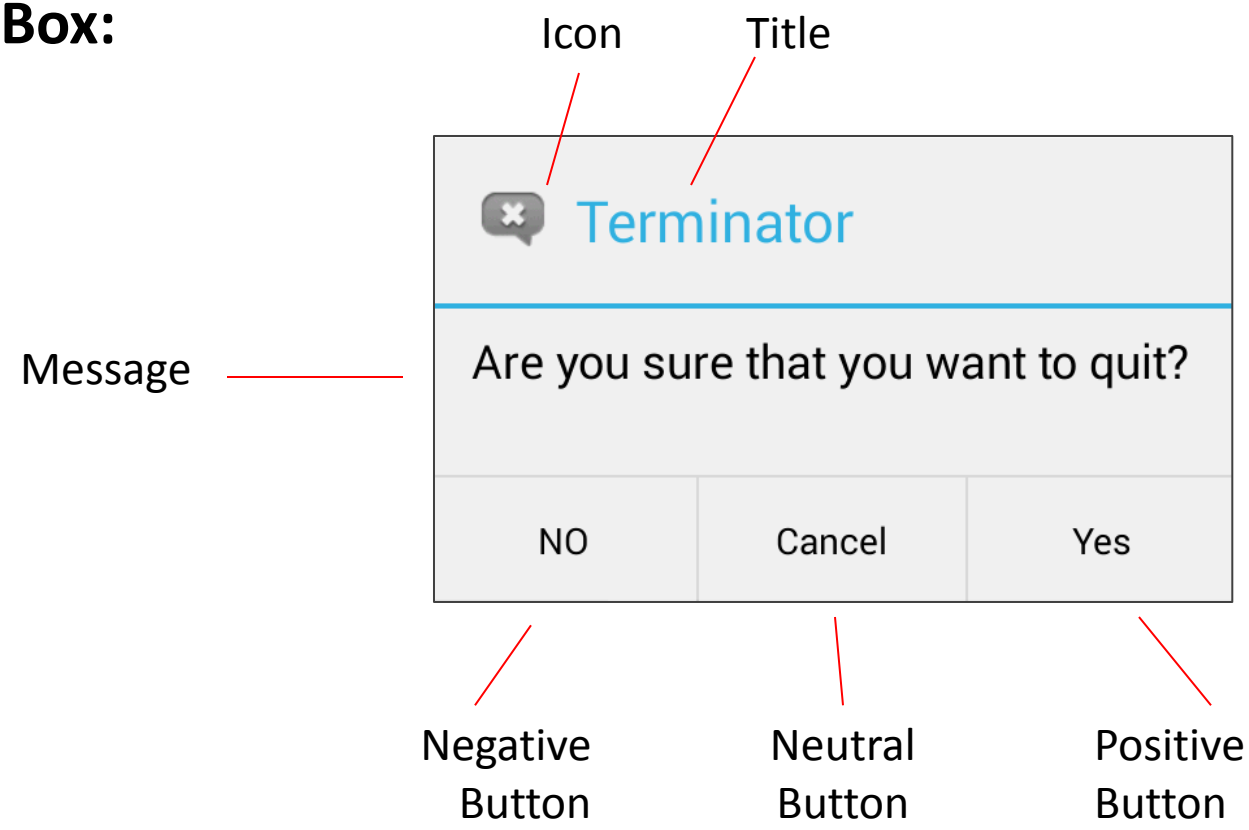
(3) The box is dismissed by either clicking on the exposed buttons or touching any portion of the UI outside the borders of the DialogBox.



Note:
Android's DialogBoxes are NOT modal views!
A fully *modal* view remains on the screen waiting for user's input while t*he rest of the application is on hold* (which is *not* the case of Android's DialogBoxes). A modal view (including Android's) has to be dismissed by an explicit user's action.

The AlertDialog

Dissecting an AlertDialog Box:



The image shown here uses:
Theme_Holo_Light_Dialog and
STYLE_NORMAL

AlertDialog

Example 1. AlertDialog Boxes

In this example the application's UI shows three buttons. When you click on them a different type of AlertDialog box is shown.

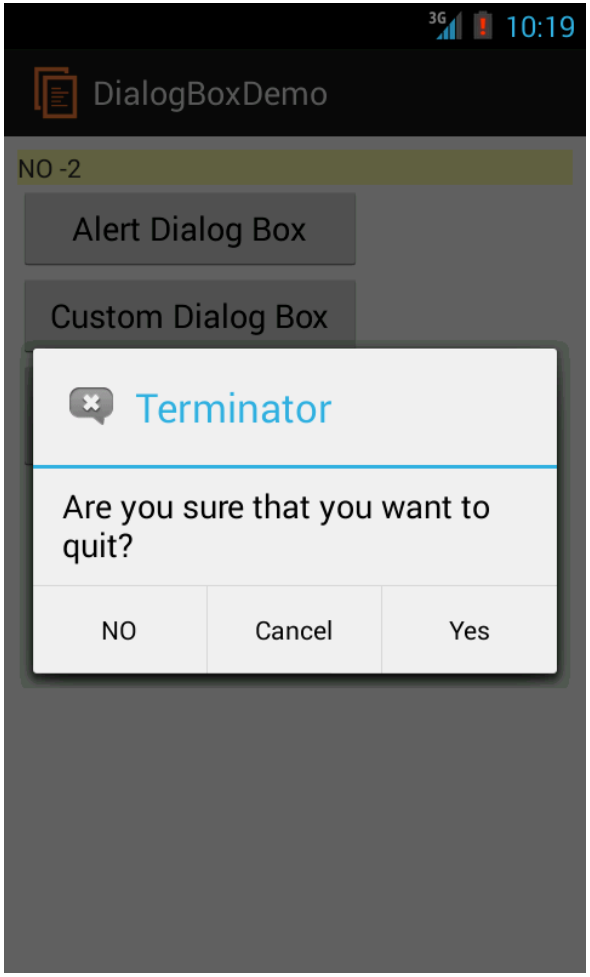
1. The first to be shown is a simple **AlertDialog** box with a message and buttons.

2. The second option is a **custom** DialogBox on which the user could type in a piece of data.

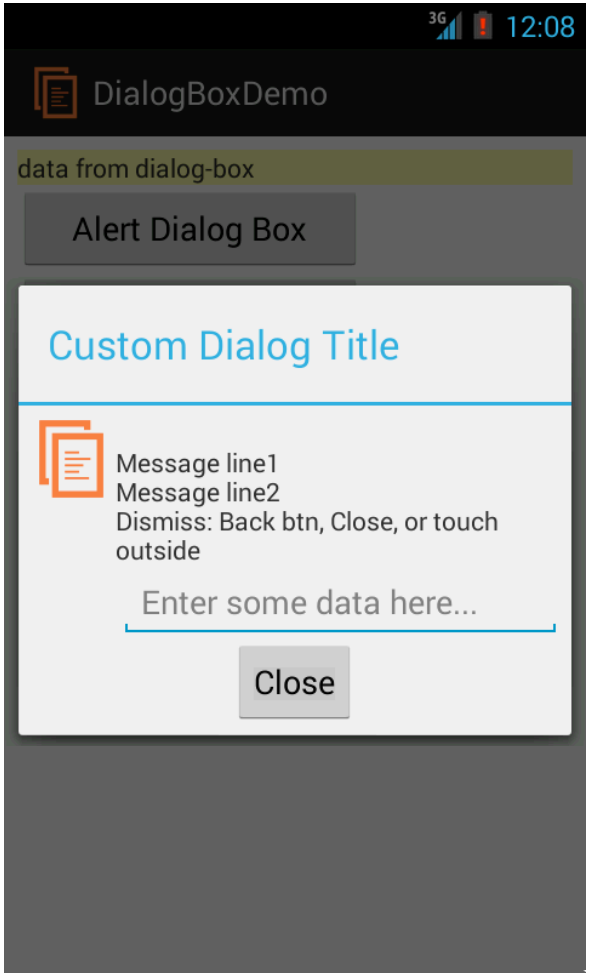
3. The last option shows a **DialogFragment** interacting with the main activity

AlertDialog

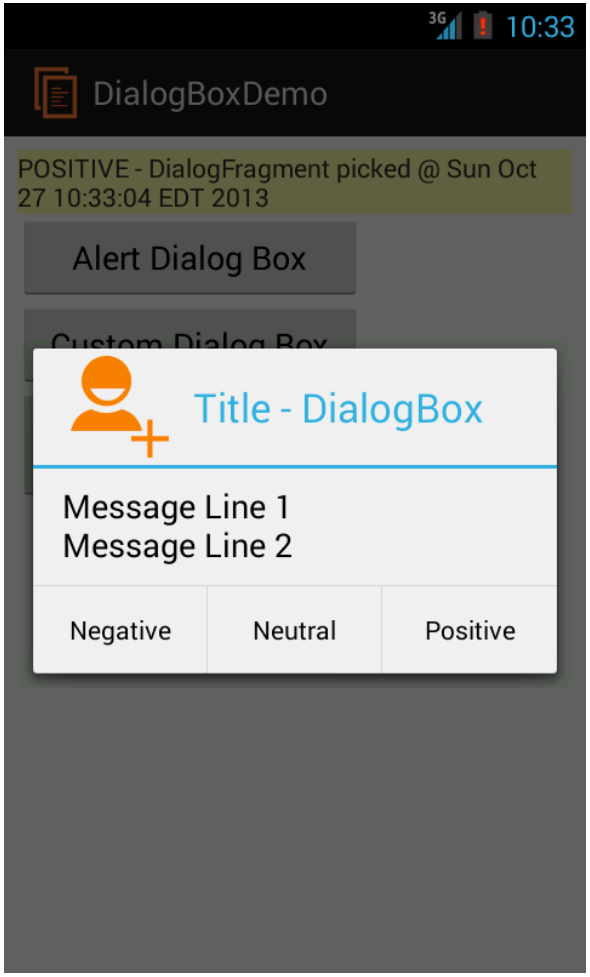
Example 1. AlertDialog Boxes



A simple **AlertDialog** offering three choices.



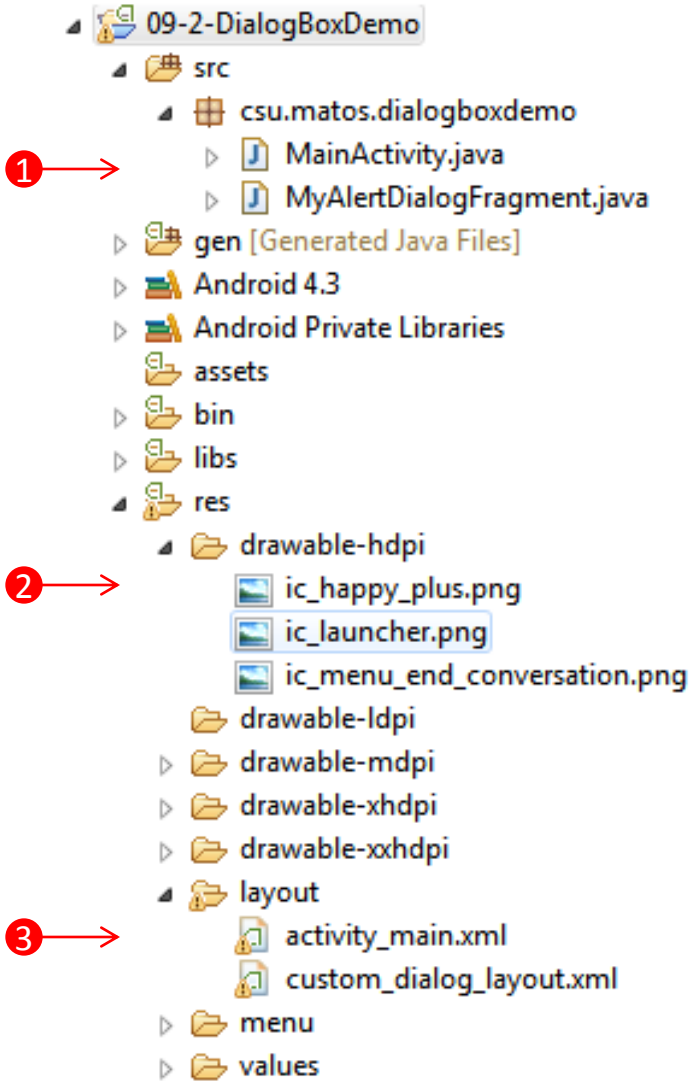
A **custom** AlertDialog allowing data to be typed.



A **DialogFragment** exposing three buttons.

AlertDialog

Example 1. App Structure



1. MainActivity shows main GUI and provides a frame for the DialogFragment to be displayed.

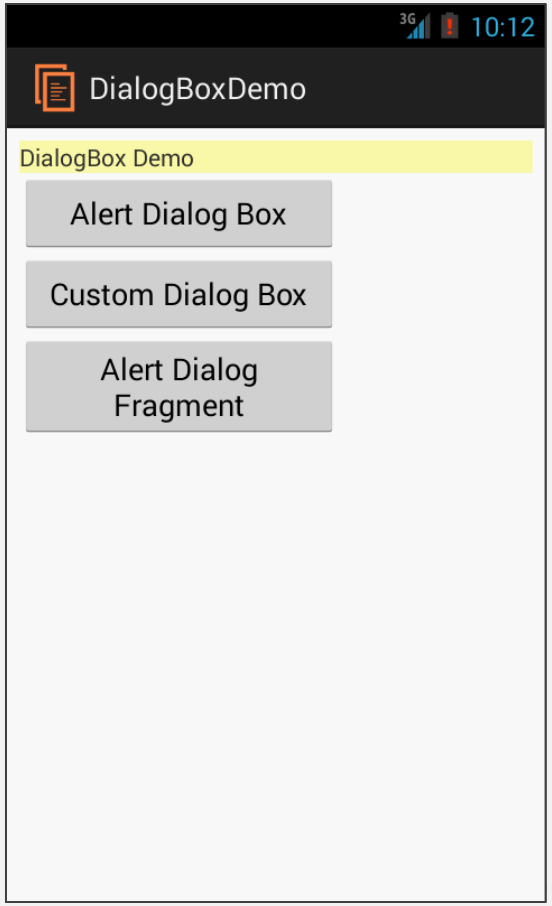
2. You want to enhance the appearance of dialog-boxes by adding meaningful icons. More details and tools at **Android Asset studio** (http://j.mp/androidassetstudio)

3. Add your XML design indicating the way your custom AlertDialog looks like.

AlertDialog

Example 1. XML Layout – activity_main.xml

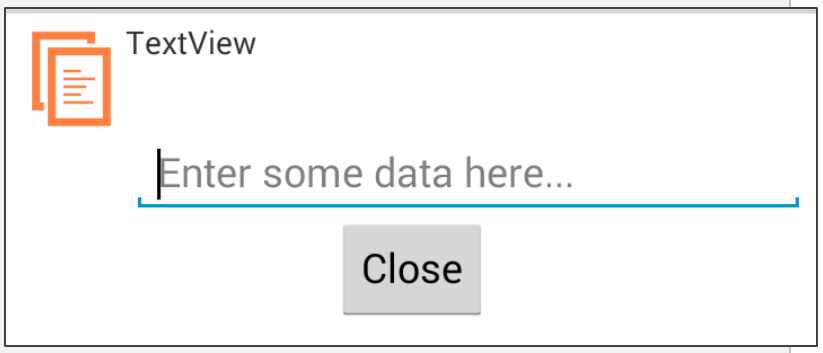
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"     android:layout_height="match_parent"
    android:orientation="vertical"          android:padding="7dp"  >

    <TextView
        android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#55ffff00"
        android:text="DialogBox Demo" />

    <Button
        android:id="@+id/btn_alert_dialog1"
        android:layout_width="190dp"
        android:layout_height="wrap_content"
        android:text="Alert Dialog Box" />

    <Button
        android:id="@+id/btn_custom_dialog"
        android:layout_width="190dp"
        android:layout_height="wrap_content"
        android:text="Custom Dialog Box" />

    <Button
        android:id="@+id/btn_alert_dialog2"
        android:layout_width="190dp"
        android:layout_height="wrap_content"
        android:text="Alert Dialog Fragment" />
</LinearLayout>
```



AlertDialog

Example 1. XML Layout – custom_dialog_layout.xml

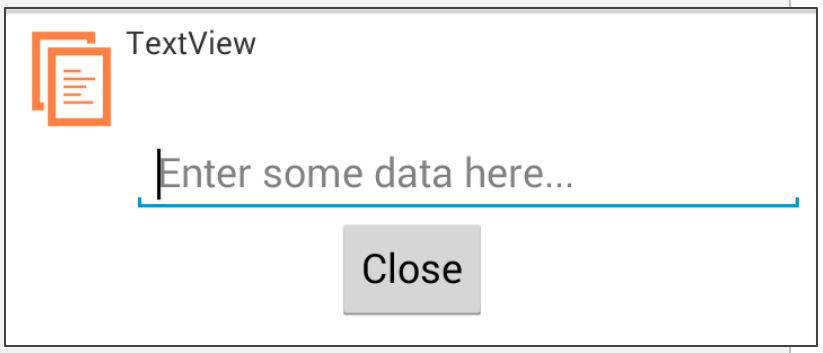
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="5dp" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <ImageView
            android:id="@+id/imageView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_launcher" />

        <TextView
            android:id="@+id/sd_textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="TextView" />
    </LinearLayout>
```



AlertDialog

Example 1. XML Layout – custom_dialog_layout.xml cont. 1

```
    <EditText
        android:id="@+id/sd_editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="50dp"
        android:ems="15"
        android:hint="Enter some data here..." >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/sd_btnClose"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Close" />

</LinearLayout>
```



AlertDialog

Example 1. MainActivity.java

```
// example adapted from:
// http://developer.android.com/reference/android/app/DialogFragment.html

public class MainActivity extends Activity implements OnClickListener {
   TextView txtMsg;
   Button btnCustomDialog;
   Button btnAlertDialog;
   Button btnDialogFragment;
   Context activityContext;
   String msg = "";

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      activityContext = this;

      txtMsg = (TextView) findViewById(R.id.txtMsg);
      btnAlertDialog = (Button) findViewById(R.id.btn_alert_dialog1);
      btnCustomDialog = (Button) findViewById(R.id.btn_custom_dialog);
      btnDialogFragment = (Button) findViewById(R.id.btn_alert_dialog2);

      btnCustomDialog.setOnClickListener(this);
      btnAlertDialog.setOnClickListener(this);
      btnDialogFragment.setOnClickListener(this);

   }
```

1

AlertDialog

Example 1. MainActivity.java cont. 1

```
@Override
public void onClick(View v) {
   if (v.getId() == btnAlertDialog.getId()) {
      showMyAlertDialog(this);
   }
   if (v.getId() == btnCustomDialog.getId()) {
      showCustomDialogBox();
   }
   if (v.getId() == btnDialogFragment.getId()) {
      showMyAlertDialogFragment(this);
   }
}// onClick

private void showMyAlertDialog(MainActivity mainActivity) {
   new AlertDialog.Builder(mainActivity)
         .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) {
                     // actions serving "YES" button go here
                     msg = "YES " + Integer.toString(whichButton);
                     txtMsg.setText(msg);
                  }
               })// setPositiveButton
```

2 → `if (v.getId() == btnAlertDialog.getId()) {`

3 → `.setTitle("Terminator")`

AlertDialog

Example 1. MainActivity.java cont. 2

```
            .setNeutralButton("Cancel",
                  new DialogInterface.OnClickListener() {
                     public void onClick(DialogInterface dialog,
                           int whichButton) {
                        // actions serving "CANCEL" button go here
                        msg = "CANCEL " + Integer.toString(whichButton);
                        txtMsg.setText(msg);
                     }// OnClick
                  })// setNeutralButton

            .setNegativeButton("NO", new DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int whichButton) {
                  // actions serving "NO" button go here
                  msg = "NO " + Integer.toString(whichButton);
                  txtMsg.setText(msg);
               }
            })// setNegativeButton

            .create()
            .show();

      }// showMyAlertDialog
```

AlertDialog

Example 1. MainActivity.java cont. 3

```
private void showCustomDialogBox() {

   final Dialog customDialog = new Dialog(activityContext);
   customDialog.setTitle("Custom Dialog Title");

   // match customDialog with custom dialog layout
4  customDialog.setContentView(R.layout.custom_dialog_layout);

   ((TextView) customDialog.findViewById(R.id.sd_textView1))
               .setText("\nMessage line1\nMessage line2\n"
               +"Dismiss: Back btn, Close, or touch outside");

   final EditText sd_txtInputData = (EditText) customDialog
                                          .findViewById(R.id.sd_editText1);

   ((Button) customDialog.findViewById(R.id.sd_btnClose))
         .setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
               txtMsg.setText(sd_txtInputData.getText().toString());
               customDialog.dismiss();
            }
         });

   customDialog.show();

}
```

AlertDialog

Example 1. MainActivity.java cont. 4

```
    private void showMyAlertDialogFragment(MainActivity mainActivity) {

5 →    DialogFragment dialogFragment = MyAlertDialogFragment
                                       .newInstance(R.string.title);

       dialogFragment.show(getFragmentManager(), "TAG_MYDIALOGFRAGMENT1");

    }

6 →  public void doPositiveClick(Date time) {
       txtMsg.setText("POSITIVE - DialogFragment picked @ " + time);

    }

    public void doNegativeClick(Date time) {
       txtMsg.setText("NEGATIVE - DialogFragment picked @ " + time);

    }

    public void doNeutralClick(Date time) {
       txtMsg.setText("NEUTRAL - DialogFragment picked @ " + time);

    }

}
```

AlertDialog

Example 1. MainActivity.java

Comments

1. The main UI shows three buttons and a TextView on which data coming from the executing dialog-boxes is to be written.
2. When a button is clicked the proper DialogBox is shown.
3. **showMyAlertDialog** uses a builder class to create a new AlertDialog adding to it a title, icon, message and three action buttons. Each action button has an onClick() method responsible for services to be rendered on behalf of the selection. We update the main UI's top TextView with the button's id.
4. The **custom** dialog-box is *personalized* when the `.setContentView(R.layout.custom_dialog_layout)` method is executed. Later, its "Close" button is given a listener, so the data entered in the dialog's EditText view could be sent to the UI's top TextView and, the box is finally dismissed.
5. A **DialogFragment** is instanciated. It's title is supplied as an argument to be 'bundled' when the fragment is created. Later the dialog will be show on top of the containing activity.
6. **Callback** methods (doPositive(), doNegative()…) are provided to empower the DialogFragment to pass data (a timestamp) back to the main activity.

AlertDialog

Example 1. MyAlertDialogFragment.java

```
public class MyAlertDialogFragment extends DialogFragment {

  public static MyAlertDialogFragment newInstance(int title) {

    MyAlertDialogFragment frag = new MyAlertDialogFragment();

    Bundle args = new Bundle();
    args.putInt("title", title);
    args.putString("message", "Message Line 1\nMessage Line 2");
    args.putInt("icon", R.drawable.ic_happy_plus);

    frag.setArguments(args);
    return frag;
  }

  @Override
  public Dialog onCreateDialog(Bundle savedInstanceState) {

    int title = getArguments().getInt("title");
    int icon = getArguments().getInt("icon");
    String message = getArguments().getString("message");

    return new AlertDialog.Builder(getActivity())
        .setIcon(icon)
        .setTitle(title)
        .setMessage(message)
```

AlertDialog

Example 1. MyAlertDialogFragment.java cont. 1

```
        .setPositiveButton("Positive",
            new DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog,
                  int whichButton) {
                ((MainActivity) getActivity())
                    .doPositiveClick(new Date());
              }
            })
        .setNegativeButton("Negative",
            new DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog,
                  int whichButton) {
                ((MainActivity) getActivity())
                    .doNegativeClick(new Date());
              }
            })
        .setNeutralButton("Neutral",
            new DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog,
                  int whichButton) {
                ((MainActivity) getActivity())
                    .doNeutralClick(new Date());
              }
            }).create();
  }
}
```

3

AlertDialog

Example 1. MyAlertDialogFragment.java

Comments

1. The class extends **DialogFragment**. It's *instantiator* accepts a title, message and icon arguments. As customary with fragments, the arguments are placed into a single bundle which is then associated to the fragment.

2. The **onCreateDialog** method extracts the arguments (title, icon, and message) from the DialogFragment's bundle. A common AlertDialog builder is called to prepare the dialog box using the supplied arguments.

3. Three option buttons are added to the DialogFragment. Each has a listener that when activated, makes its onClick method interact with a callback method in the MainActivity. To illustrate that data from the fragment could be passed from the dialog-box, a timestamp is supplied to the callbacks.

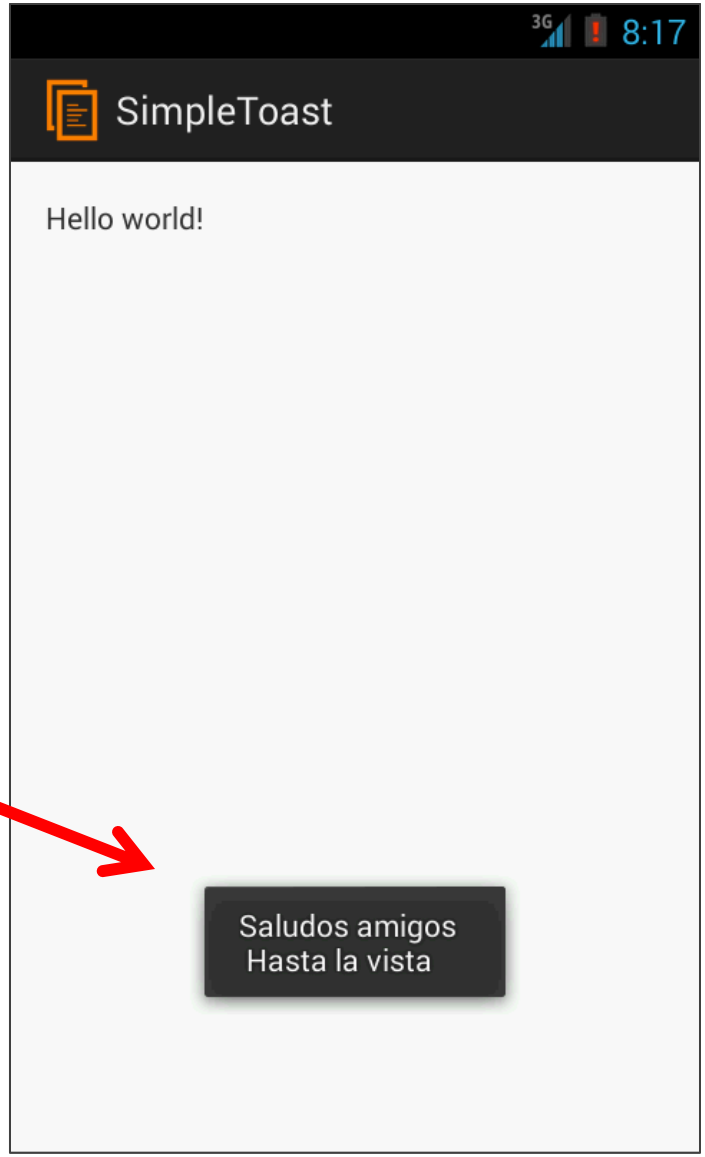
The Toast Widget

Toasts are very simple one-way message boxes.

Typically they are used in situations in which a **brief message** should be flashed to the user.

A toast is shown as a semi-opaque floating view over the application ‘s UI. It’s lifetime is between 2-4 sec.

Notoriously, *Toasts never receive focus !*



The Toast Widget

Example 2. Toast's Syntax

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

Context: A reference to the view's environment (where am I, what is around me…)

Message: The message you want to show

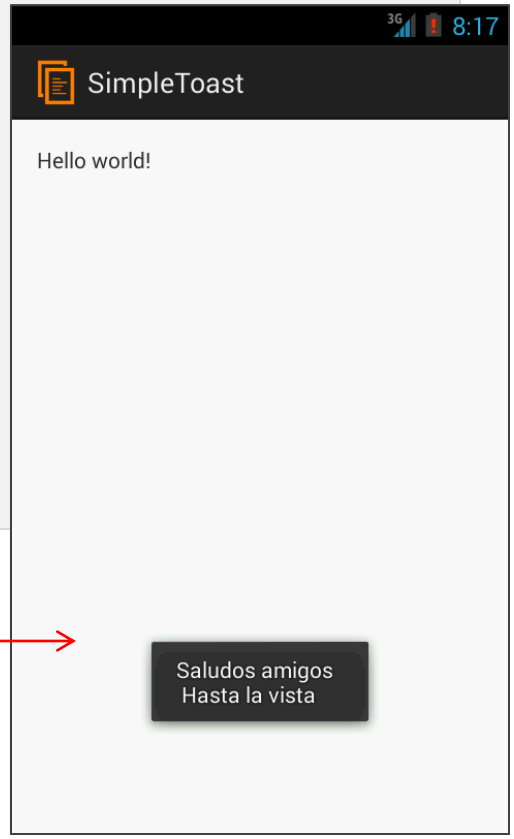
Duration: Toast.*LENGTH_SHORT* (0) about 2 sec
Toast.*LENGTH_* LONG (1) about 3.5 sec

The Toast class has only a few methods including: *makeText*, *show, setGravity,* and *setMargin.*

The Toast Widget

Example 2. A Simple Toast

```
public class MainActivity extends Activity {

    @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();
    }
}
```



In this simple application, passing the **context** variable could be done using: `getApplicationContext()`, `MainActivity.this`, or simply using **`this`**.

The Toast Widget

Example 3. Re-positioning 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)

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

The Toast Widget

Example 3. Re-positioning a Toast View

Method 1

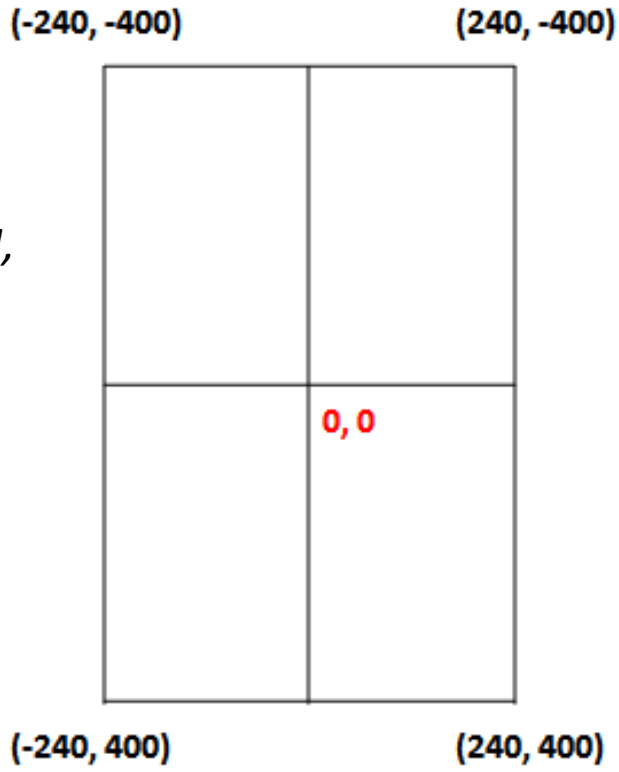
```
void setGravity (int gravity, int xOffset, int yOffset)
```

(Assume the phone has a **480x800** screen density)

gravity: Overall placement. Typical values include: *Gravity.CENTER, Gravity.TOP, Gravity.BOTTOM, (see Apendix B)*

xOffset: The *xOffset* range is -240,…,0,…240 left, center, right

yOffset: The *yOffset* range is: -400,…,0,…400 top, center, bottom



The Toast Widget

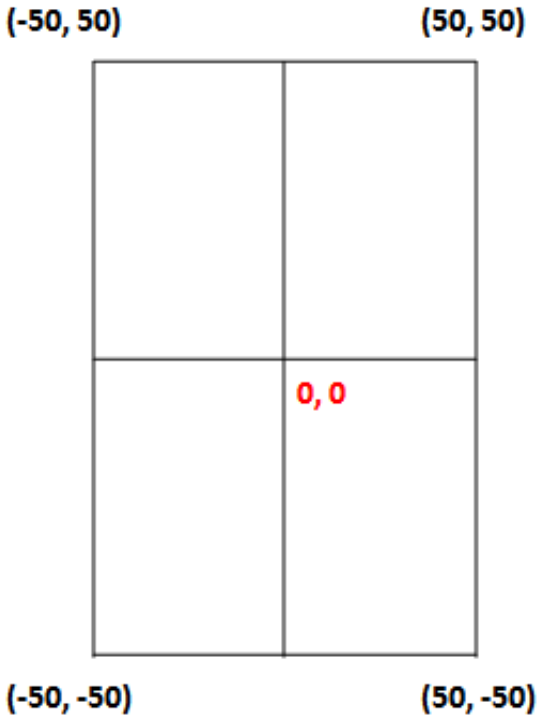
Example 3. Re-positioning a Toast View

Method 2

- The (0,0) point – *Center of the screen* –occurs where horizontal and vertical center lines cross each other.
- There is 50% of the screen to each side of that center point
- Margins are expressed as a percent value between: -50,…, 0, …, 50.

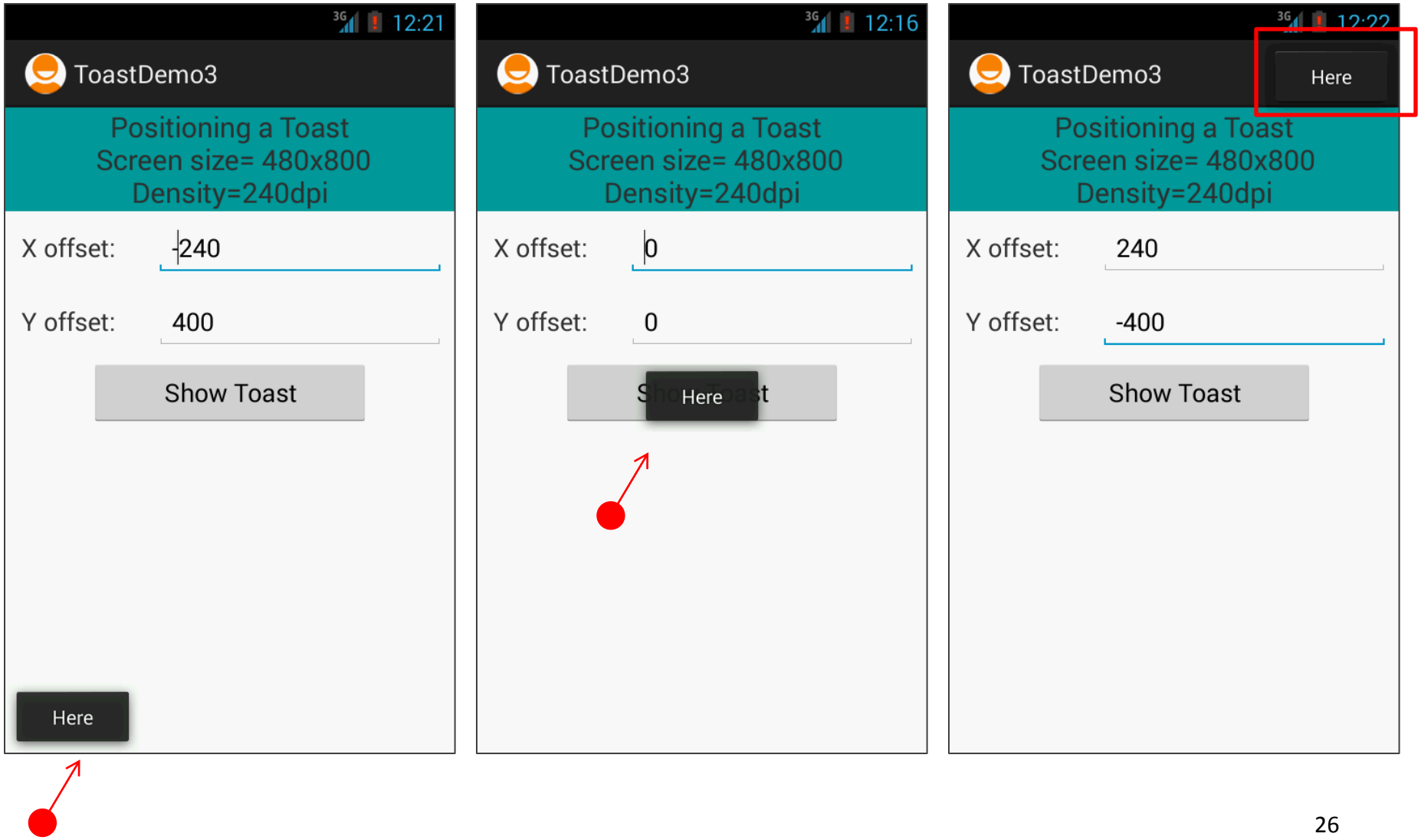
```
void setMargin (float horizontalMargin,
                float verticalMargin)
```

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



The Toast Widget

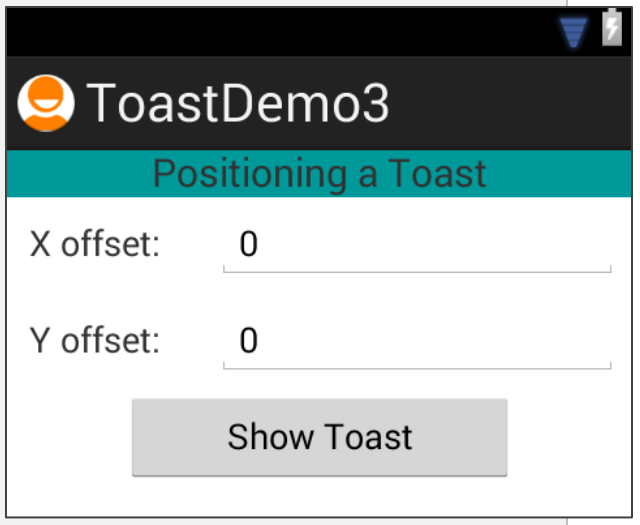
Example 3. Re-positioning a Toast View



The Toast Widget

Example 3. XML Layout: activity_main.xml

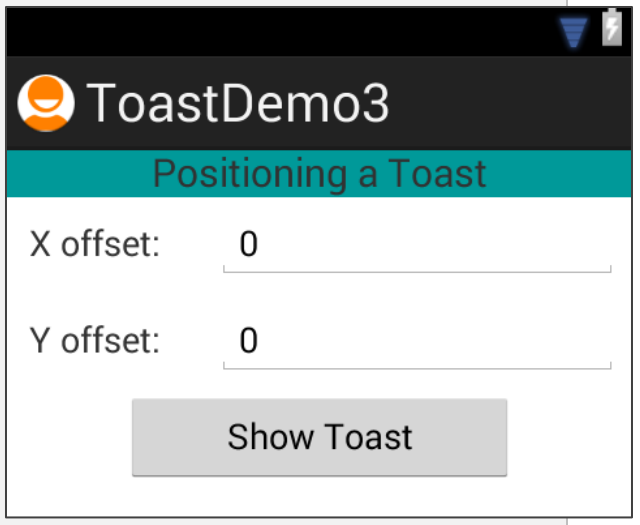
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/txtCaption"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff009999"
        android:gravity="center"
        android:text="Positioning a Toast"
        android:textSize="20sp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp" >

        <TextView
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:text=" X offset: "
            android:textSize="18sp" />

 <EditText
            android:id="@+id/txtXCoordinate"
```



The Toast Widget

Example 3. XML Layout: activity_main.xml cont. 1

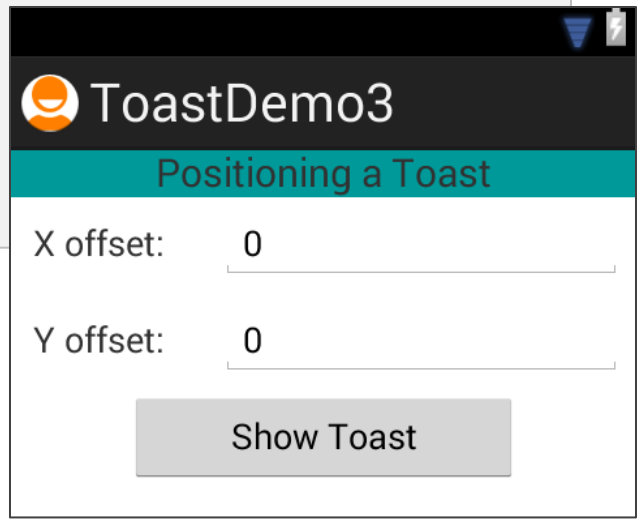
```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:inputType="numberSigned"
            android:text="0"
            android:textSize="18sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp" >

        <TextView
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:text=" Y offset: "
            android:textSize="18sp" />

        <EditText
            android:id="@+id/txtYCoordinate"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:inputType="numberSigned"
            android:text="0"
            android:textSize="18sp" />
    </LinearLayout>
```



The Toast Widget

Example 3. XML Layout: activity_main.xml cont. 2

```
    <Button
        android:id="@+id/btnShowToast"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text=" Show Toast " >
    </Button>

</LinearLayout>
```



The Toast Widget

Example 3. MainActivity: ToastDemo3.java

```
public class ToastDemo3 extends Activity {
   EditText txtXCoordinate;
   EditText txtYCoordinate;
   TextView txtCaption;

   Button btnShowToast;

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

1 →   // bind GUI and Java controls
      txtCaption = (TextView) findViewById(R.id.txtCaption);
      txtXCoordinate = (EditText) findViewById(R.id.txtXCoordinate);
      txtYCoordinate = (EditText) findViewById(R.id.txtYCoordinate);
      btnShowToast = (Button) findViewById(R.id.btnShowToast);

2 →   // find screen-size and density(dpi)
      int dpi  = Resources.getSystem().getDisplayMetrics().densityDpi;
      int width=  Resources.getSystem().getDisplayMetrics().widthPixels;
      int height = Resources.getSystem().getDisplayMetrics().heightPixels;
      txtCaption.append("\n Screen size= " + width + "x" + height
                  +" Density=" + dpi + "dpi");
```

The Toast Widget

Example 3. MainActivity: ToastDemo3.java cont. 1

```
      // show toast centered around selected X,Y coordinates
      btnShowToast.setOnClickListener(new OnClickListener() {
         @Override
         public void onClick(View v) {
            try {
               Toast myToast = Toast.makeText(getApplicationContext(),
                     "Here", Toast.LENGTH_LONG);

               myToast.setGravity(
                     Gravity.CENTER,
                     Integer.valueOf(txtXCoordinate.getText().toString()),
                     Integer.valueOf(txtYCoordinate.getText().toString()));

               myToast.show();

            } catch (Exception e) {
               Toast.makeText(getApplicationContext(), e.getMessage(),
                     Toast.LENGTH_LONG).show();
            }
         }
      });


   }// onCreate

}// class
```

3

The Toast Widget

Example 3. MainActivity: ToastDemo3.java

Comments

1. Plumbing. GUI objects are bound to their corresponding Java controls. When the button is clicked a Toast is to be shown.

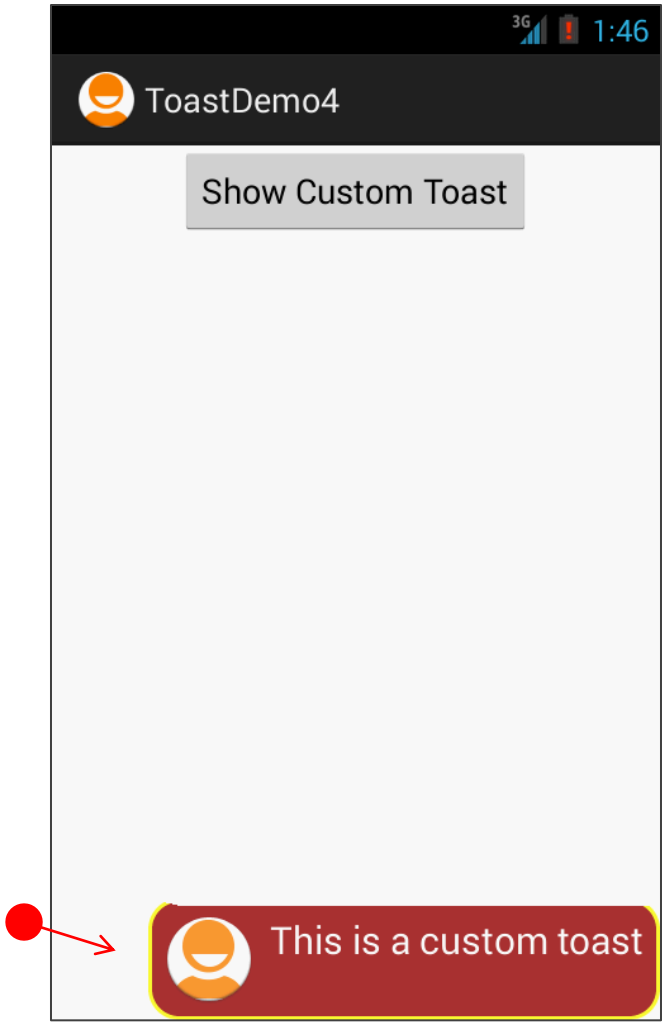
2. The call `Resources.getSystem().getDisplayMetrics()` is used to determine the screen size (Height, Width) in pixels, as well as its density in dip units.

3. An instance of a Toast is created with the *makeText* method. The call to `setGravity` is used to indicate the (X,Y) coordinates where the toast message is to be displayed. X and Y refer to the actual horizontal/vertical pixels of a device's screen.

The Toast Widget

Example 4. A Custom-Made Toast View

Toasts could be modified to display a custom combination of color, shape, text, image, and background.



Steps

To create a custom Toast do this:

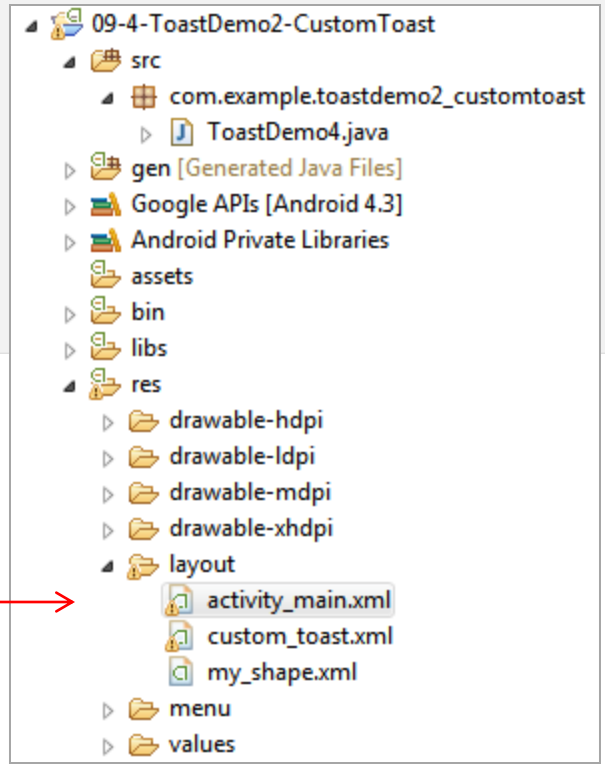
1. Define the XML layout you wish to apply to the custom toasts.
2. In addition to a TextView where the toast's message will be shown, you may add other UI elements such as an image, background, shape, etc.
3. Inflate the XML layout. Attach the new view to the toast using the `setView()` method.

Example based on:
http://hustleplay.wordpress.com/2009/07/23/replicating-default-android-toast/
http://developer.android.com/guide/topics/ui/notifiers/toasts.html

The Toast Widget

Example 4. XML Layout - activity_main.xml

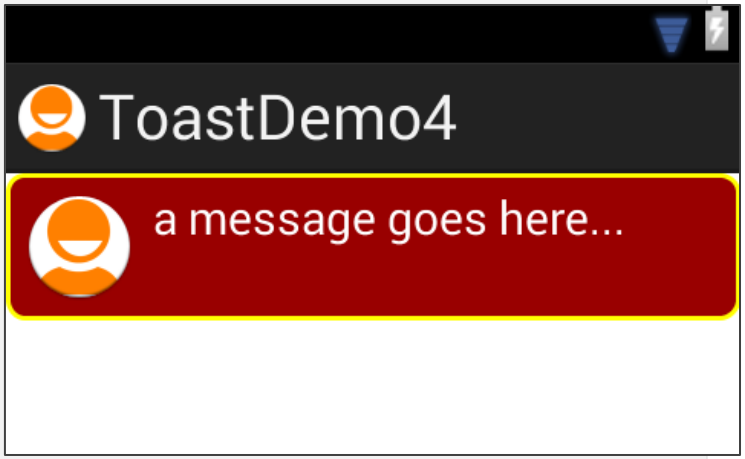
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="showCustomToast"
        android:text="Show Custom Toast"
        android:layout_gravity="center"
        tools:context=".ToastDemo4" />

</LinearLayout>
```



The Toast Widget

Example 4. XML Layout - custom_toast.xml

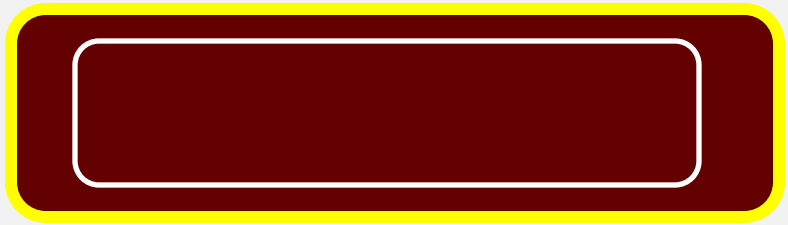
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@layout/my_shape"
    android:orientation="horizontal"
    android:padding="8dp" >

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="8dp"
        android:src="@drawable/ic_launcher" />

    <TextView
        android:id="@+id/toast_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="a message goes here..."
        android:textColor="#ffffffff"
        android:textSize="20sp" />

</LinearLayout>
```



The Toast Widget

Example 4. XML Layout - my_shape.xml

```
<?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:bottom="4dp"
        android:left="10dp"
        android:right="10dp"
        android:top="4dp" />

    <corners android:radius="15dp" />

</shape>
```

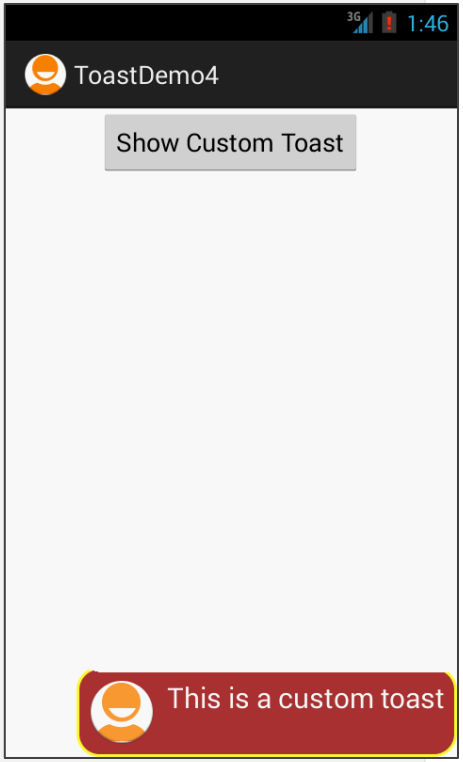


Note: A basic shape is a drawable such as a rectangle or oval. Defining attributes are stroke(border) , solid(interior part of the shape), corners, padding, margins, etc. Save this file in the **res/layout** folder. For more information see **Appendix A**.

The Toast Widget

Example 4. MainActivity - ToastDemo4.java

```
public class ToastDemo4 extends Activity {

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

    }//onCreate
    public void showCustomToast(View v){
      // ////////////////////////////////////////////////
      // this fragment creates a custom Toast showing
      // image + text + shaped_background
      // triggered by XML button's android:onClick=...

      Toast customToast = makeCustomToast(this);

      customToast.show();

    }
```



The Toast Widget

Example 4. MainActivity - ToastDemo4.java cont. 1

```
   protected Toast makeCustomToast(Context context) {
      // Reference:
      // http://developer.android.com/guide/topics/ui/notifiers/toasts.html

      LayoutInflater inflater = getLayoutInflater();

      View layout = inflater.inflate( R.layout.custom_toast, null);

      TextView text = (TextView) layout.findViewById(R.id.toast_text);
      text.setText("This is a custom toast");

      Toast toast = new Toast(context);
      toast.setMargin(50,-50); //lower-right corner
      toast.setDuration(Toast.LENGTH_LONG);

      toast.setView(layout);

      return toast;

   }//makeCustomToast

}//ToastDemo2
```

1 → 2 → 3 →

The Toast Widget

Example 4. MainActivity - ToastDemo4.java

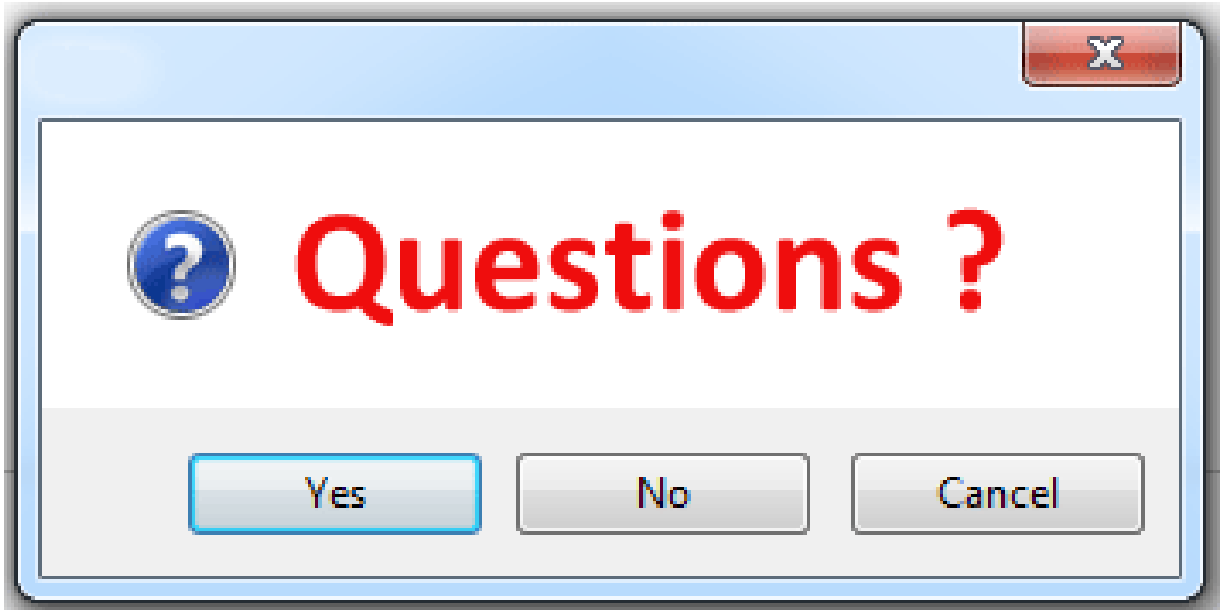
Comments

1. After the custom toast layout is inflated, you gain control to its TextView in which the user's message will be held.

2. The toast is positioned using the `setMargin()` method to the lower right corner of the screen (50, -50)

3. The inflated view is attached to the newly created Toast object using the `.setView()` method.

Dialog Boxes & Toast Widget



Dialog Boxes

Appendix A.

Shape Drawable

Is an XML file that defines a geometric figure, including colors and gradients.

Some basic shapes are: rectangle, oval, ring, line

References:

http://developer.android.com/reference/android/graphics/drawable/shapes/Shape.html

http://developer.android.com/guide/topics/resources/drawable-resource.html#Shape

http://developer.android.com/reference/android/graphics/drawable/ShapeDrawable.html

```
<?xml version="1.0" encoding="utf-8"?>
<Shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape=["rectangle" | "oval" | "line" | "ring"] >
    <corners
        android:radius="integer"
        android:topLeftRadius="integer"
        android:topRightRadius="integer"
        android:bottomLeftRadius="integer"
        android:bottomRightRadius="integer" />
    <gradient
        android:angle="integer"
        android:centerX="integer"
        android:centerY="integer"
        android:centerColor="integer"
        android:endColor="color"
        android:gradientRadius="integer"
        android:startColor="color"
        android:type=["linear" | "radial" | "sweep"]
        android:useLevel=["true" | "false"] />
    <padding
        android:left="integer"
        android:top="integer"
        android:right="integer"
        android:bottom="integer" />
    <size
        android:width="integer"
        android:height="integer" />
    <solid
        android:color="color" />
    <stroke
        android:width="integer"
        android:color="color"
        android:dashWidth="integer"
        android:dashGap="integer" />
</shape>
```