5

UI 进阶

5-1　Menus

大部分 Android 设备上都有一个 Menu 按钮，方便用户与设备互动并做相应设置。如果在设备的首页（Home）按下 Menu 按钮，会弹出如图 5-1 所示的选项菜单，这也是一般人最常用到的 Menu 功能。

图 5-1

实际上 Android 提供了 3 种 Menu 的功能：

（1）Options Menu（选项菜单）：图 5-1 即属于 Options Menu。

（2）Context Menu（上下文菜单）：当用户在指定组件上长按（long-press），就会弹出 Context Menu。

（3）Submenu（子菜单）：当用户单击 Options Menu 或 Context Menu 上面的菜单项时可以显示第二层菜单，第二层菜单即属于 Submenu。

5–1–1　Options Menu

虽然可以在程序代码内直接利用 Menu 类来建立 Options Menu，但是除非特殊情况，否则一般

建议通过 XML 文件来定义 Options Menu。要产生一个使用者可以操作的 Options Menu，需要下列 2 个步骤：

Step 1 使用 XML 文件建立 Menu 组件：如上所述，一般会使用 XML 文件来建立 Menu 组件相关内容，所以下面范例将会说明如何通过 XML 文件来建立 Menu 组件。

Step 2 改写与 Options Menu 有关的方法：虽然已经使用 XML 文件建立 Menu 组件，但界面仍不会显示该 Menu 组件，必须改写与 Options Menu 有关的方法才能真正在界面上显示 Menu。

使用 XML 文件建立 Menu 组件

将定义 Menu 组件的 XML 文件放在 res/menu 目录中，在该文件内会使用下列 2 个重要标签：

- <menu> 标签：建立 Menu 组件，该组件专门用来存放 MenuItem 组件（菜单项），图 5-1 中弹出来的选项菜单就是利用 <menu> 标签建立。
- <item> 标签：建立 MenuItem 组件，也就是建立 Menu 的菜单项；图 5-1 中的“添加”、“搜索”等菜单项就是利用 <item>标签标建立的菜单项。而 <item> 菜单项则有 3 个重要属性：
  - android:id——MenuItem 组件的资源 ID，方便在程序代码中存取指定的 MenuItem 组件。
  - android:icon——菜单项上显示的图标。
  - android:title——菜单项上显示的文字。

范例 OptionsMenuEx

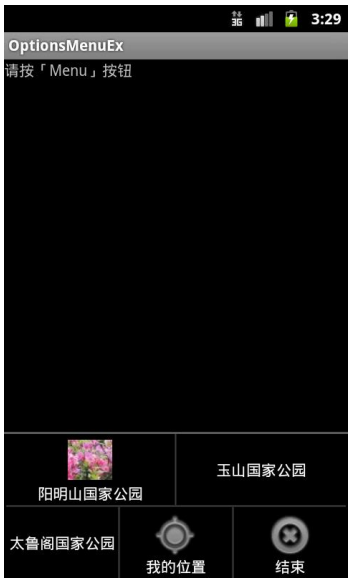

图 5-2

范例说明：

- 界面显示：请按 Menu 按钮。
- 按下实体 Menu 按钮后会弹出 5 个菜单项，其中“阳明山国家公园”、“我的位置”、“结束”等 3 个菜单项有图标。
- 单击“结束”菜单项会结束该程序；单击其余 4 个菜单项会以 Toast 信息方式显示菜单项上面的文字。

OptionsMenuEx/res/menu/mymenu.xml

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <menu xmlns:android="http://schemas.android.com/apk/res/android">
3.     <item android:id="@+id/yangmingshan" android:title="@string/yangmingshan"
4.        android:icon="@drawable/yangmingshan" />
5.     <item android:id="@+id/yushan" android:title="@string/yushan" />
6.     <item android:id="@+id/taroko" android:title="@string/taroko" />
7.     <item android:id="@+id/myloc" android:title="@string/myloc"
8.        android:icon="@drawable/ic_menu_mylocation" />
9.     <item android:id="@+id/exit" android:title="@string/exit"
10.       android:icon="@drawable/ic_menu_close_clear_cancel" />
11. </menu>
```

mymenu.xml 文件主要设置 Menu 的 MenuItem 组件（菜单项），3、5、6、7、9 行共有 5 个 <item> 标签，代表 Menu 组件有 5 个 MenuItem 组件。也可以在程序代码内调用 Menu 的 add() 来动态新增 MenuItem 组件。

3 行：

- android:id：值为“"@+id/yangmingshan"”，代表该菜单项的资源 ID 为 yangmingshan，方便在程序代码中通过 id 存取此组件。
- android:title：值为“"@string/yangmingshan"”，代表取得放在文本文件内的“阳明山国家公园”文字，所以图 5-2 的第一个菜单项会显示该文字。
- android:icon：值为“"@drawable/yangmingshan"”，代表取得放在 res/drawable 目录内主文件名为 yangmingshan 的图形文件，所以图 5-2 的第一个菜单项会显示该图形。

不可不知

菜单项为 6 个以下（含 6 个），会全部显示出来。选项超过 6 个，则第 6 个菜单项会显示“更多”（More），如图 5-3 所示；当单击“更多”选项，才会将真正的第 6 个菜单项以及剩下的其他菜单项显示出来。

图 5-3

改写与 Options Menu 有关的方法

要在界面上呈现可以让使用者操作的 Options Menu，还需要改写下列 Activity 类的 2 个方法：

public boolean onCreateOptionsMenu(Menu menu) 当 Options Menu 要显示时，会调用此方法。此方法目的在于初始化 Options Menu，所以必须加载定义 Menu 组件的 XML 文件。此方法只会被调用一次（意味着产生 Option Menu 后，就无法再修改其菜单项），并产生一个真正会在界面上显示的 Options Menu。 • menu：会显示在界面上的 Menu 组件，所以必须将菜单项放置在此组件上。 • 返回值：返回 true，Options Menu 才能显示在界面上；返回 false 则无法显示 Options Menu。
public boolean onOptionsItemSelected(MenuItem item) Options Menu 上的菜单项被单击时，此方法会被调用。 • item：被选取的 MenuItem 组件会被传递进来。 • 返回值：返回 true 代表开发者已经对菜单项被选取的情况作了对应的处理，所以无须系统作后续处理。false 代表允许系统作后续处理。

OptionsMenuEx/src/org/optionsMenuEx/OptionsMenuEx.java

```
18.    @Override
19.    public boolean onCreateOptionsMenu(Menu menu) {
20.       MenuInflater inflater = getMenuInflater();
21.       inflater.inflate(R.menu.mymenu, menu);
22.       return true;
23.    }
```

```
24.
25.    @Override
26.    public boolean onOptionsItemSelected(MenuItem item) {
27.       String msg = "";
28.       switch (item.getItemId()) {
29.       case R.id.yangmingshan:
30.          msg = getString(R.string.yangmingshan);
31.          break;
32.       case R.id.yushan:
33.          msg = getString(R.string.yushan);
34.          break;
35.       case R.id.taroko:
36.          msg = getString(R.string.taroko);
37.          break;
38.       case R.id.myloc:
39.          msg = getString(R.string.myloc);
40.          break;
41.       case R.id.exit:
42.          this.finish();
43.       default:
44.          return super.onOptionsItemSelected(item);
45.       }
46.       Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
47.       return true;
48.    }
```

20-21 行：调用 getMenuInflater() 取得 MenuInflater 对象，有了该对象才能调用 inflate() 装载 mymenu.xml 资源文件，并将文件内容转化成显示在界面上的 Menu 组件。

26 行：当用户单击菜单项时，会将被选取的 MenuItem 组件传递给 item 参数。

28-47 行：调用 getItemId() 可以取得选项所对应的资源 ID。用户选取选项后，使用 switch-case 判断被选取的是哪个选项，并使用 Toast 消息框呈现该选项的文字。因为都有作对应的处理（43-44 行的情况除外），所以 47 行可以直接返回 true，代表无须系统作后续处理。

43-44 行：一般建议如果没有对应处理方式，应该调用父类（Activity）的 onOptionsItemSelected()，由该方法返回 false，而非直接返回 false[1]。

5–1–2　Context Menu

Context Menu 与 Options Menu 的创建方式十分类似，要创建一个使用者可以操作的 Context Menu，需要下列 3 个步骤：

Step 1　使用 XML 文件建立 Menu 组件：这点与 Options Menu 相同，不再赘述。另外，Context Menu 可以使用 <group> 标签将数个菜单项组起来设置成复选框（checkbox）与单选按钮（radio

[1] 请参考 http://developer.android.com/guide/topics/ui/menus.html 的“Creating an Options Menu”部分，有详细地说明。

button）。

Step 2 改写与 Context Menu 有关的方法：虽然改写的方法名称与 Options Menu 不同，但功能大致相同。

Step 3 注册指定 UI 组件：必须指定在何种 UI 组件上久按才会弹出 Context Menu。Options Menu 没有此项设置。

使用 XML 文件建立 Menu 组件

范例 ContextMenuEx

图 5-4

范例说明：

- 界面显示：请长按本例界面。
- 长按界面后会弹出 5 个菜单项，其中“阳明山国家公园”、“玉山国家公园”、“太鲁阁国家公园”等 3 个选项为同一组菜单。
- 单击“结束”菜单项会结束该程序；单击其余 4 个菜单项会以 Toast 消息框呈现菜单项上面的文字。

ContextMenuEx/res/menu/mymenu.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <menu xmlns:android="http://schemas.android.com/apk/res/android">
3.     <group android:checkableBehavior="single">
```

```
4.        <item android:id="@+id/yangmingshan" android:title="@string/yangmingshan" />
5.        <item android:id="@+id/yushan" android:title="@string/yushan" />
6.        <item android:id="@+id/taroko" android:title="@string/taroko" />
7.    </group>
8.        <item android:id="@+id/myloc" android:title="@string/myloc" />
9.        <item android:id="@+id/exit" android:title="@string/exit" />
10. </menu>
```

3-7 行：使用 <group> 标签将 4、5、6 行的选项组起来并设置成单选按钮。3 行的 android:checkableBehavior 属性有 3 个值可选择：

- single：设置成单选按钮。
- all：设置成复选框。
- none：设置成普通选项。

不可不知

Context Menu 不支持图标菜单（icon menu），但支持复选框与单选按钮；而 Options Menu 则正好相反，支持图标菜单，但不支持复选框与单选按钮。

改写与 Context Menu 组件有关的方法

要在界面上呈现可以让使用者操作的 Context Menu，还需要改写下列 Activity 类的 2 个方法：

public void onCreateContextMenu (ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) 当 Context Menu 要显示时，会调用此方法。此方法目的在于初始化 Context Menu，所以必须加载定义 Menu 组件的 XML 文件。每次要显示 Context Menu 时，此方法就会被调用一次，意味着创建 Context Menu 后，还可以修改其选项，这点与 Options Menu 不同。 • menu：须将菜单放置在此 Menu 组件上。 • v：被长按的 UI 组件。 • menuInfo：依据 v 所属的 View 类不同会显示不同的额外信息。
public boolean onContextItemSelected (MenuItem item) Context Menu 上的菜单项被单击时，此方法会被调用。 • item：被选取的 MenuItem 组件会被传递进来。 • 返回值：返回 true 代表开发者已经对选项被选取的情况作了对应的处理，所以无须系统作后续处理。false 代表允许系统作后续处理。

ContextMenuEx/res/menu/mymenu.xml

```
15.   @Override
16.   public void onCreate(Bundle savedInstanceState) {
17.      super.onCreate(savedInstanceState);
18.      setContentView(R.layout.main);
19.      LinearLayout linear = (LinearLayout)findViewById(R.id.linear);
20.      registerForContextMenu(linear);
21.   }
22.
```

```
23.   @Override
24.   public void onCreateContextMenu(ContextMenu menu, View v,
25.                               ContextMenuInfo menuInfo) {
26.      MenuInflater inflater = getMenuInflater();
27.      inflater.inflate(R.menu.mymenu, menu);
28.   }
29.
30.   @Override
31.   public boolean onContextItemSelected(MenuItem item){
32.      String msg = "";
33.      switch (item.getItemId()) {
34.      case R.id.yangmingshan:
35.         msg = getString(R.string.yangmingshan);
36.         break;
37.      case R.id.yushan:
38.         msg = getString(R.string.yushan);
39.         break;
40.      case R.id.taroko:
41.         msg = getString(R.string.taroko);
42.         break;
43.      case R.id.myloc:
44.         msg = getString(R.string.myloc);
45.         break;
46.      case R.id.exit:
47.         this.finish();
48.      default:
49.         return super.onContextItemSelected(item);
50.      }
51.      Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
52.      return true;
53.   }
```

20 行：调用 registerForContextMenu() 并指定在 LinearLayout 组件上长按时会弹出 Context Menu。

26-27 行：调用 getMenuInflater() 取得 MenuInflater 对象，有了该对象才能调用 inflate() 载入 mymenu.xml 资源文件，并将文件内容转化成显示在界面上的 Menu 组件。

33-52 行：调用 getItemId() 可以取得菜单项所对应的资源 ID，通过 switch-case 判断被选取的菜单项，并使用 Toast 消息框显示该该当项的文字。因为都有作对应的处理（48-49 行除外），所以 52 行可以直接返回 true，代表无须系统作后续处理。

48-49 行：如果没有对应处理方式，应该调用父类的 onContextItemSelected()。

5–1–3 Submenu

单击 Options Menu 与 Context Menu 的菜单项后还可以再显示子选单，也就是所谓的 Submenu，可以提供更多的菜单项让用户单击，不过在 Submenu 内不可再建立 Submenu。要产生一个使用者

可以操作的 Submenu，需要下列 2 个步骤：

Step 1　使用 XML 文件建立 Submenu 组件：前述的 <item> 标签内再以嵌套方式嵌入 <menu> 与对应的 <item> 标签即可。

Step 2　改写与 Menu 有关的方法：依据 Submenu 依附的对象（Options Menu 或 Context Menu）来改写对应的方法。

使用 XML 文件建立 Submenu 组件

范例 SubmenuEx

范例说明：

- 界面显示：请按 Menu 按钮，按下实体 Menu 按钮后会弹出 5 个菜单项，如图 5-5 所示。
- 单击“阳明山国家公园”会弹出子选单，其上有“园区导览”、“交通位置”2 个子菜单项，如图 5-6 所示。
- 单击任何一个子菜单项都会以 Toast 消息框显示父菜单与子菜单上面的文字。

图 5-5

图 5-6

SubmenuEx/res/menu/mymenu.xml

```
1.   <?xml version="1.0" encoding="utf-8"?>
2.   <menu xmlns:android="http://schemas.android.com/apk/res/android">
3.      <item android:id="@+id/yangmingshan" android:title="@string/yangmingshan" >
4.         <menu>
```

```
5.            <item android:id="@+id/guide" android:title="@string/guide" />
6.            <item android:id="@+id/traffic" android:title="@string/traffic" />
7.         </menu>
8.      </item>
9.      <item android:id="@+id/yushan" android:title="@string/yushan" />
10.     <item android:id="@+id/taroko" android:title="@string/taroko" />
11.     <item android:id="@+id/myloc" android:title="@string/myloc" />
12.     <item android:id="@+id/exit" android:title="@string/exit" />
13.  </menu>
```

3-7 行：第 3 行的菜单建立子选单，并在 5、6 行定义子菜单。也可以在程序代码内调用 Menu 的 addSubMenu() 动态新增 SubMenu（Menu 的子接口）组件，然后通过 SubMenu 组件调用 add() 动态新增子选项。

改写与 Menu 有关的方法

如果 Submenu 依附的对象是 Options Menu，应改写 onCreateOptionsMenu()、onOptionsItemSelected()；如果依附的对象是 Context Menu，则应改写 onCreateContextMenu()、onContextItemSelected()。

SubmenuEx/src/org/submenuEx/SubmenuEx.java

```
25.  @Override
26.  public boolean onOptionsItemSelected(MenuItem item) {
27.     String msg = "";
28.     switch (item.getItemId()) {
29.     case R.id.yangmingshan:
30.        msg = getString(R.string.yangmingshan);
31.        break;
32.     case R.id.guide:
33.        msg = getString(R.string.yangmingshan) + " > " +
34.              getString(R.string.guide);
35.        break;
36.     case R.id.traffic:
37.        msg = getString(R.string.yangmingshan) + " > " +
38.              getString(R.string.traffic);
39.        break;
40.
41.     case R.id.yushan:
42.        msg = getString(R.string.yushan);
43.        break;
44.     case R.id.taroko:
45.        msg = getString(R.string.taroko);
46.        break;
47.     case R.id.myloc:
48.        msg = getString(R.string.myloc);
49.        break;
50.     case R.id.exit:
51.        this.finish();
52.     default:
53.        return super.onOptionsItemSelected(item);
```

```
54.         }
55.         Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
56.         return true;
57.     }
```

26 行：无论单击主菜单（Options Menu）选项或是子选单（Submenu）选项都会调用此方法。

32-39 行：通过 switch-case 判断被选取的菜单项，如果是子选单的菜单项被选取，使用 Toast 消息框显示被选取的子菜单项与所属父菜单的文字。

5-2　对话窗口

当使用者要删除某个联系人时，为了避免删错，通常会弹出一个对话窗口再次询问“删除此联系人？”，让使用者有反悔的机会，如图 5-7 所示。

图 5-7

以上所述的对话窗口属于 AlertDialog（提醒对话窗口），除此之外 Android 还提供 DatePickerDialog（日期选择对话框）与 TimePickerDialog（时间选择对话框）等其他功能的对话窗口。

5–2–1　AlertDialog

AlertDialog 是 Dialog 的子类，可能是最常见的对话窗口。建立 AlertDialog 时必须设置 3 个部分：

- 对话窗口的标题文字。
- 对话窗口的消息正文。
- 对话窗口的按钮（包含 Button 事件处理）及按钮上面的文字。

使用 AlertDialog.Builder 类可以快速建立并设置 AlerDialog，其常用方法如表 5-1 所示：

▼ 表 5-1

AlertDialog.Builder 类
建构式
public AlertDialog.Builder (Context context) 建立 AlertDialog.Builder 对象。 • context：对话窗口所依附的对象，通常是指 Activity 对象。
方法
public AlertDialog.Builder setTitle (int titleId) 设置对话窗口的标题文字。 • titleId：标题文字的资源 ID，通常是指文本文件内对应的文字 ID。
public AlertDialog.Builder setMessage (int messageId) 设置对话窗口欲显示的消息正文。 • messageId：消息正文的资源 ID，通常是指文本文件内对应的文字 ID。
public AlertDialog.Builder setPositiveButton (int textId, DialogInterface.OnClickListener listener) 设置对话窗口的 Positive 按钮[2] 及上面的文字。 • textId：按钮文字的资源 ID，通常是指文本文件内对应的文字 ID。 • listener：监听 Positive 按钮是否被按下的监听器。
public AlertDialog.Builder setNegativeButton (int textId, DialogInterface.OnClickListener listener) 设置对话窗口的 Nagative 按钮及上面的文字。 • textId：按钮文字的资源 ID，通常是指文本文件内对应的文字 ID。 • listener：监听 Negative 按钮是否被按下的监听器。
AlertDialog.Builder 类
方法
public AlertDialog.Builder setCancelable (boolean cancelable) 设置对话窗口是否可以被取消。 • cancelable：true 代表可以取消，例如按下实体的返回按钮可以取消对话窗口；false 代表不可以取消。预设为 true。
public AlertDialog show () 依照 AlertDialog.Builder 的设置建立 AlertDialog，并显示在界面上。

范例 AlertDialogEx

范例说明：

- 按下“结束程序”按钮后会弹出 AlertDialog 并显示对应的标题、信息与按钮。

[2] AlertDialog 一共可设 3 种按钮：PositiveButton、NegativeButton 和 NeutralButton，此种分类与按钮真正的功能不一定有关系，但建议功能与按钮的分类尽量相符。例如：将“确定”按钮设定在 PositiveButton。

- 按下“确定”按钮会结束并离开此应用程序。
- 按下“取消”按钮会回到主窗口。

图 5-8

AlertDialogEx/src/org/alertDialogEx/AlertDialogEx.java

```
19.   private void findViews() {
20.      btnExit = (Button)findViewById(R.id.btnExit);
21.      btnExit.setOnClickListener(new OnClickListener() {
22.         @Override
23.         public void onClick(View v) {
24.            new AlertDialog.Builder(AlertDialogEx.this)
25.            .setTitle(R.string.title)
26.            .setMessage(R.string.prompt)
27.            .setPositiveButton(R.string.submit,
28.               new DialogInterface.OnClickListener() {
29.                  public void onClick(DialogInterface dialog, int id) {
30.                     AlertDialogEx.this.finish();
31.                  }
32.               })
33.            .setNegativeButton(R.string.cancel,
34.               new DialogInterface.OnClickListener() {
35.                  public void onClick(DialogInterface dialog, int id) {
36.                        dialog.cancel();
37.                  }
38.               })
```

```
39.              .setCancelable(false)
40.              .show();
41.          }
42.      });
43.  }
```

25 行：设置对话窗口的标题文字。

26 行：设置对话窗口欲显示的消息正文。

27 行：设置对话窗口的 Positive 按钮及上面的文字。

28-32 行：利用匿名内部类实现 DialogInterface.OnClickListener.onClick()，当 Positive 按钮被按下时，onClick() 会自动被调用，运行“AlertDialogEx.this.finish();”并关闭 Activity。

36 行：调用 cancel() 会取消对话窗口。

39 行：false 代表不可以取消对话窗口。

40 行：将建立好的 AlertDialog 显示在界面上。

5–2–2 DatePickerDialog 与 TimePickerDialog

为了让使用者能够可视化地选择日期/时间，开发者可以使用 DatePickerDialog/ TimePickerDialog 可视化选择组件来达到此目的。建立 DatePickerDialog/TimePickerDialog 组件所需使用到的方法说明如表 5-2 所示：

▼ 表 5-2

Activity 类
public final void showDialog (int id) 调用此方法会自动调用另一个 overloading 方法 - showDialog (int id, Bundle args)，而参数 args 会传入 null。 • id：为了管理对话窗口而建立的标识符。
protected Dialog onCreateDialog (int id, Bundle args) 调用 showDialog()之后会自动调用此方法。如果返回 null 则不会建立对话窗口。 • id：当初调用 showDialog()所设置的 ID。 • args：当初调用 showDialog()所设置的 Bundle 对象。
DatePickerDialog 类
public DatePickerDialog (Context context, DatePickerDialog.OnDateSetListener callBack, int year, int monthOfYear, int dayOfMonth) 建立 DatePickerDialog 对话窗口。 • context：通常是当前 Activity 对象。 • callBack：OnDateSetListener 对象，为了调用已实现的 onDateSet()。 • year：DatePickerDialog 对话窗口显示时，预选的年。 • monthOfYear：当前的月。 • dayOfMonth：当前的日。

续表

DatePickerDialog.OnDateSetListener 界面
public abstract void onDateSet (DatePicker view, int year, int monthOfYear, int dayOfMonth) 用户选定日期并按下“确定”按钮后会自动调用此方法。 • view：发生事件的 DatePicker 组件。 • year：选定的年。 • monthOfYear：选定的月，值为 0-11（zero-based）；换句话说，1 月是以 0 代表，这是为了配合 Calendar 类。 • dayOfMonth：选定的日。
TimePickerDialog 类
public TimePickerDialog (Context context, TimePickerDialog.OnTimeSetListener callBack, int hourOfDay, int minute, boolean is24HourView) 建立 TimePickerDialog 对话窗口。 • context：通常是当前 Activity 对象。 • callBack：OnTimeSetListener 对象，为了调用已实现的 onTimeSet()。 • hourOfDay：TimePickerDialog 对话窗口显示时，当前的时。 • minute：当前的分。 • is24HourView：是否为 24 时制。
TimePickerDialog.OnTimeSetListener 界面
public abstract void onTimeSet (TimePicker view, int hourOfDay, int minute) 用户选定时间并按下“确定”按钮后会自动调用此方法。 • view：发生事件的 TimePicker 组件。 • hourOfDay：当前的时。 • minute：当前的分。

创建 DatePickerDialog 对话窗口的步骤如下：

Step 1 要弹出 DatePickerDialog 让使用者设置日期必须先调用 Activity 的 showDialog()。因为一个 Activity 可能有多个对话窗口，所以必须帮每个对话窗口设置 ID 并当作参数传递，以方便之后判断要建立哪一种对话窗口。

```
static final int DATE_DIALOG_ID = 0; //替每个对话窗口建立一个独一无二的 ID
showDialog(DATE_DIALOG_ID);
```

Step 2 调用 showDialog() 之后会自动调用 Activity 的 onCreateDialog() 来建立对话窗口，所以必须改写该方法。在 Step 1 设置的 ID 会被当作参数传递过来，通过 switch-case 的判断，可以知道要建立何种对话窗口。

```
@Override
protected Dialog onCreateDialog(int id, Bundle args) {
    switch (id) {
    case DATE_DIALOG_ID:
        //建立 DatePickerDialog
    }
    return null;
}
```

Step 3 要创建 DatePickerDialog 对话窗口，必须建立 DatePickerDialog 对象并实现 OnDateSetListener 的 onDateSet()。当按下日期设置按钮时，会自动调用 OnDateSetListener 的 onDateSet()。

```
private DatePickerDialog.OnDateSetListener dateSetListener =
       new DatePickerDialog.OnDateSetListener() {
          @Override
          public void onDateSet(DatePicker view, int year,
                            int monthOfYear, int dayOfMonth) {
             //程序内容
          }
       };

@Override
protected Dialog onCreateDialog(int id)  {
    switch (id) {
    case DATE_DIALOG_ID:
       return new DatePickerDialog(this,
             dateSetListener, //将实现好的 OnDateSetListener 对象放在这个位置
             mYear, mMonth, mDay);
    }
    return null;
}
```

创建 TimePickerDialog 对话窗口的方式与 DatePickerDialog 大致相同，不再赘述。

范例 DatePickerEx

图 5-9

范例说明：

- 按下“改变日期”按钮会弹出 DatePickerDialog 让使用者设置日期。
- 按下“设置”按钮会将用户选定的日期显示在主窗口的 TextView 组件上。按下“取消”按钮则取消日期设置并回到主窗口。

DatePickerEx/src/org/datePickerEx/DatePickerEx.java

```
23.   @Override
24.   public void onCreate(Bundle savedInstanceState) {
25.      super.onCreate(savedInstanceState);
26.      setContentView(R.layout.main);
27.      findViews();
28.      final Calendar c = Calendar.getInstance();
29.      mYear = c.get(Calendar.YEAR);
30.      mMonth = c.get(Calendar.MONTH);
31.      mDay = c.get(Calendar.DAY_OF_MONTH);
32.      updateDisplay();
33.   }
34.
35.   private void updateDisplay() {
36.      tvDateDisplay.setText(
37.         new StringBuilder()
38.               .append(mYear).append("-")
39.               .append(mMonth + 1).append("-")
40.               .append(mDay));
41.   }
42.
43.   private void findViews() {
44.      tvDateDisplay = (TextView)findViewById(R.id.tvDateDisplay);
45.      btnPickDate = (Button)findViewById(R.id.btnPickDate);
46.      btnPickDate.setOnClickListener(new OnClickListener() {
47.         @Override
48.         public void onClick(View v) {
49.            showDialog(DATE_DIALOG_ID);
50.         }
51.      });
52.   }
53.
54.   private DatePickerDialog.OnDateSetListener dateSetListener =
55.         new DatePickerDialog.OnDateSetListener() {
56.            @Override
57.            public void onDateSet(DatePicker view, int year,
58.                                  int monthOfYear, int dayOfMonth) {
59.               mYear = year;
60.               mMonth = monthOfYear;
61.               mDay = dayOfMonth;
62.               updateDisplay();
63.            }
```

```
64.         };
65.
66.    @Override
67.    protected Dialog onCreateDialog(int id, Bundle args) {
68.       switch (id) {
69.       case DATE_DIALOG_ID:
70.          return new DatePickerDialog(this,
71.                   dateSetListener,
72.                   mYear, mMonth, mDay);
73.       }
74.       return null;
75.    }
```

28 行：取得当前日期。

32 行：调用 36 行 updateDisplay()。

35-41 行：将指定的日期显示在 TextView 上。39 行“mMonth + 1”是因为月的值是 0-11。

46-51 行：按下“改变日期”按钮会调用 Activity 的 showDialog() 并传递 DATE_DIALOG_ID 标识符。之后 67 行的 onCreateDialog() 会被调用。

67-69 行：49 行设置的 ID 会被当作参数传递过来，通过 switch-case 的判断，可以知道要建立 DatePickerDialog。

70-72 行：建立 DatePickerDialog 对象并将实现 OnDateSetListener 的对象 dateSetListener（在 54 行建立）设置成参数，按下 DatePickerDialog 确定按钮时会调用 57 行 onDateSet()。mYear、mMonth、mDay 会成为 DatePickerDialog 一开始预选的年月日。

范例 TimePickerEx

图 5-10

范例说明：

- 按下“改变时间”按钮会弹出 TimePickerDialog 让使用者设置时间。
- 按下“设置”按钮会将用户选定的时间显示在主窗口的 TextView 组件上。按下“取消”按钮则取消时间设置并回到主窗口。

TimePickerEx/src/org/timePickerEx/TimePickerEx.java

```
40.   private static String pad(int c) {
41.      if (c >= 10)
42.         return String.valueOf(c);
43.      else
44.         return "0" + String.valueOf(c);
45.   }
46.
47.   private void findViews() {
48.      tvTimeDisplay = (TextView)findViewById(R.id.tvTimeDisplay);
49.      btnPickTime = (Button)findViewById(R.id.btnPickTime);
50.      btnPickTime.setOnClickListener(new OnClickListener() {
51.         @Override
52.         public void onClick(View v) {
53.            showDialog(TIME_DIALOG_ID);
54.         }
55.      });
56.   }
57.
58.   private TimePickerDialog.OnTimeSetListener timeSetListener =
59.      new TimePickerDialog.OnTimeSetListener() {
60.         public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
61.            mHour = hourOfDay;
62.            mMinute = minute;
63.            updateDisplay();
64.         }
65.      };
66.
67.   @Override
68.   protected Dialog onCreateDialog(int id, Bundle args) {
69.      switch (id) {
70.      case TIME_DIALOG_ID:
71.         return new TimePickerDialog(this,
72.               timeSetListener, mHour, mMinute, false);
73.      }
74.      return null;
75.   }
```

40-45 行：若数字有十位数，则直接显示；若只有个位数则补 0 后再显示。例如 7 会改成 07 后再显示。

50-55 行：按下“改变时间”按钮会调用 Activity 的 showDialog() 并传递 TIME_DIALOG_ID 标识符，之后 68 行的 onCreateDialog() 会被调用。

68-70 行：53 行设置的 ID 会被当作参数传递过来，通过 switch-case 的判断，可以知道要建立 TimePickerDialog。

71-72 行：建立 TimePickerDialog 对象并将实现 OnTimeSetListener 的对象 timeSetListener（在 58 行建立）设置成参数，按下 TimePickerDialog 确定按钮时会调用 60 行 onTimeSet()。mHour、mMinute 会成为 TimePickerDialog 一开始预选的时与分。

5-3 Spinner 与 AutoCompleteTextView

5–3–1 Spinner

Spinner 是一个非常类似下拉列表框（drop-down list）的 UI 组件，其优点是节省显示空间，因为用户尚未单击时，仅显示一条数据。如同其他 UI 组件，一般建议通过 layout 文件建立 Spinner 组件。使用 layout 文件（可参考 SpinnerEx/res/layout/main.xml）建立 Spinner 组件以及选项被单击后的事件处理步骤如下：

Step 1 建立 Spinner 选项：在文本文件内建立字符串数组[3]（请参考 SpinnerEx/res/values/strings.xml），而数组内容即为 Spinner 选项。

```
<string-array name="food_array">
  <item>蚵仔面线</item>
  <item>臭豆腐</item>
  <item>葱油饼</item>
</string-array>
```

Step 2 建立 Spinner 组件：使用 <Spinner> 标签。

Step 3 设置 Spinner 的提示文字：利用 android:prompt 属性设置提示文字，例如“android:prompt="爱吃什么？""会显示如图 5-11 的提示文字。

Step 4 设置 Spinner 选项：利用 android:entries 属性设置选项。例如 android:entries="@array/food_array"，而 array/food_array 就是 Step 1 建立好的字符串数组。

Step 5 取得 Spinner 组件后调用 setOnItemSelectedListener() 向 OnItemSelectedListener 监听器注册，并实现 onItemSelected() 以响应选项改变的情况：

```
spFood = (Spinner)findViewById(R.id.spFood);
spFood.setOnItemSelectedListener(listener);
```

[3] 也可以在程序代码内使用数组或 List 集合建立选项，例如：String[] foods = {"蚵仔面线", "臭豆腐", "葱油饼"};，然后使用 ArrayAdapter 来获取内容。

```
Spinner.OnItemSelectedListener listener =
    new Spinner.OnItemSelectedListener(){
    @Override
    public void onItemSelected(AdapterView parent,
        View view, int pos, long id) {
            //ItemSelected
        }

    @Override
    public void onNothingSelected(AdapterView parent) {
        //NothingSelected
    }
};
```

图 5-11

建立 Spinner 组件所需使用到的方法说明如表 5-3 所示：

▼ 表 5-3

AdapterView 类[4]
public void setOnItemSelectedListener (AdapterView.OnItemSelectedListener listener) 向 OnItemSelectedListener 监听器注册，要求监听选项是否被选取。 • listener：实现 OnItemSelectedListener 接口的对象。

[4] Spinner 继承 AbsSpinner；AbsSpinner 继承 AdapterView。

续表

AdapterView.OnItemSelectedListener 界面
public abstract void onItemSelected (AdapterView<?> parent, View view, int position, long id) 当选项被选取时会自动调用此方法。 • parent：发生选项被单击的 UI 组件，这里是指 Spinner 组件。 • view：在 AdapterView 中被单击的组件，通常是 TextView 组件。 • position：选项的 index。 • id：选项的 row ID。
public abstract void onNothingSelected (AdapterView<?> parent) 当选项消失时会自动调用此方法。 • parent：发生选项消失的 UI 组件，这里是指 Spinner 组件。
ArrayAdapter 类[5]
建构式
public ArrayAdapter (Context context, int textViewResourceId, T[] objects) 建立 ArrayAdapter 组件。ArrayAdapter 组件主要用来管理整个选项的内容与样式。 • context：通常是现行 Activity 对象。 • textViewResourceId：选项的样式主要是以 TextView 组件组成，所以可以在 layout 文件内建立好 TextView 组件的样式，然后通过资源 ID 套用在选项上。如果不想自行建立选项样式，也可以直接套用 Android 系统内建的样式：android.R.layout.simple_spinner_item。 • objects：选项的内容。
ArrayAdapter 类
方法
public void setDropDownViewResource (int resource) 设置整个下拉列表的样式。 • resource：通过资源 ID 套用欲呈现的下拉列表样式。可以直接套用 Android 系统内建的样式：android.R.layout.simple_spinner_dropdown_item。
AbsSpinner 类
public void setAdapter (SpinnerAdapter adapter) 将设置好的选项内容与样式套用在 Spinner 组件上。 • adapter：可以套用上述 ArrayAdapter 对象，也就是选项的内容与样式。

范例 SpinnerEx

范例说明：

- 单击任何一个 Spinner 组件都会将被选取的选项文字以 Toast 消息框显示出来。

[5] ArrayAdapter 继承 SpinnerAdapter。

图 5-12

SpinnerEx/src/org/spinnerEx/SpinnerEx.java

```
22.   private void findViews() {
23.      spFood = (Spinner)findViewById(R.id.spFood);
24.      spFood.setOnItemSelectedListener(listener);
25.
26.      spPlace = (Spinner)findViewById(R.id.spPlace);
27.      String[] places = {"Australia", "U.K.", "Japan", "Thailand"};
28.      ArrayAdapter<String> adapterPlace =
29.         new ArrayAdapter<String>(this,
30.               android.R.layout.simple_spinner_item, places);
31.      adapterPlace.setDropDownViewResource(
32.                android.R.layout.simple_spinner_dropdown_item);
33.      spPlace.setAdapter(adapterPlace);
34.      spPlace.setOnItemSelectedListener(listener);
35.   }
36.
37.   Spinner.OnItemSelectedListener listener =
38.      new Spinner.OnItemSelectedListener(){
39.         @Override
40.         public void onItemSelected(AdapterView parent,
41.            View view, int pos, long id) {
42.            Toast.makeText(parent.getContext(),
43.                  parent.getItemAtPosition(pos).toString(),
44.                  Toast.LENGTH_SHORT).show();
45.         }
46.
47.         @Override
48.         public void onNothingSelected(AdapterView parent) {
49.            Toast.makeText(parent.getContext(),
```

```
50.                     "Nothing Selected!",
51.                     Toast.LENGTH_SHORT).show();
52.         }
53.   };
```

24、34、37-53 行：Spinner 组件调用 setOnItemSelectedListener() 向 OnItemSelectedListener 注册，OnItemSelectedListener 专门监听选项是否被选取。接下来 37 行利用匿名内部类实现 OnItemSelectedListener 的 onItemSelected()、onNothingSelected() 等 2 个方法。当选项被选取时，onItemSelected() 会自动被调用，而被选取的选项文字将以 Toast 消息框显示出来。

27-30 行：调用 ArrayAdapter 构造方法以建立选项的内容与样式。选项的内容来自于 27 行的 places 字符串数组；样式则套用系统内置布局：android.R.layout.simple_spinner_item。

31-32 行：调用 setDropDownViewResource() 套用系统内置的下拉列表样式：android.R.layout.simple_spinner_dropdown_item。

33 行：Spinner 组件调用 setAdapter()套用指定的 ArrayAdapter 以加载对应的选项内容与样式。

5–3–2 AutoCompleteTextView

AutoCompleteTextView 非常类似 EditText，都是方便用户输入的组件。不过 AutoCompleteTextView 组件另外提供提示文字列表，当用户输入的局部文字符合提示文字时，应用程序就会自动列出符合的提示文字列表，让用户可以直接选择想输入的文字而不必将全部文字输入完毕，是一种方便使用者输入的设计。

AutoCompleteTextView 组件的提示列表与 Spinner 组件的选项建立方式相同，都是在文本文件内建立字符串数组来储存欲提示的文字。

范例 AutoCompleteEx

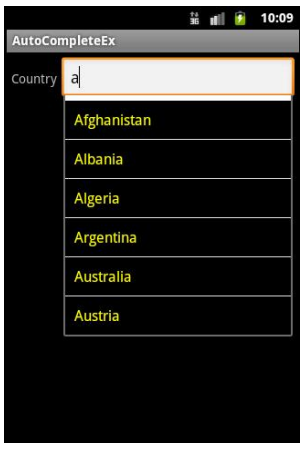

图 5-13

范例说明：

- 输入“a”，应用程序会作比对，并自动将符合的提示文字以列表方式呈现，方便使用者以选择方式输入。

AutoCompleteEx/res/layout/main.xml

```
13.   <AutoCompleteTextView android:id="@+id/acCountry"
14.       android:layout_width="match_parent"
15.       android:layout_height="wrap_content"
16.       android:layout_marginLeft="5dp"
17.       android:completionThreshold="1" />
```

13 行：定义 AutoCompleteTextView 组件。

17 行：设置至少要输入 1 个字符才会显示提示文字。如果未设置则默认为 2 个字符。

AutoCompleteEx/src/org/autoCompleteEx/AutoCompleteEx.java

```
18.   private void findViews() {
19.       acCountry = (AutoCompleteTextView)findViewById(R.id.acCountry);
20.       String[] countries =
21.           getResources().getStringArray(R.array.countries_array);
22.       ArrayAdapter<String> adapterCountry =
23.           new ArrayAdapter<String>(this, R.layout.list_item, countries);
24.       acCountry.setAdapter(adapterCountry);
25.   }
```

21 行：调用 Context 的 getResources() 会取得 Resources 对象，再调用 getStringArray() 会取得字符串数组，其内容将用来作为 AutoCompleteTextView 组件的提示文字。R.array.countries_array 代表定义在文本文件内，名为 countries_array 的字符串数组。

23 行：R.layout.list_item 代表提示文字将套用“list_item”layout 文件所定义的样式。ArrayAdapter 已在 Spinner 一节说明，不再赘述。

5-4　自定义 View 组件与 2D 绘图

Android 提供 2D 绘图功能，开发者所需组件为 android.graphics。自定义 View 组件与 2D 绘图说明如下：

（1）继承 View 类并改写 onDraw()：想绘图必须有个可显示的组件供绘制，要取得该组件最简单的方式就是自定义类（例如 GeometricView 类）去继承 View 类，并且改写 onDraw()，将想要绘制的图形置入 onDraw() 方法内，如下列程序代码：

```
public class GeometricView extends View {
  public GeometricView(Context context){
    super(context);
    //初始化组件
  }

  @Override
```

```
  protected void onDraw(Canvas canvas) {
    //绘制图形的程序代码
  }
}
```

利用构造方法创建该类的对象时，系统便会自动调用改写好的 onDraw() 方法并完成图形绘制，而产生的对象便属于 View 组件的一种。

（2）通过 layout 文件作界面配置：如果想要像其他 UI 组件一样通过 layout 文件来作界面配置，还必须通过 GeometricView(Context, AttributeSet)构造方法将 AttributeSet（该组件的 XML 属性）参数传递给父类对应的构造方法，如下列程序代码：

```
public class GeometricView extends View {
  public GeometricView(Context context){
    super(context);
    //初始化组件
  }

  public GeometricView(Context context, AttributeSet attrs){
super(context, attrs); //attrs 就是 layout 文件内为该组件设置的属性
    //初始化组件
  }

  @Override
  protected void onDraw(Canvas canvas) {
    //绘制图形的程序代码
  }
}
```

以 layout 文件建立此组件会自动调用 GeometricView(Context, AttributeSet)构造方法，而 layout 文件的设置方式如下：

```
<org.draw2D.GeometricView
  android:id="@+id/geomView"
  android:layout_width="match_parent"
  android:layout_height="wrap_content" />
```

（3）调用 View.invalidate() 重绘组件：如果自定义组件产生后想要重新绘制该组件上面的图形，可以调用 View.invalidate()，系统会先清楚原来的图形然后自动调用 onDraw()，以重新绘制此 View 的内容。

范例 Draw2D

范例说明：

- 按下“按我就变色”按钮会让下面的几何图形随机变色。

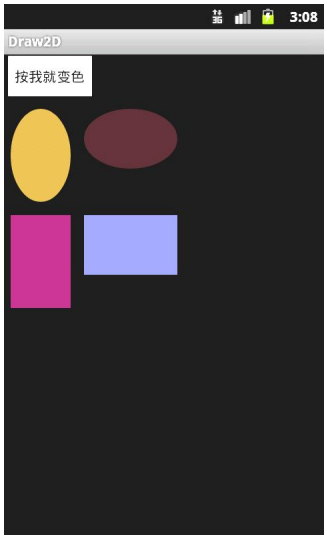

图 5-14

Draw2D/src/org/draw2D/GeometricView.java

```
 9.  public class GeometricView extends View {
10.     private ShapeDrawable[] shapes;
11.     public GeometricView(Context context) {
12.        super(context);
13.        makeShapes();
14.     }
15.
16.     public GeometricView(Context context, AttributeSet attrs){
17.        super(context, attrs);
18.        makeShapes();
19.     }
20.     public void makeShapes(){
21.        shapes = new ShapeDrawable[4];
22.        shapes[0] = new ShapeDrawable(new OvalShape());
23.        shapes[0].setBounds(10, 10, 100, 150);
24.        shapes[1] = new ShapeDrawable(new OvalShape());
25.        shapes[1].setBounds(120, 10, 260, 100);
26.        shapes[2] = new ShapeDrawable(new RectShape());
27.        shapes[2].setBounds(10, 170, 100, 310);
28.        shapes[3] = new ShapeDrawable(new RectShape());
29.        shapes[3].setBounds(120, 170, 260, 260);
30.     }
31.
32.     @Override
33.     protected void onDraw(Canvas canvas) {
34.        for (ShapeDrawable shape : shapes) {
35.           int r = (int) (256 * Math.random());
```

```
36.             int g = (int) (256 * Math.random());
37.             int b = (int) (256 * Math.random());
38.             shape.getPaint().setARGB(255, r, g, b);
39.             shape.draw(canvas);
40.         }
41.     }
42.  }
```

9 行：要自定义 UI 组件最简单的方式就是先继承 View 类。

10 行：如果需要绘制自定义的几何图形，可以利用 ShapeDrawable 类。

16 行：通过 layout 文件建立 GeometricView 组件会调用此建构式。

20 行：调用 makeShapes() 以绘制 4 个几何图形。

22 行：OvalShape 对象代表椭圆形。

23 行：调用 setBounds() 设置该图形的四周边界。

26 行：RectShape 对象代表矩形。

33 行：改写 onDraw() 以绘制自定义图形。系统会传入组件上的 Canvas（画布）对象供绘图之用。

35-37 行：随机产生红、绿、蓝颜色的值。

38 行：调用 ShapeDrawable 的 getPaint() 会取得 Paint 对象，再调用 setARGB() 可以设置颜色的 alpha 值与红、绿、蓝 3 原色。

39 行：调用 ShapeDrawable 的 draw() 会将图形绘制在画布上。

Draw2D/src/org/draw2D/Draw2D.java

```
 8.  public class Draw2D extends Activity {
 9.     private Button btnSubmit;
10.     private GeometricView geomView;
11.     @Override
12.     public void onCreate(Bundle savedInstanceState) {
13.        super.onCreate(savedInstanceState);
14.        setContentView(R.layout.main);
15.        findViews();
16.
17.        //GeometricView geomView = new GeometricView(this);
18.        //setContentView(geomView);
19.     }
20.     public void findViews(){
21.        btnSubmit = (Button)findViewById(R.id.btnSubmit);
22.        btnSubmit.setOnClickListener(new OnClickListener() {
23.           @Override
24.           public void onClick(View v) {
25.              geomView.invalidate();
26.           }
27.        });
28.        geomView = (GeometricView)findViewById(R.id.geomView);
```

```
29.         }
30.     }
```

17-18 行：通过此 2 行程序代码可以直接在 Activity 建立 GeometricView 组件。

25 行：调用 invalidate()，系统会先废弃原来在 GeometricView 组件上的画布，然后自动调用 onDraw() 并传送新的画布以便重新绘制。

28 行：此范例通过 layout 文件来建立 GeometricView 组件，所以可以利用 findViewById() 来取得对应对象。

5-5　补间动画

补间动画（tweening，就是 in between 的意思）是指填补两个图形之间的变化，让第一个图形逐渐改变成第二个图形。Android 提供位移、缩放、旋转、透明化等补间动画的功能，而这些功能可以应用在大部分的 UI 组件上，让开发者可以很简单地将 UI 组件加上动画，使得 UI 界面更加丰富活泼，并提高使用者与操作界面的互动性。建议在 XML 文件内设置补间动画的各种效果，而不要直接以程序代码编写。这点其实就与之前所述将界面设置放在 layout 文件内的概念相同，因为使用 XML 文件来设置，可以增加可读性与重复利用性。一般而言，会将设置补间动画的 XML 文件放在 Android 项目的 res/anim 目录内。设置补间动画常用到的 XML 属性大都定义在 Animation 类内，说明如表 5-4 所示：

▼ 表 5-4

Animation 类的 XML 属性

属性名称	说明	属性值
android:duration	动画播放的时间，单位为毫秒。对应的方法为 setDuration(long)。	整数，不可为负值。
android:interpolator[6]	指定动画的运行效果。设置位移补间动画后，还需要指定整个位移过程的效果是 accelerate（加速）还是 decelerate（减速）。对应的方法为 setInterpolator(Interpolator)。	• linear_interpolator（线性） • accelerate_interpolator（加速） • decelerate_interpolator（减速） • anticipate_interpolator（先退后进） • overshoot_interpolator（冲过头） • bounce_interpolator（反弹） • cycle_interpolator（以曲线方式加快重复次数）[7] 预设为 linear_interpolator
android:repeatCount	动画重复播放次数。对应的方法为 setRepeatCount(int)。	整数。-1 代表无限重复播放。 预设为 0（不重复播放）。

[6] 所有的动画特效请参看 R.anim 类的常数。

[7] 请参考 CycleInterpolator 类说明。

续表

Animation 类的 XML 属性		
属性名称	说明	属性值
android:repeatMode	动画重复播放模式。对应的方法为 setRepeatMode(int)。	• restart：重新播放。 • reverse：反向播放。 预设为 restart。
android:startOffset	设置主动画开始后多久才运行此动画，单位为毫秒。要播放多个动画时，可以使用这个属性来指定各个动画播放的相对时间。对应的方法为 setStartOffset(long)。	整数，不可为负值。

除了前述 Animation 类所定义的 XML 属性外，在设置补间动画时还需要了解其他相关 XML 属性，常用的属性分类说明如表 5-5 所示：

▼ 表 5-5

通用 XML 属性		
属性名称	说明	属性值
android:pivotX /android:pivotY	组件的哪个位置会发生补间动画，指定该位置的 X 轴/ Y 轴坐标[8]。	浮点数。 • 50% 代表在该组件的中央位置。 • 50 代表在父组件的 50% 的位置，也就是父组件的中央位置。
与位移有关的 XML 属性		
android:fromXDelta /android:fromYDelta	位移开始时组件的 X 轴/ Y 轴坐标。	浮点数。
android:toXDelta /android:toYDelta	位移结束时组件的 X 轴/ Y 轴坐标。	
与缩放有关的 XML 属性		
android:fromXScale /android:fromYScale	缩放开始时组件的水平/垂直尺寸。	浮点数。 • 1.0 代表维持原来尺寸。 • < 1.0 代表缩小。 • > 1.0 代表放大。
android:toXScale /android:toYScale	缩放结束时组件的水平/垂直尺寸。	
与旋转有关的 XML 属性		
android:fromDegrees	旋转开始时组件的角度。	浮点数。 • < 0.0 代表逆时针。 • > 0.0 代表顺时针。
android:toDegrees	旋转结束时组件的角度。	
与透明化有关的 XML 属性		
android:fromAlpha	动画开始时组件的透明度。	介于 0.0~1.0 之间的浮点数。 • 0.0 代表完全透明。 • 1.0 代表完全不透明。
android:toAlpha	动画结束时组件的透明度。	

[8] 原点 (0,0) 在左上角，向右则 X 轴的值增加；向下则 Y 轴的值增加。

Android 提供许多与补间动画设置有关的 XML 标签以方便开发者快速实现补间动画的功能，每个卷标都会对应到 Android.view.animation 组件内的类，说明如下[9]：

- <translate>：提供位移补间动画的功能，对应的类为 TranslateAnimation。
- <scale>：提供缩放补间动画的功能，对应的类为 ScaleAnimation。
- <rotate>：提供旋转补间动画的功能，对应的类为 RotateAnimation。
- <alpha>：提供透明化补间动画的功能，对应的类为 AlphaAnimation。
- <set>：使用此标签可以将数个补间动画设置成同一群组而一同播放，而且可以共享相同的设置，对应的类为 AnimationSet。

使用补间动画所需使用到的方法说明如表 5-6 所示：

▼ 表 5-6

AnimationUtils 类
public static Animation loadAnimation (Context context, int id) 加载资源目录内的动画配置文件案。 • context：加载资源文件需给予指定的 Context 对象，一般为 Activity 对象。 • id：动画配置文件对应的资源 ID。
View 类
public void startAnimation (Animation animation) 开始运行指定的动画。 • animation：要播放的 Animation 对象。

范例 TweenAnimEx

图 5-15

[9] TranslateAnimation、ScaleAnimation、RotateAnimation、AlphaAnimation、AnimationSet 等类都是 Animation 的子类。

范例说明：

- 按下“振动”按钮，EditText 会左右快速摇晃。
- 单击中间部分的 Spinner，可以选择“一二三四”文字要播放的特效。
- 单击下面部分的 Spinner，可以选择“百战不殆”文字要播放的特效。

TweenAnimEx/res/layout/main.xml

```
56.   <ViewFlipper android:id="@+id/fpText"
57.      android:layout_width="match_parent"
58.      android:layout_height="wrap_content"
59.      android:flipInterval="1500"
60.      android:layout_marginTop="10dp"
61.      android:layout_marginBottom="10dp" >
62.      <TextView
63.         android:layout_width="match_parent"
64.         android:layout_height="wrap_content"
65.         android:gravity="center_horizontal"
66.         android:textSize="20sp"
67.         android:text="@string/anim_text1"/>
68.      <TextView
69.         android:layout_width="match_parent"
70.         android:layout_height="wrap_content"
71.         android:gravity="center_horizontal"
72.         android:textSize="20sp"
73.         android:text="@string/anim_text2"/>
74.      <TextView
75.         android:layout_width="match_parent"
76.         android:layout_height="wrap_content"
77.         android:gravity="center_horizontal"
78.         android:textSize="20sp"
79.         android:text="@string/anim_text3"/>
80.   </ViewFlipper>
```

56 行：ViewFlipper 是一个简易的动画组件，可以加入多个 View 组件进来。62-79 行共加入 3 个 TextView 组件，不过 1 次仅能显示 1 个。ViewFlipper 可以设置每个子组件显示出来的间隔时间。

59 行：设置每隔 1500 毫秒显示 1 个子组件。

TweenAnimEx/src/org/tweenAnimEx/TweenAnimEx.java

```
36.   private void findViews_anim01() {
37.      etUserName = (EditText)findViewById(R.id.etUserName);
38.      btnSubmit = (Button)findViewById(R.id.btnSubmit);
39.      btnSubmit.setOnClickListener(new OnClickListener() {
40.         @Override
41.         public void onClick(View v) {
42.             Animation anim = AnimationUtils.loadAnimation(
43.                         TweenAnimEx.this, R.anim.anim_edittext);
44.             etUserName.startAnimation(anim);
```

```
45.          }
46.       });
47.    }
```

42-43 行：载入“res/anim/anim_edittext.xml”动画配置文件。

44 行：将 etUserName（EditText 组件）套用并播放 anim 所代表的动画效果。

```
49.    private void findViews_anim02() {
50.       tvAnim = (TextView)findViewById(R.id.tvAnim);
51.       spInter = (Spinner)findViewById(R.id.spInter);
52.       spInter.setOnItemSelectedListener(new OnItemSelectedListener() {
53.          @Override
54.          public void onItemSelected(AdapterView parent,
55.                View view, int pos, long id) {
56.             View parentView = (View)tvAnim.getParent();
57.             TranslateAnimation anim = new TranslateAnimation(
58.                0.0f,
59.                parentView.getWidth() - parentView.getPaddingLeft() -
60.                parentView.getPaddingRight() - tvAnim.getWidth(),
61.                0.0f, 0.0f);
62.             anim.setDuration(1000);
63.             anim.setStartOffset(300);
64.             anim.setRepeatMode(Animation.RESTART);
65.             anim.setRepeatCount(Animation.INFINITE);
66.
67.             int inter_id = android.R.anim.accelerate_interpolator;
68.             switch (pos) {
69.                case 0:
70.                   inter_id = android.R.anim.accelerate_interpolator;
71.                   break;
72.                case 1:
73.                   inter_id = android.R.anim.decelerate_interpolator;
74.                   break;
75.                case 2:
76.                   inter_id = android.R.anim.accelerate_decelerate_interpolator;
77.                   break;
78.                case 3:
79.                   inter_id = android.R.anim.anticipate_interpolator;
80.                   break;
81.                case 4:
82.                   inter_id = android.R.anim.overshoot_interpolator;
83.                   break;
84.                case 5:
85.                   inter_id = android.R.anim.anticipate_overshoot_interpolator;
86.                   break;
87.                case 6:
88.                   inter_id = android.R.anim.bounce_interpolator;
89.                   break;
90.             }
```

```
91.             anim.setInterpolator(AnimationUtils.loadInterpolator(
92.                   TweenAnimEx.this, inter_id));
93.             tvAnim.startAnimation(anim);
94.          }
95.
96.          @Override
97.          public void onNothingSelected(AdapterView parent) {}
98.       });
99.    }
```

57 行：TranslateAnimation (float fromXDelta, float toXDelta, float fromYDelta, float toYDelta) 属于位移补间动画，建构式的 4 个参数分别代表：fromXDelta——位移开始时组件的 X 轴坐标；toXDelta——位移结束时组件的 X 轴坐标；fromYDelta——位移开始时组件的 Y 轴坐标；toYDelta——位移结束时组件的 Y 轴坐标。

58-61 行：只有第 2 个参数值不是 0.0 代表组件作水平移动，而“parentView.getWidth() - parentView.getPaddingLeft() - parentView.getPaddingRight() - tvAnim.getWidth()”是指“父组件的宽度 减 父组件左边填充宽度 减 父组件右边填充宽度 减 动画组件宽度”，剩下的宽度即为位移的宽度；这代表组件位移时，从 X 轴 0.0 的位置向右移至不会超出父组件的极限，因为不希望动画组件向右位移时被父组件遮蔽。

62 行：动画持续播放 1000 毫秒。

63 行：主动画开始 300 毫秒后才运行此动画。

64 行：设置重复播放（Animation.RESTART）动画。

65 行：设置无限（Animation.INFINITE）重复播放动画。

68-90 行：根据 Spinner 选择的结果决定要采用何种内建的 interpolator 特效。“android.”开头代表系统内置。

```
101.   private void findViews_anim03() {
102.      fpText = (ViewFlipper)findViewById(R.id.fpText);
103.      fpText.startFlipping();
104.      spAnim = (Spinner)findViewById(R.id.spAnim);
105.      spAnim.setOnItemSelectedListener(new OnItemSelectedListener() {
106.         @Override
107.         public void onItemSelected(AdapterView parent,
108.               View view, int pos, long id) {
109.
110.            int anim_in = R.anim.translate_up_in;
111.            int anim_out = R.anim.translate_up_out;
112.            switch (pos) {
113.               case 0:
114.                  anim_in = R.anim.translate_up_in;
115.                  anim_out = R.anim.translate_up_out;
116.                  break;
117.               case 1:
118.                  anim_in = R.anim.translate_left_in;
```

```
119.                    anim_out = R.anim.translate_left_out;
120.                    break;
121.                case 2:
122.                    anim_in = android.R.anim.fade_in;
123.                    anim_out = android.R.anim.fade_out;
124.                    break;
125.                case 3:
126.                    anim_in = R.anim.abstract_in;
127.                    anim_out = R.anim.abstract_out;
128.                    break;
129.            }
130.            fpText.setInAnimation(AnimationUtils.loadAnimation(
131.                    TweenAnimEx.this, anim_in));
132.            fpText.setOutAnimation(AnimationUtils.loadAnimation(
133.                    TweenAnimEx.this, anim_out));
134.        }
135.
136.        @Override
137.        public void onNothingSelected(AdapterView parent) {}
138.    });
139.  }
```

112-129 行：根据 Spinner 选择结果决定要采用何种自定义特效配置文件案（放在该项目的“res/anim”目录内）。