```
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
<uses-sdk android:minSdkVersion="7" />
</manifest>
```

5. 在 Android 模拟器中运行项目，将显示主菜单界面，单击图标后，将显示对应的信息提示对话框，如图 3.3 所示。

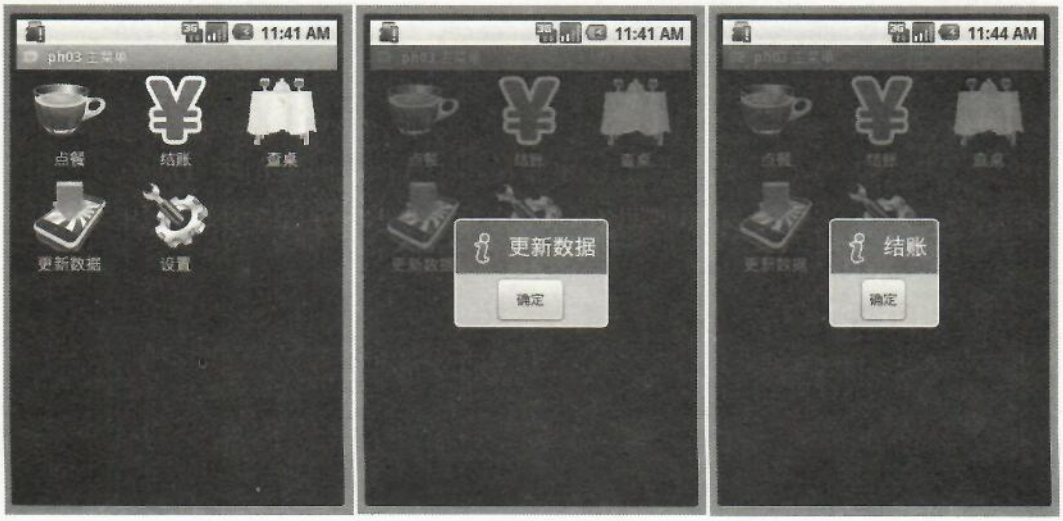

图 3.3

知识拓展

1. 自动完成文本框

自动完成文本框 AutoCompleteTextView 提供了辅助输入的功能，用户只需输入几个文字，搜索框就会显示相近的关键字。AutoCompleteTextView 可以对用户输入的文本进行有效的扩充提示，而不需要用户输入整个文本内容，只需输入一部分内容，其余的内容系统就会即时显示出来给予提示。

下列内容使用 AutoCompleteTextView 完成电话号码的自动录入功能，编写布局文件如下：

```
<?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">
    <AutoCompleteTextView
```

```
        android:id="@+id/autoCompleteTextView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="请输入电话号码" />
</LinearLayout>
```

上述布局文件中定义了一个 AutoCompleteTextView，再编写 Activity 代码如下：

```
public class AutoInput extends Activity {
    private static final String[] phonenumberStr = new String[] { "88888888",
            "86668888", "7777777", "86666666", "86145689", "86143429",
            "86123889" };

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

        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_dropdown_item_1line, phonenumberStr);
        AutoCompleteTextView autoCompleteTextView = (AutoCompleteTextView)
findViewById(R.id.autoCompleteTextView);
        autoCompleteTextView.setAdapter(adapter);
    }
}
```

上述代码中，使用字符串数组存储了一组电话号码，使用 ArrayAdapter 作为 AutoCompleteTextView 的适配器，当用户输入字符串时，AutoCompleteTextView 就会在 ArrayAdapter 中查找字符串，并显示匹配的字符串供用户选择，运行结果如图 3.4 所示。

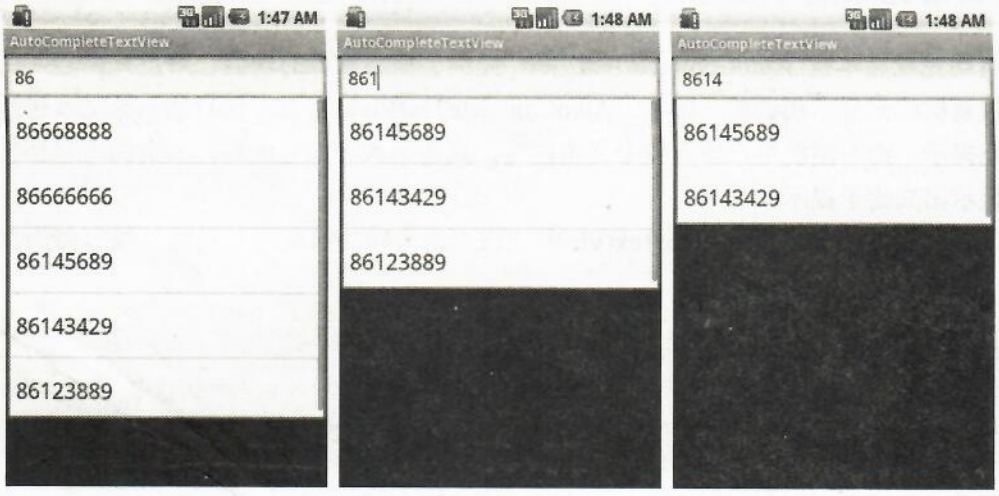

图 3.4

2. 时间相关的控件

Android 提供的与日期、时间有关的控件主要有以下四个。

- DatePicker：用于选择日期；
- TimePicker：用于选择时间；
- DigitalClock：数字时钟；
- AnalogClock：表状时钟。

这些控件的使用都比较简单，例如，下列布局文件定义了这四种控件：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:a="http://schemas.android.com/apk/res/android"
    a:orientation="vertical"
    a:layout_width="fill_parent"
    a:layout_height="fill_parent">
    <DatePicker
        a:id="@+id/dp"
        a:layout_width="fill_parent" a:layout_height="wrap_content"/>
    <TimePicker
        a:id="@+id/tp"
        a:layout_width="fill_parent" a:layout_height="wrap_content"/>
    <DigitalClock
        a:id="@+id/dc"
        a:layout_width="fill_parent" a:layout_height="wrap_content"/>
    <LinearLayout
        a:layout_width="fill_parent" a:layout_height="wrap_content">
        <AnalogClock
            a:id="@+id/ac"
            a:layout_width="wrap_content" a:layout_height="wrap_content"/>
        <TextView
            a:id="@+id/tv" a:textSize="30sp"
            a:layout_width="wrap_content" a:layout_height="wrap_content"/>
    </LinearLayout>
</LinearLayout>
```

编写 Activity 代码如下：

```
public class A extends Activity {

    DatePicker dp;
    TimePicker tp;
    DigitalClock dc;
```

```
    AnalogClock ac;
    TextView tv;

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

        dp = (DatePicker) findViewById(R.id.dp);
        tp = (TimePicker) findViewById(R.id.tp);
        dc = (DigitalClock) findViewById(R.id.dc);
        ac = (AnalogClock) findViewById(R.id.ac);
        tv = (TextView) findViewById(R.id.tv);

        Calendar c = Calendar.getInstance();
        dp.init(c.get(Calendar.YEAR), c.get(Calendar.MONTH),
                c.get(Calendar.DAY_OF_MONTH), new OnDateChangedListener() {
                    @Override
                    public void onDateChanged(DatePicker view, int year,
                            int monthOfYear, int dayOfMonth) {
                        display();
                    }
                });

        tp.setOnTimeChangedListener(new OnTimeChangedListener() {
            @Override
            public void onTimeChanged(TimePicker view, int hourOfDay, int minute){
                display();
            }
        });
    }

    void display() {
        int y = dp.getYear();
        int m = dp.getMonth() + 1;
        int d = dp.getDayOfMonth();
        int h = tp.getCurrentHour();
        int mi = tp.getCurrentMinute();
        int s = Calendar.getInstance().get(Calendar.SECOND);
        tv.setText(y + "-" + m + "-" + d + " " + h + ":" + mi + ":" + s);
    }
}
```


上述代码中，
了 OnTimeChang
DatePicker 和 Tir
运行项目，结果
单击 DatePi
在右下方的 Text

3. 进度条

进度条是常
条控件通常有两

- 圆形进
- 条形进

如果操作的
用圆形进度条；
ProgressBar
其他线程中调用
程中不能直接调
息来实现。下列

```
<?xml version
<LinearLayout
    a:orienta
    a:layout_
    <Button
```

上述代码中，为 DatePicker 控件设置了 OnDateChangedListener，为 TimePicker 控件设置了 OnTimeChangedListener，当用户修改 DatePicker 和 TimePicker 控件的值时，将调用 DatePicker 和 TimePicker 的相关方法获取日期和时间，并在 TextView 中显示修改后的结果。运行项目，结果如图 3.5 所示。

单击 DatePicker 和 TimePicker 控件上下的“+”和“-”按钮，可以修改日期和时间，并在右下方的 TextView 中显示修改后的值，如图 3.6 所示。

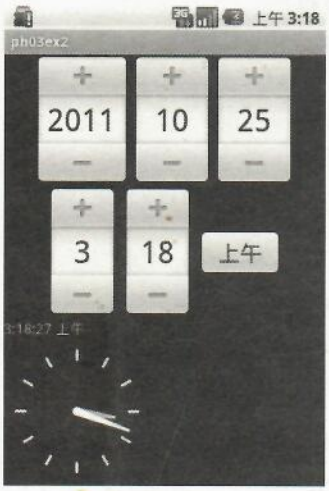

图 3.5

图 3.6

3. 进度条

进度条是常用的控件，Android 提供了 ProgressBar 控件用于表示耗时操作的进度。进度条控件通常有两种表现形式。

- 圆形进度条；
- 条形进度条。

如果操作的进度无法量化或无法预知，即在操作完成前无法得知当前的进度，则通常使用圆形进度条；反之则使用条形进度条，因为条形进度条可以直观地显示当前的完成情况。

ProgressBar 的 getProgress()和 setProgress()方法用于获取和更改当前的进度，可以直接在其他线程中调用，但是如果需要更改 ProgressBar 的显示状态（例如使其隐藏），则在其他线程中不能直接调用 ProgressBar 的相关方法，而应该使用 Handler 和 Message 对象通过传递消息来实现。下列内容演示了进度条的用法，编写布局文件如下：

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

```
        a:id="@+id/startBtn" a:text="开始"
        a:layout_width="fill_parent" a:layout_height="wrap_content"/>
    <ProgressBar
        a:id="@+id/pb1" a:visibility="gone"
        a:layout_width="wrap_content" a:layout_height="wrap_content"/>
    <ProgressBar
        a:id="@+id/pb2" a:visibility="gone"
        style="?android:attr/progressBarStyleSmall"
        a:layout_width="wrap_content" a:layout_height="wrap_content"/>
    <ProgressBar
        a:id="@+id/pb3" a:visibility="gone"
        style="?android:attr/progressBarStyleLarge"
        a:layout_width="wrap_content" a:layout_height="wrap_content"/>
    <ProgressBar
        a:id="@+id/pb4" a:visibility="gone"
        style="?android:attr/progressBarStyleHorizontal"
        a:layout_width="fill_parent" a:layout_height="wrap_content"/>
</LinearLayout>
```

上述布局文件定义了四个进度条，第一个没有指定 style 属性，采用默认样式，即中等大小的圆形进度条；第二个 style 值为“?android:attr/progressBarStyleSmall”，是小的圆形进度条；第三个 style 值为“?android:attr/progressBarStyleLarge”，是大的圆形进度条；第四个值为“?android:attr/progressBarStyleHorizontal”，是条形进度条。四个进度条的 visibility 值为“gone”，因此初始都是不显示的。编写 Activity 代码如下：

```
public class A extends Activity {

    ProgressBar pb1;
    ProgressBar pb2;
    ProgressBar pb3;
    ProgressBar pb4;
    Button startBtn;

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

        pb1 = (ProgressBar) findViewById(R.id.pb1);
        pb2 = (ProgressBar) findViewById(R.id.pb2);
        pb3 = (ProgressBar) findViewById(R.id.pb3);
        pb4 = (ProgressBar) findViewById(R.id.pb4);
        startBtn = (Button) findViewById(R.id.startBtn);
```

```
final int max = pb4.getMax(); // 最大值

final Handler handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        int p = msg.what;
        // 如果达到最大值，隐藏圆形进度条
        if (p == max) {
            pb1.setVisibility(View.GONE);
            pb2.setVisibility(View.GONE);
            pb3.setVisibility(View.GONE);
        }
        pb4.setProgress(p); // 修改条形进度条的进度
        pb4.setSecondaryProgress(p * 2); // 修改条形进度条的第二进度
    };
};

startBtn.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        // 显示四个进度条
        pb1.setVisibility(View.VISIBLE);
        pb2.setVisibility(View.VISIBLE);
        pb3.setVisibility(View.VISIBLE);
        pb4.setVisibility(View.VISIBLE);

        // 模拟耗时操作
        new Thread() {
            @Override
            public void run() {
                for (int i = 1; i <= max; i++) {
                    try {
                        Thread.sleep(50);
                        Message msg = new Message();
                        msg.what = i;
                        handler.sendMessage(msg); // 发送消息
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }.start();
```

```
            }
        });
    }
}
```

上述代码中，定义了 Handler 对象，并覆盖 handleMessage()方法，其中根据消息的内容改变四个进度条的状态；按钮的单击事件中，新线程模拟了一个耗时操作，操作过程中会定时向 Handler 对象发送消息；当 Handler 对象接收到消息时，会调用其 handleMessage()方法，从而改变进度条的状态。运行项目，初始界面不会显示进度条；单击“开始”按钮后，四个进度条出现并开始滚动；模拟的操作执行完毕后，圆形进度条被隐藏，只显示已经完成的条形进度条。效果如图 3.7 所示。

图 3.7

4. 拖动条

拖动条控件 SeekBar 与普通进度条 ProgressBar 类似（SeekBar 继承于 ProgressBar），但是支持用户拖动修改进度，下列内容演示了 SeekBar 的用法。

编写布局文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:a="http://schemas.android.com/apk/res/android"
    a:orientation="vertical" a:gravity="center_horizontal"
    a:layout_width="fill_parent" a:layout_height="fill_parent">
    <SeekBar
        a:id="@+id/sb"
        a:layout_width="fill_parent" a:layout_height="wrap_content"/>
    <TextView
        a:id="@+id/tv" a:textSize = "30sp"
        a:layout_width="fill_parent" a:layout_height="wrap_content"/>
```

```
</LinearLayout>
```

上述文件定义了一个 SeekBar 和一个 TextView，编写 Activity 代码如下：

```
public class A extends Activity {

    SeekBar sb;
    TextView tv;

    boolean finished; // 是否已完成
    boolean dragging; // 是否正在拖动

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

        sb = (SeekBar) findViewById(R.id.sb);
        tv = (TextView) findViewById(R.id.tv);
        final int max = sb.getMax(); // 最大值

        sb.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
                dragging = false; // 停止拖动
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
                dragging = true; // 开始拖动
            }

            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
                    boolean fromUser) {
                int p = sb.getProgress();
                tv.setText("进度:" + p * 100 / max + "%");
                if (p == max)
                    finished = true;
            }
        });

        new Thread() {
```

```
        @Override
        public void run() {
            while (!finished)
                if (!dragging)
                    try {
                        Thread.sleep(50);
                        sb.incrementProgressBy(1);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
        }
    }.start();
  }
}
```

上述代码中，模拟了耗时的操作，其中循环判断当没有完成时，如果没有再拖动，则修改 SeekBar 的进度。为 SeekBar 注册了状态改变监听器 OnSeekBarChangeListener，当用户开始拖动、停止拖动、进度发生改变时，会调用 OnSeekBarChangeListener 的 onStartTrackingTouch()、onStopTrackingTouch()以及 onProgressChanged()方法；在 onStartTrackingTouch()和 onStopTrackingTouch()方法中修改拖动标志的值，在 onProgressChanged()方法中更新 TextView 的显示，并判断当达到终点时修改完成标志，从而使模拟耗时操作的线程结束。

运行项目，将显示状态持续变化的 SeekBar，如图 3.8 所示。

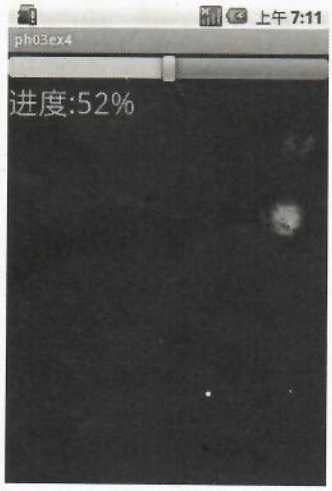

图 3.8

可以向前、向后拖动 SeekBar，拖动的过程中，进度会停止前进，松开拖动后，进度会从当前位置继续前进。

5. 图片切换效果

使用 Android 提供的 ImageSwitcher 和 Gallery 类可以方便地实现常见的图片切换效果，ImageSwitcher 通常用于显示并切换图片，而 Gallery 是存储图片的一个容器。编写布局文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:a="http://schemas.android.com/apk/res/android"
    a:orientation="vertical"
    a:layout_width="fill_parent" a:layout_height="fill_parent">
    <ImageSwitcher
        a:id="@+id/imgSwt"
        a:layout_width="fill_parent" a:layout_height="wrap_content"/>
    <Gallery
        a:id="@+id/glr"
        a:layout_width="fill_parent" a:layout_height="wrap_content"
        a:gravity="center" a:spacing="0dp"/>
</LinearLayout>
```

上述文件中，定义了一个 ImageSwitcher 和一个 Gallery。编写 Activity 代码如下：

```
public class A extends Activity {

    ImageSwitcher imgSwt;
    Gallery glr;

    // 需要显示的图片资源，保存在 drawable 目录下
    int[] imgs = { R.drawable.a, R.drawable.b, R.drawable.c, R.drawable.d,
            R.drawable.e, R.drawable.f, R.drawable.g };

    // 缓存底部在 Gallery 中使用的 ImageView
    ImageView[] galleryViews = new ImageView[imgs.length];

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

        imgSwt = (ImageSwitcher) findViewById(R.id.imgSwt);
        glr = (Gallery) findViewById(R.id.glr);

        // 设置 ImageSwitcher 显示图片的工厂
        imgSwt.setFactory(new ViewFactory() {
```

```
        @Override
        public View makeView() {
            ImageView iv = new ImageView(A.this);
            iv.setBackgroundColor(0xFF000000);
            iv.setScaleType(ImageView.ScaleType.FIT_CENTER);
            iv.setLayoutParams(new ImageSwitcher.LayoutParams(
                    LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT));
            return iv;
        }
    });
    // 设置 ImageSwitcher 的进入动画
    imgSwt.setInAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.fade_in));
    // 设置 ImageSwitcher 的退出动画
    imgSwt.setOutAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.fade_out));

    // 初始化缓存 Gallery 使用的 ImageView
    for (int i = 0; i < galleryViews.length; i++) {
        ImageView iv = new ImageView(A.this);
        iv.setImageResource(imgs[i]);
        iv.setAdjustViewBounds(true);
        iv.setLayoutParams(new Gallery.LayoutParams(
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
        galleryViews[i] = iv;
    }
    // 设置 Gallery 的适配器
    glr.setAdapter(new BaseAdapter() {
        @Override
        public int getCount() {
            return imgs.length;
        }

        @Override
        public Object getItem(int position) {
            return null;
        }

        @Override
        public long getItemId(int position) {
            return 0;
        }
```



```
        });
        // G
        glr.
        });
    }
}
```

上述代码

- Image
- Galle

更新

运行项目

```
            @Override
            public View getView(int position, View convertView, ViewGroup parent){
                return galleryViews[position];
            }
        });
        // Gallery 被选中的事件
        glr.setOnItemSelectedListener(new OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> arg0, View arg1,
                    int arg2, long arg3) {
                imgSwt.setImageResource(imgs[arg2]);
            }

            @Override
            public void onNothingSelected(AdapterView<?> arg0) {
                // TODO
            }
        });
    }
}
```

上述代码中，主要针对 ImageSwitcher 和 Gallery 对象执行了下列操作。

- ImageSwitcher：设置显示图片的工厂、进入动画和退出动画；
- Gallery：设置适配器，其中主要需要覆盖 getView()方法；设置选中事件，其中需要更新 ImageSwitcher 显示的图片。

运行项目，将显示第一张图片，效果如图 3.9 所示。

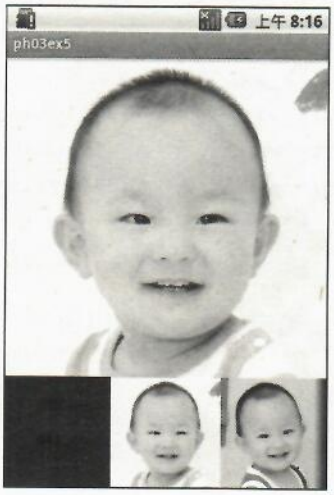

图 3.9

单击图 3.9 下方的小图片，上部将显示对应的图片，如图 3.10 所示。

图 3.10

6. 进度对话框

进度对话框实际上就是一个包含进度条的对话框，其使用方式与单纯的进度条非常类似，也有两种表现形式。

- 圆形进度条对话框；
- 条形进度条对话框。

下列内容演示了进度对话框的用法，编写布局文件如下：

```
<?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">
    <Button android:id="@+id/startBtn1" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="圆形进度条对话框" />
    <Button android:id="@+id/startBtn2" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="条形进度条对话框" />
</LinearLayout>
```

上述布局中声明了两个按钮，编写 Activity 代码如下：

```
public class A extends Activity {
    Button startBtn1;
    Button startBtn2;
    ProgressDialog pd1;
    ProgressDialog pd2;
```

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

    startBtn1 = (Button) findViewById(R.id.startBtn1);
    startBtn2 = (Button) findViewById(R.id.startBtn2);
    pd1 = new ProgressDialog(this);
    pd2 = new ProgressDialog(this);

    pd1.setMessage("请稍候......");
    pd2.setMessage("请稍候......");
    pd2.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);//设置进度条为条形

    final int max = 100;// 进度最大值
    final Handler handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            int p = msg.what;
            // 如果达到最大值，隐藏进度条对话框
            if (p == max) {
                pd1.hide();
                pd2.hide();
            }
            // 修改条形进度条对话框的进度
            pd2.setProgress(p);
        };
    };

    startBtn1.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            pd1.show();
            new Thread() {
                @Override
                public void run() {
                    for (int i = 1; i <= max; i++) {
                        try {
                            Thread.sleep(50);
                            Message msg = new Message();
                            msg.what = i;
                            handler.sendMessage(msg); // 发送消息
```

```
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
            }.start();
        }
    });

    startBtn2.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            pd2.show();
            new Thread() {
                @Override
                public void run() {
                    for (int i = 1; i <= max; i++) {
                        try {
                            Thread.sleep(50);
                            Message msg = new Message();
                            msg.what = i;
                            handler.sendMessage(msg); // 发送消息
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
            }.start();
        }
    });
  }
}
```

上述代码中，定义了两个进度条对话框 pd1 和 pd2，其中 pd2 通过调用 setProgressStyle (ProgressDialog.STYLE_HORIZONTAL)使其显示条形进度条，pd1 则采用默认的圆形进度条。与普通的进度条类似，在按钮单击事件中模拟耗时操作，并通过 Handler 对象发送 Message，在 Handler 对象的 handleMessage()方法中，根据消息的内容改变 pd1 和 pd2 的状态。运行项目，分别单击两个按钮，将显示两种风格的进度对话框，如图 3.11 所示。

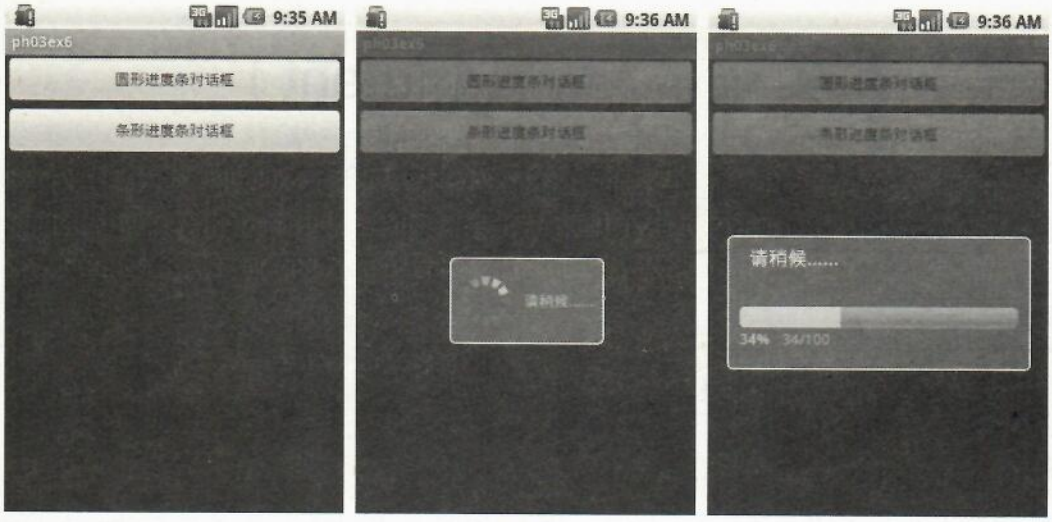

图 3.11

拓展练习

练习 3.E.1

修改进度条知识拓展中的代码，使进度条到达最大值后再逐渐缩小，缩小到最小之后，再逐渐变大，如此循环执行。

练习 3.E.2

修改拖动条知识拓展中的代码，添加下列四个按钮，并完成对应的功能。

- 暂停：使进度暂停；
- 执行：继续执行；
- 加速：执行速度变为原来的 2 倍；
- 减速：执行速度变为原来的一半。

练习 3.E.3

修改图片切换效果知识拓展中的代码，使下部 Gallery 中的图片自动
部图片也相应地自动切换。

```
                                                    .ame">
                                                .ion.MAIN" />
                                            .t.category.LAUNCHER" />


                                        .ty"></activity>
```