第5章 第二个activity

本章，我们将为GeoQuiz应用增加第二个activity。activity控制着当前屏幕界面，新增加的activity将增加第二个用户界面，以方便用户查看当前问题的答案，如图5-1所示。

图5-1 CheatActivity提供了偷看答案的机会

如用户选择先查看答案，然后再返回QuizActivity回答问题，则会收到一条新的信息，如图5-2所示。

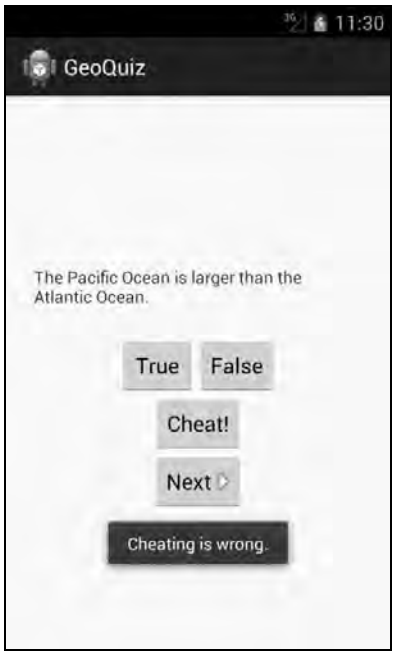

图5-2　有没有偷看答案，别想瞒过QuizActivity

通过本章GeoQuiz应用的升级开发，我们可以从中学到以下知识点。

- 不借助应用向导，创建新的activity及配套布局。
- 从一个activity中启动另一个activity。启动activity意味着请求操作系统创建新的activity实例并调用它的onCreate(Bundle)方法。
- 在父activity（启动方）与子activity（被启动方）间进行数据传递。

5.1　创建第二个 activity

要创建新的activity，接下来要做的事不少。首先创建CheatActivity所需的布局文件，然后创建CheatActivity类本身。

不过，现在我们还是先打开strings.xml文件，添加本章需要的所有字符串资源，如代码清单5-1所示。

代码清单5-1　添加字符串资源（strings.xml）

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

  ...
  <string name="question_asia">Lake Baikal is the world\'s oldest and deepest
      freshwater lake.</string>
  <string name="cheat_button">Cheat!</string>
```

```
    <string name="warning_text">Are you sure you want to do this?</string>
    <string name="show_answer_button">Show Answer</string>
    <string name="judgment_toast">Cheating is wrong.</string>

</resources>
```

5.1.1 创建新布局

本章开头的屏幕截图展示了CheatActivity视图的大致样貌。图5-3展示了它的组件定义。

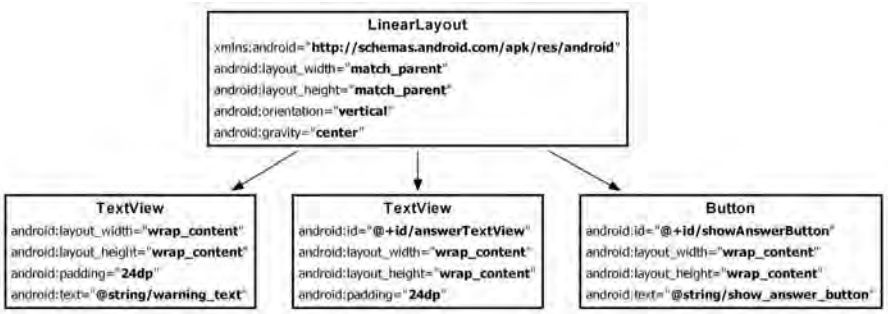

图5-3 CheatActivity的布局图示

为创建布局文件，在包浏览器中右键单击res/layout目录，选择New → Other...菜单项。在Android文件夹里，找到并选择Android XML Layout File，如图5-4所示。然后单击Next按钮。

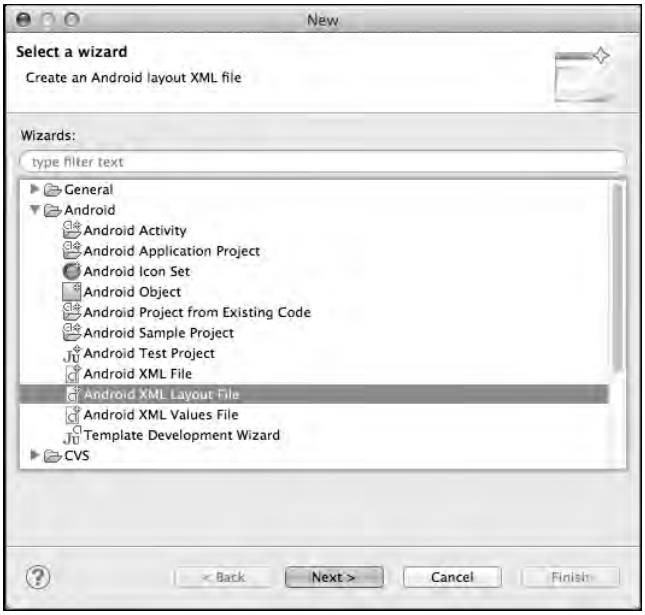

图5-4 创建新的布局文件

在接下来弹出的对话框中，输入布局文件名activity_cheat.xml并选择LinerLayout作为根元素，最后单击Finish按钮完成，如图5-5所示。

图5-5　命名并配置新布局文件

观察已打开的activity_cheat.xml布局文件，我们发现该XML文件头部包含了以下一行代码：

```
<?xml version="1.0" encoding="utf-8"?>
```

XML布局文件不再需要该行代码。不过，通过布局向导等方式创建布局文件，这一行代码还是会被默认添加。

（如不习惯GUI的开发方式，可不使用布局向导。例如，要创建新布局文件，可直接在res/layout目录新建activity_cheat.xml文件，然后刷新res/layout目录让Eclipse识别它。该做法适用于大多数Eclipse开发向导。我们可按照自己的方式创建XML文件以及Java类文件。记住，唯一必须使用的开发向导是新建Android应用向导。）

布局向导已经添加了LinerLayout根元素。接下来只需添加一个android:gravity属性和其他三个子元素即可。

第8章以后，我们将不再列示大段的XML代码，而仅以图5-3的方式给出布局组件图示。最好现在就开始习惯参照图5-3创建布局XML文件。完成创建activity_cheat.xml布局文件后，记得对照代码清单5-2进行检查核对。

代码清单5-2 第二个activity的布局组件定义（activity_cheat.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"
  android:gravity="center">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="24dp"
    android:text="@string/warning_text" />

  <TextView
    android:id="@+id/answerTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="24dp" />

  <Button
    android:id="@+id/showAnswerButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/show_answer_button" />

</LinearLayout>
```

保存布局文件，切换到图形工具模式预览新建布局。

虽然没有创建供设备横屏使用的布局文件，不过，借助开发工具，我们可以预览默认布局横屏时的显示效果。

在图形布局工具中，找到预览界面上方工具栏里的一个设备（带绿色箭头）模样的按钮。单击该按钮切换布局预览方位，如图5-6所示。

图5-6 水平方位预览布局（activity_cheat.xml）

可以看到，默认布局在竖直与水平方位下效果都不错。布局搞定了，接下来我们来创建新的activity子类。

5.1.2　创建新的activity子类

在包浏览器中，右键单击com.bignerdranch.android.geoquiz包，选择New → Class菜单项。

在随后弹出的对话框中，将类命名为`CheatActivity`。在Superclass栏输入android.app.Activity，如图5-7所示。

图5-7　创建`CheatActivity`类

点击Finish按钮，Eclipse随即在代码编辑区打开了CheatActivity.java文件。

覆盖`onCreate(...)`方法，将定义在activity_cheat.xml文件中的布局资源ID传入`setContentView(...)`方法，如代码清单5-3所示。

代码清单5-3　覆盖`onCreate(...)`方法（CheatActivity.java）

```
public class CheatActivity extends Activity {

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cheat);
    }

}
```

CheatActivity还有更多任务需要在onCreate(...)方法中完成。不过现在我们先进入下一环节，即在应用的manifest配置文件中声明CheatActivity。

5.1.3 在manifest配置文件中声明activity

manifest配置文件是一个包含元数据的XML文件，用来向Android操作系统描述应用。该文件总是以AndroidManifest.xml命名，可在项目的根目录找到它。

通过包浏览器，在项目的根目录中找到并打开它。忽略GUI编辑器，选择编辑区底部的AndroidManifest.xml标签切换到代码展示界面。

应用的所有activity都必须在manifest配置文件中声明，这样操作系统才能够使用它们。

创建QuizActivity时，因使用了新建应用向导，向导已自动完成声明工作。而CheatActivity则需手工完成声明工作。

在AndroidManifest.xml配置文件中，完成CheatActivity的声明，如代码清单5-4所示。

代码清单5-4 在manifest配置文件中声明CheatActivity（AndroidManifest.xml）

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.bignerdranch.android.geoquiz"
  android:versionCode="1"
  android:versionName="1.0" >

  <uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="17" />

  <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
      android:name="com.bignerdranch.android.geoquiz.QuizActivity"
      android:label="@string/app_name" >
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <activity
      android:name=".CheatActivity"
      android:label="@string/app_name" />
  </application>

</manifest>
```

这里的android:name属性是必需的。属性值前面的“.”可告知OS：在manifest配置文件头部包属性值指定的包路径下，可以找到activity的类文件。

manifest配置文件里还有很多有趣的东西。不过，我们现在还是先集中精力把CheatActivity配置并运行起来吧。在后续章节中，我们还将学习到更多有关manifest配置文件的知识。

5.1.4　为QuizActivity添加cheat按钮

按照开发设想，用户在QuizActivity用户界面上点击某个按钮，应用立即产生CheatActivity实例，并显示其用户界面。因此，我们需要在layout/activity_quiz.xml以及 layout-land/activity_quiz.xml布局文件中定义需要的按钮。

在默认的垂直布局中，添加新按钮定义并设置其为根LinearLayout的直接子类。新按钮应该定义在Next按钮之前，按钮添加方法如代码清单5-5所示。

代码清单5-5　默认布局中添加cheat按钮（ layout/activity_quiz.xml ）

```
    ...
</LinearLayout>

<Button
  android:id="@+id/cheat_button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="@string/cheat_button" />

<Button
  android:id="@+id/next_button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="@string/next_button" />

</LinearLayout>
```

在水平布局模式中，将新按钮定义在根FrameLayout的底部居中位置，如代码清单5-6所示。

代码清单5-6　水平布局中添加cheat按钮（ layout-land/activity_quiz.xml ）

```
    ...
</LinearLayout>

<Button
  android:id="@+id/cheat_button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_gravity="bottom|center"
  android:text="@string/cheat_button" />

<Button
  android:id="@+id/next_button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_gravity="bottom|right"
  android:text="@string/next_button"
```

```
        android:drawableRight="@drawable/arrow_right"
        android:drawablePadding="4dp" />

    </FrameLayout>
```

保存修改后的布局文件。然后重新打开QuizActivity.java文件，添加新按钮变量以及资源引用代码。最后再添加View.onClickListener监听器代码存根。启用新按钮的做法如代码清单5-7所示。

代码清单5-7 启用Cheat按钮（QuizActivity.java）

```
public class QuizActivity extends Activity {

    ...

    private Button mNextButton;
    private Button mCheatButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        ...

        mCheatButton = (Button)findViewById(R.id.cheat_button);
        mCheatButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // Start CheatActivity
            }
        });

       updateQuestion();
    }

 ...

}
```

准备工作完成了，下面我们来学习如何启动CheatActivity。

5.2 启动 activity

一个activity启动另一个activity最简单的方式是使用以下Activity方法：

```
public void startActivity(Intent intent)
```

我们可能会以为startActivity(...)方法是一个类方法，启动activity就是针对Activity子类调用该方法。实际并非如此。activity调用startActivity(...)方法时，调用请求实际发给了操作系统。

准确地说，该方法调用请求是发送给操作系统的ActivityManager。ActivityManager负责创建Activity实例并调用其onCreate(...)方法。activity的启动示意图如图5-8所示。

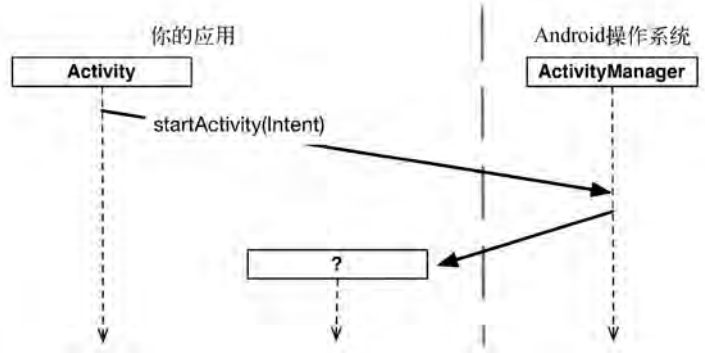

图5-8　启动activity

ActivityManager如何知道该启动哪一个Activity呢？答案就在于传入startActivity(...)方法的Intent参数。

基于intent的通信

intent对象是component用来与操作系统通信的一种媒介工具。目前为止，我们唯一见过的component就是activity。实际上还有其他一些component：service、broadcast receiver以及content provider。

Intent是一种多功能通信工具。Intent类提供了多个构造方法，以满足不同的使用需求。

在GeoQuiz应用中，我们使用intent告知ActivityManager该启动哪一个activity，因此可使用以下构造方法：

```
public Intent(Context packageContext, Class<?> cls)
```

传入该方法的Class对象指定ActivityManager应该启动的activity；Context对象告知ActivityManager在哪一个包里可以找到Class对象，关系图如图5-9所示。

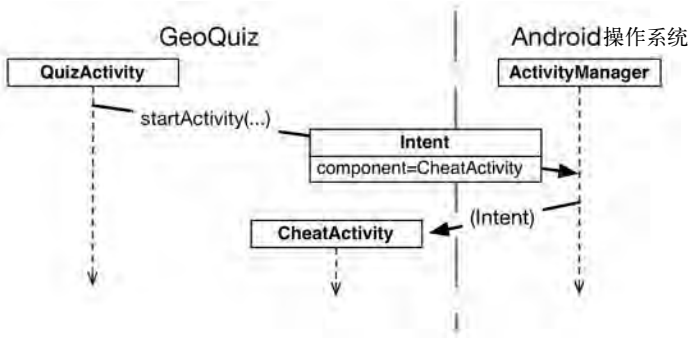

图5-9　intent：ActivityManager的信使

在mCheatButton的监听器代码中，创建包含CheatActivity类的Intent实例，然后将其传入startActivity(Intent)方法，如代码清单5-8所示。

代码清单5-8 启动CheatActivity活动（QuizActivity.java）

```
        ...

        mCheatButton = (Button)findViewById(R.id.cheat_button);
        mCheatButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                Intent i = new Intent(QuizActivity.this, CheatActivity.class);
                startActivity(i);
            }
        });

        updateQuestion();
    }
```

在启动activity以前，ActivityManager会检查确认指定的Class是否已在配置文件中声明。如已完成声明，则启动activity，应用正常运行。反之，则抛出ActivityNotFoundException异常。这就是我们必须在manifest配置文件中声明应用全部activity的原因所在。

显式与隐式intent

如通过指定Context与Class对象，然后调用intent的构造方法来创建Intent，则创建的是显式intent。同一应用中，我们使用显式intent来启动activity。

同一应用里的两个activity间，通信却要借助于应用外部的ActivityManager，这可能看起来有点奇怪。不过，这种模式会使不同应用间的activity交互变得容易很多。

一个应用的activity如需启动另一个应用的activity，可通过创建隐式intent来处理。我们会在第21章学习到隐式intent的使用。

运行GeoQuiz应用。单击Cheat按钮，新activity实例的用户界面将显示在屏幕上。单击后退按钮，CheatActivity实例会被销毁，继而返回到QuizActivity实例的用户界面中。

5.3 activity 间的数据传递

既然CheatActivity与QuizActivity都已经就绪，接下来就可以考虑它们之间的数据传递了。图5-10展示了两个activity间传递的数据信息。

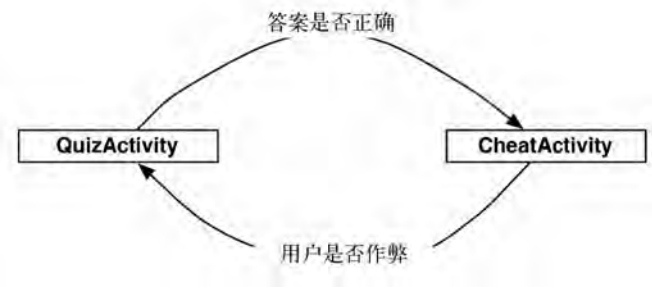

图5-10 QuizActivity与CheatActivity间的对话

CheatActivity启动后，QuizActivity会将当前问题的答案通知给它。

用户知道答案后，单击后退键回到QuizActivity，CheatActivity随即会被销毁。在被销毁前的瞬间，它会将用户是否作弊的数据传递给QuizActivity。

接下来，我们首先要学习的是如何将数据从QuizActivity传递到CheatActivity。

5.3.1　使用intent extra

为将当前问题答案通知给CheatActivity，需将以下语句的返回值传递给它：

```
mQuestionBank[mCurrentIndex].isTrueQuestion();
```

该值将作为extra信息，附加在传入startActivity(Intent)方法的Intent上发送出去。

extra信息可以是任意数据，它包含在Intent中，由启动方activity发送出去。接受方activity接收到操作系统转发的intent后，访问并获取包含在其中的extra数据信息。关系图如图5-11所示。

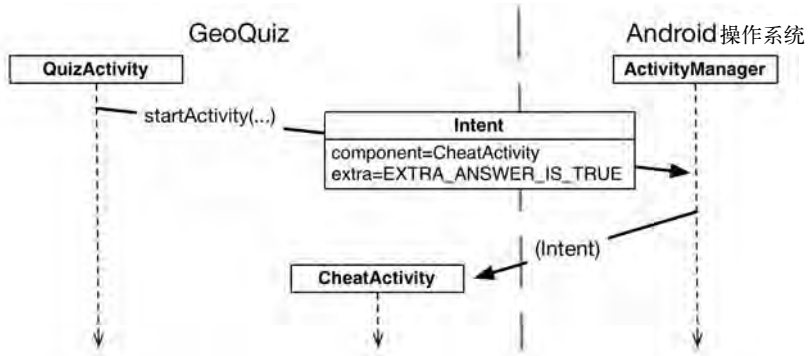

图5-11　Intent extra:activity间的通信与数据传递

如同QuizActivity.onSaveInstanceState(Bundle)方法中用来保存mCurrentIndex值的key-value结构，extra也同样是一种key-value结构。

将extra数据信息添加给intent，我们需要调用Intent.putExtra(...)方法。确切地说，是调用如下方法：

```
public Intent putExtra(String name, boolean value)
```

Intent.putExtra(...)方法有多种形式。不变的是，它总是有两个参数。一个参数是固定为String类型的key，另一个参数值可以是多种数据类型。

在CheatActivity.java中，为extra数据信息新增key-value对中的key，如代码清单5-9所示。

代码清单5-9　添加extra常量（CheatActivity.java）

```
public class CheatActivity extends Activity {

    public static final String EXTRA_ANSWER_IS_TRUE =
        "com.bignerdranch.android.geoquiz.answer_is_true";

    ...
```

activity可能启动自不同的地方，我们应该为activity获取和使用的extra定义key。如代码清单5-9所示，使用包名来修饰extra数据信息，这样可以避免来自不同应用的extra间发生命名冲突。

接下来，再回到QuizActivity，将extra附加到intent上，如代码清单5-10所示。

代码清单5-10 将extra附加到intent上（QuizActivity.java）

```
        ...
        mCheatButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                Intent i = new Intent(QuizActivity.this, CheatActivity.class);
                boolean answerIsTrue = mQuestionBank[mCurrentIndex].isTrueQuestion();
                i.putExtra(CheatActivity.EXTRA_ANSWER_IS_TRUE, answerIsTrue);
                startActivity(i);
            }
        });

        updateQuestion();
    }
```

这里只需一个extra。但如有需要，也可以附加多个extra到同一个Intent上。

要从extra获取数据，会用到如下方法：

```
public boolean getBooleanExtra(String name, boolean defaultValue)
```

第一个参数是extra的名字。getBooleanExtra(...)方法的第二个参数是指定默认值（默认答案），它在无法获得有效key值时使用。

在CheatActivity代码中，编写代码实现从extra中获取信息，然后将信息存入成员变量中，如代码清单5-11所示。

代码清单5-11 获取extra信息（CheatActivity.java）

```
public class CheatActivity extends Activity {

    ...

    private boolean mAnswerIsTrue;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cheat);

        mAnswerIsTrue = getIntent().getBooleanExtra(EXTRA_ANSWER_IS_TRUE, false);
    }

}
```

请注意，Activity.getIntent()方法返回了由startActivity(Intent)方法转发的Intent对象。

最后，在CheatActivity代码中，编码实现单击Show Answer按钮后可获取答案并将其显示在TextView上，如代码清单5-12所示。

代码清单5-12　启用作弊模式（CheatActivity.java）

```
public class CheatActivity extends Activity {

    ...

    private boolean mAnswerIsTrue;

    private TextView mAnswerTextView;
    private Button mShowAnswer;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cheat);

        mAnswerIsTrue = getIntent().getBooleanExtra(EXTRA_ANSWER_IS_TRUE, false);

        mAnswerTextView = (TextView)findViewById(R.id.answerTextView);

        mShowAnswer = (Button)findViewById(R.id.showAnswerButton);
        mShowAnswer.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (mAnswerIsTrue) {
                    mAnswerTextView.setText(R.string.true_button);
                } else {
                    mAnswerTextView.setText(R.string.false_button);
                }
            }
        });
    }

}
```

TextView相关的代码还是很直观的。可通过使用TextView.setText(int)方法来设置TextView要显示的文字。TextView.setText(int)方法有多种变体。这里，我们通过传入资源ID来调用该方法。

运行GeoQuiz应用。单击Cheat按钮弹出CheatActivity的用户界面。然后单击Show Answer按钮查看当前问题的答案。

5.3.2　从子activity获取返回结果

现在用户可以毫无顾忌地偷看答案了。如果CheatActivity可以把用户是否偷看过答案的情况通知给QuizAcitivity就更好了。下面我们来修正这个问题。

若需要从子activity获取返回信息时，可调用以下Activity方法：

```
public void startActivityForResult(Intent intent, int requestCode)
```

该方法的第一个参数同前述的intent。第二个参数是*请求代码*。请求代码是先发送给子activity，然后再返回给父activity的用户定义整数值。当一个activity启动多个不同类型的子activity，且需要判断区分消息回馈方时，我们通常会用到该请求代码。

在QuizActivity中，修改mCheatButton的监听器，调用startActivityForResult (Intent, int)方法，如代码清单5-13所示。

代码清单5-13 调用startActivityForResult(...)方法（QuizActivity.java）

```
        ...
        mCheatButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                Intent i = new Intent(QuizActivity.this, CheatActivity.class);
                boolean answerIsTrue = mQuestionBank[mCurrentIndex].isTrueQuestion();
                i.putExtra(CheatActivity.EXTRA_ANSWER_IS_TRUE, answerIsTrue);
                startActivity(i);
                startActivityForResult(i, 0);
            }
        });

        updateQuestion();
    }
```

QuizActivity只会启动一个类型的子activity。具体发送信息是什么都无所谓，因此对于需要的请求代码参数，传入0即可。

1. 设置返回结果

实现子activity发送返回信息给父activity，有以下两种方法可供调用：

```
public final void setResult(int resultCode)
public final void setResult(int resultCode, Intent data)
```

通常来说，参数result code可以是以下两个预定义常量中的任何一个：

- Activity.RESULT_OK;
- Activity.RESULT_CANCELED。

（如需自己定义结果代码，还可使用另一个常量：RESULT_FIRST_USER）

在父activity需要依据子activity的完成结果采取不同操作时，设置结果代码很有帮助。

例如，假设子activity有一个OK按钮及一个Cancel按钮，并且为每个按钮的单击动作分别设置了不同的结果代码。根据不同的结果代码，父activity会采取不同的操作。

子activity可以不调用setResult(...)方法。如不需要区分附加在intent上的结果或其他信息，可让操作系统发送默认的结果代码。如果子activity是以调用startActivityForResult(...)方法启动的，结果代码则总是会返回给父activity。在没有调用setResult(...)方法的情况下，如果用户单击了后退按钮，父activity则会收到Activity.RESULT_CANCELED的结果代码。

2. 返还intent

GeoQuiz应用中，数据信息需要回传给QuizActivity。因此，我们需要创建一个Intent，附加上extra信息后，调用Activity.setResult(int, Intent)方法将信息回传给QuizActivity。

前面，我们已经为CheatActivity接收的extra定义了常量。CheatActivity要回传信息给QuizActivity，我们同样需要为回传的extra做类似的定义。为什么不在接收信息的父activity中定义extra常量呢？这是因为，传入及传出extra针对CheatActivity定义了统一的接口。这样，如果在应用的其他地方使用CheatActivity，我们只需要关注使用定义在CheatActivity中的那些常量。

在CheatActivity代码中，为extra增加常量key，再创建一个私有方法，用来创建intent，附加extra并设置结果值。然后在Show Answer按钮的监听器代码中调用该方法。设置结果值的方法如代码清单5-14所示。

代码清单5-14　设置结果值（CheatActivity.java）

```
public class CheatActivity extends Activity {

    public static final String EXTRA_ANSWER_IS_TRUE =
        "com.bignerdranch.android.geoquiz.answer_is_true";
    public static final String EXTRA_ANSWER_SHOWN =
        "com.bignerdranch.android.geoquiz.answer_shown";

    ...

    private void setAnswerShownResult(boolean isAnswerShown) {
        Intent data = new Intent();
        data.putExtra(EXTRA_ANSWER_SHOWN, isAnswerShown);
        setResult(RESULT_OK, data);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...

        // Answer will not be shown until the user
        // presses the button
        setAnswerShownResult(false);
        ...
        mShowAnswer.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (mAnswerIsTrue) {
                    mAnswerTextView.setText(R.string.true_button);
                } else {
                    mAnswerTextView.setText(R.string.false_button);
                }
                setAnswerShownResult(true);
            }
        });
    }

}
```

用户单击Show Answer按钮时，CheatActivity调用setResult(int, Intent)方法将结果代码以及intent打包。

然后，在用户单击后退键回到QuizActivity时，ActivityManager调用父activity的以下方法：

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)
```

该方法的参数来自于QuizActivity的原始请求代码以及传入SetResult(...)方法的结果代码和intent。

图5-12展示了应用内部的交互时序。

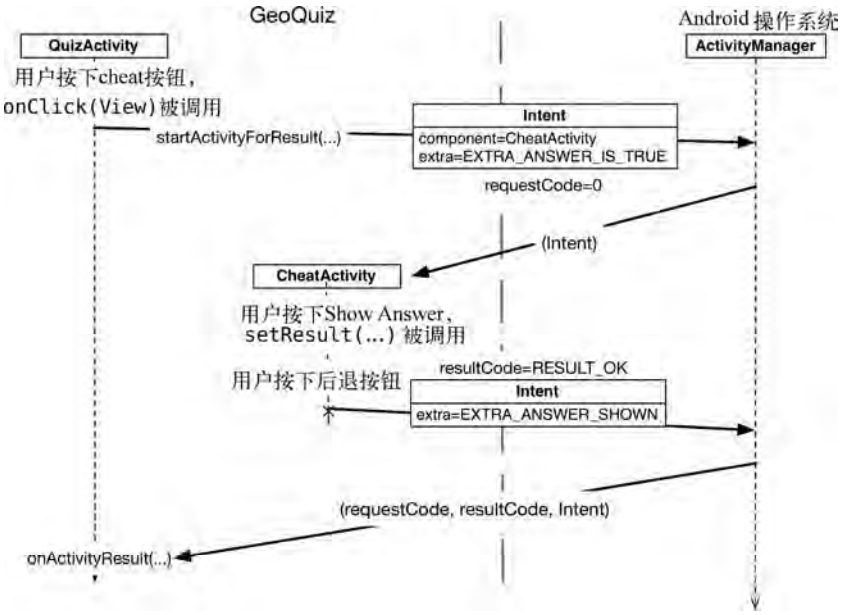

图5-12 GeoQuiz应用内部的交互时序图

最后覆盖QuizActivity的onActivityResult(int, int, Intent)方法来处理返回结果。

3. 处理返回结果

在QuizActivity.java中，新增一个成员变量保存CheatActivity回传的值。然后覆盖onActivityResult(...)方法获取它。onActivityResult(...)方法的实现如代码清单5-15所示。

代码清单5-15 onActivityResult(...)方法的实现（QuizActivity.java）

```
public class QuizActivity extends Activity {

    ...

    private int mCurrentIndex = 0;

    private boolean mIsCheater;

    ...

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (data == null) {
          return;
        }
        mIsCheater = data.getBooleanExtra(CheatActivity.EXTRA_ANSWER_SHOWN, false);
    }

    ...
}
```

观察onActivityResult(...)方法的实现代码，我们发现，QuizActivity并不关心请求代码或结果代码是什么。不过，在其他情况下，某些条件判断编码会使用到这些代码值。

最后，修改QuizActivity中的checkAnswer(boolean)方法，确认用户是否偷看答案并给出相应的反应。基于mIsCheater变量值改变toast消息的做法如代码清单5-16所示。

代码清单5-16　基于mIsCheater变量值改变toast消息（QuizActivity.java）

```
    private void checkAnswer(boolean userPressedTrue) {
        boolean answerIsTrue = mQuestionBank[mCurrentIndex].isTrueQuestion();

        int messageResId = 0;

        if (mIsCheater) {
            messageResId = R.string.judgment_toast;
        } else {
            if (userPressedTrue == answerIsTrue) {
                messageResId = R.string.correct_toast;
            } else {
                messageResId = R.string.incorrect_toast;
            }
        }

        Toast.makeText(this, messageResId, Toast.LENGTH_SHORT)
            .show();
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...

        mNextButton = (Button)findViewById(R.id.next_button);
        mNextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mCurrentIndex = (mCurrentIndex + 1) % mQuestionBank.length;
                mIsCheater = false;
                updateQuestion();
            }
        });

        ...
    }
```

运行GeoQuiz应用。偷看下答案，看看会发生什么。

5.4　activity 的使用与管理

来看看当我们在各activity间往返的时候，操作系统层面到底发生了什么。首先，在桌面启动器中点击GeoQuiz应用时，操作系统并没有启动应用，而只是启动了应用中的一个activity。确切地说，它启动了应用的launcher activity。在GeoQuiz应用中，QuizActivity就是它的launcher activity。

使用应用向导创建GeoQuiz应用以及QuizActivity时，QuizActivity默认被设置为launcher activity。配置文件中，QuizActivity声明的intent-filter元素节点下，可看到QuizActivity

被指定为launcher activity，如代码清单5-17所示。

代码清单5-17 QuizActivity被指定为launcher activity（AndroidManifest.xml）

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  ... >

  ...

  <application
    ... >
    <activity
      android:name="com.bignerdranch.android.geoquiz.QuizActivity"
      android:label="@string/app_name" >
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <activity
      android:name=".CheatActivity"
      android:label="@string/app_name" />
  </application>

</manifest>
```

QuizActivity实例出现在屏幕上后，用户可单击Cheat！按钮。CheatActivity实例在QuizActivity实例上被启动。此时，它们都处于activity栈中，如图5-13所示。

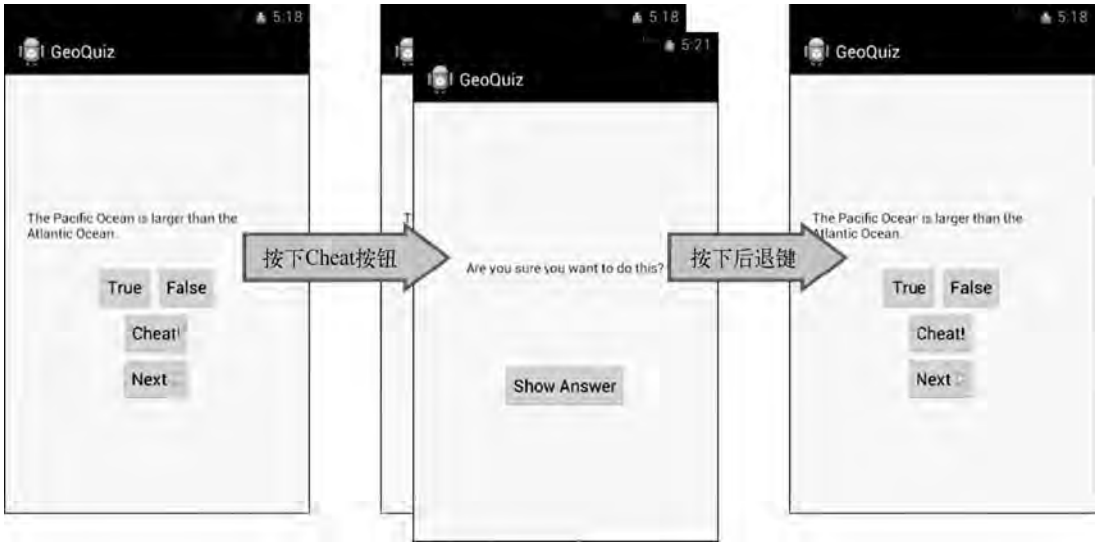

图5-13 GeoQuiz的回退栈

单击后退键，CheatActivity实例被弹出栈外，QuizActivity重新回到栈顶部，如图5-13所示。

在CheatActivity中调用Activity.finish()方法同样可以将CheatActivity从栈里弹出。

如在Eclipse中运行GeoQuiz应用，在QuizActivity界面单击后退键，QuizActivity将从栈里弹出，我们将退回到GeoQuiz应用运行前的画面，如图5-14所示。

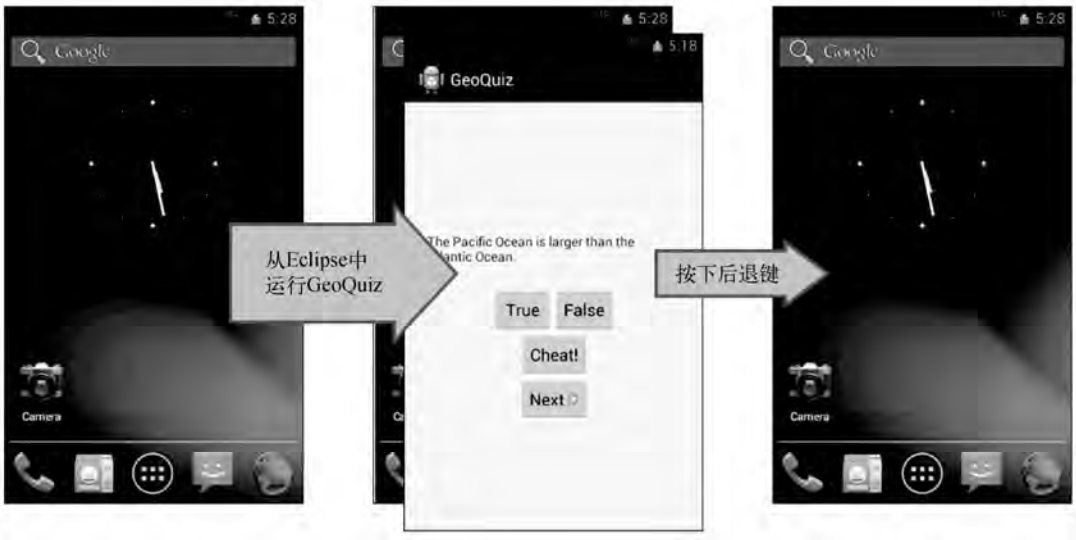

图5-14　Eclipse中运行应用，后退返回至桌面

如从桌面启动器启动GeoQuiz应用，在QuizActivity界面单击后退键，将退回到桌面启动器界面，如图5-15所示。

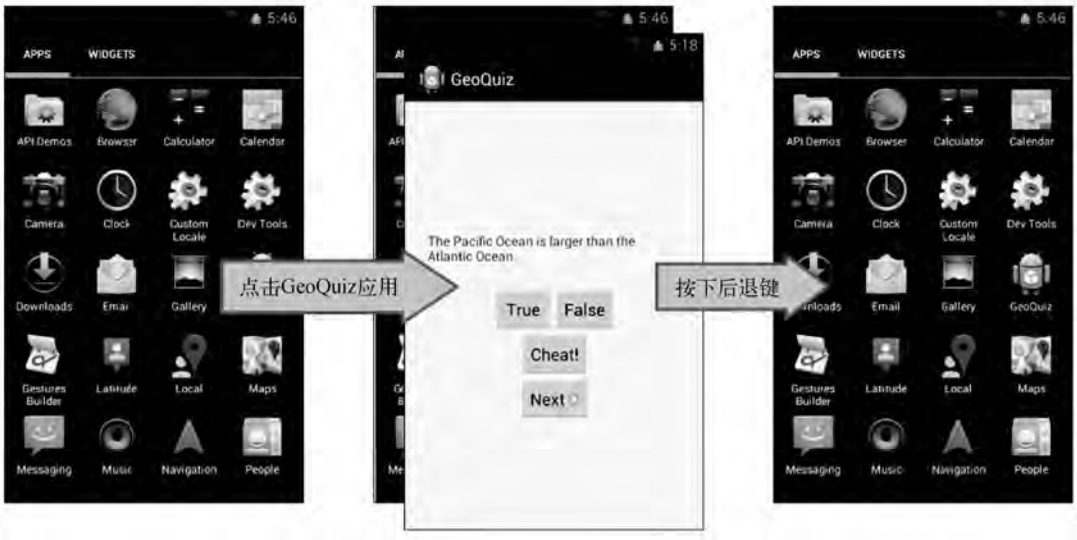

图5-15　从桌面启动器启动GeoQuiz应用

在桌面启动器界面，点击后退键，将返回到桌面启动器启动前的系统界面。

至此，我们已经看到，ActivityManager维护着一个非特定应用独享的回退栈。所有应用的activity都共享该回退栈。这也是将ActivityManager设计成操作系统级的activity管理器来负责启动应用activity的原因之一。不局限于单个应用，回退栈作为一个整体共享给操作系统及设备使用。

（想了解下“向上”按钮？第16章，我们将学习如何使用并配置它。）

5.5 挑战练习

作弊者注定会失败的。当然，如果他们能一直避开反作弊手段，那就另当别论了。也许他们能做到这一点，因为他们是作弊者嘛。

GeoQuiz应用有一些重大漏洞，我们的任务就是堵住这些漏洞。从易到难，以下为待解决的三个漏洞。

- 用户作弊后，可通过旋转CheatActivity来清除作弊痕迹。
- 作弊返回后，用户可通过旋转QuizActivity来清除mIsCheater变量的保存值。
- 用户不断单击Next按钮，直到再次遇到偷看过答案的问题，从而使作弊纪录丢失。

祝好运！