ISEL / ADEETC

Master in Communication Networks and Multimedia Engineering

Interactive Multimedia Applications

Tutorial 3

Interactive Multimedia Applications

Universe Explorer

(First Part - User Interface)

Rui Jesus

Introduction

This work aims to introduce and explore the application development environment for Android devices namely the XML editor to build the user interface. The tutorial starts from the "Hello World" application that is changed in order to build a new application with a more elaborate user interface. Below is a link that you should consult during the development of this work.

Android Developers
http://developer.android.com/training/index.html

Note: this tutorial should be done in class and the code of the resulting Android projects must be delivered through the Moodle platform until the 25th of March.

Laboratory Work

Universe Explorer

1. Create a new project in **Android Studio** by repeating the steps you took in Tutorial 1.

2. Create a new class within the same package. Give the name "**WorldGen**" to the class and create the methods and attributes as shown in Figure 1. The "**WorldGen**" class will be used to generate a new world/planet consisting of several attributes and methods that describe the characteristics of a planet.

3. Run the program and check the result that should be the same as the "Hello World" application.

4. At the end of the **onCreate()** method of the "**MainActivity**" class create an object of class "**WorldGen**", for instance: "`static WorldGen earth = news WorldGen("Earth", 5973, 9.78);`" Put the line of code, "`earth.setPlanetColonies(1);`" below to delete the "warning".

```
package com.example.hello_world_t1;
public class WorldGen {
        String planetName = "Earth";
        int planetMass;
        double planetGravity;
        int planetColonies;
        long planetPopulation;
        int planetBases;
        int planetMilitary;
        boolean planetProtection;
        publics WorldGen (String name, int mass, double gravity) {
                planetName = name;
                planetMass = mass;
                planetGravity = gravity;
                planetColonies = 0;
                planetPopulation = 0;
                planetBases = 0;
                planetMilitary = 0;
                planetProtection = false;
        }
        void setPlanetColonies (int numColonies) {
                // incrementar planetColonies de numColonies unidades
        }
        int getPlanetColonies() {
                // retorna  planetColonies;
        }
        void setPlanetMilitary (int numBases) {
                // incrementa planetBases de numBases unidades
        }
        int getPlanetMilitary() {
                // retorna planetBases
        }
        void turnForceFieldOn() {
                // coloca planetProtection a true
        }
        void turnForceFieldOff() {
                // coloca planetProtection a false
        }
        boolean getForceFieldState() {
                // retorna o valor planetProtection
        }
        void setColonyImmigration (int numColonists) {
                // incrementa planetPopulation de numColonists unidades
        }
        long getColonyImmigration() {
                // retorna planetPopulation
        }
        void setBaseProtection (int numForces) {
                // incrementa planetMilitary de numForces unidades
        }
        int getBaseProtection() {
                // retorna planetMilitary
        }
}
```

Figure 1. "WorldGen" class.

5. Following the previous point, initialize the remaining attributes of the "`earth`" object. For instance, "`planetColonies = 1; planetPopulations = 1000; planetBases = 1; planetMilitary = 100; planetProtection = true;`".

6. Create the "`protected void setStartUpWorldValues()`" method in the "**MainActivity**" class to initialize the attributes of a new planet. Use this method instead of the lines of code that you performed in the previous point.

7. Run the program to check for errors.

8. The next step is to build the layout shown in Figure 2. Begin by creating the necessary strings in the "*strings.xml*" file.

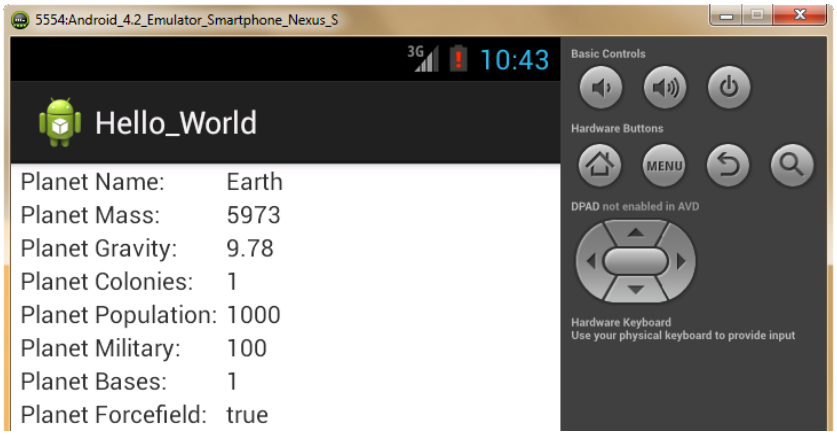

Figure 2. Main Layout of the Application.

9. Use the strings from the previous point in the file "*content_main.xml*". Use the **TextView** tag (or graphic editor) to display the necessary strings in the layout, for instance,

```
 <TextView
        android:id="@+id/textView1"
        android:layout_marginLeft="36dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/planet_name_label"
 />
<TextView
        android:id="@+id/textView2"
        android:layout_marginLeft="36dp"
        android:layout_below="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/planet_mass_label"
/>
```

Run the program. Do not forget to remove the "Hello World" string.

10. To write the value of each attribute on the screen, create another method in the "**MainActivity**" class, "`private void setStartUpScreenText(…)`". To put the value of the attributes on the screen use the following code:

```
TextView planetNameValue = (TextView) this.findViewById(R.id.dataView1);
```

```
planetNameValue.setText(earth.planetName);
TextView planetMassValue = (TextView) this.findViewById(R.id.dataView2);
planetMassValue.setText(String.valueOf(earth.planetMass));
```

11. Run de program. Do not forget that the "`setStartUpWorldValue()`" and "`setStartUpScreenText()`" methods of class "**MainActivity**" should be used inside of the **onCreate()** method.

12. Search the Web for an image of the universe with a resolution of approximately 2400x1440 pixels, to be used as the background image. In an image editing program, generate the remaining images for each generalized density group.

13. Do as in tutorial 2 and put the previous images in their **drawable** directories. Place the image as the background image of the "**MainActivity**".

14. Now let's create a menu (see Figure 4). Copy to the file "*menu_main.xml*" of the menu directory, the code in Figure 3. Run the program. Check that the menu is correct (see Figure 4).

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.example.hello_world_t1.MainActivity" >

    <item android:id="@+id/menu_add"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:title="@string/menu_add_planet"/>
    <item android:id="@+id/menu_config"
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="@string/menu_config_planet"/>
    <item android:id="@+id/menu_travel"
        android:orderInCategory="300"
        android:showAsAction="never"
        android:title="@string/menu_travel_planet"/>
    <item android:id="@+id/menu_attack"
        android:orderInCategory="400"
        android:showAsAction="nevers"
        android:title="@string/menu_attack_planet"/>
    <item android:id="@+id/menu_contact"
        android:orderInCategory="500"
        android:showAsAction="never"
        android:title="@string/menu_home_planet"/>
</menu>
```

Figure 3. Code for the "*menu_main.xml*" file.

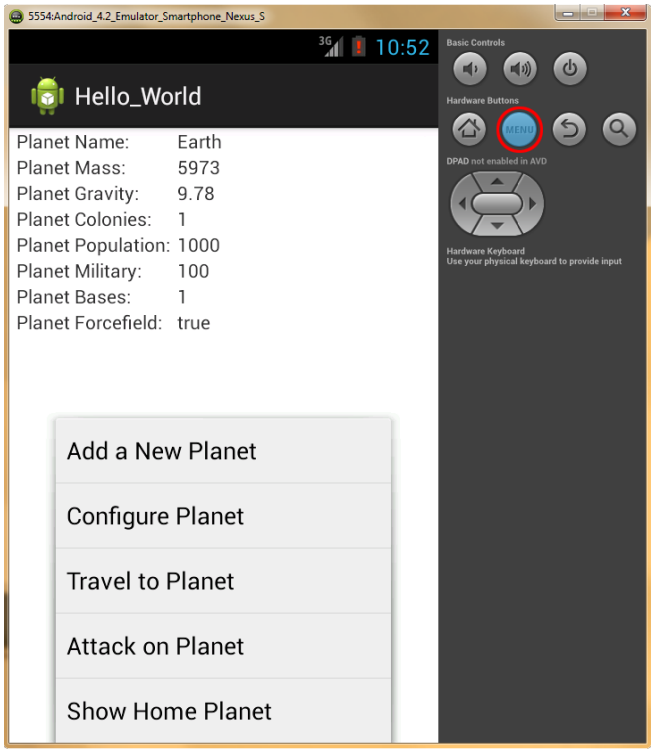

Figure 4. Menu Layout.

15. In the layout "*content_main.xml*" put a "**button**" element below, after the planet information. In the "text" attribute of the "**button**" element place a string (defined in the file "*strings.xml*") with the following text: "Planet Photos" (see Figure 5).

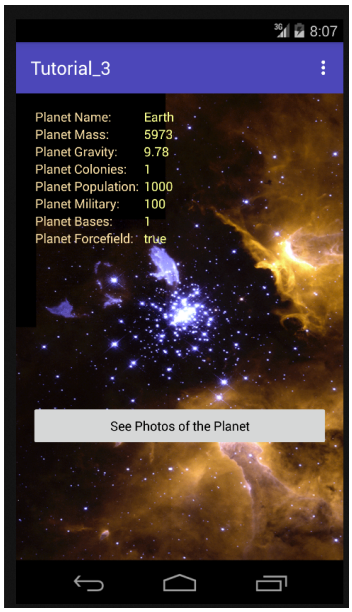

Figure 5. Layout "content_main.xml".

16. To view images of a planet we will use another activity. Proceed by enabling the **Onclick** event of the "**button**" element created the previous point. In the **Onclick** method (...) put the following code:

```
Intent intent_add = new Intent(this, GridViewPhotos.class);
this.startActivity(intent_add);
```

17. Create the "**GridViewPhotos**" class with the code below.

```
public class GridViewPhotos extends AppCompatActivity {
    private GridView gridView;
    private GridViewAdapter gridAdapter;

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

      }
}
```

18. In order to create the activity "**GridViewPhotos**" it is necessary to create the "*activity_gridph.xml*" layout (for now, it can be left blank). Comment the line of code with the variable "gridAdapter". Run the program and check what happens when you click on **button "Planet Photos…"**.

19. To launch a new activity, **add the new activity** after the activity "**MainActivity**" in the file "*AndroidManifest.xml*".
<activity android:name=".GridViewPhotos"/>

20. Run the program and verify that the new activity is blank.

21. The next step consist of building the "*activity_gridph.xml*" layout (see Figure 6). Start by putting a <GridView> element inside the <RelativeLayout> element. For this activity you need another background image so repeat points 12 and 13. Change the value of the following attributes of the <GridView> element:

android:layout_width - "match_parent"

android:layout_height - "wrap_content"

android:layout_centerHorizontal - "true"

android:gravity - "center"

android:stretchMode -“columnWidth"

android:drawSelectorOnTop -“true"

android:focusable - “true"

android:clickable - “true"

android:columnWidth - “100dp"

android:numColumns - “auto_fit"

android:verticalSpacing - “5dp"

android:layout_marginLeft - “5dp"

android:layout_marginRight - “5dp"

android:layout_marginTop - “5dp"

android:layout_marginBottom - “40dp”

Tip: Use the visual editor and try to understand the effect of each attribute.

Figure 6. Layout “*activity_gridph.xml*”.

Note: best practices indicate that the "constrainLayout" container should be used as a rule. You only should not use this container when it is not possible to make a feature that the other containers have and which is useful for our application. However, in this tutorial I will ask you to use other containers in order to you realize the characteristics of each of the containers available.

22. To complete the layout of Figure 6 it is necessary to put the "**button**" below. Run the program.

```
package com.example.ruimfjesus.gridview_photos;

import android.graphics.Bitmap;


public class ImageItem {
    private Bitmap image;
    private String title;

    public ImageItem(Bitmap image, String title) {
        super();
        this.image = image;
        this.title = title;
    }

    public Bitmap getImage() {
        // retorna image;
    }

    public void setImage(Bitmap image) {
        // Atribui ao atributo image o valor image
passado como parâmetro
    }

    public String getTitle() {
        // retorna title;
    }

    public void setTitle(String title) {
        // Atribui ao atributo title o valor title
passado como parâmetro

    }
}
```

23. Register the **OnClick**() event and in the method that responds to the event place *finish()*; Run the program.

24. At each position of the <GridView> element we will place an image with text underneath, so let's create an "**ImageItem"** class as described in the box below.

25. Search the Web for 9 images of the planet Earth (for example, images of space in which you see the earth). Each image must be at least 400x400 pixels. Repeat steps 12 and 13. It should be named "image_1.png", "image_1.png", ..., "image_9.png" to the image files.