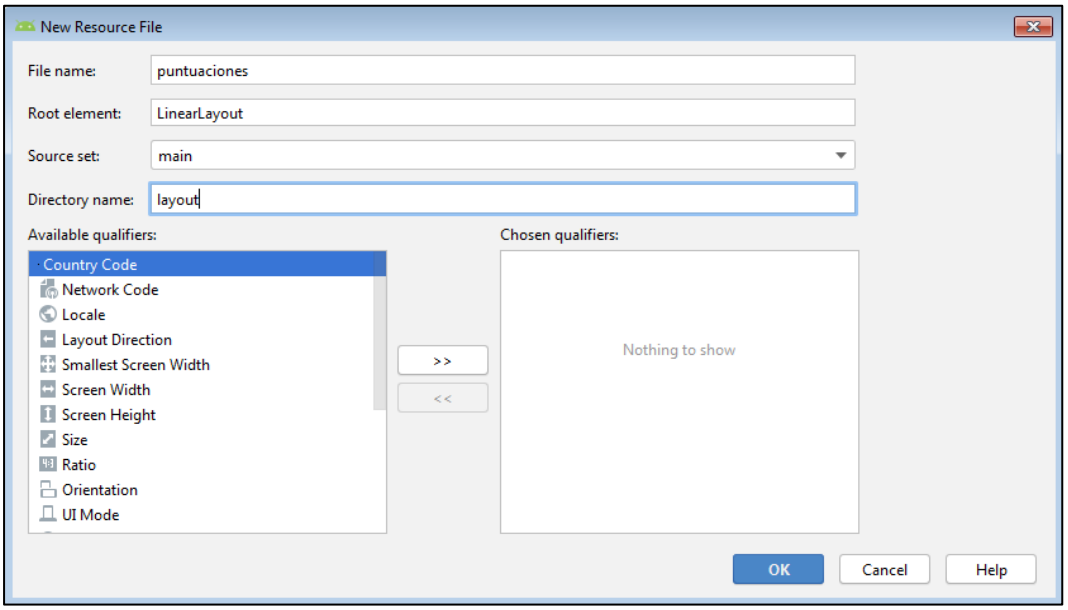
PRACTICA 14: ASTEROIDES

Parte XIV: *Un ListView que visualiza una lista de Strings (I)*

Paso 1. El *Layout* que utilizaremos en Asteroides para mostrar las puntuaciones se llamará puntuaciones.xml. En él se incluye una vista ListView. Crea el *Layout* con el siguiente código:



puntuaciones.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Puntuaciones"
            android:gravity="center"
            android:layout_margin="10px"
            android:textSize="10pt"/>
        <FrameLayout
            android:layout_width="match_parent"
            android:layout_height="0dip"
            android:layout_weight="1">
            <ListView
                android:id="@android:id/list"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:drawSelectorOnTop="false" />
            <TextView
                android:id="@android:id/empty"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="No hay puntuaciones" />
        </FrameLayout>
</LinearLayout>
```

Paso 2. Necesitamos ahora crear la actividad Puntuaciones para visualizar el Layout anterior. Crea una nueva clase en tu proyecto e introduce el siguiente código:

Puntuaciones.java

```
package com.example.asteroides;

import android.app.ListActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;

public class Puntuaciones extends ListActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.puntuaciones);
        setListAdapter(new ArrayAdapter( context: this,
                android.R.layout.simple_list_item_1,
                MainActivity.almacen.listaPuntuaciones( cantidad: 10)));

    }
}
```

Toda actividad que vaya a visualizar un ListView ha de heredar de ListActivity. Además, ha de llamar al método setListAdapter() para indicar el adaptador con la lista de elementos a visualizar. En el ejemplo se ha utilizado una de las posibilidades más sencillas, para crear un adaptador, usar la clase ArrayAdapter<clase>. Un ArrayAdapter crea las vistas del ListView a partir de los datos almacenados en un array. Puedes utilizar un array que contenga datos de cualquier clase, no tienes más que indicar en <Clase> la clase deseada. En este caso se utiliza de un array de String[1]. El constructor de ArrayAdapter<clase> tiene tres parámetros: El primer parámetro es un Context con información sobre el entorno de la aplicación. Utilizaremos como contexto la misma actividad que hace la llamada. El segundo parámetro es un Layout, utilizado para representar cada elemento de la lista. En este ejemplo, en lugar de definir uno nuevo, utilizaremos una ya definido en el sistema. El último parámetro es un array con los strings a mostrar. Para ello, llamamos al método listaPuntuaciones() que nos devuelve esta lista del objeto estático almacen de la clase MainActivity.

Paso 3. Recuerda que toda nueva actividad ha de ser registrada en AndroidManifest.xml.

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".AcercaDeActivity"
    android:label="Acerca de ..."
    android:theme="@style/Theme.AppCompat.Light.Dialog" />
<activity
    android:name=".Preferencias"
    android:label="Preferencias" />
<activity android:name=".Puntuaciones"></activity>
```

Paso 4. Prueba si funcionan las modificaciones introducidas.

Parte XV: *Un ListView que visualiza layouts personalizados*

Paso 1. Reemplaza la clase anterior por:

```
public class Puntuaciones extends ListActivity {

    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.puntuaciones);
        setListAdapter(
                new ArrayAdapter(this,
                        R.layout.elemento_lista,
                        R.id.titulo,
                        MainActivity.almacen.listaPuntuaciones( cantidad: 10)));
    }
}
```

Como hemos explicado, la classe ArrayAdapter<String>permite insertar los datos desde un array de String en nuestro ListView. En este ejemplo se utiliza un constructor con cuatro parámetros:

- **this**: es el contexto, con información sobre el entorno de la aplicación.
- **R.layout.*elemento_lista***: es una referencia de recurso a la vista que será utilizada repetidas veces para formar la lista. Se define a continuación.
- **R.id.*titulo***: identifica un id de la vista anterior que ha de ser un TextView. Su texto será reemplazado por el que se indica en el siguiente parámetro.
- MainActivity.*almacen*.listaPuntuaciones(10): vector de String con los textos que serán visualizados en cada uno de los TextView. Esta lista es obtenida accediendo a la claseAsteroides a su variable estàtica almacen llamando a su método listaPuntuaciones().

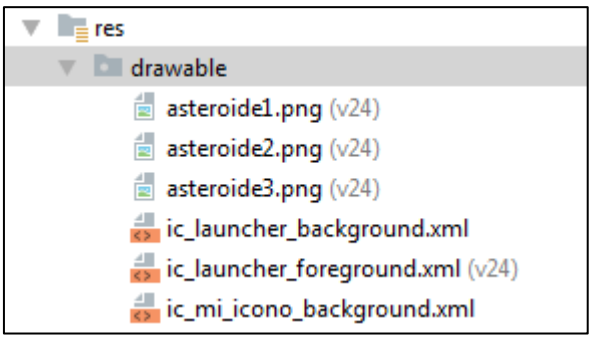
Paso 2. Ahora hemos de definir el Layout que representará cada uno de los elementos de la lista. Crea el fichero *res/Layout/elemento_lista.xml* con el siguiente código:

elemento_lista.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="?android:attr/listPreferredItemHeight">
    <ImageView android:id="@+id/icono"
        android:layout_width="?android:attr/listPreferredItemHeight"
        android:layout_height="match_parent"
        android:layout_alignParentLeft="true"
        android:src="@drawable/asteroide2"/>
    <TextView android:id="@+id/titulo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/icono"
        android:layout_alignParentTop="true"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:singleLine="true" />
    <TextView android:id="@+id/subtitulo"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Otro Texto"
        android:layout_toRightOf="@id/icono"
        android:layout_below="@id/titulo"
        android:layout_alignParentBottom="true"
        android:gravity="center"/>
</RelativeLayout>
```

Este Layout representa una imagen a la izquierda con dos textos a la derecha, uno de mayor tamaño en la parte superior. Para combinar estos elementos se ha escogido un RelativeLayout, donde el ancho se establece a partir de un parámetro de configuración del sistema ?android:attr/listPreferredItemHeight. El primer elemento que contiene es un ImageView alineado a la izquierda. Su alto es la misma que el contenedor (match_parent) mientras que el ancho se establece con el mismo parámetro que el alto del contenedor. Por lo tanto la imagen será cuadrada.

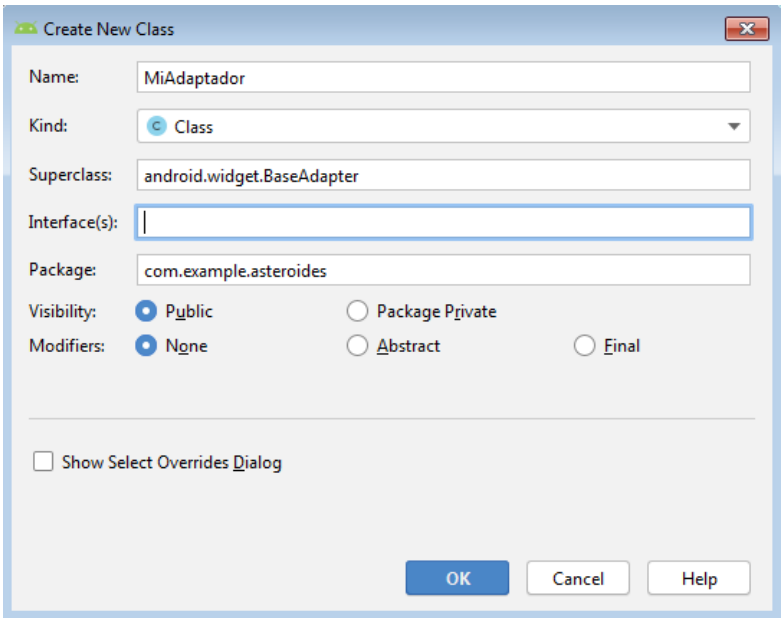
Paso 3. Puedes descargarte las imágenes utilizadas en la aplicación Asteroides de www.androidcurso.com. En el menú El gran libro de Android / Ficheros usados en ejercicios dentro de Graficos.zip. Copia el fichero asteriode1.png, asteriode2.png y asteriode3.png a la carpeta res/drawable.



Paso 4. Ejecuta la aplicación y verifica el resultado.

Parte XVI: *Un ListView con nuestro propio adaptador*

Paso 1. Crea la clase *MiAdaptador.java* en el proyecto con el siguiente código:



```
public class MiAdaptador extends BaseAdapter {
    private final Activity actividad;
    private final Vector lista;
    public MiAdaptador(Activity actividad, Vector lista) {
        super();
        this.actividad = actividad;
        this.lista = lista;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = actividad.getLayoutInflater();
        View view = inflater.inflate(R.layout.elemento_lista,  root: null,  attachToRoot: true);
        TextView textView =(TextView)view.findViewById(R.id.titulo);
        textView.setText(lista.elementAt(position).toString());
        ImageView imageView=(ImageView)view.findViewById(R.id.icono);
        switch (Math.round((float)Math.random()*3)){
            case 0:
                imageView.setImageResource(R.drawable.asteroide1);
                break;
            case 1:
                imageView.setImageResource(R.drawable.asteroide2);
                break;
            default:
                imageView.setImageResource(R.drawable.asteroide3);
                break;
        }
        return view;
    }
    @Override
    public int getCount() {
        return lista.size();
    }
    @Override
    public Object getItem(int arg0) {
        return lista.elementAt(arg0);
    }
    @Override
    public long getItemId(int position) {
        return position;
    }
}
```

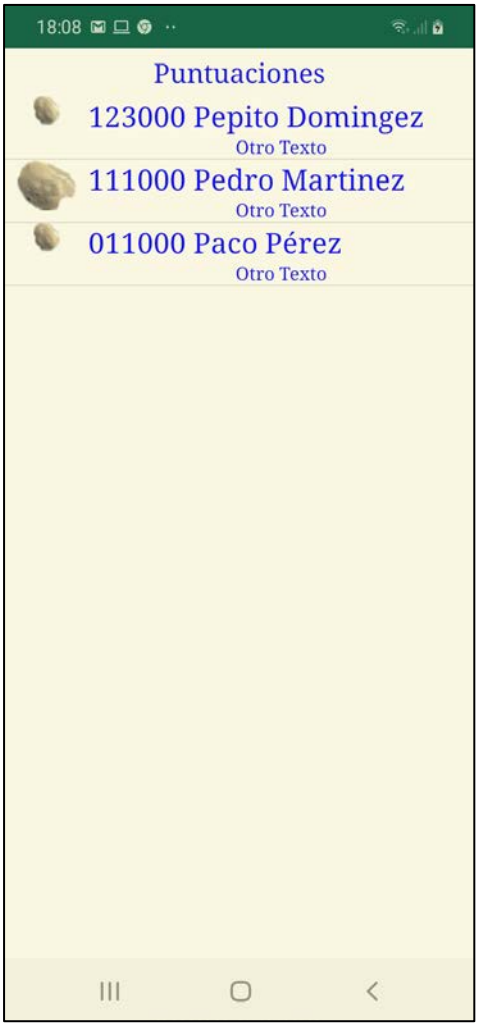
En el constructor de la clase se indica la actividad donde se ejecutará y la lista de datos a visualizar. El método más importante de esta clase es getView() el cual tiene que construir los diferentes *Layouts* que serán añadidos en la lista. Comenzamos construyendo un objeto View a partir del código xml definido en *elemento_lista.xml*. Este trabajo se realiza por medio de la clase LayoutInflater. Luego, se modifica el texto de uno de los TextView según el array que se pasó en el constructor. Finalmente, se obtiene un número al azar (Math.round()) y se asigna uno de los tres gráficos de forma aleatoria.

Paso 2. Reemplaza en la clase Puntuaciones la llamada al constructor de ArrayAdapter<String> por:

```java
public class Puntuaciones extends ListActivity {

    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.puntuaciones);
        //setListAdapter( new ArrayAdapter(this,
        //                R.layout.elemento_lista,
        //                R.id.titulo,
        //                MainActivity.almacen.listaPuntuaciones(10)));
        setListAdapter(new MiAdaptador( actividad: this,
                MainActivity.almacen.listaPuntuaciones( cantidad: 10)));
    }
}
```

Paso 3. Ejecuta la aplicación y verifica el resultado.



Parte XVII: *Detectar una pulsación sobre un elemento de la lista*

Paso 1. Añade el siguiente método a la clase *Puntuaciones.java*:

```
@Override
protected void onListItemClick(ListView listView,
                               View view, int position, long id) {
    super.onListItemClick(listView, view, position, id);
    Object o = getListAdapter().getItem(position);
    Toast.makeText( context: this,  text: "Selección: " + Integer.toString(position)
            +  " - " + o.toString(),Toast.LENGTH_LONG).show();
}
```

Paso 2. Ejecuta la aplicación, pulsa en “Puntuaciones” y luego en una puntuación para verificar el resultado.