Uso de recursos alternativos:
Layouts alternativos y traducción a otro idioma

Una aplicación Android va a poder ser ejecutada en una gran variedad de dispositivos.

El **tamaño de pantalla**, la **resolución** o el **tipo de entradas** puede variar mucho de un dispositivo a otro.

Por otra parte, nuestra aplicación ha de estar preparada para diferentes modos de funcionamiento, como el **modo automóvil** o el **modo noche**, y **para poder ejecutarse en diferentes idiomas.**

A la hora de crear el interfaz de usuario hemos de tener en cuenta todas estas circunstancias.

Afortunadamente la plataforma Android nos proporciona una herramienta de gran potencia para resolver este problema, el uso de los recursos alternativos.

Práctica: Recursos alternativos en Asteroides

1. Ejecuta la aplicación Asteroides creada anteriormente.

2. Los teléfonos móviles basados en Android permiten cambiar la configuración en apaisado y en vertical. Para conseguir este efecto con el emulador pulsa **Ctrl+F11**.

Si observas el resultado de la vista que acabas de diseñar en vertical, no queda todo lo bien que desearíamos cuando la ponemos en horizontal.

FALTA EL BOTÓN Salir. No lo vemos

Para resolver este problema Android te permite **diseñar una vista diferente para la configuración horizontal y otra para vertical.**

3. Crea la carpeta *res/layout-land.* *Para ello puedes pulsa botón derecho sobre /res y elegir NEW→Android Resource Directory y completar el formulario que aparece eligiendo "layout" en "resource type" y "orientation" en "available qualifiers".*

4. Copia en ella el fichero activity*_main.xml.Para ello selecciona el fichero y pulsa Ctrl-C. Selecciona la carpeta destino y pulsa Ctrl-V.*

5. Crea una vista similar a la que ves a continuación: formada por un *LinearLayout* que contiene un *TexView* y un *TableLayout* con dos *Button* por columna.

6. Ejecuta de nuevo la aplicación y observa como la vista se ve correctamente en las dos orientaciones.

Solución:

Has de obtener un código XML similar al siguiente:

```
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center"
android:orientation="vertical"
android:padding="30dp"
tools:context=".Asteroides">

<TextView
android:id="@+id/titulo_activity"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginBottom="20dp"
```

```
android:gravity="center"
android:text="@string/texto_tituloAplicacion"
android:textSize="25sp"/>

<TableLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center"
android:stretchColumns="*">

<TableRow>

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

<Button
android:id="@+id/btnConfigurar"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/Configurar"/>
</TableRow>

<TableRow>

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

<Button
android:id="@+id/btnSalir"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/Salir"/>
</TableRow>
</TableLayout>

</LinearLayout>
```

NOTA: *Para conseguir en un TableLayout, que las columnas se ajusten a todo el ancho de la tabla poner stretchColumns="*". stretchColumns="0"significa que asigne el ancho sobrante a la primera columna. stretchColumns="1"significa que asigne el ancho sobrante a la segunda columna. stretchColumns="*"significa que asigne el ancho sobrante entre todas las columnas.*

Android utiliza una lista de sufijos para expresar recursos alternativos. Estos sufijos pueden hacer referencia a la orientación del dispositivo, al lenguaje, la región, la densidad de píxeles, la resolución, el método de entrada,…

Por ejemplo, **si queremos traducir nuestra aplicación al inglés, español y francés**. **Siendo el primer idioma el usado por defecto, crearíamos tres versiones del fichero *strings.xml* y lo guardaríamos en los siguientes tres directorios**:

```
res/values/strings.xml

res/values-es/strings.xml

res/values-fr/strings.xml
```

Ejercicio paso a paso: *Traducción de Asteroides*

1. Crea la carpeta *res/values-en.*

2. Copia en ella el fichero *strings.xml.*

3. Traduce en este fichero todas las cadenas al inglés.

4. Ejecuta la aplicación en el emulador y verifica que la aplicación está en inglés.

5. Vamos a **cambiar la configuración de idioma del emulador**.

Para ello, accede a *Ajustes* del dispositivo (*Settings)* y selecciona la opción Mi dispositivo->*Idioma e introducción*. Dentro de esta opción selecciona *como idioma Español*.

NOTA: Observa que en otros idiomas permite seleccionar tanto el idioma como la región. Por desgracia, para el español solo permite dos regiones España y Estados Unidos.

6. Ejecuta de nuevo la aplicación y observa cómo ha traducido el texto.

7. ¿Qué pasaría si nuestra aplicación se ejecuta en un terminal configurado en chino? ¿Aparecerán las cadenas en castellano o en inglés? ¿Piensas que esta es una buena decisión de diseño? ¿Cómo lo podrías solucionar?

Respuestas: *Aparecería en castellano. Mala decisión, mejor si aparece en inglés, que es un idioma conocido universalmente. Habría que poner el fichero strings.xml con los textos en ingles en la carpeta de recurso por defecto (res/values) y el fichero con los textos en castellano en la carpeta res/values-es.*

Otro ejemplo de utilización de recursos diferenciados lo podemos ver al crear una aplicación con Android Studio. En este caso no creará una única carpeta *"mipmap"* para almacenar el icono de la aplicación, sino cinco según la densidad de píxeles del dispositivo:

```
res/mipmap-hdpi/ic_launcher.png
```

```
res/mipmap-mdpi/ic_launcher.png

res/mipmap-xhdpi/ic_launcher.png

res/mipmap-xxhdpi/ic_launcher.png

res/mipmap-xxxhdpi/ic_launcher.png
```

Resulta posible indicar varios sufijos concatenados; por ejemplo:

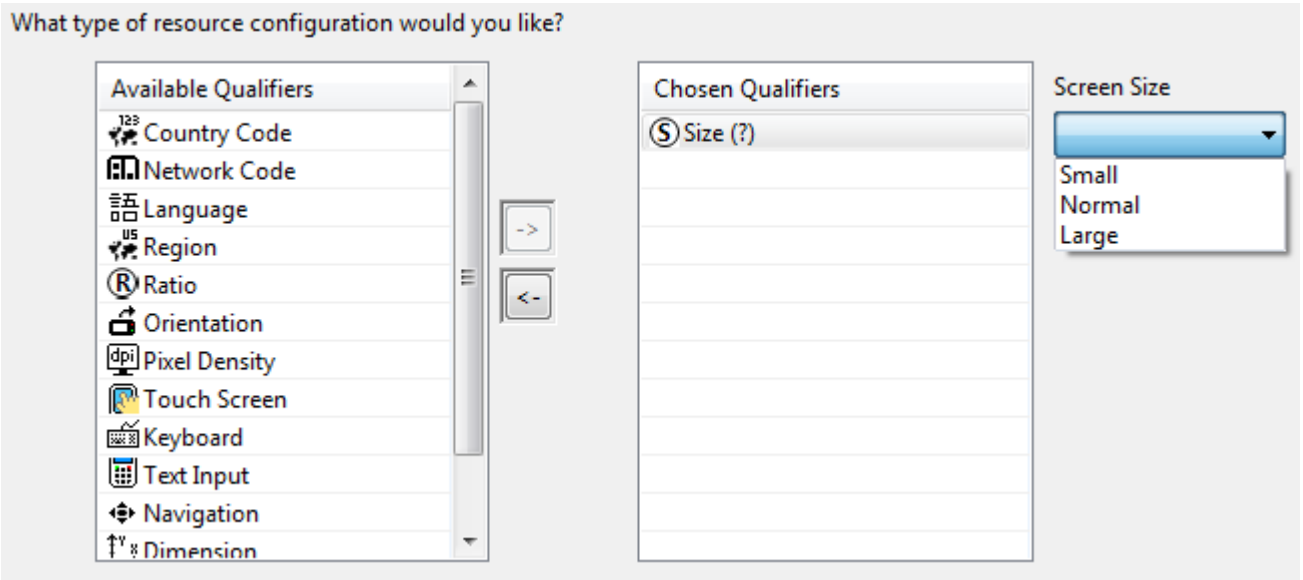
```
res/values-en-rUS/strings.xml

res/values-en-rUK/strings.xml
```

Pero cuidado, Android establece un orden a la hora de encadenar sufijos. Puedes encontrar una lista de estos sufijos en la documentación del SDK:

http://developer.android.com/guide/topics/resources/providing-resources.html

Para ver los sufijos disponibles también puedes pulsar con el botón derecho sobre la carpeta de recursos /res y seleccionar *New* > *Android Resource File*. Esta opción te permite crear un nuevo fichero XML y poner el sufijo deseado de forma y orden correcto.



Práctica: *Creando un Layout para tabletas en Asteroides*

Si ejecutas la aplicación Asteroides en una tableta observarás que el tamaño de los botones es demasiado grande (alargados).

APLICACIÓN DE EJEMPLO:

ASTEROIDES

1. Trata de hacer un *Layout* alternativo a *main.xml*, que sea utilizado en pantallas de tamaño `xlarge` (7 - 10,5 pulgadas) en orientación "land (apaisado).
Simplemente poniendo un margen mayor en el Layout se conseguirá unos botones más proporcionados.

2. Si lo deseas también puedes personalizar el fondo de la pantalla (atributo `background`), los tipos de letras, colores…

3. Verifica que la aplicación se visualiza correctamente en todos los tipos de pantalla, tanto en horizontal como en vertical.

Solución:

1. Crea la carpeta *layout-xlarge-land* y copia en ella el fichero *activity_main.xml*.

2. En el nuevo fichero modifica el valor que se usa como margen interno:

```
<LinearLayout
    …
    android:padding="150dp"… >
```

Esta solución sería mejor implementarla usando el fichero dimens.xml y guardando allí las diferentes dimensiones deseadas.

3. Ejecuta en un emulador o tableta real de 10 pulgadas y verifica que el resultado.