13. RelativeLayout

`RelativeLayout` es un contenedor que organiza sus elementos en posiciones relativas. Por ejemplo, un elemento se puede colocar a la izquierda o debajo de otro, o pegado a un lado del contenedor o alineado con otro elemento. Se utiliza para evitar anidar contenedores, lo cual es costoso desde el punto de vista computacional.

Para utilizar este contenedor hemos de identificar los elementos de la interfaz mediante el atributo `android:id`. Estos identificadores se asocian a los elementos de la interfaz y nos permiten acceder a los botones desde el fichero Java. La sintaxis es `android:id="@+id/f2"`, donde `f2` es un identificador válido elegido por el programador y el signo `+` indica que `f2` es un nuevo identificador que se añade automáticamente al fichero `R.java`.

Como se puede comprobar a continuación, la clase `R` está formada por subclases, una de las cuales se denomina `id` y contiene los identificadores enteros como miembros públicos, estáticos y finales. Esto impide alterar el valor del identificador y, a la vez, permite su acceso como `R.id.f2`:

/gen/R.java

```
public final class R {
    public static final class attr {
    }
    public static final class dimen {
    public static final int activity_horizontal_margin=0x7f040000;
        public static final int activity_vertical_margin=0x7f040001;
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class id {
        public static final int action_settings=0x7f080021;
        public static final int f1=0x7f080001;
        public static final int f10=0x7f080006;
        public static final int f11=0x7f08000a;
        public static final int f12=0x7f08000b;
        public static final int f13=0x7f08000c;
        public static final int f14=0x7f08000d;
        public static final int f15=0x7f08000e;
        public static final int f16=0x7f08000f;
        public static final int f17=0x7f080010;
        public static final int f18=0x7f080011;
        public static final int f19=0x7f080012;
        public static final int f2=0x7f080000;
        public static final int f20=0x7f080013;
        public static final int f21=0x7f080014;
        public static final int f22=0x7f080015;
        public static final int f23=0x7f080016;
        public static final int f24=0x7f080017;
        public static final int f25=0x7f080018;
        public static final int f26=0x7f080019;
        public static final int f27=0x7f08001a;
         ...
    }
    public static final class layout {
        public static final int activity_main=0x7f030000;
    }
    ...
}
```

Veamos cómo reconstruir la interfaz de nuestro juego con `RelativeLayout` en lugar de `LinearLayout`. El código es bastante extenso, así que solo mostramos una parte a continuación, la correspondiente a las dos primeras filas del tablero:

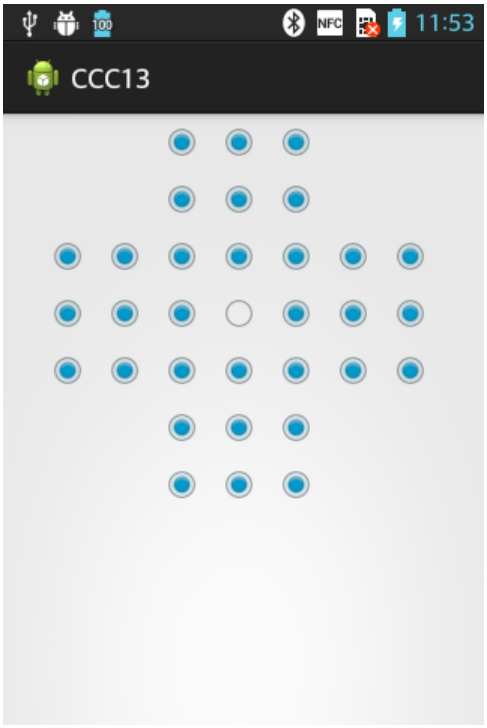
`/res/layout/activity_main.xml`

```
<?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="match_parent" >
    <RadioButton
        android:id="@+id/f2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:layout_centerHorizontal="true"/>
    <RadioButton
        android:id="@+id/f1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:layout_toLeftOf="@id/f2"/>
    <RadioButton
        android:id="@+id/f3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:layout_toRightOf="@id/f2"/>

    <RadioButton
        android:id="@+id/f4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:layout_below="@id/f1"
        android:layout_alignLeft="@id/f1"/>
    <RadioButton
        android:id="@+id/f5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:layout_below="@id/f1"
        android:layout_toRightOf="@id/f4"/>
    <RadioButton
        android:id="@+id/f6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:layout_below="@id/f1"
        android:layout_toRightOf="@id/f5"/>

    ...

</RelativeLayout>
```



El botón identificado como `f2` se centra horizontalmente:

```
<RadioButton
    android:id="@+id/f2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:layout_centerHorizontal="true"/>
```

El botón identificado como `f1` se sitúa a su izquierda:

```
<RadioButton
    android:id="@+id/f1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:layout_toLeftOf="@id/f2"/>
```

Finalmente, el botón identificado como `f3` se sitúa la derecha del botón `f2`:

```
<RadioButton
    android:id="@+id/f3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:layout_toRightOf="@id/f2"/>
```

Encontrarás el código completo del fichero de diseño en la carpeta `/res/layout/` del proyecto `ccc13` que acompaña a esta unidad. Aunque el aspecto final del tablero es indistinguible del de la unidad 8, esta versión del diseño es preferible pues hemos evitado anidar contenedores, lo cual hace que la interfaz se dibuje mas rápidamente.

A continuación encontrarás la lista completa de atributos XML correspondientes a la clase `RelativeLayout.LayoutParams` y organizada por categorías:

Atributos utilizados para especificar la posición de una vista con respecto a su contenedor:

- `android:layout_alignParentTop`: si `true`, hace que el lado superior de la vista se alinee con el lado superior del contenedor.
- `android:layout_alignParentBottom`: si `true`, hace que el lado inferior de la vista se alinee con el lado inferior del contenedor.
- `android:layout_alignParentLeft`: si `true`, hace que el lado izquierdo de la vista se alinee con el lado izquierdo del contenedor.
- `android:layout_alignParentRight`: si `true`, hace que el lado derecho de la vista se alinee con el lado derecho del contenedor.
- `android:layout_centerHorizontal`: si `true`, centra la vista horizontalmente dentro del contenedor.
- `android:layout_centerVertical`: si `true`, centra la vista verticalmente dentro del contenedor.
- `android:layout_centerInParent`: si `true`, centra la vista horizontal y verticalmente dentro del contenedor.

Atributos utilizados para controlar la posición de una vista respecto a la de otra (el identificador de la segunda vista se especifica como en el siguiente ejemplo: `android:layout_toRightOf="@id/f2"`):

- `android:layout_above`: coloca el lado inferior de la vista por encima de la vista especificada.
- `android:layout_below`: coloca el lado superior de la vista por debajo de la vista especificada.
- `android:layout_toLeftOf`: coloca el lado derecho de la vista a la izquierda de la vista especificada.
- `android:layout_toRightOf`: coloca el lado izquierdo de la vista a la derecha de la vista especificada.
- `android:layout_alignWithParentIfMissing`: si `true`, el padre se utilizará como referencia si la referencia a la segunda vista no se puede encontrar para `layout_toLeftOf`, `layout_toRightOf`, …

Atributos utilizados para alinear una vista con otra (el identificador de la segunda vista se especifica como en este ejemplo: `android:layout_alignRight="@id/f1"`):

- `android:layout_alignBaseLine`: la línea base de las dos vistas se alinea.
- `android:layout_alignBottom`: el lado inferior de las dos vistas se alinea.
- `android:layout_alignTop`: el lado superior de las dos vistas se alinea.
- `android:layout_alignLeft`: el lado izquierdo de las dos vistas se alinea.
- `android:layout_alignRight`: el lado derecho de las dos vistas se alinea.