Proyecto final de curso
Android: Fundamentos de Programación
(Septiembre-Diciembre 2016)

Nombre de la aplicación: Ajedrez Infantil

Autor: Madina Iñigo, Usua

Qué hace la aplicación:

Aplicación didáctica infantil para que niños aprendan a jugar al ajedrez, pretende ser un complemento de los videos Aprende con Rey.

Licencia:

Tareas realizadas:

B) Ejercicio base:

B.1. Crear un layout con las fichas a la izquierda y el tablero a la derecha.

B.3. Resaltar posiciones de una coordenada.

Para la animación de resaltado de casilla utilizamos dos drawables de tipo AnimationList, uno para las casillas blancas alternando los colores de la casilla entre blanco y verde claro y otro para las casillas negras, alternando los colores entre negro y verde oscuro.

La función se encarga de dada una casilla, comprobar si la casilla es blanca o negra y aplicar a ésta la animación correspondiente.

G) Actividad Inicial:

G.2. Actividad y Layout Capitulo: Se muestran a la izquierda el personaje, a la derecha los botones de Ver Vídeo y de los ejercicios correspondientes.

Figura1. Ejemplo de Capitulo

G.3. Actividad Ver Vídeo:

Para la reproducción de los videos creamos un activity que recibirá el enlace de youtube que se quiere reproducir, reutilizando la misma actividad cada vez que tengamos que reproducir un video. Nos planteamos dos opciones a la hora de realizar la reproducción, mediante la clase VideoView que nos proporciona Android o mediante la API que nos facilita Google, YouTube Api. La clase que nos facilita Android es bastante limitada, debemos proporcionarle el enlace .3gp del video para la versión móvil sacrificando la calidad de éste. Por lo que finalmente nos decantamos por incorporar la librería de Google que nos ofrece las siguientes ventajas para nuestra aplicación entre otras:

- Soporte de reproducción de alta calidad de vídeo con Android 2.2 y posteriores
- Sencilla integración
- Soporte para pantalla completa

Pasos para la integración:

- Añadir el archivo .jar a la carpeta /libs del proyecto
- Incluir la librería como dependencia en nuestro proyecto
- Hacer que la actividad VerVideo extienda de YouTubeBaseActivity
- Hacer que la actividad VerVideo implemente la interfaz YouTubePlayer.OnInitializeListener que implementa las devoluciones de llamada que se invocan cuando la inicialización del reproductor se realiza correctamente o presenta errores

Figura 2. Actividad VerVideo

G.4. Ajustes para móviles y otros arreglos:

1. Ajuste del tamaño de los textos para la versión móvil.

2. Centrado de los botones en main_activity.xml y resto de capítulos.

3. Añadir título en la parte superior indicando en qué capítulo, parte estamos o el título en la actividad principal (Aprende con Rey).

La fuente utilizada para resaltar los títulos es Gorditas-Bold.ttf de Google Fonts.

Android no nos proporciona herramientas para poder asignar fuentes externas a vistas, por lo que tenemos dos opciones para realizar esta tarea: utilizar librerías externas que nos permitan añadir una fuente a un layout o hacerlo mediante código. En nuestro proyecto lo hemos hecho mediante código pero pienso que es uso de una librería externa es algo que se debería considerar más adelante de cara a la publicación de la aplicación y a futuras versiones de ésta. Para establecer una fuente diferente a la que nos proporciona Android por defecto, debemos añadir la fuente a la carpeta assets, instanciar la fuente y aplicarla a la vista correspondiente. Un proceso bastante tedioso si se utilizan diferentes tipos de fuente aplicados a muchas vistas (ejemplo: activity AcercaDe)

I) Recursos:

I.1. Conversión de las fichas blancas y negras a VectorDrawables

A la hora de trabajar con los archivos vectoriales (.svg) que nos facilitó el diseñador de las piezas nos encontramos con bastantes problemas. Android nos permite reemplazar los recursos png por gráficos vectoriales definidos en un archivo .xml y escalables en cualquier tipo de pantalla mediante los VectorDrawables. Antes estaban limitados a dispositivos con versiones con Lollipop o posteriores, pero ahora los tenemos disponibles mediante la biblioteca de compatibilidad: support-vector-drawable. Android nos ayuda en la conversión del archivo .svg a .xml permitiéndonos crear

VectorDrawables pero la conversión no es 100% compatible con este tipo de ficheros no reconoce los gradientes, ni ciertas etiquetas q están incluidas en los diseños de las piezas, mostrándolas totalmente negras como resultado y perdiendo la mayor parte de los contornos de éstas como se muestra en la siguiente imagen.

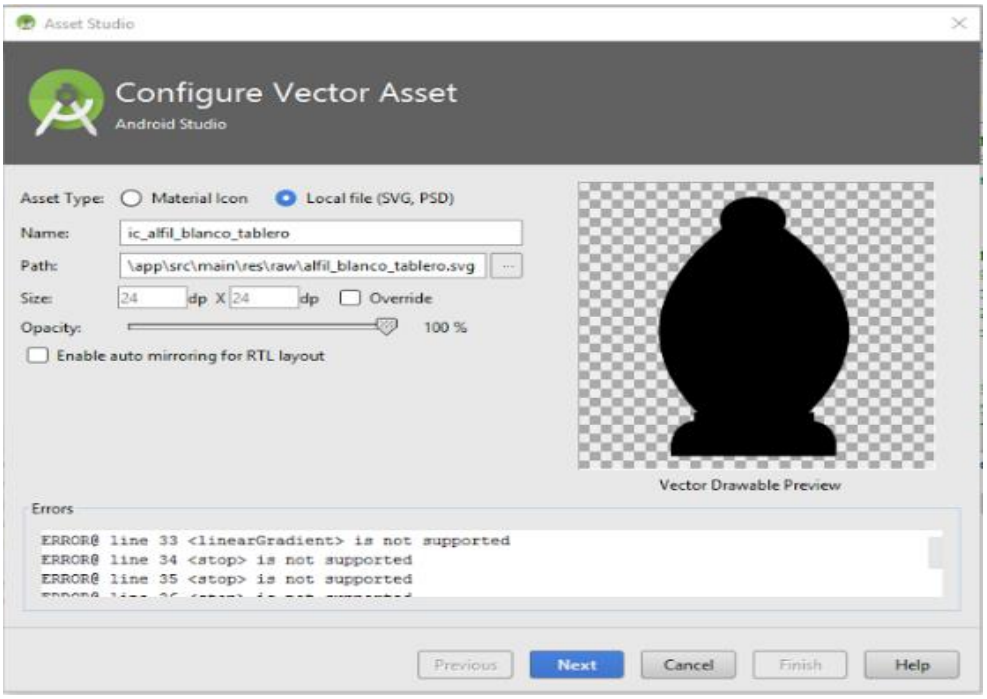

Figura 3. xml resultante de la conversión de svg a VectorDrawable

Debido a la problemática que se nos presentaba con los vectorDrawables estudiamos diferentes alternativas. Valoramos el incorporar una librería externa, SVG-Kit for Android, que nos permitiera mostrar las piezas originales y escalarlas en función de la pantalla frente al uso de los vectorDrawables. Esta librería calcula en tiempo de ejecución el acrchivo Bitmap necesario para la resolución con la que se va a trabajar, pudiendo afectar en el rendimiento de la aplicación. En el siguiente esquema mostramos las ventajas e inconvenientes de cada uno de ellos:

Opción A: VectorDrawable:

- Ventajas:
  - Usamos tecnología 100% Android.
  - No hace falta añadir librerías externas.
  - Aprendemos cómo hacer algo a más bajo nivel que si usamos una librería externa.

- Inconvenientes
  - Hay que retocar los gráficos SVG antes de poderlos usar.
  - No se muestran los gradientes.

Opción B: Librería SVG Kit for Android

- Ventajas:
  - Podemos usar los ficheros SVG tal cual están.
  - Los gráficos se muestran con gradientes.

- Inconvenientes:
  - Hay que usar e incluir una librería externa (con lo cual el APK será más grande)
  - Hay que valorar cómo afecta en tiempo y consumo de recursos tener que convertir los SVG en Bitmap en tiempo de ejecución
  - No sabemos cómo va a funcionar del drag & drop.

Finalmente, tras valorar las ventajas e inconvenientes decidimos usar los VectorDrawables, por estabilidad, fiabilidad, dudas en cuanto a rendimiento de la librería externa y por aprender más sobre el uso de tecnología Android.

Una vez convertidos los archivos mediante VectorAssets, que quedaban como en la imagen superior, para poder recuperar el color de las piezas trabajamos cada uno de los ficheros de forma independiente, detectando qué línea de código pertenece a qué parte de la pieza y restaurando el color a cada zona. Al perder los contornos, para que la pieza no quedará demasiado plana, jugamos con diferentes tonos de grises (elegidos de los que componen el grandiente original) para tratar de dar más volumen a la pieza. Posteriormente uno de los compañeros del equipo de trabajo descubrió como generar los .xml sin perder los contornos, adjunto el link de su método.

I.3. Recorte del tictac de cuando se hace una pregunta utilizando el programa SoundForge

I.4. Iconos en vectorDrawables para ajustes y acerca de.

J) Tareas Varias

J1. Creación de las preferencias de la aplicación mediante el uso de fragments, activity y layout

J2. Creación de la actividad y layout Acerca de, usando un cuadro de diálogo como tema de la actividad.

Para crear la actividad y darle un aire más informal acerca de utilizamos diferentes tipografías del estilo “escrito a mano”:

IndieFlower.ttf, LoveYaLikeASister.ttf y Dekko-Regular.ttf

J3. Cambio del nombre del paquete: fusionar inicialmente los trabajos de todos conllevó a tener que modificar el nombre del paquete cambiándolo a es.upv.mmoviles.ajedrez

J4. Hacer que la aplicación sea full screen.