Software para Dispositivos Móviles
Grado en Ingeniería Informática del Software
Escuela de Ingeniería Informática – Universidad de Oviedo

Gestión de la vista

Juan Ramón Pérez Pérez y Mª del Puerto Paule Ruiz

Departamento de Informática

jrpp@uniovi.es

Gestión imágenes

- Imagen 2D
- Imagen 3D (OpenGL)

Vista en Android

- Dos maneras de crear una vista:
  - Creando componentes en diseño (a través de ficheros XML)
  - Creando componentes en ejecución
- Tipo de vista:
  - Componentes estándar (Layout, componentes).
  - Vista personalizada

Recursos de tipo imagen

- Provisión de recursos (incorporar distintos tipos de recursos a la aplicación)
- Carpeta de recursos: drawable/
- Archivos de mapas de bits (.png, .9.png, .jpg, .gif) o archivos XML que se han compilado en los subtipos de **recursos de elemento de diseño**.

Recursos tipo transición

- Archivo XML que define un elemento de diseño capaz de fundirse entre dos recursos de elementos de diseño. Crea un TransitionDrawable.

```
<?xml version="1.0" encoding="utf-8"?>
<transition
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/guitar" />
    <item android:drawable="@drawable/flute" />
</transition>
```

Acceso a recursos

- Una vez declarado el recurso se puede usar haciendo referencia al ID de recurso.
- El ID de recurso siempre está compuesto por:
  - El *tipo de recurso*
  - El *nombre del recurso*, que es el nombre de archivo sin la extensión o el valor en el atributo XML android:name
- Dos maneras de acceder a un recurso:
  - En código: Usando un valor entero de una subclase de tu clase R; por ejemplo: R.string.hello
  - En XML: Usando una sintaxis XML que también corresponde al ID de recurso definido en tu clase R; por ejemplo: @string/hello

Componente ImageView – Creación en Ejecución

```
LinearLayout mLinearLayout;

protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);

  // Crea un LinearLayout en el que vamos a introducir el ImageView
  mLinearLayout = new LinearLayout(this);

  // Instanciamos un ImageView e incluimos sus propiedades
  ImageView i = new ImageView(this);
  i.setImageResource(R.drawable.guitar);

  // hacemos que el ImageView se adapte a las dimensiones del Drawable
  i.setAdjustViewBounds(true);
  i.setLayoutParams(new Gallery.LayoutParams(LayoutParams.WRAP_CONTENT,
      LayoutParams.WRAP_CONTENT));

  // Añadimos ImageView al layout y lo mostramos como la vista activity
  mLinearLayout.addView(i);
  setContentView(mLinearLayout);
}
```

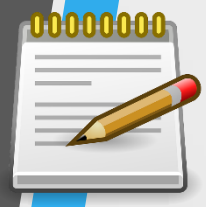
Componente ImageView – Creación en Diseño

- A través del layout podemos mostrar una imagen en pantalla:

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:srcCompat="@drawable/guitar" />
```

- Recuperar referencia al componente / Vista desde el programa

```
imagen= (ImageView)findViewById(R.id.imageView);
```



Ejercicio 1a

1. Al pinchar en la imagen, cambia el contenido del componente con otro *drawable* diferente a la original.
2. Aplicar una transición para realizar el cambio (*startTransition*)
3. Conservar el estado de la transición para hacer una transición directa o inversa (*reverseTransition*)

Operaciones sobre la imagen- Escalar imagen

- Representación a Bitmap:

```
Bitmap
bmap=BitmapFactory.decodeResource(getResources(),R.drawable.guita
r);
```

- Escalado del Bitmap:

```
Bitmap bMapScaled= Bitmap.createScaledBitmap(bmap,
imagen.getWidth()/2, imagen.getHeight()/2, true);
```

- Colocar en el componente la nueva imagen escalada:

```
imagen.setImageBitmap(bMapScaled);
```

Ejercicio 1b

1. Al pinchar en un botón la imagen reduce su tamaño
2. ¿Se podría volver a aumentar imagen?

Operaciones sobre la imagen- Rotación

- Los dos primeros pasos igual que en el escalado, pero además…
- Se precisa una matriz y ángulo de rotación

```
Matrix mat=new Matrix();
rotacion=rotacion+10;
mat.postRotate(rotacion);
Bitmap bMapRotated=Bitmap.createBitmap(bmap, 0, 0,
bmap.getWidth(), bmap.getHeight(), mat, false);
```

- Colocar en el componente la nueva imagen rotada:
- `imagen.setImageBitmap(bMapRotated);`

Ejercicio 1c – Rotación de la imagen

1. Al pinchar en un botón, la imagen rota
2. Mejorar código para rotar la imagen escalada o cambiada

Para ampliar sobre gestión de imágenes

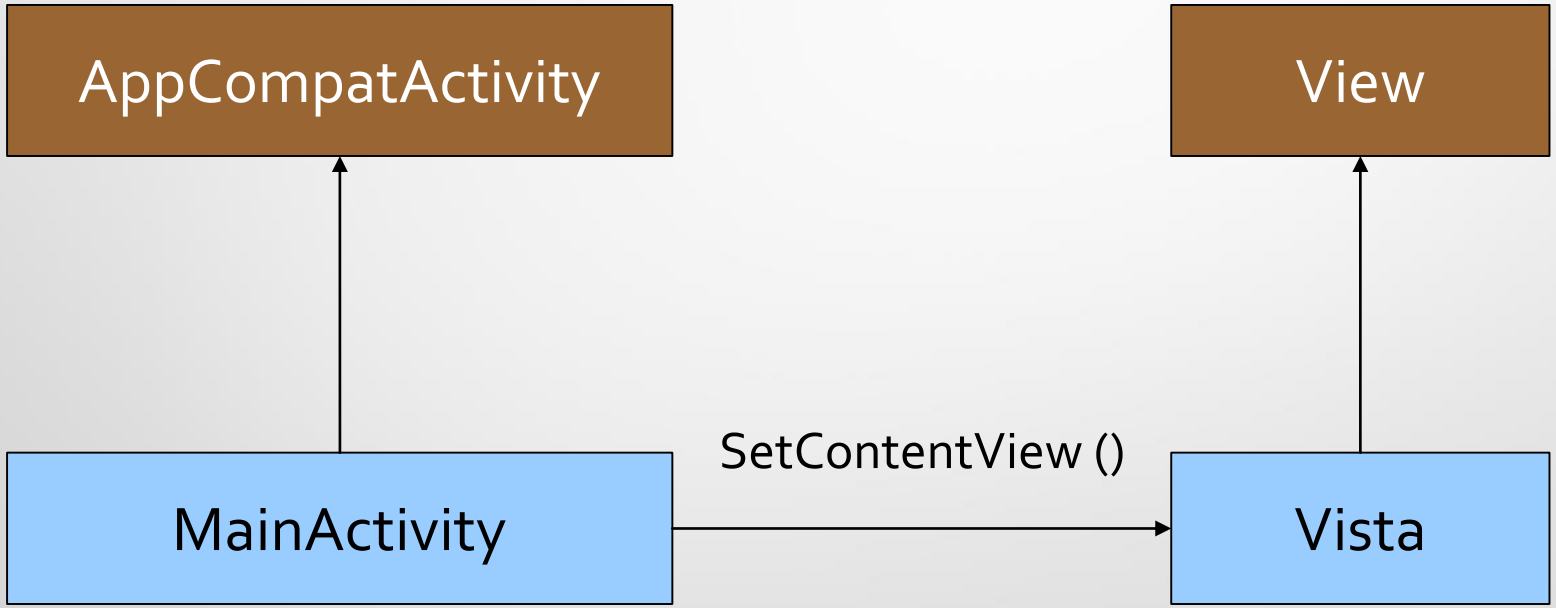
- Drawable
  - https://developer.android.com/reference/android/graphics/drawable/Drawable.html
- ImageView
  - https://developer.android.com/reference/android/widget/ImageView.html
- Trabajar con interfaces dibujables
  - https://developer.android.com/training/material/drawables.html
- Manejar Bitmaps:
  - https://developer.android.com/topic/performance/graphics/index.html

Gestión manual de la vista

Gestión manual de la vista

- Implementar una clase que herede de View
- Redefinir el Método OnDraw()
- Permite dibujar libremente sobre un canvas

Diagrama de clases



Clase Vista

- Clase padre que gestiona todos los eventos de la vista
- Establece la imagen a visualizar → Drawable imagen

```
getResources().getDrawable(R.drawable.imagen,null);
```

- Método onDraw
  - Es llamado por Android cada vez que quiere dibujar la vista
  - setBounds: Establece donde se va a dibujar la imagen

Implementación

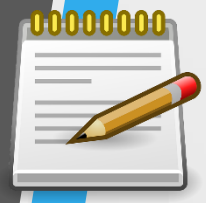
- Establecer imagen en el constructor de Vista

```
imagen =
    getResources().getDrawable(R.drawable.imagen,null);
```

- Clase MainActivity
  - Establece la vista con el objeto de Vista (personalizado)

```
Vista vista =new Vista(this);

setContentView(vista);
```



Ejercicio 2a

1. Establecer la vista de una actividad que contenga una imagen (VistaManual)
2. Dibuja un circulo en el centro de la actividad (canvas)

Touch

- La funcionalidad “Touch” consiste en arrastrar un objeto por pantalla
- El método “onTouchEvent” se ejecuta cada vez que se toca la pantalla, se mueve algo y se suelta

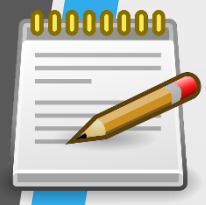
Touch en la clase Vista

- Método - acciones:
  - onTouchEvent: Evento que gestiona el "touch" . Acciones:
  - ACTION_DOWN: El usuario inicia el touch.
  - ACTION_UP: El usuario finaliza el touch.
  - ACTION_MOVE: El usuario se desplaza por la pantalla.
- Atributos:
  - mX: Coordenada X dónde comienza el touch
  - mY: Coordenada Y dónde comienza el touch
- Importante: Invocar *invalidate()* para redibujar la imagen

Métodos auxiliares

- Touch_start: El usuario presiona la pantalla. Inicialización de las variables mX y mY.
- Touch_up: El usuario levanta el dedo de la pantalla. Liberación de mX y mY.
- Touch_move: El usuario mueve el dedo por la pantalla.

- Mediante dx y dy conocemos la distancia movida por el usuario.
- Actualizar las coordenadas de la imagen para cuando se vuelva a pintar.
- Actualización de mX y mY por si el usuario sigue moviéndose



Ejercicio 2b

1. Dado el ejercicio 4, hacer que la imagen se mueva por la pantalla(VistaManualTouch)
2. Que la imagen sólo se mueva cuando el usuario esté pulsando dentro de la misma

Para ampliar sobre gestión de la vista

- Canvas y Drawables
  - https://developer.android.com/guide/topics/graphics/2d-graphics.html
- Creación de nuevos controles con vistas personalizadas
  - https://developer.android.com/training/custom-views/index.html