Nuevos Paradigmas de Interacción
Práctica 3

Aplicación Android con los sensores:

- Cámara
- MultiTouch
- Gestos

Autores:

- Eva María Almansa Aránega
- Luis Alberto Segura Delgado
- Samuel López Liñán

Contenido

1- Descripción de la Aplicación Realizada

La aplicación desarrollada detectará gestos multitouch para iniciar la cámara y tomar una foto o grabar un video. Si se realiza un desplazamiento hacia abajo con **dos** dedos se activará la cámara para la realización de fotos. Si el desplazamiento con dos dedos se realiza hacía arriba, se activará la cámara para la grabación de un video.

Además, al inicio de la aplicación se podrá detectar, si el usuario lo desea, el trazo de las siguientes letras: t, F y W. Cada trazo de letra detectado será para abrir cada una de las distintas aplicaciones "sociales": t será para "Twitter", la letra F será para "Facebook" y la letra W para "Whatsapp". En caso de que no estén alguna de las aplicaciones instaladas en el teléfono móvil del usuario, redirecciona a la aplicación playStore de Android, para que pueda, si lo desea, descargarse la aplicación en cuestión.

Para mostrar el funcionamiento del inicio de la aplicación al usuario y añadir el nombre que se mostrará en la instalación de la aplicación, se modifica el archivo strings.xml que se encuentra en la subcarpeta res/values y se añade las siguientes líneas:

```
<string name="app_name">MegaLauncher</string>
<string name="tutorial">Bienvenido a la app de fotos/vídeos!\nPara usar la app:
\n
        * Desliza con dos dedos hacia arriba para hacer una foto o hacia abajo
para hacer un vídeo.\n\n
        ** Dibuja una t para acceder a Twitter, una F para acceder a Facebook o
una W para iniciar WhatsApp.
</string>
```

2- Descripción de la Solución

2.1- Cámara

Para la configuración del proyecto se ha de modificar el archivo AndroidManifest.xml y solicitar permisos de: Cámara, vídeo, audio y almacenamiento en la SD:

```
  <manifest... >
   ...
<!--- Permisos para la camara, audio, video y SD >-->
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.RECORD_VIDEO" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-feature android:name="android.hardware.camera" />

</manifest>
```

Si surge algún problema de configuración del proyecto se recomienda mirar el enlace [2].

Lo siguiente que se ha de hacer es agregar funcionalidad a los permisos anteriormente mencionados. Para ello se crea una nueva clase CameraController.java:

-Variables para el control de la actividad:

```
  //Control de solicitud de actividad
  private static final int CAMERA_CAPTURE_IMAGE_REQUEST_CODE = 100;
  private static final int CAMERA_CAPTURE_VIDEO_REQUEST_CODE = 200;
  public static final int MEDIA_TYPE_IMAGE = 1;
  public static final int MEDIA_TYPE_VIDEO = 2;

  private Context c;
```

- Constructor:

```
  public CameraController(Context cont)
  {
      c = cont;
  }
```

-Capturar una imagen, añadir un método:

```
/*
* Capturing Camera Image
*/
public void captureImage()
{
      // Dar un nombre a la fotografia para almacenarla en la carpeta por
      defecto del movil
      String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new
      Date());

      ContentValues values = new ContentValues();
      values.put(MediaStore.Images.Media.TITLE, "IMG_" + timeStamp + ".jpg");

      Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

      //fileUri = ((Activity)c).getOutputMediaFileUri(MEDIA_TYPE_IMAGE);
      fileUri =
      ((Activity)c).getContentResolver().insert(MediaStore.Images.Media.EXTERNAL
      _CONTENT_URI, values); // store content values

      intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);

      // start el intent" de captura

      ((Activity)c).startActivityForResult(intent,CAMERA_CAPTURE_IMAGE_REQUEST_C
      ODE);
}
```

-Grabar un vídeo, añadir un método:

```
/*
* Recording video
*/
public void recordVideo()
{
      Intent intent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);

      //fileUri = getOutputMediaFileUri(MEDIA_TYPE_VIDEO);

      // set video quality
      intent.putExtra(MediaStore.EXTRA_VIDEO_QUALITY, 1);

      //intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);

      // start el intent" de captura de vï¿½deo
```

```
        ((Activity)c).startActivityForResult(intent,CAMERA_CAPTURE_VIDEO_REQUEST_C
        ODE);
}
```

-Actividad que se produce una vez finalizado la Captura/Grabación de la cámara:

```
/**
* Receiving activity result after closing the camera
* */
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
        //Comprobar la solicitud del usuario de guardar la fotografia
        if (requestCode == CAMERA_CAPTURE_IMAGE_REQUEST_CODE)
        {
              Toast.makeText(((Activity)c).getApplicationContext(),"CAMERA_CAPTURE
              _IMAGE_REQUEST_CODE", Toast.LENGTH_SHORT).show();
              if (resultCode == ((Activity)c).RESULT_OK)
              {
                    Toast.makeText(((Activity)c).getApplicationContext(),"RESULT_O
                    K", Toast.LENGTH_SHORT).show();
                    previewCapturedImage();
              } else if (resultCode == ((Activity)c).RESULT_CANCELED)
              {
                    Toast.makeText(((Activity)c).getApplicationContext(),"User
                    cancelled image capture", Toast.LENGTH_SHORT).show();
              } else {
                    Toast.makeText(((Activity)c).getApplicationContext(),
                    "Sorry! Failed to capture image", Toast.LENGTH_SHORT).show();
              }
        }
        else
        {
             Toast.makeText(((Activity)c).getApplicationContext(),"NO ::::
             CAMERA_CAPTURE_IMAGE_REQUEST_CODE", Toast.LENGTH_SHORT).show();
        }
}
```

-Almacenar la imagen desde la ruta, añadir método:

```
/*
* Display image from a path to ImageView
*/
private void previewCapturedImage()
{
     try
     {
          // hide video preview
          //videoPreview.setVisibility(View.GONE);

          //imgPreview.setVisibility(View.VISIBLE);

          // bimatp factory
          BitmapFactory.Options options = new BitmapFactory.Options();

          // downsizing image as it throws OutOfMemory Exception for larger
          // images
          options.inSampleSize = 8;

          Bitmap bitmap = BitmapFactory.decodeFile(fileUri.getPath(),options);

          System.out.println(fileUri.getPath());

          //imgPreview.setImageBitmap(bitmap);

     } catch (NullPointerException e) {
```

```
      e.printStackTrace();
      }
}
```

2.2- MultiTouch

Para la gestión de gestos multitouch, se ha realizado una clase llamada “MultiTouchView”, que extiende a la clase “View”. La idea es que la actividad principal incorpore en la interfaz una vista basada en esta clase. En esta clase se “escuchan” los eventos de toque de la vista. Una vez recibido un evento de toque, se analiza según los parámetros recibidos.
La función “onTouchEvent” recibe una variable de tipo “`MotionEvent`”, que contiene toda la información necesaria para saber lo ocurrido. Concretamente se hace uso de la información sobre el puntero (dedo) que ha disparado el evento. Podemos obtener su identificador, el cual guardaremos en un array junto a la posición inicial (la posición que tuvo cuando realizo la pulsación). Esta variable contiene información sobre el tipo de acción que ha sido realizada.
Se distinguen tres acciones clave:

- Movimiento del puntero (`MotionEvent.`*`ACTION_MOVE`*): Cuando nos encontramos con esta acción quiere decir que nuestro puntero se ha movido, por lo que actualizaremos la información almacenada sobre él. (Comprobando el movimiento realizado desde la posición inicial hasta la posición actual).

```
// Si un dedo se ha movido..
    case MotionEvent.ACTION_MOVE: {
      for (int size = event.getPointerCount(), i = 0; i < size; i++) {
      // Para cada dedo, lo buscamos en nuestros arrays y actualizamos la
informacion
        PointF point = mActivePointers.get(event.getPointerId(i));
        PointF initialPoint = initialPos.get(event.getPointerId(i));
        if (point != null) {

            // Realizamos la diferencia para saber el desplazamiento que se ha
realizado
            float moveX = initialPoint.x - event.getX(i);
            float moveY = initialPoint.y - event.getY(i);


            System.out.println("("+initialPoint.x+", "+initialPoint.y+")
("+event.getX(i)+", "+event.getY(i)+") (" +moveX+", "+moveY+")");

            // Si nos movemos 20 unidades hacia arriba o hacia abajo tendremos
que hemos realizado el gesto
            if( moveY < -20 )
            {     // Only down move!
                  detectedFingerMoveDown.setValueAt(event.getPointerId(i),
true);
            }
            else if( moveY > 20 )
            {     // Only up move!
                  detectedFingerMoveUp.setValueAt(event.getPointerId(i), true);
            }

        }
       }
      break;}
```

- Pulsación del puntero (*MotionEvent.ACTION_DOWN* o *MotionEvent.ACTION_POINTER_DOWN*): Cuando nos encontramos con esta acción quiere decir que un nuevo puntero ha establecido contacto con la pantalla del dispositivo. Por tanto debemos almacenar la información sobre el nuevo puntero. (Su posición inicial)

```
// Si lo que ha ocurrido es que un nuevo dedo a "pulsado" la pantalla..
    case MotionEvent.ACTION_DOWN:
    case MotionEvent.ACTION_POINTER_DOWN: {
      // Añadimos la informacion a nuestros arrays
      PointF f = new PointF();
      f.x = event.getX(pointerIndex);
      f.y = event.getY(pointerIndex);
      mActivePointers.put(pointerId, f);
      initialPos.put(pointerId, f);
      detectedFingerMoveUp.put(pointerId, false);
      detectedFingerMoveDown.put(pointerId, false);
      break;
    }
```

- Un puntero deja de estar en contacto con la pantalla (*MotionEvent.ACTION_POINTER_UP* o *MotionEvent.ACTION_POINTER_UP* o *MotionEvent.ACTION_CANCEL*): Cuando nos encontramos con esta acción quiere decir que un puntero ha dejado de estar en contacto con la pantalla. Por tanto debemos eliminar toda la información que habíamos almacenado sobre el.

```
// Si el dedo deja de estar en contacto con la pantalla..
    case MotionEvent.ACTION_UP:
    case MotionEvent.ACTION_POINTER_UP:
    case MotionEvent.ACTION_CANCEL: {
      // Eliminamos la informacion del dedo de nuestros arrays
      mActivePointers.remove(pointerId);
      detectedFingerMoveUp.remove(pointerId);
      detectedFingerMoveDown.remove(pointerId);
      initialPos.remove(pointerId);
      break;
    }
```

Una vez que sabemos cuándo podemos guardar la información que necesitamos y como, lo único que necesitamos es ir comprobando si se ha detectado el movimiento. Para ello haremos uso del evento “onDraw”, que también detecta este tipo de eventos. En él lo que haremos será dibujar circulitos para que se pueda saber cuántos dedos se han detectado y comprobar si se ha detectado desplazamiento de dos dedos hacía arriba o hacia abajo. Si se ha detectado desplazamiento, lo único que hacemos es hacer llamadas a la clase encargada de controlar la cámara.

```
// Si detecto movimiento hacia arriba, aviso y llamo a la clase encargada de la
camara
    if( detectedUp() )
    { // Two fingers move detection
      canvas.drawText("Detected Up!", 10, 40 , textPaint);
      Toast.makeText(((Activity)c).getApplicationContext(),
                "Image Detected! (Up)", Toast.LENGTH_SHORT)
                .show();
      resetDetection(); // Reseteo la deteccion para evitar que en el mismo
intante habra mas veces la camara
      cam.captureImage();
    }
    // Si detecto movimiento hacia abajo, aviso y llamo a la clase encargada de
la camara
    else if( detectedDown() )
    {
      canvas.drawText("Detected Down!", 10, 40 , textPaint);
      Toast.makeText(((Activity)c).getApplicationContext(),
                "Video Detected! (Down)", Toast.LENGTH_SHORT)
                .show();
      resetDetection(); // Reseteo la deteccion para evitar que en el mismo
intante habra mas veces la camara
      cam.recordVideo();
    }
```

Para el desarrollo de esta parte se ha reutilizado el código de [1]. Se ha usado como base porque está muy bien documentado y además viene con la parte de dibujar los dedos detectados en pantalla. Se recomienda que se lea esa documentación para comprender mejor el manejo de este tipo de eventos en Android.

2.3- Gestos

Para la gestión de gestos o gestos onetouch por diferenciarlos de los multitouch, se ha creado una vista de tipo <android.gesture.GestureOverlayView> el activity_main.xml que es donde está el layout de la actividad principal pudiendo ponerle las características que sean necesarias para cada gusto, con esta vista podremos ver el patrón conforme lo vayamos dibujando mediante una línea amarilla, para este caso se ha puesto la característica de que pueda detectar múltiples trazas ya que se han utilizado patrones de varias trazas (android:gestureStrokeType="multiple")
Los gestos que queremos que se detecten se almacenarán en la carpeta res/raw/ de nuestro proyecto para poder acceder a ellos fácilmente utilizando el siguiente código dentro del onCreate de nuestra actividad principal:
gesturelib = GestureLibraries.fromRawResource(this, R.raw.gestures);
Una vez cargada la biblioteca de gestos la cual hemos creado nosotros (hay una aplicación en la página de Android que te puedes descargar que se llama gestureBuilder y te permite crear una biblioteca de gestos) pasamos a definir la capa de gestos, en este caso es la vista anteriormente mencionada que le hemos asignado una id (android:id="@+id/gesture_view") para acceder a ella fácilmente desde la actividad con la siguiente línea:
GestureOverlayView gestureview = (GestureOverlayView)findViewById(R.id.gesture_view);
Seguidamente vamos a asignar a la vista el Listener de gestos para cada acción:
gestureview.addOnGesturePerformedListener(gesturelistener);

Y ya solo nos queda crear el Listener de gestos que será el encargado de interpretar que gesto se produce para luego añadírsele una acción a cada patrón efectuado, o sea, a cada gesto:

```
// Crea el Listener de Gestos
    private GestureOverlayView.OnGesturePerformedListener gesturelistener = new
GestureOverlayView.OnGesturePerformedListener() {
        @Override
        // Detecta la realizacion de un gesto en la Vista de Gestos
        public void onGesturePerformed(GestureOverlayView overlay, Gesture
gesture) {
            // Almacena una lista con las posibles soluciones y la puntuacion de
cada una (ordenada de mayor a menor puntuacion)
            ArrayList<Prediction> predictions = gesturelib.recognize(gesture);
            // Si obtengo una solución decente con una puntuación mayor que 4 es
que hay algún patron que tiene bastante parecido al dibujado
            /*gesturesText.setTextSize(12);
            gesturesText.setText("Gestos:\n");
            for(Prediction prediction : predictions)
              gesturesText.append("N: " + prediction.name + ", P: " +
prediction.score +", S: "+gesture.getStrokesCount()+"\n");*/
            if (predictions.get(0).score > 3) {
                Boolean instalado = false;
                String appName;
                Intent LaunchIntent;
                // Lista de aplicaciones del dispositivo
                List<ApplicationInfo> apps =
getPackageManager().getInstalledApplications(0);
                // Comprobamos cual es el patrón dibujado
                if(predictions.get(0).name.contains("F"))
                    appName="com.facebook.katana";
                else
                    if(predictions.get(0).name.contains("W"))
                        appName="com.whatsapp";
                    else
                    if(predictions.get(0).name.contains("t"))
                        appName="com.twitter.android";
                    else
                        appName="Nothing";
                if(appName!="Nothing") {
                    // Si se ha detectado un patrón buscamos en la lista a que
aplicación corresponde
                    for (ApplicationInfo app : apps) {
                        // Si la aplicación aparece significa que esta
instalada, entonces la abrimos
                        if (app.className != null &&
app.className.contains(appName)) {
                            //gesturesText.append("Name: " + app.className +
"\n");
                            instalado = true;
LaunchIntent = getPackageManager().getLaunchIntentForPackage(appName);
                            startActivity(LaunchIntent);
                        }
                    } // Si no esta instalada la aplicacion le mandamos al
market para que se la descargue
                    if(!instalado){
                        LaunchIntent = new Intent(Intent.ACTION_VIEW);
            LaunchIntent.setData(Uri.parse("market://details?id="+appName));
                        startActivity(LaunchIntent);
                    }
                }
            }
        }
    };
```

3- Lecturas Recomendadas y Referencias

[1]- http://www.vogella.com/tutorials/AndroidTouch/article.html
[2]- http://developer.android.com/training/camera/photobasics.html