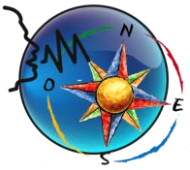
Tutoriales Android

Nuevos Paradigmas de Interacción

Mario Ruiz Calvo

(https://github.com/mrucal)

BrujulaVoz



Descripción

La aplicación consta de dos partes. En la primera, presionando sobre el botón *INICIAR* se lanza el reconocimiento de voz en el que deberemos decir un punto cardinal y un margen de error.

Si el reconocimiento y el contenido son correctos se abre una nueva actividad en la que se muestra una brújula. Buscaremos el punto cardinal que hemos dicho previamente girando el dispositivo hasta que coincida con la línea vertical situada arriba. Cuando estemos en la dirección correcta el color de fondo cambiará a verde.

Funcionamiento

En la clase *reconocimiento*: Comenzamos inicializando el botón que creará un nuevo intent para el reconocimiento cuando hagamos click en el.

En el método *onActivityResult* el intent que hemos iniciado con el botón nos devolverá un ArrayList con los posibles strings que se han escuchado. El primer de estos es strings será el que se ajuste con mayor probabilidad a lo que se ha dicho. Anlizamos este String devolviendo un mensaje de error en los siguientes casos: que se hayan dicho mas (o menos) de dos palabras, que la primera palabra no sea un punto cardinal o que la segunda palabra no sea un número o que el numero sea menor de 100.

Si el reconocimiento es correcto iniciamos un nuevo intent al que pasamos el punto cardinal (ya en grados) y el error. En el método *onCreate* obtenemos estos parámetros y se inicializa el sensor que obtiene los grados.

El método *onSensorChanged* se encargará de obtener los grados de la brújula cada vez que cambien. Si nos encontramos en la dirección pedida anteriormente cambiaremos el color de fondo de la apliación.

Con los grados obtenidos y un objeto de tipo *RotateAnimation* rotaremos la imagen de la brújula para que señale la dirección correcta.

GestoFoto

Descripción

La aplicación consta de dos partes. En la primera, debemos realizar un dibujo aproximado del gesto que nos dan en la imagen. Si el gesto tiene suficiente precisión (mayor a 4.7) se abre una nueva actividad en la que se muestra la cámara. Tras una espera de 3 segundos la aplicación tomará una foto mostrando un mensaje de confirmación y guardando la imagen en la memoria interna del dispositivo.

Funcionamiento

En el método *onCreate* de la clase gestos cargamos la librería de gestos del fichero de gestos obtenido con la aplicación *Gesture Tool*. Inicializamos también el *GestureOverlayView* que capturará el gesto.

El método *onGesturePerformed* se ejecuta cuando se dibuja un nuevo gesto. Una vez dibujado el gesto, en la variable *predictions* se almacenaran la lista de posibles coincidencias ordenadas de mayor a menor probabilidad de acierto. Si el primero de ellos es igual al gesto que estábamos buscando con una precisión mayor a 4.7 iniciamos una nueva actividad que ejecutará la cámara.

En el método *onCreate* de la clase cámara inicializamos el *SurfaceView* que mostrará la cámara y creamos un *Handler* que se encarga de ejecutar el método para tomar una foto, *takePicture*, tras una espera de 3 segundos.

PuntoGPSQR

Descripción

Al iniciar la aplicación y presionar sobre el botón *ESCANEAR* se abre la aplicación *BarcodeScaner*. Una vez escaneado el código QR se muestra un mapa centrado en la posición actual y un marker que muestra el *GOAL* obtenido. Conforme vayamos avanzando se dibujará una línea roja indicando el camino recorrido que cambiará a verde cuando hayamos alcanzado el *GOAL*.

Funcionamiento

Al presionar el botón *ESCANEAR*, el método *onClinck* inicia el escaneo del código QR con *scanIntegrator.initiateScan* que se encarga de abrir la aplicación *BarcodeScaner*. Para ello utiliza las clases *IntentIntegrator* y *IntentResult* que han sido añadidas al proyecto siguiendo las indicaciones del siguiente tutorial.

Una vez finalizado el proceso de escaneo, obtenemos el resultado en un string en *onActivityResult* y creamos e iniciamos un nuevo intent al que pasamos el string que contiene el *GOAL*.

En *onCreate* inicializamos los botones y ajustamos algunos parámetros iniciales del mapa con la función *setUpMapIfNeeded* como el tipo de mapa (MAP_TYPE_NORMAL), la ubicación inicial del mapa y el zoom. Después obtenemos el string con la latitud y longitud del goal que nos ha enviado la actividad anterior. Convertimos el string a un objeto del tipo *LatLng* con el que añadimos un marker azul (*setMarker*) con la etiqueta *GOAL* e iniciamos la escucha con *setOnMyLocationChangeListener*. Cada vez que cambiemos de posición se irá añadiendo al objeto de tipo ruta que hemos creado y dibujando la ruta en rojo hasta que lleguemos al *GOAL* que se dibujara en verde.

Se ha creado la clase Ruta para facilitar el dibujo del recorrido. Tiene variables para almacenar la posición del *GOAL* y las posiciones cercanas al *GOAL* para que no sea necesario alcanzar la posición exacta para dar el recorrido como bueno. También tiene un método que comprueba si se ha llegado ya al *GOAL*.

MovimientoSonido

Descripción

La aplicación utiliza el acelerómetro del dispositivo para detectar un movimiento en el teléfono. En este caso el movimiento será: levantar el dispositivo (X:0,Y:10,Z:0) y girarlo a la izquierda (simulando el movimiento de un latigo). Si se reconoce correctamente el movimiento se escuchará un sonido y se mostrara brevemente la imagen de un látigo.

Para ayudar al usuario a colocar el dispositivo en la posición correcta la aplicación mostrara los valores de los 3 ejes.

Funcionamiento

Para facilitar la aplicación y detección del gesto se ha creado una clase Posición. Estará compuesto por un float por cada eje y un método *between* con el que dada una posición a y 3 valores umbrales devuelve true si la posición está cercana a la posición a con respecto a los umbrales.

En el método *onCreate* de la clase principal se inicializa el sensor acelerómetro y se inicia la escucha con *sm.registerListener*.

El método *onSensorChanged* detectará cuando se ha movido el dispositivo. Los valores del sensor se encuentran en *event.values*. Además almacenaremos el tiempo actual.

En este caso el gesto está formado por dos posiciones. Si la posición actual está cercana a la primera posición del gesto (lo comprobamos con el método *between* de la clase *Posicion*) incrementamos el índice a las posiciones del gesto y actualizamos el tiempo. Detectaremos entonces el siguiente movimiento, que deberá hacerse en menos de un segundo (de no ser así el índice se reinicia a 0 para detectar de nuevo la primera posición). Si la segunda posición es correcta y se hace en el tiempo requerido reproducimos el sonido del látigo y mostramos una imagen (durante un segundo).

De manera orientativa los *TextView* con los valores de los tres ejes se actualizan siempre que se detecte movimiento en el dispositivo, aunque este movimiento no forme todavía parte del gesto que hay que realizar.

PuntoSorpresa

Descripción

La aplicación hace uso del sensor de proximidad para avanzar o retroceder imágenes en una galería de fotos, o para pausar o reanudar una presentación de las imágenes.

Los gestos detectados con el sensor de proximidad son:

- Avanzar (pasada corta): Pasar la mano o el dedo por encima del sensor (deteniéndolo brevemente).
- Retroceder (doble pasada): Pasar la mano o el dedo dos veces rápidas por encima del sensor.
- Pausar/Reanudar (pasada larga): Dejar la mano o el dedo encima del sensor al menos un segundo y medio.

Funcionamiento

La clase menú es una interfaz sencilla que muestra dos botones, para mostrar la galería o una presentación. En cualquier caso lanzará un intent con un booleano como parámetro indicando la opción elegida.

En el método de *onCreate* de la clase sorpresa se inicializan los *TextView*, la imagen y el sensor de proximidad. Se inicia la escucha del sensor. Además obtenemos el intent anterior y el booleano que indica si hay que iniciar la galería o la presentación. Si hay que iniciar la presentación ejecutamos el runnable que cambiará la imagen con *setImageResource* e incrementa el índice de las imágenes cada 3 segundos.

En el método *onSensorChanged* tenemos dos opciones dependiendo de si se ejecuta la galería o la presentación.

El sensor de proximidad devuelve en *event.values* un valor 0 cuando ha detectado un objeto y un 10 cuando no ha detectado nada. En la galería cada vez que se detecta un objeto se incrementa el contador *n_pasadas*.

Cuando se ha detectado un objeto, si ya es la segunda vez y han pasado menos de 500 milisegundos desde la última detección consideramos que se ha realizado el gesto de retroceder por lo que se decrementa el índice de la imagen y se reinicia el contador de objetos detectados a 0 para seguir detectando gestos. Si se ha detectado un segundo objeto pero el tiempo ha sido mayor a 500 milisegundos descartamos que haya sido un gesto, por lo que solo se contabiliza como un objeto mas poniendo el contador *n_pasadas* a 1 (para poder contarlo

como posible primer movimiento del gesto de doble pasada).

Si hemos entrado en *onSensorChanged* y no se ha detectado un objeto (v=10), pero ya hay una detección previa. Si el tiempo es mayor a 60 milisegundos quiere decir que el objeto ha estado en el sensor de forma prolongada. Esto se considera como el gesto de avanzar por lo que se incrementa el índice de la imagen y se reinicia *n_pasadas* a 0. Para que se acepte como el gesto de avanzar no valdría una simple pasada, pues se podría confundir con una primera pasada del gesto de retroceder. Debemos obligar a que el objeto permanezca al menos 60 milisegundos para poder distinguirlos.

Si estamos en modo presentación el único gesto posible es el de pasada larga que reproduce y pausa la presentación. La detección es similar al gesto de avanzar. Solo hay que comprobar que cuando no se ha detectado nada (v=10), el anterior objeto estuvo expuesto al senos al menos 1400 milisegundos. La distinción del play y la pausa se controla con un booleano que va actualizándose cada vez que se realiza una de estas acciones. Y las acciones se realizan mediante un *handler.removeCallbacks* para la pausa y *handler.post(runnable)* para el play.

Referencias

BrujulaVoz

http://agamboadev.esy.es/como-crear-un-brujula-en-android/

Seminario: Aplicaciones orales en Android, Zoraida Callejas

GestoFoto

http://nuevos-paradigmas-de-interaccion.wikispaces.com/file/view/NPI-%20Pr%C3%A1ctica%203_2.pdf/536562512/NPI-%20Pr%C3%A1ctica%203_2.pdf

http://www.tutorialeshtml5.com/2013/02/tutorial-crear-y-reconocer-gestos-con.html

http://nuevos-paradigmas-de-interaccion.wikispaces.com/Tutorial+sobre+c%C3%A1mara+en+Android

http://itechmasters.blogspot.com.es/2011/11/how-to-make-programmed-self-timer.html

PuntoGPSQR

http://expocodetech.com/usar-google-maps-en-aplicaciones-android-2/

http://expocodetech.com/usar-google-maps-en-aplicaciones-android-mapa/

http://expocodetech.com/usar-google-maps-en-aplicaciones-android-geolocalizacion/

http://expocodetech.com/usar-google-maps-en-aplicaciones-android-polilineas/

http://expocodetech.com/lectura-de-codigo-de-barras-en-android/

MovimientoSonido

http://nuevos-paradigmas-de-interaccion.wikispaces.com/Aceler%C3%B3metro+en+Android

http://www.maestrosdelweb.com/curso-android-sensores-trabajar-con-acelerometro/

PuntoSorpresa

http://mycyberacademy.com/creando-una-app-que-use-el-sensor-de-proximidad-de-tu-dispositivo-android/

https://www.youtube.com/watch?v=sEzNHlTO-S0