|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Materia:** |  | Programación Móvil II | | |
| **Nombre Informe:** |  | Sensores: Futbolito Pocket | | |
| **Alumno(s):** |  | Isabel Contreras Rico | **Fecha:** | 03/05/2021 |

**Objetivo**

Implementar los conocimientos adquiridos en cuanto a manejo de sensores. Y con ello, desarrollar una aplicación móvil que permita acceder a un juego semejante a Futbolito Pocket, donde se movilice un balón de acuerdo a algún sensor del dispositivo.

**Temas del plan de estudios**

4. Sensores

4.1 SensorManager

4.2 Sensores de movimiento

4.3 Sensores de orientación

**Material**

Para el desarrollo de esta práctica es necesario contar con un IDE para diseñar aplicaciones en Android, como lo es *Android Studio*.

**Marco Teórico**

Sensores

La mayoría de los dispositivos con Android tienen sensores integrados que miden el movimiento, la orientación y diversas condiciones ambientales. Estos sensores son capaces de proporcionar datos sin procesar con alta precisión y exactitud, y son útiles para supervisar el movimiento o posicionamiento tridimensional del dispositivo, o si quieres supervisar los cambios en el entorno ambiental cerca de un dispositivo.

La plataforma de Android admite tres amplias categorías de sensores:

* Sensores de movimiento. Estos sensores miden las fuerzas de aceleración y las fuerzas de rotación en tres ejes. Esta categoría incluye acelerómetros, sensores de gravedad, giroscopios y sensores del vector de rotación.
* Sensores ambientales. Estos sensores miden varios parámetros ambientales, como la temperatura y la presión del aire ambiental, la iluminación y la humedad. Esta categoría incluye barómetros, fotómetros y termómetros.
* Sensores de posición. Estos sensores miden la posición física de un dispositivo. Esta categoría incluye sensores de orientación y magnetómetros.

El marco de trabajo del sensor de Android permite acceder a muchos tipos de sensores. Algunos de estos sensores se basan en hardware y otros en software. Los sensores basados en hardware son componentes físicos integrados en un dispositivo de mano o tablet. Para obtener los datos, miden directamente propiedades ambientales específicas, como la aceleración, la intensidad del campo geomagnético o el cambio angular. Los sensores basados en software no son dispositivos físicos, aunque imitan los sensores basados en hardware. Los sensores basados en software derivan sus datos de uno o más de los sensores basados en hardware y, a veces, se denominan sensores virtuales o sensores sintéticos.

SensorManager

Esta clase se usa para crear una instancia del servicio del sensor. Proporciona varios métodos para acceder a sensores y escucharlos, registrar y cancelar el registro de objetos de escucha de sensores de eventos y adquirir información de orientación. También proporciona varias constantes del sensor que se usan para informar la exactitud del sensor, definir las velocidades de adquisición de datos y calibrar sensores.

Sensor

Esta clase se emplea para crear una instancia de un sensor específico. Proporciona varios métodos que permiten determinar las capacidades de un sensor.

SensorEvent

El sistema usa esta clase para crear un objeto de evento de sensor, que proporciona información sobre un evento del sensor. Un objeto de evento de sensor incluye la siguiente información: los datos sin procesar del sensor, el tipo de sensor que generó el evento, la exactitud de los datos y la marca de tiempo del evento.

SensorEventListener

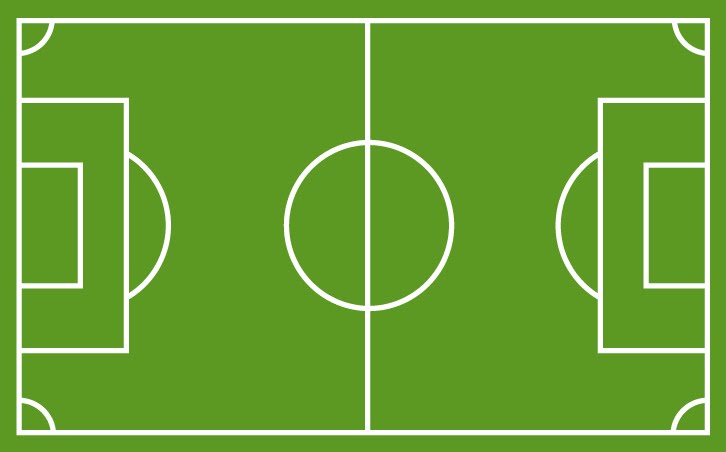
Esta interfaz es usada para crear dos métodos de devolución de llamada que reciben notificaciones (eventos del sensor) cuando cambian los valores del sensor o cuando cambia la exactitud del sensor.

**Desarrollo**

Durante el desarrollo de esta práctica se crearán 4 clases: MainActivity, Juego, Bola, y Puntos; se diseñará una vista bajo el nombre activity\_main.

El objetivo de esta aplicación, es que el balón se introduzca en las porterías a través de la línea amarilla, y no las líneas rojas (las cuales deben estar bloqueadas como si se tratase de un obstáculo), sin embargo, el balón puede salir del área de portería por donde sea.





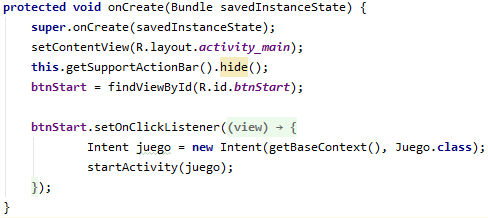
Activity\_main.xml

Esta vista se conforma de un título, un botón de inicio de juego, y una imagen animada (GIF).

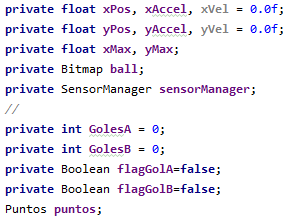
El botón lleva al usuario a la actividad Juego, donde está la cancha y el balón.

MainActivity.java

Esta es una clase muy sencilla, que esconde la barra de acciones, asigna el botón del activity\_main.xml a un objeto btnStart, el cual tiene un evento click que crea un Intent de la clase Juego, y la inicia.



Juego.java

Tiene algunas variables globales, como: xPos representa la posición en eje X, xAccel representa la aceleración del eje X, yPos es la posición en eje Y, yAccel es la aceleración en el eje Y. Por otro lado, xMax es el valor máximo que puede tomar xPos en el eje de las X, mientras que yMax es el máximo en el eje Y.

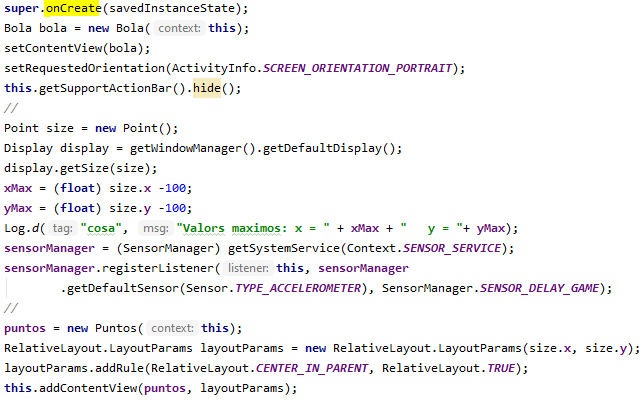
Se tiene un Bitmap llamado ball que será la imagen del balón rodante.

Un objeto SensorManager para acceder y manipular sensores.

Para la funcionalidad del juego, se tienen dos variables enteras GolesA y GolesB que representan los goles que se han metido en cada portería, siendo la A la de la izquierda y B la de la derecha. También hay dos variables booleanas flagGolA y flagGolB que señalan si se ha metido un gol en alguna de las porterías.

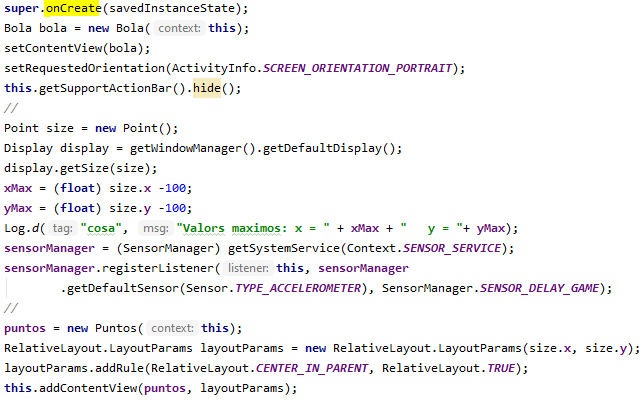
Y una instancia de la clase Puntos.

El método onCreate de la clase se conforma con una vista (View) de la clase Bola, ya que esta es una vista. Tiene una orientación portrait equivalente al dispositivo vertical. Y de igual forma esconde la barra de acciones.

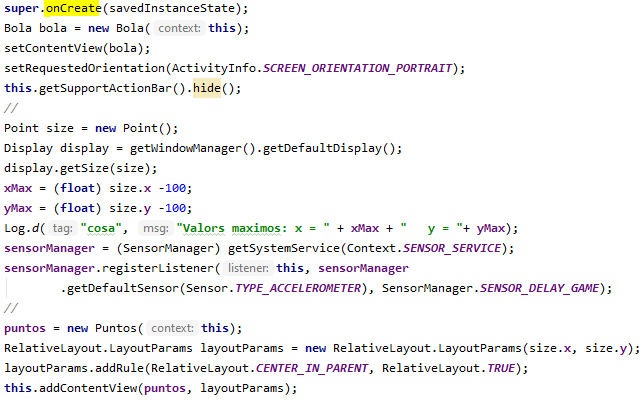


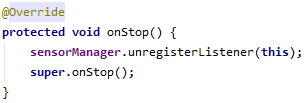
Después, Se crea una instancia de la clase Point, la cual se llama size. Y una instancia de la clase Display que obtendrá información de la ventana del dispositivo, del área donde se despliega la aplicación, el objeto tiene un método para obtener el tamaño de size, tanto en eje x como en eje y, estos valores se asignan a xMax e yMax restándole a ambos 100, esto evita que la imagen del balón salga de los límites, ya que tiene 100px de alto y ancho.

Se instancia la clase SensorManager, de acuerdo a los servicios del dispositivo, y se registra el sensor Acelerometro.

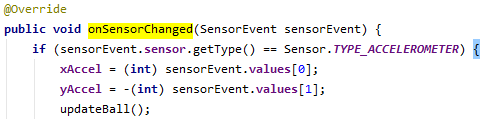


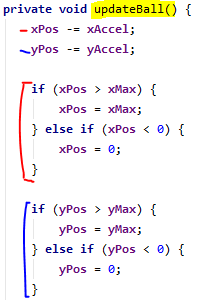
Finalmente, se termina de instanciar la clase Puntos. Se crea una variable de parámetro de layout donde se especifican las dimensiones del dispositivo como las dimensiones de los parámetros, y se agrega como regla que esté centrado. A la clase actual se le añade otro View de contenido que será la clase (o vista) puntos con los parámetros especificados.



En el método onStop de la clase se quita el sensor de los registros.

También debe sobre escribirse el método onSensorChanged() que se dispara cuando cambia el estado de alguno de los sensores del dispositivo. Dentro del evento, se cuestiona si el sensor que ha cambiado es el Acelerometro, el cual es el que nos interesa y el que está registrado. En caso de ser así, se asignan los valores que contenga el argumento, el primero corresponde a la aceleración en X, y el segundo a la aceleración en Y, estos se asignan a sus variables respectivas. Y se manda llamar al método updateBall() que se encarga de asignar correctamente las posiciones.



El método updateBall() se conforma de varias líneas de código, comenzando por la asignación de los valores de xPos y yPos.

Se marca en rojo todo lo referente al eje X y en azul lo del eje Y.

Primeramente se asignan las aceleraciones como una adición negativa a las variables. Se cuestiona si la posición en mayor al máximo valor posible en su eje, y de ser así, tomará dicho valor.

Por el contrario, si el valor del eje es menor a 0, se asigna 0 como valor a la variable en cuestión.

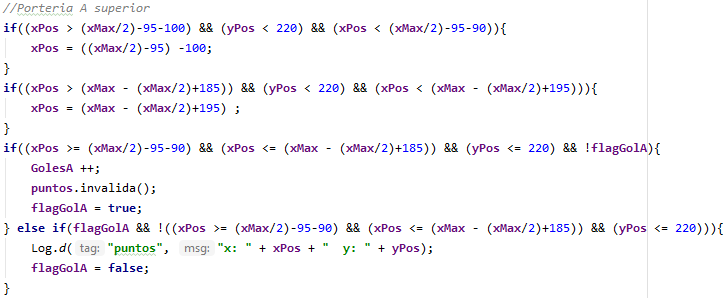
De este modo se evita que la imagen del balón exceda los límites de la pantalla y desaparezca o quede incompleta.

Hay que recordar que es necesario que la imagen del balón ingrese a las porterías para contar un Gol, pero que esto debe ser por la entrada de las porterías como se conocen normalmente. Por lo que es preciso bloquear los otros límites de las porterías para que el balón no acceda por lado izquierdo o derecho de alguna.

Para hacerlo se consideran por separado las dos porterías, primeramente la portería superior si se ve el teléfono de forma vertical. Si la posición en x es mayor a la mitad del eje X menos 195 pixeles, que es lo equivalente al tamaño de media portería y el balón, y además la posición en y es menor a 220 pixeles (que equivale al punto más bajo del poste de la portería), y también, la posición en x es menor a la mitad de la portería menos 185px (a fin de darle una especie de ancho al poste imaginario). De ser así, se asigna la posición en x a lo correspondiente al poste izquierdo de la portería.

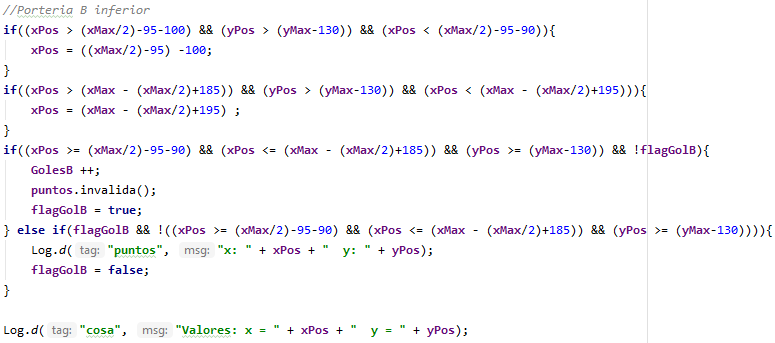
De igual forma se calcula el poste derecho de la portería A, cambiando los valores de la distancia de la mitad de la portería ahora hacia la derecha (se suman pixeles).

Si la posición en X está entre los valores de los postes de la portería y es menor a 220px indica que se ha metido un Gol, pero mientras el balón se mueva dentro de la portería se cumplen con los criterios, así que se debe limitar también con una bandera que señala si ya se marcó o no el gol. Cuando se hace la suma del gol se manda llamar al método onDraw de la vista puntos a través del método invalida. Si el criterio no se cumple se apaga la bandera, esto es, cuando el balón sale de la portería.



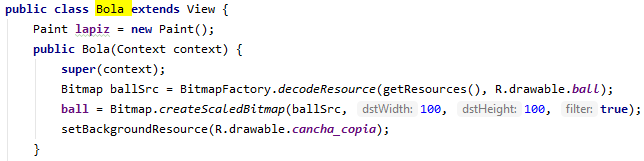
Lo mismo ocurre con la portería inferior de la pantalla o portería B, con el primer bloque if se define el poste izquierdo. Con el segundo se detallan los criterios del poste derecho. Y con el tercero los criterios para contar un gol, marcarlo y desmarcarlo.

Y con eso finaliza la implantación del método updateBall().



Clase Bola

La clase extiende o hereda de View, lo que lo convierte en un componente o una vista. Tiene un objeto Paint llamado lápiz que ayuda a dibujar sobre el Canva. El constructor de la clase tiene un objeto Bitmap que viene de los recursos, y es la imagen de la pelota de futbol. Esta se crea con las dimensiones de 100px por 100px. Y también se asigna como fondo del componente la imagen de la cancha de futbol.



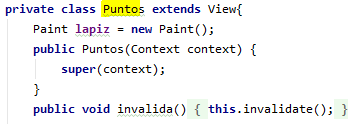
La primera línea (azul) del método onDraw corresponde al posicionamiento de la imagen de la pelota sobre la vista, esto será donde xPos y yPos señalen, estas variables son globales y se modifican en la clase, hay que recordar que la clase Bola está dentro de la clase Juego.

Las siguientes líneas (rojo) son solo para calcular las dimensiones de las porterías, es decir, el posicionamiento de los postes y la entrada a la misma.



Clase Puntos

La clase Puntos también extiende de View y contiene un objeto Paint, para dibujar sobre el Canvas. Contiene un método llamado invalida() que permite a otras clases solicitar que se redibuje la vista.

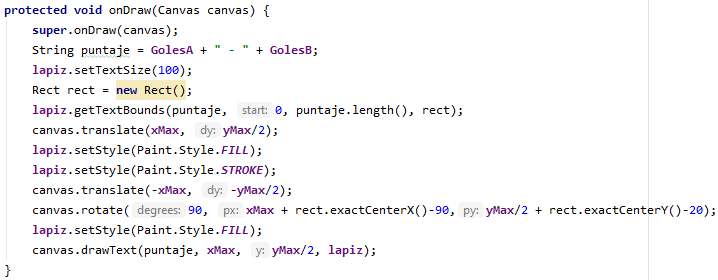


El objetivo de esta clase es dibujar el marcador el juego. Para ello, se crea una cadena llamada puntaje, con los valores de la cantidad de Goles registrados en la portería A y los que hay en la portería B, separados por un guion medio.

Se configuran las características del lápiz, como el tamaño del texto, estilo y grosor.

Y se crea un objeto Rect que forma un rectángulo. Se obtienen los bordes de dicho objeto acorde al texto.

Se traslada el Canvas a la posición donde se desea poner el puntaje, esto es, a la mitad del eje Y, y en el lado extremo máximo de X. De acuerdo a esto, el texto queda vertical de acuerdo a la pantalla, pero se desea que este esté horizontal, por lo que se traslada a los valores mencionados en negativo, y se rota 90° el Canvas, para finalmente dibujar el puntaje que define el texto, en las posiciones que se ha mencionado anteriormente.



Eso es todo en cuanto a la codificación.

**Resultados**

A continuación se muestran capturas del funcionamiento de la aplicación. La primera imagen representa la pantalla inicial con el GIF en movimiento, y la segunda la cancha una vez que el juego ha comenzado.

|  |  |
| --- | --- |
| C:\Users\Lenovo T540P\OneDrive\Documentos\ITSUR\VIII Semestre\MOVIL II - 2\U4\Imagenes\1.jpg | C:\Users\Lenovo T540P\OneDrive\Documentos\ITSUR\VIII Semestre\MOVIL II - 2\U4\Imagenes\2.jpg |

Las figuras 3 y 4 muestran la validación de no exceder los bordes izquierdo y derecho de la portería B, resaltando que se forzó el dispositivo a hacerlo. La figura 5 muestra lo equivalente a un gol dentro de la portería B, y como este se suma al marcador.

|  |  |  |
| --- | --- | --- |
| C:\Users\Lenovo T540P\OneDrive\Documentos\ITSUR\VIII Semestre\MOVIL II - 2\U4\Imagenes\6.jpg | C:\Users\Lenovo T540P\OneDrive\Documentos\ITSUR\VIII Semestre\MOVIL II - 2\U4\Imagenes\7.jpg | C:\Users\Lenovo T540P\OneDrive\Documentos\ITSUR\VIII Semestre\MOVIL II - 2\U4\Imagenes\8.jpg |

Las Figuras 6 y 7 muestran el funcionamiento de los límites de la portería A, izquierda y derecha respectivamente, cabe destacar que se estaba forzando el sensor para que pasara los límites. La Figura 8 muestra un gol, es decir, el ingreso del balón a la portería por la entrada de la misma.

|  |  |  |
| --- | --- | --- |
| C:\Users\Lenovo T540P\OneDrive\Documentos\ITSUR\VIII Semestre\MOVIL II - 2\U4\Imagenes\3.jpg | C:\Users\Lenovo T540P\OneDrive\Documentos\ITSUR\VIII Semestre\MOVIL II - 2\U4\Imagenes\4.jpg | C:\Users\Lenovo T540P\OneDrive\Documentos\ITSUR\VIII Semestre\MOVIL II - 2\U4\Imagenes\5.jpg |

**Bibliografía**

Google developers. (27 de 12 de 2019). Android developer. Obtenido de https://developer.android.com/guide/topics/sensors/sensors\_overview?hl=es-419

Google developers. (13 de 03 de 2021). Android developer. Obtenido de https://developer.android.com/guide/topics/sensors/sensors\_motion