CAPÍTULO 4.
Entradas en Android: teclado, pantalla táctil y sensores

4.1. **Manejando eventos de usuario**

4.1.1. Escuchador de eventos

```
protected void onCreate(Bundle savedValues) {
    ...
   Button boton = (Button)findViewById(R.id.boton);
   boton.setOnClickListener( new OnClickListener() {
      public void onClick(View v) {
         // Acciones a realizar
      }
   ...
}
```

```
public class Ejemplo extends Activity implements
OnClickListener {
    protected void onCreate(Bundle savedValues) {
        ...
       Button boton = (Button)findViewById(R.id.boton);
       boton.setOnClickListener(this);
    }

    public void onClick(View v) {
     // Acciones a realizar
    }
    ...
}
```

4.1.2. Manejadores de eventos

4.2. El teclado

```
@Override
public boolean onKeyDown(int codigoTecla, KeyEvent evento) {
    super.onKeyDown(codigoTecla, evento);
     // Suponemos que vamos a procesar la pulsación
    boolean procesada = true;
    switch (codigoTecla) {
    case KeyEvent.KEYCODE_DPAD_UP:
        aceleracionNave = +PASO_ACELERACION_NAVE;
        break;
    case KeyEvent.KEYCODE_DPAD_LEFT:
        giroNave = -PASO_GIRO_NAVE;
        break;
    case KeyEvent.KEYCODE_DPAD_RIGHT:
        giroNave = +PASO_GIRO_NAVE;
        break;
    case KeyEvent.KEYCODE_DPAD_CENTER:
    case KeyEvent.KEYCODE_ENTER:
        ActivaMisil();
        break;
    default:
        // Si estamos aquí, no hay pulsación que nos interese
        procesada = false;
        break;
    }
    return procesada;
}
```

```
@Override
   public boolean onKeyUp(int codigoTecla, KeyEvent evento) {
      super.onKeyUp(codigoTecla, evento);
      // Suponemos que vamos a procesar la pulsación
      boolean procesada = true;
      switch (codigoTecla) {
      case KeyEvent.KEYCODE_DPAD_UP:
         aceleracionNave = 0;
         break;
      case KeyEvent.KEYCODE_DPAD_LEFT:
      case KeyEvent.KEYCODE_DPAD_RIGHT:
         giroNave = 0;
         break;
      default:
         // Si estamos aquí, no hay pulsación que nos interese
         procesada = false;
         break;
      }
      return procesada;
   }
```

4.3. **La pantalla táctil**

```
   private float mX=0, mY=0;
   private boolean disparo=false;

   @Override
   public boolean onTouchEvent (MotionEvent event) {
        super.onTouchEvent(event);
        float x = event.getX();
        float y = event.getY();
        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            disparo=true;
            break;
        case MotionEvent.ACTION_MOVE:
            float dx = Math.abs(x - mX);
            float dy = Math.abs(y - mY);
            if (dy<6 && dx>6){
                giroNave = Math.round((x - mX) / 2);
                disparo = false;
            } else if (dx<6 && dy>6){
```

```
                aceleracionNave = Math.round((mY - y) / 25);
                disparo = false;
            }
            break;
        case MotionEvent.ACTION_UP:
            giroNave = 0;
            aceleracionNave = 0;
            if (disparo){
                ActivaMisil();
            }
            break;
        }
        mX=x; mY=y;
        return true;
    }
```

4.4. *Gestures*

4.4.1. Creación de una librería de gestures

4.4.2. Añadiendo la librería de gestures a nuestra aplicación

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center_horizontal"
    android:text="Introduce una gesture"
    android:textSize="8pt"
    android:layout_margin="10dip"/>
  <TextView
    android:id="@+id/salida"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"/>
  <android.gesture.GestureOverlayView
```

```
        android:id="@+id/gestures"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:gestureStrokeType="multiple"
        android:fadeOffset="800"/>
    </LinearLayout>
```

```
public class Gestures extends Activity implements
OnGesturePerformedListener {
    private GestureLibrary libreria;
    private TextView osalida;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        libreria = GestureLibraries.fromRawResource(this,
                                            R.raw.gestures);
        if (!libreria.load()) {
          finish();
        }
        GestureOverlayView gesturesView = (GestureOverlayView)
                                  findViewById(R.id.gestures);
        gesturesView.addOnGesturePerformedListener(this);
        salida = (TextView) findViewById(R.id.salida);
    }

    public void onGesturePerformed(GestureOverlayView ov,
                                        Gesture gesture) {
        ArrayList<Prediction> predictions =
                                  libreria.recognize(gesture);
        salida.setText("");
        for (Prediction prediction : predictions){
          salida.append(prediction.name+" " +
                                        prediction.score+"\n");
         }
    }
}
```

4.4.3. Añadiendo *gestures* a Asteroides

3) Añade al principio de `res/layout/main.xml` el siguiente código. Cierra la etiqueta al final del fichero.

```
<android.gesture.GestureOverlayView
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/gestures"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:gestureStrokeType="multiple"
   android:fadeOffset="800">
```

4) En el fichero `Asteroides.java` añade en la definición de la clase:

```
public class Asteroides extends Activity
implements OnGesturePerformedListener{
    private GestureLibrary libreria;
    …
```

5) Añade al final del método onCreate:

```
libreria = GestureLibraries.fromRawResource(this,
                                                R.raw.gestures);
if (!libreria.load()) {
   finish();
}
GestureOverlayView gesturesView =
   (GestureOverlayView) findViewById(R.id.gestures);
gesturesView.addOnGesturePerformedListener(this);
```

6) Añade el siguiente método:

```
public void onGesturePerformed(GestureOverlayView ov,
                                            Gesture gesture) {
   ArrayList<Prediction> predictions=libreria.recognize(gesture);
   if (predictions.size()>0){
      String comando = predictions.get(0).name;
      if (comando.equals("play")){
         lanzarJuego();
      } else if (comando.equals("configurar")){
         lanzarPreferencias();
      } else if (comando.equals("acerca_de")){
         lanzarAcercaDe();
      } else if (comando.equals("cancelar")){
         finish();
      }
   }
}
```

4.5. **Los sensores**

4.5.1. Listar los sensores del dispositivo

```
public class Sensores extends Activity {
    private TextView salida;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        salida = (TextView) findViewById(R.id.salida);
        SensorManager mSensorManager = (SensorManager)
                            getSystemService(SENSOR_SERVICE);
        List<Sensor> listSensors = mSensorManager.
                            getSensorList(Sensor.TYPE_ALL);
        for(Sensor sensor: listSensors) {
             log(sensor.getName());
        }
    }


      private void log(String string) {
          salida.append(string + "\n");
      }
  }
```

Añade la siguiente propiedad al *TextView* de res/layout/main.xml:

```
android:id="@+id/salida"
```

4.5.2. Acceso a los datos del sensor

```
listSensors = mSensorManager.getSensorList(Sensor.TYPE_ ORIENTATION);
if (!listSensors.isEmpty()) {
    Sensor orientationSensor = listSensors.get(0);
    mSensorManager.registerListener(this, orientationSensor,
                                    SensorManager.SENSOR_DELAY_UI);}
listSensors = mSensorManager.getSensorList(Sensor.TYPE_ACCELEROMETER);
if (!listSensors.isEmpty()) {
    Sensor acelerometerSensor = listSensors.get(0);
    mSensorManager.registerListener(this, acelerometerSensor,
                                    SensorManager.SENSOR_DELAY_UI);}
listSensors = mSensorManager.getSensorList(Sensor.TYPE_MAGNETIC_FIELD);
if (!listSensors.isEmpty()) {
    Sensor magneticSensor = listSensors.get(0);
    mSensorManager.registerListener(this, magneticSensor,
                                    SensorManager.SENSOR_DELAY_UI);}
listSensors = mSensorManager.getSensorList(Sensor.TYPE_TEMPERATURE);
if (!listSensors.isEmpty()) {
    Sensor temperatureSensor = listSensors.get(0);
    mSensorManager.registerListener(this, temperatureSensor,
                                    SensorManager.SENSOR_DELAY_UI);}
```

4.5.2.1. Obtención de datos de los sensores

```
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
}

@Override
public void onSensorChanged(SensorEvent event) {
   // Cada sensor puede provocar que un thread pase por aquí
   // así que sincronizamos el acceso
   synchronized (this) {
      switch(event.sensor.getType()) {
      case Sensor.TYPE_ORIENTATION:
         for (int i=0 ; i<3 ; i++) {
            log("Orientación "+i+": "+event.values[i]);
         }
         break;
      case Sensor.TYPE_ACCELEROMETER:
         for (int i=0 ; i<3 ; i++) {
            log("Acelerómetro "+i+": "+event.values[i]);
         }
```

```
            break;
        case Sensor.TYPE_MAGNETIC_FIELD:
            for (int i=0 ; i<3 ; i++) {
                log("Magnetismo "+i+": "+event.values[i]);
            }
            break;
        default:
            for (int i=0 ; i<event.values.length ; i++) {
                log("Temperatura "+i+": "+event.values[i]);
            }
        }
    }
}


public class Sensores extends Activity implements SensorEventListener {
    private LinearLayout mLinearLayout;
    private TextView aTextView[][] = new TextView[20][3];;
    private SensorManager mSensorManager;
    private Sensor aSensor[] = new Sensor[20];
    private List<Sensor> listSensors;

@Override public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mLinearLayout = (LinearLayout) findViewById(R.id.LinearLayout01);
    mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
    listSensors = mSensorManager.getSensorList(Sensor.TYPE_ALL);
    int n = 0;
    for (Sensor sensor : listSensors) {
        aSensor[n] = sensor;
        TextView mTextView = new TextView(this);
        mTextView.setText(sensor.getName());
        mLinearLayout.addView(mTextView);
        LinearLayout nLinearLayout = new LinearLayout(this);
        mLinearLayout.addView(nLinearLayout);
        for (int i = 0; i < 3; i++) {
            aTextView[n][i] = new TextView(this);
            aTextView[n][i].setText("?");
            aTextView[n][i].setWidth(87);
        }
        TextView xTextView = new TextView(this);
        xTextView.setText("  X: ");
        nLinearLayout.addView(xTextView);
        nLinearLayout.addView(aTextView[n][0]);
        TextView yTextView = new TextView(this);
```

```
        yTextView.setText("  Y: ");
        nLinearLayout.addView(yTextView);
        nLinearLayout.addView(aTextView[n][1]);
        TextView zTextView = new TextView(this);
        zTextView.setText("  Z: ");
        nLinearLayout.addView(zTextView);
        nLinearLayout.addView(aTextView[n][2]);
        mSensorManager.registerListener(this, sensor,
                                        SensorManager.SENSOR_DELAY_UI);
        n++;
    }
}

@Override public void onAccuracyChanged(Sensor sensor, int accuracy) {}

@Override public void onSensorChanged(SensorEvent event) {
    synchronized (this) {
        int n = 0;
        for (Sensor sensor: listSensors) {
            if (event.sensor == sensor) {
                for (int i=0; i<event.values.length; i++) {
                  aTextView[n][i].setText(Float.toString(event.values[i]));
                }
            }
            n++;
        }
    }
}}
```

4.5.3. Utilización de los sensores en Asteroides

```
SensorManager mSensorManager = (SensorManager)
context.getSystemService(context.SENSOR_SERVICE);
List<Sensor> listSensors = mSensorManager.getSensorList(
                                        Sensor.TYPE_ORIENTATION);
if (!listSensors.isEmpty()) {
    Sensor orientationSensor = listSensors.get(0);
    mSensorManager.registerListener(this, orientationSensor,
                SensorManager.SENSOR_DELAY_UI);
}
```

```
public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    private boolean hayValorInicial = false;
    private float valorInicial;

    public void onSensorChanged(SensorEvent event) {
        float valor = event.values[1];
        if (!hayValorInicial){
            valorInicial = valor;
            hayValorInicial = true;
        }
        giroNave=(int) (valor-valorInicial)/3 ;
    }
```

4.6. **Introduciendo un misil en Asteroides**

Para poder disparar a los asteroides va ha ser necesario introducir un misil en el juego. En primer lugar añade las siguientes variables a la clase `VistaJuego`:

```
    // //// MISIL //////
    private Grafico misil;
    private static int PASO_VELOCIDAD_MISIL = 12;
    private boolean misilActivo = false;
    private int distanciaMisil;
```

Si queremos trabajar con gráficos vectoriales, puedes crear la variable `drawableMisil` de la siguiente forma, en el constructor:

```
    ShapeDrawable dMisil = new ShapeDrawable(new RectShape());
    dMisil.getPaint().setColor(Color.WHITE);
    dMisil.getPaint().setStyle(Style.STROKE);
    dMisil.setIntrinsicWidth(15);
    dMisil.setIntrinsicHeight(3);
    drawableMisil = dMisil;
```

En el método `actualizaFisica()` añade las siguientes líneas:

```
    // Actualizamos posición de misil
    if (misilActivo) {
        misil.incrementaPos();
```

```
            distanciaMisil--;
            if (distanciaMisil < 0) {
                misilActivo = false;
            } else {
                for (int i = 0; i < Asteroides.size(); i++)
                if (misil.verificaColision(Asteroides.elementAt(i))) {
                    destruyeAsteroide(i);
                    break;
                }
            }
        }
```

Añade los siguientes dos métodos:

```
private void destruyeAsteroide(int i) {
    Asteroides.remove(i);
    misilActivo = false;
}

private void ActivaMisil() {
    misil.setPosX(nave.getPosX()+ nave.getAncho()/2-misil.getAncho()/2);
    misil.setPosY(nave.getPosY()+ nave.getAlto()/2 - misil.getAlto()/2);
    misil.setAngulo(nave.getAngulo());
    misil.setIncX(Math.cos(Math.toRadians(misil.getAngulo()))
                                                * PASO_VELOCIDAD_MISIL);
    misil.setIncY(Math.sin(Math.toRadians(misil.getAngulo()))
                                                * PASO_VELOCIDAD_MISIL);
    distanciaMisil = (int) Math.min(
                this.getWidth()  / Math.abs(misil.getIncX()),
                this.getHeight() / Math.abs(misil.getIncY())) - 2;
    misilActivo = true;
}
```