How to receive input from the double tap sensor in the context of a game with Java ME

This article describes how to receive input from the newly introduced double tap sensor on a Series 40, full touch device, with Java Runtime 2.0.0.

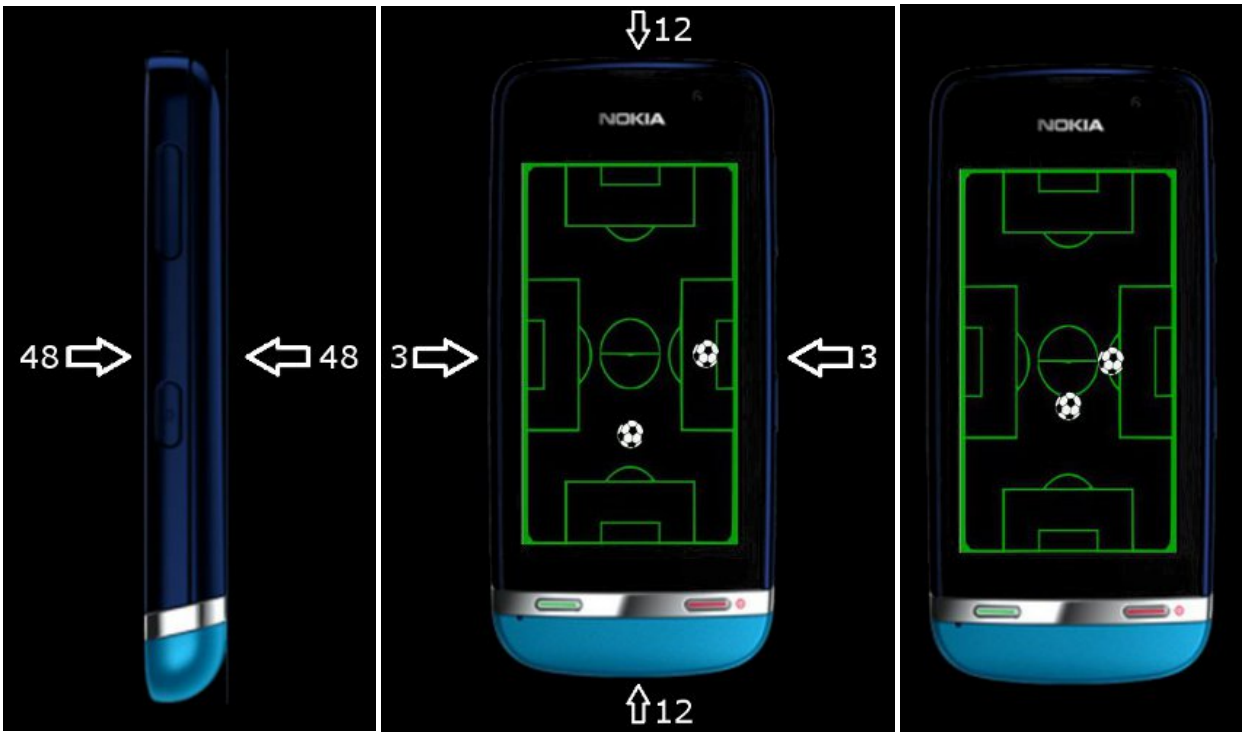
Introduction

The double tap sensor provides a new interaction channel between the end user and the device, that differs from the standard input methods. The example in this article, uses exclusively double tapping as input, and demonstrates how a developer does not always need to rely on touch events, virtual keyboard input or gestures in order to interact with the end user. What differentiates the double tap sensor from the other input methods is that, it can receive events from parts of the device other than the screen, such as the left, right, top and bottom sides of the phone as well as the back side.

This example uses a custom soccer field with two moving soccer balls, one along the vertical and one on the horizontal axis. The purpose of the game is to keep changing the direction of each ball, by double tapping one of the four sides of the device, so that the balls do not reach the goal posts, located on the left, right, bottom and top side of the screen. The following logic is in place:

- If one of the balls reaches one of the goal posts, this causes a game over
- Game over returns back to the start screen
- In order to start a new game from the start screen, one needs to double tap either the screen or the back side (cover side) of the device
- A new game starts with both soccer balls located at the center of the soccer field. Their original direction is from top to bottom and from left to right.
- The direction of the ball can change by double tapping only after it has reached or crossed over to the half that brings the ball closer to that half's goal post.
- The balls change color depending on the half of the screen they move along. Movement in the top and left halves causes the ball to be painted red. The color changes to standard white for movement in the bottom and right halves.

Note: Double tap sensitivity differs from the level of sensitivity present in resistive or capacitive screens. More physical effort is typically required to generate a double tap event



Values returned by the double tap sensor

These are the possible values returned by the double tap sensor:

Return INT Value	Corresponding Event
12	A double tap received from the top or bottom side of the device

(C) Copyright Nokia 2014. All rights reserved.

3	A double tap received from the left or right side of the device
48	Either the screen or the back side of the device were double tapped

Please note that, the application cannot distinguish between a left or right side double tap. The same applies for top or bottom as well as front or back side double taps.

The sensor connection and the data listener

A sensor connection is established by using the sensor manager to query for the "double_tap" sensor and by opening the URL returned by the first item in the search results. A DataListener is attached to the sensor connection in order to listen asynchronously for incoming double tap events:

```
//query the double_tap sensor
SensorInfo infos[] = SensorManager.findSensors("double_tap", null);
//If the double tap sensor is found
if (infos.length > 0) {
    //open a connection and attach a data listener to the sensor
    try {
        sensor = (SensorConnection) Connector.open(infos[0].getUrl());
        sensor.setDataListener(this, 1);
        }
    catch (Exception e) {
         //error
     }
   }
```

The `dataReceived()` method, gives asynchronous call backs that contain a single `TYPE_INT` data value from the double tap sensor. After the value is extracted it is sent over to the Game's main two screens:

```
    //this receives sensor events via the data listener
    public void dataReceived(SensorConnection sensor, Data[] data, boolean isDataLost) {
        int[] intValues = data[0].getIntValues();
        int tapValue = intValues[intValues.length - 1];
        //this sends the received sensor events to the start screen
        infoCanvas.receivedDoubleTap(tapValue);
        //this sends the received sensor events to the main screen
        soccerCanvas.receivedDoubleTap(tapValue);
    }
```

The start screen

The start screen is a simple canvas instance, with a green background color, displaying the message "Double tap!". The message is painted in the middle of the screen. The entire canvas is used in full screen mode for displaying the start screen:

```
public void paint(Graphics g) {
        int width = getWidth();
        int height = getHeight();
        //Green background
        g.setColor(0, 160, 0);
        g.fillRect(0, 0, width, height);
        //White foreground text
        g.setColor(255, 255, 255);
        //display the message in the middle of the sceen
        int x = width / 2;
        int y = height / 2;
        g.drawString(message, x, y, Graphics.BASELINE | Graphics.HCENTER);
```


(C) Copyright Nokia 2014. All rights reserved.

```
    }
```

When the end user double taps the screen (or the back side of the device), the sensor sends the value 48 to the start screen's canvas instance. The application uses this value only if the start screen is shown (i.e. is current) in order to launch the Game's main screen. This control is necessary because double tapping accidentally the screen when the Game's main screen is current, would result in resetting the soccer balls' position, which is the first action that takes place when the main screen is loaded.

```
    //If double tap is received on the front or back of the device
    public void receivedDoubleTap(int tapValue) {
        //switch to main screen only if the user, current sees the start screen
        if(tapValue == 48 && this.isShown()) {
            midlet.displayField();
        }
    }
```

The main screen

The application uses variables that keep track of the X and Y coordinates for the movements on the horizontal and vertical axes. The point defined by these set of coordinates corresponds to the center of the ball:

```
        //places the balls in the middle of the field
        horizBallX = (screenWidth - whiteBall.getWidth()) / 2;
        horizBallY = (screenHeight - whiteBall.getHeight()) / 2;

        verBallX = (screenWidth - whiteBall.getWidth()) / 2;
        verBallY = (screenHeight - whiteBall.getHeight()) / 2;
```

Boolean variables are used in order to keep track of the direction of each ball as well as handling the termination of the thread inside which the animation/movement takes place:

```
        //initially the vertical direction is down and the horizontal one is right
        goingLeft = false;
        goingUp = false;
        exitThread = false;
```

Inside the animation thread, we prevent the screen's light from turning off by calling the setLights method. Please note, that the outcome from calling this method, depends also on whether the screen's lights are set to be controlled by the application or the system profile, from the device's menu settings. The thread simply increases or decreases the X and Y coordinates of each soccer ball depending on the ball's direction:

```
public void run() {
        while(!exitThread) {
            try {
                //In order for the lights not to go off, the user needs to set
                //the lights to "app defined"
                DeviceControl.setLights(0, 100);
                //move the balls
                if(goingLeft) {
                    horizBallX = horizBallX - step;
                }
                else {
                    horizBallX = horizBallX + step;
                }

                if(goingUp) {
                    verBallY = verBallY - step;
                }
```


(C) Copyright Nokia 2014. All rights reserved.

```
                else {
                    verBallY = verBallY + step;
                }
```

The barrier, beyond which movement means that the Game is Over, is calculated from the screen's and resource images' resolution when the main screen is instantiated. Inside the animation thread, a control is performed, that checks whether the user has failed to redirect the ball away from the barrier. The display is then returned back to the start screen and the thread terminates:

```
                //initiate the barrier coordinates based on this device's screen size
                 rightLimit = screenWidth - (whiteBall.getWidth() / 2);
                 leftLimit = 0 - (whiteBall.getWidth() / 2);
                 topLimit = 0 - (whiteBall.getHeight() / 2);
                 btmLimit = screenHeight - (whiteBall.getHeight() / 2);

                ...

               //if any of the balls have reached a barrier, it is Game over!
                if( horizBallX - (whiteBall.getWidth() / 2) <= leftLimit ||
                    horizBallX + (whiteBall.getWidth() / 2) >= rightLimit ||
                    verBallY - (whiteBall.getHeight() / 2) <= topLimit ||
                    verBallY + (whiteBall.getHeight() / 2) >= btmLimit) {
                    //return back to start screen
                    midlet.displayInfo();
                    //allow this thread to exit for performance improvement reasons
                    exitThread = true;
```

Double tapping from the Game's main screen, changes the direction of each ball

```
public void receivedDoubleTap(int tapValue) {
        /*
         * A left/right double tap when the ball in the horizontal direction is located
in the right half
         * results in changing the direction from right to left
         */
        if (( (horizBallX + (whiteBall.getWidth() / 2)) > centerX) && (tapValue == 3)) {
            goingLeft = true;
        }

        /*
         * A left/right double tap when the ball in the horizontal direction is located
in the left half
         * results in changing the direction from left to right
         */
        if (( (horizBallX + (whiteBall.getWidth() / 2)) <= centerX) && (tapValue == 3))
{
            goingLeft = false;
        }

        /*
         * A top/btm double tap when the ball in the vertical direction is located in
the bottom half
         * results in changing the direction from down to up
         */
        if( ( (verBallY + (whiteBall.getHeight() / 2)) > centerY) && (tapValue == 12)) {
            goingUp = true;
        }

        /*
```


(C) Copyright Nokia 2014. All rights reserved.

```
         * A top/btm double tap when the ball in the vertical direction is located in
the upper half
         * results in changing the direction from up to down
         */
        if( ( (verBallY + (whiteBall.getHeight() / 2)) <= centerY) && (tapValue == 12))
{
            goingUp = false;
        }
    }
```

Resources

The source code of this MIDlet is available for download from here: File:DoubleTapSoccerSource.zip

The binary files of this MIDlet are available for download from here: File:DoubleTapSoccerBinaries.zip

See also

- The Developer's guide - Channel Info for Sensors

(C) Copyright Nokia 2014. All rights reserved.