Collision detection in Java ME

Overview

This code snippet demonstrates how to implement the logic for basic collision detection which is useful, for example, in simple games. In the snippet five blocks are drawn onto the screen and sent moving (using the methods `MyCanvas.startBouncing()` and `MyCanvas.run()`). Collisions against walls and other blocks are detected (using method `MyCanvas.paint()`).

Another possibility to implement collision detection is by using the MIDP 2.0 Game API, and specifically the `javax.microedition.lcdui.game.Sprite.collidesWith()` method.

Source file: Block.java

```
import javax.microedition.lcdui.Graphics;

/**
 * Represents a bouncing block.
 */
public class Block {
    public static final int WIDTH = 10;
    public static final int HEIGHT = 10;
    public static final int MAX_SPEED = 3;

    private int[] color = new int[3];     // Three RGB components
    private int[] position = new int[2];  // X and Y components
    private int[] speed = new int[2];     // X and Y components

    public Block(int colorR, int colorG, int colorB,
            int posX, int posY,
            int speedX, int speedY) {
        color[0] = colorR;
        color[1] = colorG;
        color[2] = colorB;
        position[0] = posX;
        position[1] = posY;
        speed[0] = speedX;
        speed[1] = speedY;
    }

    public int[] getPosition() {
        return position;
    }

    /**
     * Bounces the block in X direction.
     */
    public void bounceX() {
        speed[0] = -speed[0];
    }

    /**
     * Bounces the block in Y direction.
     */
    public void bounceY() {
        speed[1] = -speed[1];
    }

    /**
     * Updates the block position.
     */
    public void move() {
        position[0] = position[0] + speed[0];
        position[1] = position[1] + speed[1];
    }

    /**
     * Determines whether or not this Block and the specified Block
     * intersect.
     * @param b the specified Block
     * @return true if the specified Block and this Block intersect;
     *   false otherwise.
     */
    public boolean intersects(Block b) {
        int[] tPos = this.getPosition();
        int tw = tPos[0] + WIDTH;
        int th = tPos[1] + HEIGHT;
        int[] bPos = b.getPosition();
        int bw = bPos[0] + WIDTH;
        int bh = bPos[1] + HEIGHT;
    return (bw > tPos[0] && bh > tPos[1] && tw > bPos[0] && th > bPos[1]);
    }

    /**
     * Draws the block using the specified graphics context.
     */
    public void draw(Graphics g) {
        g.setColor(color[0], color[1], color[2]);
        g.fillRect(position[0], position[1], WIDTH, HEIGHT);
    }
}
```

Source file: MyCanvas.java

```
import java.util.Random;
```


(C) Copyright Nokia 2013. All rights reserved.

```
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Graphics;

/**
 * Represents a drawing canvas. Also contains the "bouncer thread".
 */
public class MyCanvas extends Canvas implements Runnable {
    private final int NUMBER_OF_BLOCKS = 5;

    private Block[] blocks;
    private boolean running;

    public MyCanvas() {
        Random rnd = new Random();
        int width = getWidth();
        int height = getHeight();

        // Initialize the blocks (color, position and speed)
        blocks = new Block[NUMBER_OF_BLOCKS];
        for (int i = 0; i < NUMBER_OF_BLOCKS; i++) {
            int colorR = Math.abs(rnd.nextInt()) % 256;
            int colorG = Math.abs(rnd.nextInt()) % 256;
            int colorB = Math.abs(rnd.nextInt()) % 256;
            int posX = Math.abs(rnd.nextInt()) % width;
            int posY = Math.abs(rnd.nextInt()) % height;
            int speedX = rnd.nextInt() % (2 * Block.MAX_SPEED + 1) - Block.MAX_SPEED;
            int speedY = rnd.nextInt() % (2 * Block.MAX_SPEED + 1) - Block.MAX_SPEED;
            blocks[i] = new Block(colorR, colorG, colorB,
                posX, posY,
                speedX, speedY);
        }
    }

    /**
     * Starts the bouncer thread.
     */
    public void startBouncing() {
        if (running) {
            return;
        }
        // Let's play
        running = true;
        Thread bouncerThread = new Thread(this);
        bouncerThread.start();
    }

    /**
     * Stops the bouncer thread.
     */
    public void stopBouncing() {
        running = false;
    }

    /**
     * From Runnable. The main loop.
     */
    public void run() {
        while (running) {
            // Repaint the screen
            repaint();
            // Sleep for a while
            try {
                Thread.sleep(50);
            } catch (InterruptedException ex) {
                ex.printStackTrace();
            }
        }
    }

    /**
     * From Canvas. Does the painting.
     */
    protected void paint(Graphics g) {
        // Draw background
        int width = getWidth();
        int height = getHeight();
        g.setColor(255, 255, 255);
        g.fillRect(0, 0, width, height);

        // Update the block state and draw them
        for (int i = 0; i < NUMBER_OF_BLOCKS; i++) {
            Block block = blocks[i];

            // Detect collisions against walls
            int[] blockPos = block.getPosition();
            if (blockPos[0] + Block.WIDTH >= width || blockPos[0] < 0) {
                block.bounceX();
            }
            if (blockPos[1] + Block.HEIGHT >= height || blockPos[1] < 0) {
                block.bounceY();
            }

            // Move the block
            block.move();

            // Detect collisions against each other
            for (int j = 0; j < NUMBER_OF_BLOCKS; j++) {
                Block otherBlock = blocks[j];
                if (j != i && block.intersects(otherBlock)) {
                    block.bounceX();
                    block.bounceY();
                }
            }

            block.draw(g);
        }
    }
}
```

Source file: BouncyMIDlet.java

```
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
```


(C) Copyright Nokia 2013. All rights reserved.

```
import javax.microedition.midlet.MIDlet;

/**
 * The main MIDlet.
 */
public class BouncyMIDlet extends MIDlet implements CommandListener {
    private MyCanvas canvas;
    private Command exitCommand;

    /**
     * Constructor. Constructs the object and initializes displayables.
     */
    public BouncyMIDlet() {
        canvas = new MyCanvas();

        exitCommand = new Command("Exit", Command.EXIT, 0);
        canvas.addCommand(exitCommand);
        canvas.setCommandListener(this);
    }

    /**
     * From MIDlet.
     * Called when the MIDlet is started.
     */
    public void startApp() {
        Display.getDisplay(this).setCurrent(canvas);
        canvas.startBouncing();
    }

    // Other inherited methods omitted for brevity
    // ...

    /**
     * From CommandListener.
     * Called by the system to indicate that a command has been invoked on a
     * particular displayable.
     * @param command the command that was invoked
     * @param displayable the displayable where the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
        // Exit the MIDlet
        if (command == exitCommand) {
            canvas.stopBouncing();
            destroyApp(true);
            notifyDestroyed();
        }
    }
}
```

Postconditions

Five blocks are bouncing on the screen.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.