Book Series

of

jago{coding}

Learn & Share {Your Code}

www.jagocoding.com

Penggunaan Canvas Pada J2ME Untuk Game Programming

Oleh: Adnan w Anadrep

Pada kesempatan kali ini saya akan memberikan kamu tutorial mengenai Canvas pada J2ME yang biasanya sering digunakan di dalam pembuatan Game! Pastinya kamu sudah tahu ya mengenai game pada platform klasik ini.

Halo selamat berjumpa lagi dengan saya :) Pada kesempatan kali ini saya akan memberikan kamu tutorial mengenai **Canvas** pada J2ME yang biasanya sering digunakan di dalam pembuatan Game! Pastinya kamu sudah tahu ya mengenai game pada platform klasik ini.

Dalam tutorial kali ini kamu perlu mempersiapkan beberapa tools diantaranya:

1. Java Development Kit (JDK) v1.7
2. NetBeans dengan fitur Java Mobile Edition (ME).
3. Oracle (TM) ME SDK 3.2 Device Manager.

Sebelumnya akan saya jelaskan apa itu Canvas pada J2ME dan kegunaannya untuk pembuatan game. Lebih spesifik lagi, dalam tutorial ini saya menggunakan kelas GameCanvas yang merupakan turunan (inheritance) langsung dari kelas Canvas yang juga merupakan turunan (inheritance) dari kelas Displayable.

GameCanvas kelas menyediakan dasar untuk user interface game. Selain fitur yang diwarisi dari Canvas (commands, input events, dll) juga menyediakan kemampuan-game spesifik seperti buffer grafis off-layar dan kemampuan untuk menanyakan status kunci. Sedangkan **Canvas** adalah kelas dasar untuk menulis aplikasi yang perlu untuk menangani peristiwa-tingkat rendah dan untuk mengeluarkan panggilan grafis untuk menggambar ke layar. Aplikasi game kemungkinan akan membuat banyak penggunaan dari kelas Canvas. Dari perspektif pengembangan aplikasi, kelas Canvas dipertukarkan dengan kelas standar Screen, sehingga aplikasi dapat mencampur dan mencocokkan kanvas dengan layar tingkat tinggi yang diperlukan.

Sampel J2ME ini menunjukkan penggunaan kanvas di sebuah **MIDlet**. Sebagian besar metode grafis yang tersedia yang digunakan di sini untuk memberitahu Anda dengan cepat mempelajari konsep pemrograman grafis dalam program J2ME. Hal ini juga menunjukkan dasar-dasar membuat game J2ME menggunakan kontrol dari telepon. Misalnya penanganan kiri, tombol kanan dan atas, bawah tombol dll.

Itu tadi adalah penjelasan mengenai Canvas yang akan kita gunakan dalam pembuatan sample game ini. Selanjutnya saya bakal menjelaskan bagaimana jalannya aplikasi sample game yang akan kita buat :)

Kita akan membuat sample game yang bernama Moving Circle, dimana terdapat sebuah circle (lingkaran) dalam satu layar yang bergerak tanpa henti. Seperti yang telah kita ketahui sebelumnya bahwa aplikasi J2ME akan berjalan diatas MIDlet yang sama kegunaannya dengan Activity pada Android. Untuk itu dalam pembuatan aplikasi game ini saya akan membuat 1 kelas Canvas dan 1 kelas main (MIDlet).

Langsung saja langkah pertama yaitu buka NetBeans-nya *yaiyaalaah, dan langsung **New->Project**, pilih **Java ME Project -> Mobile Application**.

Next, beri nama project.

Next, pilih emulator dan device nya.

Setelah selesai membuat project baru, maka akan muncul dalam project explorer seperti ini:

Kemudian kita buat kelas yang pertama yaitu kelas **MovingCircleCanvas.java** dimana kegunaan kelas ini adalah untuk menggambar dan menggerakkan objek (lingkaran) dalam sebuah kanvas.

Klik, kemudian beri nama file:

Setelah file MovingCircleCanvas.java sudah dibuat, maka Let's Code! Kelas MovingCircleCanvas.java ini meng-***extends*** kelas **GameCanvas** (yang sudah dijelaskan sebelumnya) dan meng-***implements*** **Runnable** untuk menjalankan Thread.

```
public class MovingCircleCanvas extends GameCanvas implements Runnable {
```

Buat beberapa variable yang akan digunakan:

```
    public MIDlet mainMidlet;
    public Graphics g;
    public static int SLEEP_TIME = 1000 / 30; //waktu sleep Thread
    public static int ballRadius, ballX, ballY;
```

```
    public static int speedBallX = 5; //kecepatan bola terhadap koordinat
x
    public static int speedBallY = 3; //kecepatan bola terhadap koordinat
y
    public static int diameter = 10; //diameter bola
    private static final int BOX_HEIGHT = 290; //resolusi height
    private static final int BOX_WIDTH = 240; //resolusi width
```

Bangun konstruktor untuk kelas ini untuk dipanggil pada kelas Midlet.java (kelas Main)

```
    public MovingCircleCanvas(MIDlet midlet) {
        super(false);
        mainMidlet = midlet;
        //this.setFullScreenMode(true); //fullscreen
    }
```

Kemudian buat method untuk menjalankan Thread, dalam method ini kita bakal membangun game logic dari game yang akan kita buat.

```
    public void mulai() {
        Thread runner = new Thread(this);
        runner.start();
    }

    public void run() {
        // TODO Auto-generated method stub
        g = getGraphics(); //get graphics

        ballRadius = 0; //radius bola terhadap screens

        ballX = ballRadius + BOX_WIDTH / 2; // posisi awal bola (x,y)
        ballY = ballRadius + BOX_HEIGHT / 2;

        while (true) {
            g.setColor(255, 255, 255); //bgcolor
            g.fillRect(0, 0, getWidth(), getHeight()); //

            drawCircle(ballX, ballY, diameter, 0xff0000); //bentuk bola

            /**
             * membuat text di layar
             */
            g.setColor(0, 0, 0); //color untuk font dibawah
            g.drawString("A Game Test By @ndhaperdana", BOX_HEIGHT / 2,
BOX_WIDTH / 2, g.TOP | g.HCENTER);
            Font font = g.getFont();
            g.drawString("UP-DOWN to move ball", BOX_WIDTH / 2, BOX_HEIGHT
/ 2 + font.getHeight(), g.TOP | g.HCENTER);
            g.drawString("LEFT-RIGHT to change ball size", BOX_WIDTH / 2,
```

```
BOX_HEIGHT / 2 + font.getHeight() * 2, g.TOP | g.HCENTER);

            ballX += speedBallX; //biar bola bergerak searah koordinat X
            ballY += speedBallY; //biar bola bergerak searah koordinat Y

            /**
             * collision detection (tumbukan terhadap screen canvas)
             */
            if (ballX - ballRadius < 0) {
                speedBallX = -speedBallX;
                ballX = ballRadius;
            } else if (ballX + ballRadius > BOX_WIDTH) {
                speedBallX = -speedBallX;
                ballX = BOX_WIDTH - ballRadius;
            }
            if (ballY - ballRadius < 0) {
                speedBallY = -speedBallY;
                ballY = ballRadius;
            } else if (ballY + ballRadius > BOX_HEIGHT) {
                speedBallY = -speedBallY;
                ballY = BOX_HEIGHT - ballRadius;
            }
            flushGraphics(); // untuk buffer pada kanvas
            repaint(); // Permintaan repaint untuk seluruh kanvas.
            try {
                Thread.sleep(SLEEP_TIME); //sleep thread
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    /**
     * void untuk menggambar bentuk bola
     *
     * @param x //letak bola
     * @param y
     * @param diameter //diameter
     * @param color //warna
     */
    public void drawCircle(int x, int y, int diameter, int color) {
        g.setColor(color);
        g.fillArc(x - diameter / 2, y - diameter / 2, diameter, diameter,
0, 360);
    }
```

Setelah selesai membuat game logic tadi, maka lanjut membuat method untuk meng-handle ketika navigation key di tekan.

```
    /**
```

```
     * void keyPressed untuk action kalau nav.key ditekan
     */
    public void keyPressed(int keyCode) {
        int gameAction = getGameAction(keyCode);
        if (Midlet.isPaused) {
            if (gameAction == UP || gameAction == DOWN || gameAction ==
LEFT || gameAction == RIGHT) {
            } //disable nav. key
            else {
                super.keyPressed(keyCode);
            }
        } else {
            switch (gameAction) {
                case UP:
                    ballX += 5;
                    speedBallX += 5;
                    ballY += 5;
                    speedBallY += 5;
                    break;
                case DOWN:
                    ballX -= 5;
                    speedBallX -= 5;
                    ballY -= 5;
                    speedBallY -= 5;
                    break;
                case RIGHT:
                    if (diameter < 20) {
                        diameter += 1;
                    }
                    break;
                case LEFT:
                    if (diameter > 0) {
                        diameter -= 1;
                    }
                    break;
            }
        }
        repaint();
    }
```

Selesai sudah membuat dan menggerakkan suatu objek pada Canvas, sekarang kita lanjut untuk menjalankan file tadi dengan memanggil file tadi ke file utama **Midlet.java** dan beberapa tambahan untuk meng-handle jika tombol left dan right pada handphone ditekan.

```
/**
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package movingcircle;

import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.*;

/**
 * @author Nanda w Perdana
 */
public class Midlet extends MIDlet implements CommandListener {
    public MovingCircleCanvas canvas;
    public Alert alert;
    public static boolean isPaused = false;
    private Command exitCmd = new Command("Exit", Command.EXIT, 99);
    private Command pauseCmd = new Command("Pause", Command.SCREEN, 1);
    private Command resumeCmd = new Command("Resume", Command.SCREEN, 0);
    private int tempX, tempY;

    public Midlet() {
        // TODO Auto-generated constructor stub
        canvas = new MovingCircleCanvas(this);
        canvas.addCommand(exitCmd);
        canvas.addCommand(pauseCmd);
        canvas.setCommandListener(this);
    }

    protected void destroyApp(boolean arg0) throws
MIDletStateChangeException {
        // TODO Auto-generated method stub
    }

    protected void pauseApp() {
        // TODO Auto-generated method stub
    }

    protected void startApp() throws MIDletStateChangeException {
        // TODO Auto-generated method stub
        Display display = Display.getDisplay(this);
        display.setCurrent(canvas);
        canvas.mulai();
    }

    public void commandAction(Command c, Displayable s) {
        // TODO Auto-generated method stub
        if (c == exitCmd) {
            try {
                destroyApp(true);
                notifyDestroyed();
```

```
            } catch (MIDletStateChangeException e) {
                e.printStackTrace();
            }
        } else if (c == pauseCmd) {
            isPaused = true;
            canvas.addCommand(resumeCmd);
            canvas.removeCommand(pauseCmd);
            tempX = canvas.speedBallX;
            tempY = canvas.speedBallY;
            canvas.speedBallX = 0;
            canvas.speedBallY = 0;
        } else if (c == resumeCmd) {
            isPaused = false;
            canvas.addCommand(pauseCmd);
            canvas.removeCommand(resumeCmd);
            canvas.speedBallX = tempX;
            canvas.speedBallY = tempY;
//                      canvas.mulai();
        }
    }
}
```

Yap, selesai sudah kita membuat satu aplikasi sample game ini, langkah selanjutnya tinggal compile dan jalankan aplikasinya :)

Sekian tutorial dari saya, terimakasih, dan Let's Rock!

Tentang Penulis

Adnan w Anadrep