Double buffering in Java met MIDP 2.0

Auteur: drs. M.S.L.F. Manssen

http://www.manssen.eu

In de vorige tutorial hebben we *Timers en TimerTasks* behandeld. In dit document gebruiken we timers en timertasks om double buffering, zoals geïntroduceerd in MIDP 2.0, te implementeren.

Als er continu naar het scherm (Display) wordt geschreven, is de kans groot dat de gebruiker het scherm ziet flikkeren.Dit is te voorkomen met een techniek die “double buffering” wordt genoemd. Er wordt telkens geschreven naar (getekend op) een zgn. “off-screen” buffer. Als het tekenen voltooid is, wordt het off-screen buffer in één keer weggeschreven (geflushed) naar het scherm.

MIDP2.0 beschikt over de class GameCanvas, waarin double buffering is verwerkt. Met de method getGraphics() kan een referentie graphics worden opgevraagd naar het off-screen buffer. Dit buffer kan dan gedurende de rest van het leven van de midlet worden gebruikt.

Het volgende voorbeeld illustreert het gebruik van double buffering. Er wordt telkens (m.b.v. een timer en een timertask) een lijn getekend op het off-screen buffer. Dit buffer wordt pas geflushed als op de refresh button van de telefoon wordt gedrukt.

```
// Importeer de benodigde package. Voor het voorbeeld is
// vooral javax.microedition.lcdui.game.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;
import java.util.*;
import java.lang.*;

//Bestand: MyGameCanvas.java
class MyTimerTask extends TimerTask {
      private MyGameCanvas canvas;

      public MyTimerTask( ) {
      }

      // Stel het canvas in waarop getekend moet worden
      public void setCanvas ( MyGameCanvas myGameCanvas ) {
            canvas = myGameCanvas;
      }
      // Roep de methode van canvas aan waarin het tekenwerk wordt gedaan.
      public void run() {
            canvas.drawRandom();
      }
}

class MyGameCanvas extends GameCanvas {
      private Graphics graphics;
      private Random random;
      int width, height;
      int x1, y1, x2, y2;
      int color;

      public MyGameCanvas( boolean surpressKeyEvents ) {
            // Het is verplicht om eerst de constructor van de superclass
            // aan te roepen.
            super( surpressKeyEvents );
            // Vraag een referentie op naar het off-screen buffer.
            graphics = getGraphics();
```

```
            // Maak een nieuw Random object aan, dat we later zullen gebruiken
            // om random lijnen op het scherm te tekenen.
            random = new Random();
            width = getWidth();
            height = getHeight();
      }


      public void drawRandom() {
            // maak een random color aan. Gebruik alleen de laagste 24 bits
            // een color kan gedefinieerd worden als 0x00RRGGBB, met
            // RR voor de roodwaarde, GG voor de groenwaarde, en BB voor
            // de blauwwaarde;
            color = random.nextInt() & 0xFFFFFF;
            graphics.setColor( color );

            // Maak random coordinaten aan.
            x1 = Math.abs( random.nextInt() ) % width;
            y1 = Math.abs( random.nextInt() ) % height;
            x2 = Math.abs( random.nextInt() ) % width;
            y2 = Math.abs( random.nextInt() ) % height;
            // Teken een lijn met de random coordinaten
            graphics.drawLine( x1, y1, x2, y2 );
      }

      // In de refresh method roepen we flushGraphics aan om het off-screen
      // buffer naar het scherm te schrijven.
      public void refresh() {
            flushGraphics();
      }
}

public class DoubleBuffering extends MIDlet implements CommandListener {

    private Command exitCommand, refreshCommand;
    private Display display;
    private MyGameCanvas gameCanvas;
    private MyTimerTask timerTask;
    private Timer timer;
    private boolean firstTime = true;
    public void DoubleBUffering() {
    }

    public void startApp() {
      // De code van startApp hoeft maar één keer worden uitgevoerd, dus
      // gebruiken we de conditie firstTime.
      if ( firstTime ) {
            firstTime = false;
            gameCanvas = new MyGameCanvas( true );
              // Maak de commando's voor de buttons aan.

              refreshCommand = new Command("Refresh", Command.SCREEN, 1);
              exitCommand = new Command("Exit", Command.EXIT, 1);

              // Voeg de commando's toe aan de gameCanvas, en geef aan dat
              // we graag de commando's willen ontvangen.
              gameCanvas.addCommand(exitCommand);
              gameCanvas.addCommand(refreshCommand);
              gameCanvas.setCommandListener(this);

              // Stel gameCanvas in als scherm.
              display = Display.getDisplay(this);
display.setCurrent(gameCanvas);
```

```
            // Schedule MyTimerTask, die zorg draagt voor het periodiek
            // aanroepen van de drawRandom methode van MyGameCanvas, waarmee
            // een willekeurig lijn van een willekeurige kleur naar het off-
            // screen buffer wordt geschreven.
            timerTask = new MyTimerTask();
            timerTask.setCanvas( gameCanvas );
            timer = new Timer();
            timer.schedule( timerTask, 1000, 1000);
      }
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void commandAction(Command command, Displayable screen) {
        if (command == exitCommand) {
              destroyApp(false);
            notifyDestroyed();
        }
        else if (command == refreshCommand) {
            // Als er op de refresh button wordt gedrukt, wordt het off-screen
            // buffer naar het scherm geschreven.
            gameCanvas.refresh();
        }
    }
}
```