Canvas Vs Game Canvas

This articles shows some differences between Canvas and GameCanvas. The accompanying example shows an actor represented by a square that moves freely on the screen.

Canvas

Known as low-level user interface, it provides a rich variety of possibilities to draw onto the display. A class which extends Canvas has to implement the method `paint()` with instructions that draw elements in the screen.

GameCanvas

GameCanvas is a Canvas extension to simplify game development and to fix weak points from Canvas. Using `GameCanvas` is easier to control key events and update the screen. To control key events you just have to use the method `getKeyStates()` instead of using 3 methods (`keyPressed`, `keyReleased`, `keyRepeated`) used in Canvas and to update the screen you just use `flushGraphics()` method.

Example:

Let's suppose we've created an actor that is represented by a square in the screen. It has X, Y, Dx and Dy parameters. X and Y are coordinates from the upper left vertex and they cannot be less than zero. Dx and Dy represents the actor's width and height respectively.

The following snippet of code below will show the difference between Canvas and GameCanvas for handling key events.

Using Canvas

```
protected void keyPressed(int keyCode) {
        switch (getGameAction(keyCode)) {
        case UP:
            actor.changePosition(actor.getX(), actor.getY() - 5);
            break;
        case DOWN:
            actor.changePosition(actor.getX(), actor.getY() + 5);
            break;
        case RIGHT:
            actor.changePosition(actor.getX() + 5, actor.getY());
            break;
        case LEFT:
            actor.changePosition(actor.getX() - 5, actor.getY());
            break;
        }
        //This method prevents the actor to get out of the screen
        verifyPosition();

    }

    protected void keyRepeated(int keyCode) {
        switch (getGameAction(keyCode)) {
        case UP:
            actor.changePosition(actor.getX(), actor.getY() - 5);
            break;
        case DOWN:
            actor.changePosition(actor.getX(), actor.getY() + 5);
            break;
        case RIGHT:
            actor.changePosition(actor.getX() + 5, actor.getY());
            break;
```


(C) Copyright Nokia 2014. All rights reserved.

```
        case LEFT:
            actor.changePosition(actor.getX() - 5, actor.getY());
            break;
        }
        //This method prevents the actor to get out of the screen
        verifyPosition();
    }
```

Using GameCanvas

```
public void createMoviment() {

        int key = getKeyStates();

        // changing actor coordinates according to key events
        if ((key & LEFT_PRESSED) != 0) {
            actor.changePosition(actor.getX() - 5, actor.getY());

        } else if ((key & RIGHT_PRESSED) != 0) {
            actor.changePosition(actor.getX() + 5, actor.getY());

        } else if ((key & UP_PRESSED) != 0) {
            actor.changePosition(actor.getX(), actor.getY() - 5);

        } else if ((key & DOWN_PRESSED) != 0) {
            actor.changePosition(actor.getX(), actor.getY() + 5);

        }
        // This method prevents the actor to get out of the screen
        verifyPosition();
    }
```

As you can see using GameCanvas for handling key events is simplier than use Canvas because you only have to implement one method and call *getKeyStates()* to know about key events. In this example, when we use Canvas, the methods *keyPressed()* and *keyRepeated()* had to be implemented with the same code. The *keyPressed()* method is called when you press one key and release it. On the other hand, the *keyRepeated()* method is called when you press one key and holds it, dismissing after an interval time.

The following snippet of code shows the differences between Canvas and GameCanvas to update the screen.

Using Canvas

```
Image offScreenBuffer;

(...)

 protected void paint(Graphics graphics) {

        Graphics aux = offScreenBuffer.getGraphics();
        // clean the screen
        aux.setColor(0xffffffff);
        aux.fillRect(0, 0, getWidth(), getHeight());

        // draw the actor in the screen
        aux.setColor(actor.getColor());
        aux.fillRect(actor.getX(), actor.getY(), actor.getDx(), actor.getDy());
```


(C) Copyright Nokia 2014. All rights reserved.

```
        graphics.drawImage(offScreenBuffer, 0, 0, Graphics.LEFT| Graphics.TOP);
    }
```

Using GameCanvas

```
Graphics graphics;

(...)

private void paint() {
        // clean the screen
        graphic.setColor(0xffffffff);
        graphic.fillRect(0, 0, getWidth(), getHeight());

        // draw the actor in the screen
        graphic.setColor(actor.getColor());
        graphic.fillRect(actor.getX(), actor.getY(), actor.getDx(), actor.getDy());
        flushGraphics();
    }
```

Updating screen with `GameCanvas` is very simple, you just have to use `flushGraphics()` method every time you need to update the screen. However, if you are using Canvas, you will have two buffers, one is on the screen and the other you are updating all elements that you need (everything inside `paint()` method) in the end of `paint()` method you update everything. So, it is simpler to control your game cycle and update the screen using GameCanvas than using Canvas.

(C) Copyright Nokia 2014. All rights reserved.