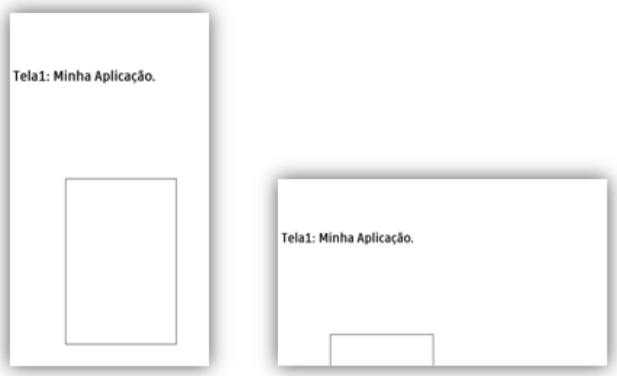
Como minimizar o problema da rotação da tela

Introdução

A funcionalidade de rotação da interface do usuário, os chamados modos retrato ou paisagem (portrait ou landscape) podem dar muita dor de cabeça para o desenvolvedor, pois a experiência do usuário com a aplicação será prejudicada. Isto ocorre, porque a maioria das aplicações que são desenvolvidas foram projetadas para visualmente atenderem a um modo de exibição, retrato ou paisagem. Veja a imagem abaixo para entender melhor sobre o problema.



NOTA: Lembrando que este problema ocorre quando é utilizado o baixo nível (*Canvas* ou *GameCanvas*). Pois, o alto nível ajusta os componentes automaticamente.

Como evitar a rotação?

Alguns celulares Nokia podem evitar que a interface do usuário rotacione. Mas, lembrando que essa funcionalidade não está disponível para todos os dispositivos. Para maiores informações, clique neste link.

Minimizando o problema da rotação da tela

O objetivo deste artigo é minimizar o problema da visualização da interface do usuário devido à rotação da tela. A solução é a exibição de uma imagem que avise ao usuário que o modo de exibição (retrato ou paisagem) não é permitido ou não é apropriado para ser visualizado.

Implementação

O método SizeChanged()

Primeiramente, precisamos conhecer o método que é chamado automaticamente quando ocorre alguma mudança na largura ou altura da tela. Este método é o `sizeChanged()`. Quando acontece a rotação da tela haverá troca dos valores entre a largura e altura, por exemplo, se no modo retrato a largura é 320 pixels e a altura é 640 pixels quando o usuário posicionar o celular em modo paisagem a largura será 640 pixels e a altura 320 pixels. Este método deve ser sobre escrito a partir de uma classe que herda de *Displayable*, exemplo, *Canvas*, ou *GameCanvas*.

A assinatura do método `sizeChanged()` é:

```
sizeChanged(int aWidth, int aHeight)
```

Onde os argumentos são os novos valores da largura e da altura.

Exemplo de código

O exemplo abaixo é uma simples implementação, porém eficaz para monitorar a rotação da tela. Neste código de exemplo, o modo paisagem(landscape) foi determinado como sendo o modo não permitido.

(C) Copyright Nokia 2013. All rights reserved.

Abaixo está um resumo das classes utilizadas:

- **CheckRotationMidlet** : Nome do MIDlet
- **Rotation**: Classe que gerencia a rotação e exibe uma mensagem de modo não permitido
- **Screen1**: Uma tela qualquer que exibe um texto e um retângulo
- **Screen2**: Uma tela que exibe somente um texto.

Vamos começar com o MIDlet:

```
public class CheckRotationMidlet extends MIDlet {

    public static Display display;


    public void startApp() {
        display = Display.getDisplay(this);

        display.setCurrent( new Screen1() );

    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

}
```

O trecho de código acima refere-se a classe `MIDlet` que está bem simples. Foi obtida uma Instância de `Display` para mudança de tela. Logo em seguida, foi dito para ser mostrada na tela do celular o `Screen1`.

Vamos agora observar a classe que gerencia as telas quando ocorre rotação.

```
/**
 * Classe que utiliza o padrão singleton
 * @author Ed Prado
 */
public class Rotation extends Canvas{


    private Displayable lastScreen;

    private static Rotation rotation;

    private final int WIDTH_LANDSCAPE;
    private final int WIDTH_PORTRAIT;

    private static Image imgModeNotAllow; //imagem de modo não permitido


    private Rotation() {
        this.setFullScreenMode(true);
        WIDTH_PORTRAIT = this.getWidth();
        WIDTH_LANDSCAPE = this.getHeight();

        try {
            lastScreen = CheckRotationMidlet.display.getCurrent();
            imgModeNotAllow = Image.createImage("/landscape.png");
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    public static Rotation getInstance(){
            if (rotation == null){
                rotation = new Rotation();
            }

        return rotation;
    }

    public void setLastScreen(Displayable last){
        lastScreen = last;
    }



    public void verifyCurrentMode(int aWidth){

        if (aWidth == WIDTH_LANDSCAPE){ //modo paisagem
            CheckRotationMidlet.display.setCurrent(this);


        }
        else if (aWidth == WIDTH_PORTRAIT) {  //modo retrato
            CheckRotationMidlet.display.setCurrent(lastScreen);
        }
           repaint();

    }

    protected void paint(Graphics g) {
        g.drawImage(imgModeNotAllow, 0, 0, Graphics.TOP | Graphics.LEFT);
    }

    protected void sizeChanged(int aWidth, int aHeight) {
        super.sizeChanged(aWidth, aHeight);

        verifyCurrentMode(aWidth);

    }
```


(C) Copyright Nokia 2013. All rights reserved.

```
}
```

Agora, vamos à explicação do código acima. Esta classe segue o padrão de projeto chamado **Singleton** que somente permite uma única instância desta classe. As variáveis `WIDTH_LANDSCAPE`, `WIDTH_PORTRAIT` são os valores da largura em modo paisagem e retrato, respectivamente.

A variável `lastScreen` armazena a última tela que estava sendo exibida antes de acontecer a rotação. Mais adiante, veremos como ela é configurada. No construtor é instanciado uma imagem que será exibida quando o celular estiver no modo que não é permitido.

O método `verifyCurrentMode()` recebe como argumento somente a largura, pois é a partir desta largura que vamos identificar em que modo o celular está quando acontecer alguma rotação. Caso o celular se posicione em modo paisagem será exibida a classe `Rotation` como tela atual, onde é mostrado ao usuário uma imagem explicando que o modo não é permitido. Caso o celular se posicione no modo retrato ele exibirá a última que o usuário estava visualizando antes de rotacionar.

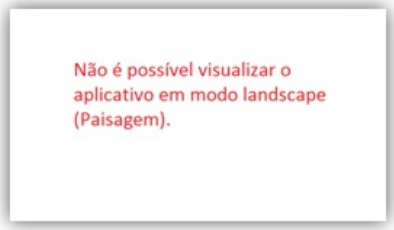
Analise o código abaixo para entender como é configurado o `lastScreen` e o que é implementado no `sizeChanged()`.

```
/**
 *
 * @author Ed Prado
 */
public class Screen1 extends GameCanvas{

    private Graphics g;

    public Screen1() {
      super(true);
        this.setFullScreenMode(true);

        g = this.getGraphics();


        g.drawString("Tela1: Minha Aplicação.", 5, 100, Graphics.TOP | Graphics.LEFT);
         g.drawRect(100, 300, 200, 300);
        flushGraphics();

        Rotation.getInstance().setLastScreen(this);
    }

    protected void pointerReleased(int x, int y) {
        super.pointerReleased(x, y);

         CheckRotationMidlet.display.setCurrent(new Screen2());

    }

//    protected void sizeChanged(int aWidth, int aHeight) {
//        super.sizeChanged(aWidth, aHeight);
//
//        Rotation.getInstance().verifyCurrentMode(aWidth);
//
//
//  }
}
```

Veja que é no construtor da classe acima que a variável `lastScreen` recebe a instância da própria classe para que a aplicação tenha conhecimento de qual tela deve retornar quando sair do modo não permitido. Essa configuração deve ser aplicada em qualquer classe que herde de *Canvas* ou *GameCanvas* podendo ser utilizada, também, nas classes que utilizam alto-nível, exemplo, *Form*, *List* etc.

A classe `Screen2` segue o mesmo raciocínio da classe `Screen1`

Veja abaixo a imagem da aplicação no modo não permitido(landscape) :



Detalhe sobre os celulares Nokia

Foi constatado que no celular Nokia(Nokia 5530) no qual a aplicação foi testada não é preciso sobrescrever o método `sizeChanged()` em todas as classes que herdam de `Displayable`. Somente é necessário tê-la na classe `Rotation`, pois quando ela é instanciada o método `sizeChanged()` será chamado mesmo que a classe `Rotation` não seja a tela em exibição.

Sugestões de melhoria

Lembre-se que você pode adicionar imagens, animações, propagandas ou até mini-jogos quando o usuário estiver no modo “não permitido”.

(C) Copyright Nokia 2013. All rights reserved.

Download

Faça o download do código-fonte completo. File:CheckRotation.zip

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.