

This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. The article is believed to be still valid for the original topic scope.



Contents

- [1 Introduction](#)
- [2 Key Handling](#)
 - ◆ [2.1 Key Action Type](#)
 - ◆ [2.2 Key Code](#)
- [3 Key Handling Example](#)
 - ◆ [3.1 widget.xml](#)
 - ◆ [3.2 key_handling.he](#)
- [4 Code Snippet](#)
- [5 See Also](#)

Introduction

This page will show you **how to handle key action in WidSets**.

Key Handling

It is quite easy to handle key action in WidSets. You can do it by just add **keyAction** system callback function to your helium source code.

```
boolean keyAction(Component source, int op, int code) {  
    ...  
    return false;  
}
```

Here is the meaning of each parameter:

- **Component source** - Currently focused component.
- **int op** ? Key action type
- **int code** ? Key code

Notice that you have to return boolean value in this function. It's used for indicate key consumption status. If you return true, it means that key event was consumed and should not be processed anywhere else.

Key Action Type

You can check action type by compare with 2nd parameter of keyAction function (**int op**).

```
boolean keyAction(Component source, int op, int code) {
    if (op == KEY_PRESSED)
    {
        ...
    }
    return false;
}
```

There are 3 key action types that WidSets widget can detect.

- **KEY_PRESSED** ? Key was pressed
- **KEY_RELEASED** ? Key was released
- **KEY_REPEATED** ? Key was repeated

Key Code

You can check that action has been sent from which key by check 3rd parameter of keyAction function (**int code**).

```
boolean keyAction(Component source, int op, int code) {
    if (op == KEY_PRESSED)
    {
        if (code == KEY_UP)
        {
            ...
        }
    }
    return false;
}
```

For normal key like **0 to 9**, * and #, you can check it using these key code directly.

```
if (code == '1' )
{
    ...
}
```

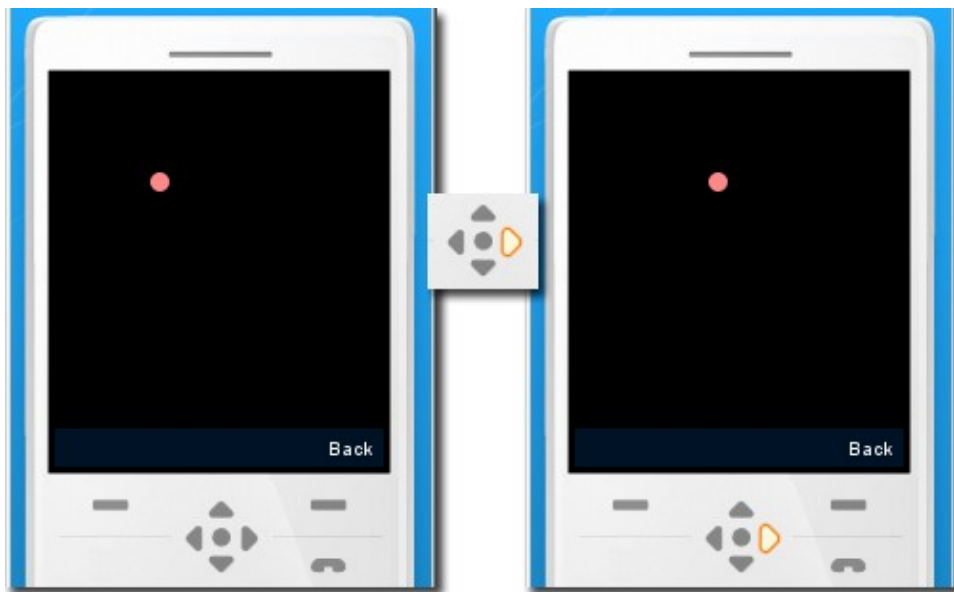
For special key, you have to use constant value for each key like listed below.

- **KEY_UP** ? Up Arrow
- **KEY_DOWN** ? Down Arrow

- **KEY_LEFT** ? Left Arrow
- **KEY_RIGHT** ? Right Arrow
- **KEY_FIRE** ? Joystick
- **KEY_OK** ? OK Softkey
- **KEY_BACK** ? Back/Cancel Softkey
- **KEY_CR** ? Carriage Return
- **KEY_ENTER** ? Line Feed
- **KEY_BACKSPACE** ? Backspace Key
- **KEY_DELETE** ? Delete Key
- **KEY_ESC** ? Escape Key
- **KEY_TAB** ? Tab Key

Key Handling Example

This example will show you how to make the ball move by pressing key.



widget.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<widget spec_version="2.1">
  <info>
    <name>Key Handling</name>
    <version>1.0</version>
    <author>Sittiphol Phanvilai</author>
    <clientversion>1.0</clientversion>
    <shortdescription>Key Handling Example</shortdescription>
    <longdescription>Example show you how to handle keypress</longdescription>
    <tags>example keypress</tags>
  </info>
```

```
<parameters>
```

```
  <parameter type="string" name="widgetname" description="Name of widget" editable="no">Key H
```

WidSets_for_Rookie_EP_10_:_Key_Handling

```
</parameters>

<resources>
  <code src="key_handling.he"/>

  <stylesheet>
    bg {
      color-1: white;
      background: solid green;
      align: hcenter vcenter;
      border: 1 1 1 1;
      border-type: rectangle white;
    }

    cenmid
    {
      align: hcenter vcenter;
    }

    canvas {
      color-1: black;
      color-2: red;
      color-3: white;
      font-3: small bold underlined;
    }
  </stylesheet>

  
</resources>

<layout minimizedheight="45sp">
  <view id="viewMini">
    
  </view>

  <view id="viewMaxi" class="bg">
    <script id="hello" class="text"/>
  </view>

  <webview>
    <weblabel class="top: 0px; left: 10px;" style="color: black;">${widgetname}</weblabel>
  </webview>
</layout>

</widget>
```

key_handling.he

```
/*
 * Copyright (C) 2008 Nokia Corporation.
 *
 * Licensed under separate Nokia Corporation End-user Software
 * Agreement (the "License").
 *
 * You may not use this Software except in compliance with the License.
 * The Software is distributed under the License on "AS IS" basis,
 * withouth warranties. See the License for the specific language
 * governing rights and limitations under the License.
 *
 * Modified by Sittiphol Phanvilai (Neois)
```

WidSets_for_Rookie_EP_10:_Key_Handling

```
*
*/

class key_handling
{
    //It's nice to store command ids to static constants
    const int CMD_BACK = 1;

    //MenuItem's are displayed over phone's soft buttons
    //Usually to go back, ok, open options menu etc
    MenuItem BACK = new MenuItem(CMD_BACK, "Back");

    Shell g_shell = null;
    Canvas g_canvas = null;

    int g_BallX = 50;
    int g_BallY = 50;

    int g_incX = 1;
    int g_incY = 1;

    //WidSets framework will call createElement() per
    //each script-element it finds from views being
    //created by createView()

    Component createElement(String viewName,
                            String elementName,
                            Style style,
                            Object context)
    {
        return null;
    }

    void startWidget()
    {
        //instantiate minimized view in startup
        setMinimizedView(createView("viewMini", getStyle("bg")));
    }

    Shell openWidget()
    {
        //instantiate maximized view when user opens this widget
        g_canvas = new Canvas(getStyle("canvas"));
        final Shell shell = new Shell(g_canvas);

        //place canvas over automatic Scrollable created by Shell
        shell[0] = shell[0][0];

        return (g_shell = shell);
    }

    void stopWidget()
    {
    }

    MenuItem getSoftKey(Shell shell, Component focused, int key)
    {
        //return the key we want to display at position=SOFTKEY_BACK
        //this usually is the Right Soft Button (RSB), for other key
        //positions return null, as we don't want other keys
        if (key == SOFTKEY_BACK) {
            return BACK;
        }
    }
}
```

WidSets_for_Rookie_EP_10_:_Key_Handling

```
    }
    return null;
}

void actionPerformed(Shell shell, Component source, int action)
{
    //when CMD_BACK event comes in, pop the current shell (this widget)
    if (action == CMD_BACK) {
        popShell(shell);
    }
}

void paint(Component c, Graphics g, Style style, int width, int height)
{
    g.setColor(style.color(0));
    g.fillRect(0, 0, width, height);
    g.setColor(0xFF8888);
    g.fillArc(g_BallX, g_BallY, 10, 10, 0, 360);
}

boolean keyAction(Component source, int op, int code) {
    if (op == KEY_PRESSED) {
        if (code == KEY_UP)
            g_BallY--;
        if (code == KEY_DOWN)
            g_BallY++;
        if (code == KEY_LEFT)
            g_BallX--;
        if (code == KEY_RIGHT)
            g_BallX++;
        if (code == KEY_FIRE)
        {
            g_BallX = 50;
            g_BallY = 50;
        }
        g_shell.repaint(false);
        flushScreen(true);
    }
    return false;
}
}
```

Code Snippet

You can download source code for this tutorial from [File:WidSets Key Handling Example.zip](#)

See Also

- [WidSets for Rookie EP 1 : First Step to WidSets SDK](#)
- [WidSets for Rookie EP 2 : First Compilation with WidSets SDK](#)
- [WidSets for Rookie EP 3 : Understand Hello World](#)
- [WidSets for Rookie EP 4 : Fasten WidSets Development](#)
- [WidSets for Rookie EP 5 : EditPlus Integration](#)
- [WidSets for Rookie EP 6 : Softkey Menu](#)
- [WidSets for Rookie EP 7 : Standard UI](#)

WidSets_for_Rookie_EP_10_:_Key_Handling

- [WidSets for Rookie EP 8 : Canvas](#)
- [WidSets for Rookie EP 9 : Timer](#)
- **[WidSets for Rookie EP 10 : Key Handling](#)**
- [WidSets for Intermediate EP 1 : HTTP Request](#)
- [WidSets for Intermediate EP 2 : HTTP with XML Filter](#)
- [WidSets for Advance EP 1 : Life Pictures Project](#)
- [WidSets SDK Tips : Emulator Language Changing](#)
- [WidSets SDK Tips : Emulator Skin Changing](#)
- [WidSets SDK Tips : Add Custom Emulator Skin](#)