Developing Metro or Panorama UI for Series 40 full touch

This code example demonstrates how to create a Panorama view (like on Nokia Lumia) for Series 40 touch devices using Nokia UI and Java ME Canvas.

Note: This is an entry in the Asha Touch Competition 2012Q3

Introduction

On most mobile UIs, apps are broken into views/pages that fit within the confines of the screen - while parts of other views may be displayed briefly during transitions, essentially each view is separate and discrete. Windows Phone provides an alternative presentation mechanism in which the paged content is displayed on a wide horizontal canvas. For some apps this works really well - allowing the different content on each "page" of content to share the common theme of the app.

This code example recreates the Windows Phone Panorama Control on Series 40 touch devices. The images below show the different screens of the UI used in this example, with the common background theme.

The UI behaviour is very similar to on Window Phones; the main difference on Series 40 is that we need to pay close attention to device limitations and optimise for reduced memory and processing power.

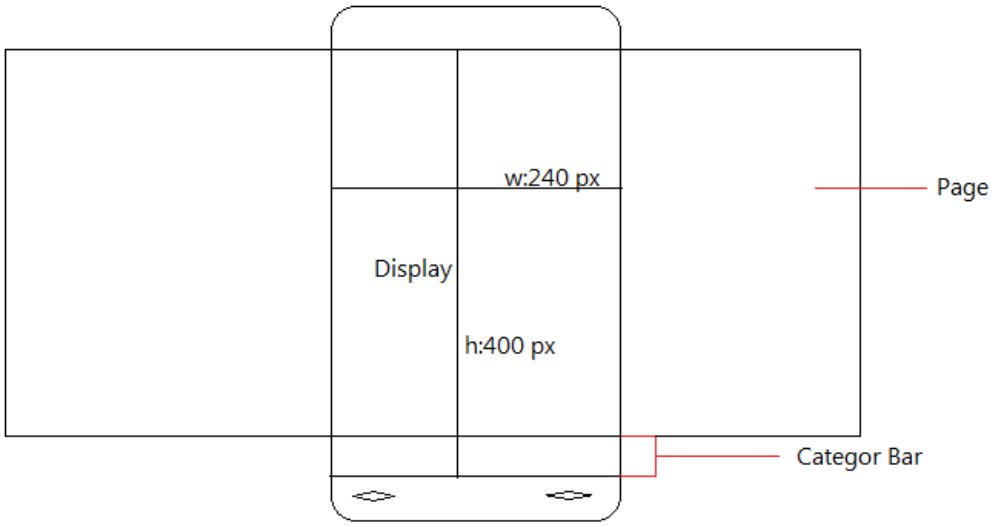
This article also provides a code component, which you can use in your applications for faster development.

Note: "Metro" is based on the design principles of classic Swiss graphic design. Animation plays a large part, with transitions, and user interactions such as presses or swipes recommended to always be acknowledged by some form of natural animation or motion. Therefore do not try to implement on non-touch devices and even on small screen resolution devices.

Design

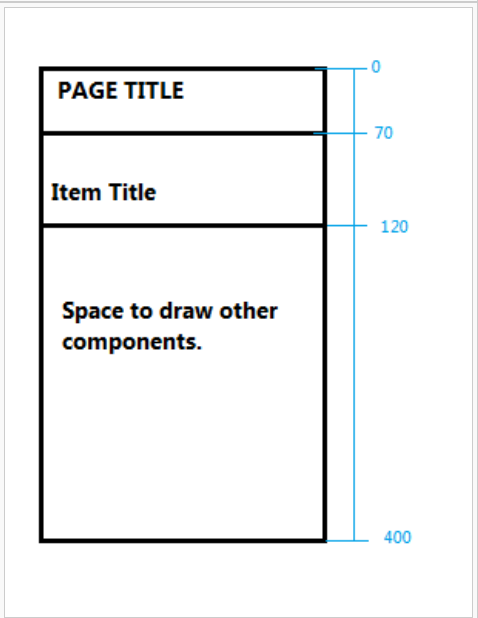
Designing a Panorama Page is simple. Its like a vertical sliding level of a game. Only the part which is in between position 0 and 240 (screen width) will be visible on screen, rest are clipped out. Series 40 full touch device have a resolution of 240x400 pixels. We can use Category Bar (new UI component available on Series 40 Developer Platform 2.0) as that of application bar (in Windows Phone).

(C) Copyright Nokia 2014. All rights reserved.



Since it includes complete canvas drawing, you should take a reference variable for x axis and construct other components depending on that reference variable. For example, If your panorama page contains four panorama items. Use item1X, item2X, item3X and item4X for panorama item1, panorama item2, panorama item3 and panorama item4. See the design specification section for complete specification used in the example provided.

Panorama item is like a container which groups all the related items together. Each panorama item consists of "Title" called *Panorama item title*. Each item/component (text box, text block, button, list, choice group etc) in a panorama item are drawn on canvas using corresponding x-reference variable.

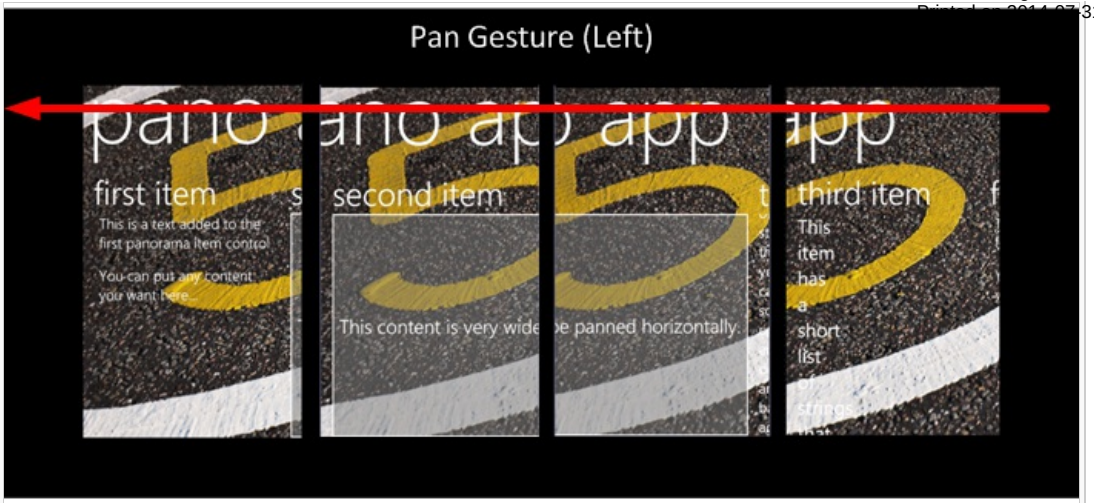


Division of screen for drawing components.

(C) Copyright Nokia 2014. All rights reserved.

Metro UI Design Principles

Metro is an design language created by Microsoft. See Design Principles for a walkthrough of the standards and design elements that embody the Metro Design.



Windows Panorama Page

UI Components

The Panorama UI is drawn on the Canvas so you cannot use the standard Form UI Components, but instead will need to create your own. I recommend you start by reading the Java Developers Library topic on Canvas.

Note: Do not use LWUIT components - while these may draw, they are incompatible with Metro design style.

This article provides a number of "compatible" components that you may find useful, including Tile Buttons and Alerts. I plan to provide other compatible components in the near future.

Tile Button

Important functions of tile `MUIButton` class are:

- `public MUIButton(String content, String imgURL)`

  Constructor, Content - button text. imgURL - Url of Image file.

- `public void setMUIButtonClickEvent(MUIButtonEvents evt);`

  Button Click Event Listener Registration function.

- `public MUIButton(String content, String imgURL)`

  Constructor, Content - button text. imgURL - Url of Image file.

- `public void setMUIButtonClickEvent(MUIButtonEvents evt);`

  Button Click Event Listener Registration function.

- `public void setPos(int x, int y);`

  Places the button on given position with respect to Panorama Item Grids.

- `public void setTextColor(int r, int g, int b);`

  To customize the text font color. default color white.

- `public int[] getTextColor();`

  Returns the current text color.

- `protected void checkPressed(int x, int y);`

  Since it is drawn on canvas, this function is called in `pointerPressed` event and this will tiger the `MUIButtonEvents`.

Creating Object of `MUIButton` is as follows:

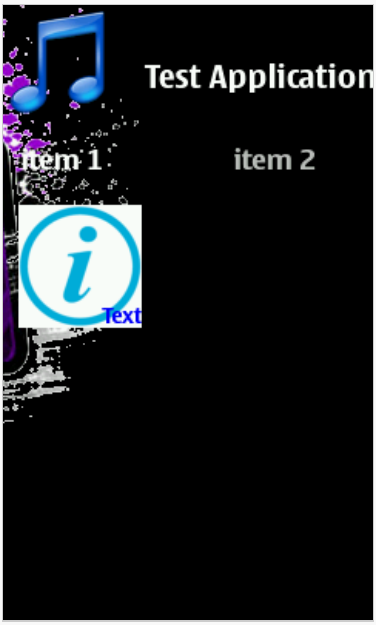
```
MUIButton btn; //Declaration.
btn = new MUIButton("Text", "/metro.png"); //Initialization.
btn.setMUIButtonClickEvent(new MUIButtonEvents() {
   public void onPress(MUIButton btn) {
   }
});  //Button Click event listener.
```


(C) Copyright Nokia 2014. All rights reserved.

```
btn.setTextColor(0, 0, 255); //To set the text color.
btn.setPos(10, 10); //Position inside PanoramaItem grid.
```

Detecting button press:

```
protected void pointerPressed(int x, int y){
 for(int i =0; i< super.total ; i++){
  panoramaItem p = (panoramaItem) super.pItems.elementAt(i);
  if(p.posX == 0 && p.itemNo == 0){
   btn.checkPressed(x, y);
  }
 }
}
```



Square tile Button.

Alerts

Important functions of `MUIAlert` Class are:

- `public MUIAlert(String title, String body, int type);`

  Constructor, title - Alert dialog title, body - Message to be displayed with alert, type - Button type.

The type can be any of the following:

- `MUIAlert_OK`

  Alert with only OK button.

- `MUIAlert_OKCANCEL`

  Alert with OK and Cancel buttons.

- `public void setMUIAlertResultListener(MUIAlertResultEventlistener evt);`

  Alert result button event listener.

The return result value will be any one of the following depending upon user selection:

- `MUIAlert_Result_OK`

  Returned on positive selection.

- `MUIAlert_Result_CANCEL;`

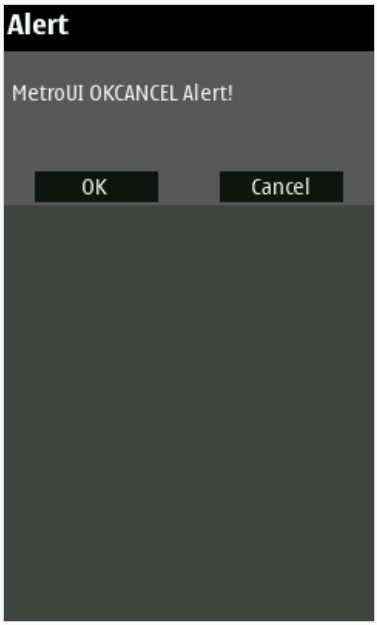
  Returned on negative selection.

Creating Object of `MUIAlert` is as follows:

(C) Copyright Nokia 2014. All rights reserved.

```
MUIAlert = new MUIAlert("Alert", "MetroUI OK Alert!", MUIAlert.MUIAlert_OK);
alt.setMUIAlertResultListener(new MUIAlertResultEventlistener() {
   public void onMUIAlertResult(MUIAlert alt, int reuslt) {
   }
});
Display.getDisplay(d).setCurrent(alt); //To display on screen. d is instance of a
midlet.
```

Alert with OK button



Alert with OK Cancel buttons

Application bar using Category Bar

You can use Category Bar as Application Bar or Task bar for navigation between different pages, if your application consists of more than one page. The article Designing Series 40 apps with Nokia UI API and Canvas for Great User Experiences explains you how to use Category Bar for Navigation (Switching between Displays) and using as Menu bar.

Page with Category Bar set visible.

Page without Category Bar

Note: Don't use Category bar with Metro UI application. However, it may be an exception for some cases.

(C) Copyright Nokia 2014. All rights reserved.

Scrolling with Flick, Selection with Tap and Animation - Gesture and Frame Animator API

Creating List menus, Scroll view and all. Most important the scrolling Panorama page, with which I started discussion. With Gesture API you can detected Tap, Flick, Drag, Drag and Drop events.

If you want to make simple or complex animation with no time, I suggest to use Frame Animator to do animations.
Using Gesture API
Using Frame Animator API, I found Demo Example. But worth of looking at it. You will find most of thing you required.

Design and Implementing Panorama Page

Design specification used in the Example Application

This section describes the main parameters used in the example application provided. These can be used as a guide, although you should use a specification that is suitable to your own app use case. Page

Object	Image Size	Text
Background Image	min: h:400, w:240 (for single item) max: h:400, w:240*no_of_item (more than 1)	no text
Page Title	70x70 pixel	Face: FACE_PROPORTIONAL Style:STYLE_ BOLD Size:SIZE_LARGE
Page Title	no image	Face: FACE_PROPORTIONAL Style:STYLE_ BOLD Size:SIZE_LARGE
Panorama Item Titles	no image	Face: FACE_MONOSPACE Style:STYLE_ BOLD Size:SIZE_MEDIUM
Body	-NA-	Face:FACE_SYSTEM Style:STYLE_PLAIN Size:SIZE_SMALL

List and List Items

Object	Image Size	Text
Icon	Size: 50x50 pixel	-NA-
Text 1	no image	Face:FACE_SYSTEM Style:STYLE_BOLD Size:SIZE_MEDIUM
Text 2	no image	Face:FACE_SYSTEM Style:STYLE_PLAIN Size:SIZE_MEDIUM
Text 3	no image	Face:FACE_SYSTEM Style:STYLE_BOLD Size:SIZE_SMALL

Sliding the page

Gesture event detection:

```
public void gestureAction(Object arg0, GestureInteractiveZone arg1,
   GestureEvent arg2) {

  switch (arg2.getType()) {
  case GestureInteractiveZone.GESTURE_FLICK:
```


(C) Copyright Nokia 2014. All rights reserved.

```
   float direction = arg2.getFlickDirection();            /* To find out the flick
direction*/
   boolean left = Math.abs(direction) < Math.PI / 2; /* i.e. right-to-left or left-to-
right */
   if (left) {
    dir = 1;         /* 1 because, we need increment in X-axis, to create left-to-right
slide effect. */
   } else {
    dir = -1;      /* -1 because, we need decrements in X-axis, to create right-to-left
slide effect. */
   }
   slide = true; /* for handling sliding animation */
                        slidecount = 0;
   break;
  case GestureInteractiveZone.GESTURE_TAP: /* this is to detect select event of Misc
Playlist item */
   m1.pointerPressed(arg2.getStartX(), arg2.getStartY());
   m2.pointerPressed(arg2.getStartX(), arg2.getStartY());
   m3.pointerPressed(arg2.getStartX(), arg2.getStartY());
  default:
   break;
  }

 }
```

Sliding animation

```
 public void run() {
  while (true) {
   if (slide) {
    if (dir == -1) {/* create right-to-left slide effect. */
     if (slidecount != 240) {/* total no of steps to move is 240, i.e screen width */
      slidecount ++;
      bgPosx --; /*
          * Background image x position reference
          * variable
          */
      item1X --; /*
          * Panorama item 1, i.e Music Playlist x
          * position reference variable
          */
      item2X --; /*
          * Panorama item 2, i.e Video Playlist x
          * position reference variable
          */
      item3X --; /*
          * Panorama item 3, i.e About x position
          * reference variable
          */
      /*
       * to create rotating effect. Background Image should
       * not scroll/slide beyond the total no. of panorama
       * item
       */
      if (bgPosx <= -(240 * itemCount)) {
       bgPosx = 0;
```


(C) Copyright Nokia 2014. All rights reserved.

```
      }
      /*
       * if the background image width is lesser than the
       * length of panorama item, i.e 240*total_item.
       */
      /* Background should repeat from beginning of Image */
      if (backgroundImage.getWidth() + bgPosx <= 0) {
       bgPosx = 240 - slidecount;
      }
      /*
       * To control the position of item and to re-position
       * back to original place, when their x position go
       * beyond the total items
       */
      if (item1X <= -(240 * itemCount)) {
       item1X = 0;
       item2X = 240;
       item3X = 480;
      }
      if (item2X <= -(240 * itemCount)) {
       item2X = 0;
       item3X = 240;
       item1X = 480;
      }
      if (item3X <= -(240 * itemCount)) {
       item3X = 0;
       item1X = 240;
       item2X = 480;
      }

     } else {
      /* to create title sliding effect */
      titleX -= 45;
      if (titleX < -(45 * (itemCount - 1))) {
       titleX = 250;
      }
      slide = false;
     }
    }
    if (dir == 1) {/*  create left-to-right slide effect. */
     if (slidecount != 240) { /* total no of steps to move is 240, i.e screen width */
      slidecount ++;
      bgPosx ++;
      item1X ++;
      item2X ++;
      item3X ++;
      if (bgPosx > 0) {
       bgPosx = -(240 * itemCount);
      }
      if (backgroundImage.getWidth() + bgPosx <= 0) {

      }
      if (item1X >= (240 * itemCount)) {
       item1X = 0;
       item2X = 240;
       item3X = 480;
      }
```


(C) Copyright Nokia 2014. All rights reserved.

```
      if (item2X >= (240 * itemCount)) {
       item2X = 0;
       item3X = 240;
       item1X = 480;
      }
      if (item3X >= (240 * itemCount)) {
       item3X = 0;
       item1X = 240;
       item2X = 480;
      }

     } else {
      titleX += 45;
      slide = false;
     }
    }
   }
   if (titleX > 2) {
    titleX -= 3;
   }
   repaint();
   System.gc();
  }
 }
```

Tip: There is nothing like you should perform sliding animation for page content too. You can skip animations for page contents. Draw them on Canvas when its parent (Panorama Item) is at visible region (i.e X position of panorama item is equal to 0). This make your application coding simple.

Component

`panoramaItem` Class: This class contains the structure to create Panorama Items.

- `panoramaItem(String title, int itemNo)`

  Constructor, title - title of panorama item. itemNo - index of item in the sequence, 0 - first panorama item and (n-1) - last panorama item.

- `int posX`

  Current x position in the page(Canvas).

- `int posY`

  Current y position in the page(Canvas). Default value 90.

`panoramapage` Class: Class extending `panoramapage` class must create objects of `panoramaItem` and should initialize the `Vector pItems` and refer this vector to draw other components in derived class. Derived class of `panoramapage` class must define these functions:

- `protected void pointerPressed(int x, int y)`
- `protected void paint(Graphics g)`

  paint method should call super class paint method. i.e this method should include: `super.paint(g);`

Important function of `panoramapage` class should be used in derived class:

- `super(String title, String bgImg, String titImg);`

  Constructor of super class. title - Page title, bgImg - path to background Image file, titImage - path to title image file.

- `super.setPanoramaItems(Vector v);`

(C) Copyright Nokia 2014. All rights reserved.

`Vector v` is an Vector containing objects of `panoramaItem` class. This initializes the `Vector pItems` of base class.

Important Variable of `panoramapage` class that will be often used in derived class:

- `Vector pItems`

  Vector containing `panoramaItem`.

- `int total`

  Total number of items in `Vector pItems`.

Initializing the `Vector pItems` of base class.

```
String[] tit = {"item 1","item 2","item 3","item 4"};
Vector items = new Vector();
for(int i = 0; i< 4; i++){
 items.addElement(new panoramaItem(tit[i], i));
}
super.setPanoramaItems(items);
```

Using `Vector pItems` and `int total` to draw components on page:

```
protected void paint(Graphics g){
 super.paint(g);
 for(int i =0; i< super.total ; i++){
 panoramaItem p = (panoramaItem) super.pItems.elementAt(i);
 if(p.posX == 0){
                if( p.itemNo == 0){
     //draw components belonging to 1st panorama item here.
       }else if(p.itemNo == 1) { ... }
                }
 }
}
```

Note: To use this component in your project, you must create project in Nokia SDK 2.0 for Java. CLDC must be 1.1 and MIDP should be 2.1.

Creating Page using Components

```
public class page extends panoramapage implements MUIButtonEvents,
MUIAlertResultEventlistener{

 MUIButton btn;
  main d;
 MUIAlert alt;
 public page(String title, String bgImg, String titImg, main d) {
  super(title, bgImg, titImg);
  this.d = d;
  btn = new MUIButton("Text", "/metro.png");
  btn.setMUIButtonClickEvent(this);
  btn.setTextColor(0, 0, 255);
  btn.setPos(10, 10);
  String[] tit = {"item 1","item 2","item 3","item 4"};
  Vector items = new Vector();
  for(int i = 0; i< 4; i++){
```


(C) Copyright Nokia 2014. All rights reserved.

```
    items.addElement(new panoramaItem(tit[i], i));
   }
   super.setPanoramaItems(items);
   alt = new MUIAlert("Alert", "MetroUI OK Alert!", MUIAlert.MUIAlert_OK);
   alt.setMUIAlertResultListener(this);
  }


  protected void paint(Graphics g){
   super.paint(g);
   for(int i =0; i< super.total ; i++){
    panoramaItem p = (panoramaItem) super.pItems.elementAt(i);
    if(p.posX == 0 && p.itemNo == 0){
     btn.paint(g);
    }
   }

  }

  protected void pointerPressed(int x, int y){
   for(int i =0; i< super.total ; i++){
    panoramaItem p = (panoramaItem) super.pItems.elementAt(i);
    if(p.posX == 0 && p.itemNo == 0){
     btn.checkPressed(x, y);
    }
   }
  }
  public void onPress(MUIButton btn) {
   Display.getDisplay(d).setCurrent(alt);
  }


  public void onMUIAlertResult(MUIAlert alt, int reuslt) {
   if(alt == this.alt){
    if(reuslt == MUIAlert.MUIAlert_Result_OK){
     Display.getDisplay(d).setCurrent(d.p1);
    }
   }

  }
 }
```

Source Code: File:SampleMetroUIComponentApp.zip, contains demo application developed using MUI (Mtero UI) components. Example demonstrate the use of MUIButton and MUIAlert also.

Component Source-code

File contains MUIButton, MUIButtonEvents, MUIAlert, MUIAlertResultEventlistener, panoramaItem and panoramapage classes. Extract to "src" folder of your project and its ready to use.
File:Panoramapage.zip

Snapshots

(C) Copyright Nokia 2014. All rights reserved.

Sample program

Required SDK: Nokia SDK 2.0 for Java Developers

The sample program is developed using "Nokia SDK 2.0 for Java (beta)" with CLDC-1.1 and MIDP 2.1. Please make sure that your target testing project implies the same before copying MIDlet/Class into your project. I suggest you to explore the sample program and use component in your project instead of using the classes from the sample program.
Attached zip file contains a basic scrolling panorama page. Flick right or left and see the amazing: File:MetroApp.zip

Tip: As I developed this as a sample, I haven't kept *OPTIMIZATION* in mind, but do not forget it.

Downloads

- Sample Metro App.
- MetroUI Component for developers.
- Sample app developed using MUI component.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.