Building the Artifact Map example application

Project setup

1. Copy the “QNXDataExample” folder to your workspace or the location where you want it.
2. In Flash Builder, choose File > New > ActionScript Mobile Project.
3. In the “Project Location” page of the wizard:
   a. enter a name for the project, such as “QNXDataExample”
   b. Under “Project Location,” choose the folder you created in step 1.
   c. Click “Next”
4. In the “Mobile Settings” page of the wizard:
   d. Under “Target Platforms” make sure only “BlackBerry Tablet OS” is checked
   e. Under “Application Settings” uncheck “Automatically reorient”
   f. Click “Next”
5. In the “Build Paths” page of the wizard, Click “Finish”

App initialization

1. In the constructor of your app’s main class, add the following code:

```
super();
stage.align = StageAlign.TOP_LEFT;
stage.scaleMode = StageScaleMode.NO_SCALE;
```

Main app container

1. Continuing in the constructor, add this code:

```
// Create layout
createLayout();
```

2. Add the createLayout() method to your class with the following code:

```
private function createLayout():void
{
    var mainContainer:Container = new Container();
    mainContainer.flow = ContainerFlow.HORIZONTAL;

    addChild(mainContainer);
    mainContainer.setSize(1024, 600);
}
```

Create the list container

1. Create a new class in the “views” package named “ListContainer” with the superclass “Container”
2. Add the following code to the ListContainer class constructor:

```
size = 224;
sizeUnit = SizeUnit.PIXELS;
debugColor = 0xFF0000;
```

3. In the body of your main application class declare the listContainer variable:

```
private var listContainer:ListContainer;
```

4. Add the lines in bold to the createLayout() method:

```
private function createLayout():void
{
    listContainer = new ListContainer();

    var mainContainer:Container = new Container();
    mainContainer.flow = ContainerFlow.HORIZONTAL;

    mainContainer.addChild(listContainer);

    addChild(mainContainer);
    mainContainer.setSize(1024, 600);
}
```

Create the image container

1. Create a new class in the “views” package named “ImageContainer” with the superclass “Container”

2. Add the following code to the ImageContainer class constructor:

```
debugColor = 0x0000FF;
align = ContainerAlign.NEAR;
```

3. In the body of your main application class declare the imageContainer variable:

```
private var imageContainer:ImageContainer;
```

4. Add the lines in bold to the createLayout() method:

```
private function createLayout():void
{
    listContainer = new ListContainer();

    imageContainer = new ImageContainer();

    var mainContainer:Container = new Container();
    mainContainer.flow = ContainerFlow.HORIZONTAL;

    mainContainer.addChild(listContainer);
    mainContainer.addChild(imageContainer);

    addChild(mainContainer);
    mainContainer.setSize(1024, 600);
}
```

Add the map image to the image container

1. In the ImageContainer class, add the following variable and method declarations:

```
private var image:Image;

private function createChildren():void
{
    if (!image)
    {
        image = new Image();
        image.setImage("assets/area.jpg");
        addChild(image);
    }
}
```

2. Add the following code to the ImageContainer() constructor:

```
public function ImageContainer()
{
    debugColor = 0x0000FF;
    align = ContainerAlign.NEAR;
    createChildren();
}
```

Load application data

1. In the main application class, add the following variable declaration:

```
private var dataDelegate:DataDelegate;
```

2. Add the following method declarations to the main application class:

```
private function loadData():void
{
    if (!dataDelegate)
    {
        dataDelegate = new DataDelegate();
        dataDelegate.addEventListener(Event.COMPLETE, loadedHandler);
    }
    dataDelegate.load();
}

private function loadedHandler(event:Event):void
{
    // Pass data to child containers
}
```

3. Add the following line to the end of the main application class's constructor:

```
loadData();
```

Create the item renderer for the map images

1. Create a new class in the "renderers" package named "SiteImage" that is a subclass of the "Image" class

2. Declare the siteVO variable in the SiteImage class as follows:

```
public var data:SiteVO;
```

3. Create the following method in the SiteImage class:

```
public function setData(data:SiteVO):void
{
    this.data = data;
    setImage("assets/" + data.icon);
    x = data.locationX - 15;
    y = data.locationY - 20;
}
```

Show the site images on the map when the data loads

1. In the ImageContainer class, add the following variable declaration:

```
private var currentSiteImages:Array = [];
```

2. Add the following method:

```
public function setData(data:Array):void
{
    var img:SiteImage;

    // Add objects
    for (var i:int = 0; i < data.length; i++)
    {
        img = new SiteImage();
        img.setData(data[i].vo);
        currentSiteImages.push(img);
        addChild(img);
    }
}
```

3. In the main application class, add the following line to the loadedHandler() method:

```
imageContainer.setData(dataDelegate.currentData);
```

Show the list control with data

1. In the ListContainer class, add the following variable declaration:

```
private var list:List;
```

2. Add the following method to the ListContainer class:

```
private function createChildren():void
{
    if (!list)
    {
        list = new List();
        list.columnWidth = 224;
        list.size = 100;
        list.sizeUnit = SizeUnit.PERCENT;
        list.selectionMode = ListSelectionMode.SINGLE;
        list.allowDeselect = true;
        addChild(list);
    }
}
```

3. Add the following line to the end of the ListContainer() constructor:

```
createChildren();
```

4. Add the following method to the ListContainer class:

```
public function setData(data:Array):void
{
    list.dataProvider = new DataProvider(data);
}
```

5. In the main application class, in the loadedHandler() method, add the following line:

```
listContainer.setData(dataDelegate.currentData);
```

Select the item on the map when it is selected in the list

1. Add the following event declaration to the ListContainer class (outside the class body):

```
[Event(name="siteChange", type="events.SiteChangeEvent")]
```

2. Add the following method to the ListContainer class:

```
private function listItemClickedHandler(event:ListEvent):void
{
    dispatchEvent(new SiteChangeEvent(event.data.vo as SiteVO));
}
```

3. Add the line in bold to the createChildren() method:

```
list.selectionMode = ListSelectionMode.SINGLE;
list.allowDeselect = true;
list.addEventListener(ListEvent.ITEM_CLICKED,
listItemClickedHandler);
addChild(list);
```

4. In the main application class, add the following method:

```
private function siteChangeHandler(event:SiteChangeEvent):void
{
    trace(event.site.name);
}
```

5. In the main application class, in the createLayout() method, add the following line of code:

```
listContainer = new ListContainer();
listContainer.addEventListener(SiteChangeEvent.SITE_CHANGE,
siteChangeHandler);

imageContainer = new ImageContainer();
```

Select the image on the map when it is selected in the list

1. In the ImageContainer class, add the following variable declarations:

```
private var glow:GlowFilter;
private var currentSelected:SiteImage;
```

2. Add the following lines to the end of the createChildren() method

```
if (!glow)
{
    glow = new GlowFilter(0xAA3333, 0.8, 10, 10, 6);
}
```

3. Add the following method to the ImageContainer class:

```
private function selectSiteImage(site:SiteImage):void
{
    if (currentSelected)
        currentSelected.filters = [];
    currentSelected = site;
    // Set the glow filter
    currentSelected.filters = [glow];
}
```

4. Add the following function to the ImageContainer class:

```
public function selectSite(selectedIndex:int):void
{
    for (var i:int = 0; i < currentSiteImages.length; i++)
    {
        if (currentSiteImages[i].data.index == selectedIndex)
        {
            selectSiteImage(currentSiteImages[i]);
            return;
        }
    }
}
```

5. In the main application class, add the following line of code to the siteChangeHandler() method:

```
imageContainer.selectSite(event.site.index);
```

Create a custom list item renderer part 1: create the class and override init() to create the display children

1. Create a new class in the “renderers” package named “ListItemRenderer.” Make it a subclass of the “CellRenderer” class.

2. In the ListItemRenderer class, declare the following variables:

```
protected var bg:Shape;
protected var img:Image;
protected var lblName:Label;
protected var lblLocation:Label;
```

3. Create an override for the init() method as follows:

```
override protected function init():void
{
    bg = new Shape();
    addChild(bg);

    img = new Image();
    img.setPosition(4, 7);
    img.setSize(30, 41);
    addChild(img);

    var format:TextFormat = new TextFormat();
    format.color = 0x333333;
    format.size = 20;
    format.bold = true;

    lblName = new Label();
    lblName.format = format;
    lblName.x = 38;
    lblName.y = 5;
    lblName.autoSize = TextFieldAutoSize.LEFT;
    lblName.filters = [new DropShadowFilter(2)];
    addChild(lblName);

    lblLocation = new Label();
    format.size = 12;
    lblLocation.format = format;
    lblLocation.x = 38;
    lblLocation.y = 34;
    lblLocation.autoSize = TextFieldAutoSize.LEFT;
    addChild(lblLocation);
}
```

Create a custom list item renderer part 2: override the setState() method to draw the contents when the state changes

1. In the ListItemRenderer class, override the setState() method as follows:

```
override protected function setState(state:String):void
{
    super.setState(state);

    var alpha:Number = 0.5;
    if (state == SkinStates.SELECTED)
    {
        alpha = 0.3;
    }

    bg.graphics.clear();
    bg.graphics.beginGradientFill(GradientType.LINEAR,
                                  [0x000000, 0x000000],
                                  [alpha, alpha / 2],
                                  [0, 254]);
    bg.graphics.drawRect(0, 0, width, height);
    bg.graphics.endFill();
    bg.graphics.lineStyle(2, 0x222222);
    bg.graphics.moveTo(0, height - 1);
    bg.graphics.lineTo(width - 1, height - 1);
    bg.graphics.lineTo(width - 1, 0);
}
```

Create a custom list item renderer part 3: override the data property setter to change the appearance when the data changes

1. Override the data property setter as follows:

```
override public function set data(value:Object):void
{
    super.data = value;
    updateCell();
}

private function updateCell():void
{
    if (this.data && data.vo is SiteVO)
    {
        var siteVO:SiteVO = data.vo as SiteVO;
        img.setImage("assets/" + siteVO.icon);
        lblName.text = siteVO.name;
        lblLocation.text = "Location: " + siteVO.locationX +
                           "/" + siteVO.locationY;
    }
}
```

Create a custom list item renderer part 4: Tell the List to use the custom item renderer

1. In the ListContainer class, add the lines in bold to the createChildren() method:

```
list.columnWidth = 224;
list.rowHeight = 60;
list.size = 100;
list.sizeUnit = SizeUnit.PERCENT;
list.setSkin(ListItemRenderer);
list.selectionMode = ListSelectionMode.SINGLE;
```

Create the options menu

1. Create a new class in the “views” package named “MenuContainer.” Make it a subclass of the “Container” class.

2. Add the following code to the MenuContainer constructor:

```
containment = Containment.UNCONTAINED;
createChildren();
setSize(1024, 81);
```

3. Add the following method to the MenuContainer class:

```
private function createChildren():void
{
    graphics.beginFill(0xDDDDDD, 0.9);
    graphics.drawRect(0, 0, 1024, 80);
    graphics.endFill();
    graphics.lineStyle(2, 0x000000);
    graphics.moveTo(0, 80);
    graphics.lineTo(1024, 80);
    graphics.lineStyle(2, 0x000000, 0.3);
    graphics.moveTo(0, 81);
    graphics.lineTo(1024, 81);
}
```

4. In the main application class, add the following variable declaration:

```
private var menuContainer:MenuContainer;
```

5. Add the lines in bold to the createLayout() method:

```
imageContainer = new ImageContainer();

menuContainer = new MenuContainer();

var mainContainer:Container = new Container();
mainContainer.flow = ContainerFlow.HORIZONTAL;

mainContainer.addChild(listContainer);
mainContainer.addChild(imageContainer);
mainContainer.addChild(menuContainer);

addChild(mainContainer);
```

Add buttons to the options menu

1. In the MenuContainer class, add the following variable declarations:

```
private var imageType1:Image;
private var imageType2:Image;
private var imageType3:Image;
```

2. Add the following code to the end of the createChildren() method:

```
if (!imageType1)
{
    imageType1 = new Image();
    imageType1.setImage("assets/source1.png");
    imageType1.containment = Containment.UNCONTAINED;
    imageType1.setPosition(1024 / 4, 15);
    addChild(imageType1);
}

if (!imageType2)
{
    imageType2 = new Image();
    imageType2.setImage("assets/source2.png");
    imageType2.containment = Containment.UNCONTAINED;
    imageType2.setPosition(1024 / 2, 15);
    addChild(imageType2);
}

if (!imageType3)
{
    imageType3 = new Image();
    imageType3.setImage("assets/source3.png");
    imageType3.containment = Containment.UNCONTAINED;
    imageType3.setPosition(1024 * 3 / 4, 15);
    addChild(imageType3);
}
```

Add a glow to the selected filter image button

1. Add the following variable declarations to the MenuContainer class:

```
private var currentImage:Image;
private var glow:GlowFilter;
```

2. In the createChildren() method, add the code in bold:

```
imageType1.setPosition(1024 / 4, 15);
imageType1.addEventListener(MouseEvent.CLICK, imgClickHandler);
addChild(imageType1);

...

imageType2.setPosition(1024 / 2, 15);
imageType2.addEventListener(MouseEvent.CLICK, imgClickHandler);
addChild(imageType2);

...

imageType3.setPosition(1024 * 3 / 4, 15);
imageType3.addEventListener(MouseEvent.CLICK, imgClickHandler);
addChild(imageType3);
```

3. Add the following lines to the end of the createChildren() method:

```
if (!glow)
{
    glow = new GlowFilter(0xAA3333, 0.8, 10, 10, 6);
}
```

4. Add the following method to the MenuContainer class:

```
private function imgClickHandler(event:MouseEvent):void
{
    if (currentImage)
        currentImage.filters = [];

    currentImage = event.target as Image;
    currentImage.filters = [glow];
}
```

Dispatch an event when a filter option button is selected

1. Add the following event declaration to the MenuContainer class (outside the class):

```
[Event(name="filter", type="events.FilterEvent")]
```

2. Add the code in bold to the imgClickHandler() method:

```
if (currentImage)
    currentImage.filters = [];

if (event.target == currentImage)
{
    dispatchEvent(new FilterEvent(0));
    currentImage = null;
    return;
}

if (event.target == imageType1)
{
    dispatchEvent(new FilterEvent(1));
}
else if (event.target == imageType2)
{
    dispatchEvent(new FilterEvent(2));
}
else if (event.target == imageType3)
{
    dispatchEvent(new FilterEvent(3));
}

currentImage = event.target as Image;
currentImage.filters = [glow];
```

Apply the filter to the list and map when a filter is chosen

1. In the ImageContainer class, add the code in bold to the setData() method:

```
var img:SiteImage;

// Remove objects
while (currentSiteImages.length > 0)
{
    img = currentSiteImages.pop();
    removeChild(img);
}

// Add objects
for (var i:int = 0; i < data.length; i++)
```

2. In the main application class, add the code in bold to the createLayout() method:

```
menuContainer = new MenuContainer();
menuContainer.addEventListener(FilterEvent.FILTER, filterHandler);

var mainContainer:Container = new Container();
```

3. Add the following method to the main application class:

```
private function filterHandler(event:FilterEvent):void
{
    if (event.filterType > 0)
    {
        var filterData:Array = [];
        for (var i:int = 0; i < dataDelegate.currentData.length; i++)
        {
            var item:Object = dataDelegate.currentData[i];
            if (item.vo.type == event.filterType)
            {
                item.vo.index = filterData.length;
                filterData.push(item);
            }
        }

        listContainer.setData(filterData);
        imageContainer.setData(filterData);
    }
    else
    {
        listContainer.setData(dataDelegate.currentData);
        imageContainer.setData(dataDelegate.currentData);
    }
}
```

Make the options menu respond to a bezel swipe

1. In the MenuContainer class, add the following variable declaration:

```
private var isVisible:Boolean = false;
```

2. Add the following lines to the end of the MenuContainer() constructor:

```
QNXApplication.qnxApplication.addEventListener
(QNXApplicationEvent.SWIPE_DOWN, swipeHandler);
y = -82;
```

3. Add the following method to the MenuContainer class:

```
private function swipeHandler(event:QNXApplicationEvent):void
{
    if (!isVisible)
        Tweener.addTween(this, {y: 0, time: 1.7});
    else
        Tweener.addTween(this, {y: -82, time: 1.7});

    isVisible = !isVisible;
}
```

Modify the code to allow testing on the desktop without getting a VerifyError

1. Modify the last two lines in the MenuContainer constructor as follows:

```
try
{
//    QNXApplication.qnxApplication.addEventListener
(QNXApplicationEvent.SWIPE_DOWN, swipeHandler);
    var q:QNXApplication;
    var c:Class = getDefinitionByName("qnx.system.QNXApplication") as
Class;
    c.qnxApplication.addEventListener(QNXApplicationEvent.SWIPE_DOWN,
swipeHandler);
    y = -82;
}
catch(error:VerifyError)
{
    // do nothing
}
```

Add an application icon

1. In the blackberry-tablet.xml file, modify the <qnx> node as follows:

```
<qnx>
    <icon>
        <image>assets/app-icon.png</image>
    </icon>
</qnx>
```

Specify a category for the application

1. Modify the <qnx> node, adding the line in bold:

```
    </icon>
    <category>core.games</category>
</qnx>
```

Add an application splash screen

1. Modify the <qnx> content by adding the code in bold:

```
    <category>core.games</category>
    <splashscreen>assets/splashscreen.jpg</splashscreen>
</qnx>
```