Integrating HERE Maps in LWUIT application

This article explains you can use the HERE Maps library in a LWUIT based application

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

Introduction

Lightweight UI Toolkit (LWUIT) is a popular UI framework and widget library for creating J2ME Apps, however in its current state it does not include a Map component to display maps, route, markers etc. Nokia has an excellent map and navigation services and they recently added support of it for their Series 40 devices, you can check out the full details about it here. My aim is to create a bridge between the library that Nokia offers for use and the LWUIT library by creating a Map component.

Note: In order to use HERE maps and the attached component you need to register your app and obtain a token for the maps

HERE maps integrated with LWUIT app.

The implementation

The HERE Maps API for Java ME is not open source which mean we only have access to what the library jar exposes to us to work with.

The implementation is divided into two classes

- `MapCanvas` - This is a simple class which inherits the `MapCanvas` of the HERE maps library. It is needed to expose the `pointerPressed()`, `pointerReleased()`, `keyPressed()`, `keyReleased()` etc. methods to call them from the LWUIT component.
- `Map` - This is the LWUIT component class. It extends the LWUIT `Component` class and implements the `MapListener` interface from the HERE Maps library. This class also initiates the `MapCanvas` and is responsible for drawing the `MapCanvas` to the display. Lastly, it also supports adding markers to the map.

The `MapCanvas` has absolutely nothing special and you can check out its full code in the attached source.

The `Map` Component most important method is actually is its `paint()` method.

Here is where the trick of binding the two seperate libs come together we use the `MapDisplay` class that we obtain from the `MapCanvas` and paint it and all other map components to a helper image graphics, we then create a LWUIT based image from the helper image and draw that to the lwuit graphics display.

(C) Copyright Nokia 2014. All rights reserved.

```
public void paint(Graphics g)
{
 super.paint(g);
 if (helperImg == null)
 {
  helperImg = javax.microedition.lcdui.Image.createImage(getWidth(), getHeight());
  helperImgGraphics = helperImg.getGraphics();
 }
 mapDisplay.paint(helperImgGraphics);
 MapComponent[] comps = mapDisplay.getAllMapComponents();
 for (int i = 0; i < comps.length; i++)
 {
  comps[i].paint(helperImgGraphics);
 }
 com.sun.lwuit.Image img = com.sun.lwuit.Image.createImage(helperImg);
 g.drawImage(img, 0, 0);
}
```

Another good piece of code to look at is the `Map` constructor, it takes as an arguement the MIDLet `Display` class since the `MapCanvas` requires it as well as the `AppID` and token.

```
public Map(Display display,String appID, String token)
{
 ApplicationContext.getInstance().setAppID(appID);
 ApplicationContext.getInstance().setToken(token);
 mapCanvas = new MapCanvas(display);
 factory = mapCanvas.getMapFactory();
 mapDisplay = mapCanvas.getMapDisplay();
 mapDisplay.setMapListener(this);
 mapDisplay.setCenter(new GeoCoordinate(51.477, 0.0, 0));
 mapDisplay.setZoomLevel(15, 0, 0);
 setFocusable(true);
 markerCount = 0;
 markerDataTable = new Hashtable();
}
```

Summary

You can use the ready made component I created to integrate HERE maps library to your own LWUIT based application, it supports both touch & joystick navigation inside the map.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.