Adding Markers to the Map with Maps API for Java ME

This article explains how to add markers onto the map with the HERE Maps API for Java ME API and also how to make them draggable.

Introduction

The HERE Maps API for Java ME makes it possible to easily embed HERE Maps service into Java ME applications. The API also includes the functionality to add markers to the map.

Constructing and adding markers

Markers are added to the Map by adding them into the instance of MapDisplay object, which can be retrieved from current MapCanvas object. Note that you should never try constructing the marker directly, instead it should be constructed utilizing functions provided by the MapFactory. An instance of the MapFactory can be retrieved with the MapCanvas object, and then the createMapMarker method can be used to construct the marker:

```
MapMarker marker = mapFactory.getMapFactory().createMapMarker(position,
Image.createImage("/nokiaicon.png"));
mapCanvas.getMapDisplay().addMapObject(marker);
```

You can use custom images with createMapMarker method as shown above, or if you want, you can also use the following predefined shapes:

- BALLOON
- HEXAGON
- OCTAGON
- PENTAGON
- RECTANGLE
- SQUARE
- STAR
- TRIANGLE

When using the shapes, you can also define the size in pixels (as second argument) as well as the text (as third argument) to be used inside the marker as illustrated below:

```
MapMarker marker = mapFactory.createStandardMarker(new GeoCoordinate(latitude,
longitude, altitude),40, "3",MapStandardMarker.SQUARE);
```


(C) Copyright Nokia 2014. All rights reserved.

Implementing Draggable markers

As shown in Handling map events with Maps API for Java ME article, if you want to catch map pointer events, you need to implement a custom `MapComponent` with an `EventListener`. Thus with draggable markers you need to implement these components as well, and in this example all logic for dragging is also implemented inside the `EventListener` component. The first thing to do with the pointer events is then to catch the event where the `Marker` is pressed before the dragging is started. This naturally is handled inside the `pointerPressed()` method:

```
public boolean pointerPressed(int x, int y) {
 if(mapDisplay != null){
         Point nowPoint = new Point(x,y);
                MapObject selObj = mapDisplay.getObjectAt(nowPoint);
                if (selObj != null && selObj instanceof MapStandardMarker) {
                 marker = (MapStandardMarker)selObj;
                     clicking = true;
                     movingMarker = false;
                     return true;
                }else {
                    return false;
                }
        } else {
  return false;
 }
}
```

Basically with the code the `getObjectAt()` method is used for determining whether there is a `Marker` under the point pressed, and if there is, then dragging is started. Then the next step is to actually drag the marked when user moves the pointer in the map.

```
public boolean pointerDragged(int x, int y) {
 if(marker != null){
  clicking = false;
                movingMarker = true;
                marker.setCoordinate(mapDisplay.pixelToGeo(new Point(x, y)));
  mapCanvas.repaint();
                return true;
            }else{
                movingMarker = clicking = false;
                return false;
 }
}
```

Note that if there is no `Marker` selected in the `pointerPressed()` method, then this function need to return `false`, otherwise the normal panning with the map would also be disabled. And once the user stops the dragging and lifts the finger from the map, then the `pointerReleased()` method is called. In this method we need to finalize the dragging, also with this example there is separate handling for clicking the `Marker`:

```
public boolean pointerReleased(int x, int y) {
 boolean retVal = false;
        if(marker != null){
         if(clicking){
                 mapDisplay.setCenter(marker.getCoordinate());
                 mapCanvas.repaint();
                 retVal = true;
                }else if(movingMarker){
```


(C) Copyright Nokia 2014. All rights reserved.

```
                    marker.setCoordinate(mapDisplay.pixelToGeo(new Point(x, y)));
                    mapDisplay.setCenter(marker.getCoordinate());
                    mapCanvas.repaint();
                    retVal = true;
                }

        }

        marker = null;
 clicking = movingMarker = false;
 return retVal;
}
```

Resources

Full source code of the MapRevGeoCoding is available here: File:Java MapsMarker.zip

Installation files are available here: File:MapsMarker binaries.zip

Summary

The HERE Maps API for Java ME offers rich functionalities that makes it possible to integrate all the HERE Maps features in a Java ME application, with just a few lines of code.

(C) Copyright Nokia 2014. All rights reserved.