OpenMap™ Development, Part 2

Don Dietrick
OpenMap™ Technical Lead
openmap@bbn.com
http://openmap.bbn.com
Agenda

- Controlling the Map Projection
- Different Projections
- Managing OMGraphics
- Symbols
- Drawing OMGraphics
- OMGrid/OMRaster Issues
- Clipping, Holes in OMGraphics
- PropertyConsumer and GUIs
- Geo (spatial indexing)
- OpenMap 5.0 changes, including time controls
Controlling the Map Projection
• Projection object in MapBean controls map view
• Change Projection, layers notified
• MapBean API methods
  – setProjection(Projection proj);
  – setCenter(lat, lon);
  – setScale(float scale);
MapBean Projection API

- Each call to MapBean Projection API causes Projection events to be sent to layers
- For multiple changes, modify Projection object instead and set it on MapBean

Instead of:

```java
mapBean.setCenter(lat, lon);
mapBean.setScale(scale);
```

Do:

```java
Proj p = (Proj) mapBean.getProjection();
p.setCenter(lat, lon);
p.setScale(scale);
mapBean.setProjection(p);
```
MapBean as Listener

- MapBean is an event listener
  - PanListener for PanEvents
  - ZoomListener for ZoomEvents
  - CenterListener for CenterEvents
Find the MapBean

• For API calls, find MapBean as source object of events
  – ProjectionEvent
  – MouseEvent

• Components that want the MapBean to listen for events should use the MapHandler.
• MapBean provides Projection, a read-only interface
• Projections need to be cast to Proj in order to modify parameters
• Call setProjection(Projection) on MapBean after modifications. Layers hold copy and won’t pick up changes.
OpenMap Projection Types

- Mercator
- Equal Arc (CADRG, LLXY)
- Orthographic
- Equal Distance (LambertConformal, UTM)
- Gnomonic
Projection Line Types

• Great Circle
  – Shortest geocentric line between two points
  – Longitude lines

• Rhumb
  – Line of constant bearing between two points
  – Latitude (and Longitude) lines

• Straight
  – Pixel lines, no geographic significance
Mercator

• Longitude and latitude lines perpendicular and straight
• Distortion - moving away from equator increases pixel distances between latitude lines
• Lines of constant direction (bearing) are straight
Equal Arc (CADRG, LLXY)

- Pixels and geocentric coordinates change at the same rate
- Good for images from generic sources
Orthographic

- Great circle lines through center are straight
- Always directed at the center point of map
Equal Distance (UTM, Conic)

- Pixels and ground distance change at same rate
- Zones based on Longitude, accuracy errors occur when away from Median longitude
Gnomonic

- Great circle lines are straight
Managing OMGraphics
Layer Manages OMGraphics

- Data View, responding to Projection
- Layer is responsible for deciding how OMGraphics are managed and organized
- If more organization is needed, OMGraphicLists to organize OMGraphics into groups
Responding to Scale Changes

• Change Layers, completely different data source

• Change rendering contents of OMGraphicList for single Layer

• Choice usually depends on:
  – data source for different scales - different source, different layers
  – Complexity of application to present to users
Switching Layers

• ScaleFilterLayer
  – A parent layer that switches between child layers, depending on the map scale
  – Need to have one less transition scale than layer count

• BufferedLayer
  – A layer that maintains a set of layers within a personal MapBean
  – Renders layers as images
ScaleFilterLayer

• Configure in API

    setLayersAndScales(Layer[], float[]);

• Configure in Properties

    scaledRoads.class=com.bbn.openmap.layer.ScaleFilterLayer
    scaledRoads.prettyName=Northeast Roads
    scaledRoads.layers=neroads neroadsDetailed
    scaledRoads.transitionScales=150000

    neroads.class=com.bbn.openmap.layer.shape.ShapeLayer
    ...

    neroadsDetailed.class=com.bbn.openmap.layer.shape.ShapeLayer
    ...
BufferedLayer

• Configure in API
  ```java
  addLayer(Layer);
  removeLayer(Layer);
  clearLayers();
  ```

• Configure in Properties
  ```java
  bufLayer.class=com.bbn.openmap.layer.BufferedLayer
  bufLayer.prettyName=My Layer Group
  bufLayer.layers=layer1 layer2 layer3
  bufLayer.visibleLayers=layer1 layer3
  bufLayer.hasActiveLayers=false...
  ```
Modifying Layer Data

- **OMGraphicHandlerLayer**
  - Filter in the prepare() method

- Initially organize data to quickly evaluate OMGraphics as sets

- Remember to consider data set size
  - Brute force might be just as good as clever organization
  - Run, evaluate, refine
Organizing OMGraphics

• OMGraphic organization dictated by rendering appearance
  – Rendering related to drawing order
  – Drawing order dependent on OMGraphicList add order

• OMGraphicList doesn’t have to be used for rendering
  – Visibility
  – Selection
Symbols
Using Symbols

• Different sources
  – Images from files
  – Mil-Std-2525B package
  – Icon package

• OMScalingIcon
  – Centers image over location
  – Can change image size for
• Use ImageIcon class to load

    ImageIcon ii = new ImageIcon("data/images/earthmap1k.jpg");
    Image image = ii.getImage();

• Can also use BufferedImageHelper
  • com.bbn.openmap.image package
  • Uses JAI if available

    URL imageURL = PropUtils.getResourceOrFileOrURL(imagePath);
    BufferedImage bi =
        BufferedImageHelper.getBufferedImage(imageURL);
Mil-Std-2525b Symbols

- Use SymbolReferenceLibrary
- com.bbn.openmap.tools.symbology.milStd2525 package
- DISA symbol set, not all possibilities included
- Symbols identified by 15 character key code
- SymbolChooser lets you figure out what’s available and get the code
SymbolReferenceLibrary

- Configure in Properties:
  
srl.class=com.bbn.openmap.tools.symbology.milStd2525.SymbolReferenceLibrary
  srl.imageMakerClass=com.bbn.openmap.tools.symbology.milStd2525.PNGSymbolImageMaker
  srl.path=data/symbols/milStd2525

- Configure via API
  
  PNGSymbolImageMaker imageMaker = new PNGSymbolImageMaker(pathToPNGs);
  SymbolReferenceLibrary srl = new SymbolReferenceLibrary(imageMaker);

- PNG, GIF and SVG available
OMIconFactory

- Creates resizable vector icons
- Defined with IconPart/IconPartList
- Rendered with DrawingAttributes
Drawing OMGraphics
OMDrawingTool

- Passes MouseEvents to OMGraphics to change OMGraphic parameters
- EditableOMGraphics wrap OMGraphic
  - Provide grab points
  - Contain state machine that control modification of OMGraphic
- Manages modification, passes result back to DrawingToolListener
OMDrawingTool and EditToolLoaders

- EditToolLoader tells OMDrawingTool which EditableOMGraphic should be used for OMGraphic
- EditToolLoaders available for circles, range rings, rectangles, polys, rasters, points, lines, splines, text, distance
EditorLayer

- OMGraphicHandlerLayer with Tool Interface
- Provides controls on ToolPanel
- Provide editing controls for single layer
- Requires EditorTool component that determines what editor does
  - DrawingEditorTool
- Integrated mouse mode directs events to layer
EditorLayer Examples

- EditorLayer with DrawingEditorTool
- GeoIntersectionLayer
OMDrawingTool via API

• Ask if DrawingTool can handle OMGraphic
  omDrawingTool.canEdit(omGraphic.getClass());

• Edit OMGraphic
  omDrawingTool.edit(omGraphic, drawingToolRequestor);

• Create OMGraphic
  omDrawingTool.create(class, drawingAttributes,
                        DrawingToolRequestor, ifGUIAllowed);
DrawingToolRequestor

- Component the DrawingTool notifies when it's done
- Usually the layer
  
  ```
drawingComplete(OMGraphic, OMAction);
  ```

- OMAction lets you know if OMGraphic should be added, deleted or moved
Spatial Filtering

• FilterSupport provides simple spatial filtering in x-y space for an OMGraphicList

FilterSupport fSupport = new FilterSupport(OMGraphicList);

• Filtering based on Shape object, can be retrieved from projected OMGraphic and passed to FilterSupport

OMGraphicList filtered = fSupport.filter(Shape, insideOrOutside);
OMGrid/OMRaster Issues
Projection Mismatch

- OMGGrid appearance is set of OMGraphics created by OMGridGenerators
- SimpleColorGenerator creates Equal Arc projected image, won’t match Mercator
- New ImageWarp/OMWarpingImage classes will take pixels and create warped image
Crossing the Dateline

- Crossing the Dateline is handled, if the entire image is rendered on map.
- Really a small-world problem - occurs when image needs to be rendered on both sides of map, i.e. if the image wraps around the map area.
Small-world Solutions

• OMScalingRaster needs to be extended/updated to paint image twice, once for each side of map
• Can take code from OMRect to make the projection calls to determine locations

```java
ArrayList rects =
    proj.forwardRect(new LatLonPoint(lat1, lon1), // NW
                     new LatLonPoint(lat2, lon2), // SE
                     lineType, nseg, !isClear(fillPaint));
int size = rects.size();
```

• Make smaller OMGrids/OMRasters
Clipping and OMGraphics

- Setting clip area on `java.awt.Graphics` limits the render area, specifies the area that gets painted.
- `java.awt.geom.Area` created from Shape, other areas can be subtracted.
- JTS Topology Suite
Using the PropertyConsumer Interface
PropertyConsumer

- Interface for component that can be configured by Properties
- Provides methods for setting and getting properties
- Also provides methods for gathering information about what properties are available to be set (editors)

```java
void setProperties(String, Properties);
void setProperties(Properties);
void setPropertyPrefix(String);
String getPropertyPrefix();
Properties getProperties(Properties);
Properties getPropertyInfo(Properties);
```
Defining Properties

• As defined in the class, properties should be unscoped, as if property prefix is undefined
  
  ```java
  public final static ShowLabelsProperty = “showLabels”; // not .showLabels
  ```

• In setProperties, use PropUtils to prepare prefix:
  
  ```java
  String prefix =
  PropUtils.getScopedPropertyPrefix(PropertyConsumer);

  String prefix =
  PropUtils.getScopedPropertyPrefix(String);

  String property = properties.get(prefix + ShowLabelsProperty);
  ```
Property Output

• getProperties method returns Properties filled with scoped key values as Strings

```java
properties.putProperty(prefix + ShowLabelsProperty, Boolean.toString(showLabels));
```

• Should always call super.getProperties() if superclass is PropertyConsumer
• Should always return a Properties object
• getPropertyInfo provides Properties that describe PropertyConsumer
• Each property requires 3 properties added
  – Non-scoped property name
  – Property name + ScopedEditorProperty
  – Property name + LabelEditorProperty

propInfo.put(ShowLabelsProperty, “Description”);
propInfo.put(ShowLabelsProperty + ScopedEditorProperty,
            com.bbn.openmap.util.PropertyEditor.YesNoProperty);
propInfo.put(ShowLabelsProperty + LabelEditorProperty, “GUI Label”);
PropertyEditors

- ScopedEditorProperty allows you to define a component that provides a GUI for a property
- PropertyEditors in package com.bbn.openmap.util.propertyEditor
- Inspector class creates GUI from getPropertyInfo properties
Using the Geo Spatial Package
Geo Spatial Index Package

- Package that represents locations, paths and areas as vectors
- A coordinate is represented by Geo
  - x, y, z tuple instead of lat, lon
- Paths and areas are arrays of Geos
Geo

- Represents location
- Can perform operations against other Geos
  - Distance
  - Midpoint between
  - Other Geo at azimuth, distance
  - Vector math - add, subtract, dot and cross products

- GeoArray is a Geo[] object
GeoExtent

- Group of Geos that represent an object
  - GeoSegment - two points
  - GeoPath - more than two points
  - GeoRegion - area of points
- All have BoundingCircle (center, radius)
ExtentIndex

- Spatial index that organizes Extents for quick searches
- ExtentIndexImpl organizes by longitude strips
Intersection

• Class that manages queries on an ExtentIndex
• Also answers direct spatial intersection queries about Extents
• For ExtentIndex queries, uses MatchFilter and MatchCollector
  – MatchFilter specifies distance between objects to be considered a hit
  – MatchCollector is where you get the answers, via iterator
Using Intersection with ExtentIndex

Intersection

MatchFilter

MatchCollector

Extent Index

Extents

Consider

Extent

Fetch Iterator from MatchCollector
Intersection Functions

- Region intersects region?
- Path intersect region?
- Point in region?
- Intersection point between GC arcs?
- Polys intersect?
- Point on segment?
- GC arc within distance of point?
OpenMap 5.0 Changes
OpenMap 5.0

- Changes will be required to move from OM4.6.5 to OM5.0
- Projection API changed
- Increased precision
- OMGeo, OMSHAPE
Starts with LatLonPoint

- LatLonPoint drastically changed
  
  com.bbn.openmap.proj.coords.LatLonPoint

- Extends Point2D, behaves like it
  - LatLonPoint abstract
  - Create LatLonPoint.Float, LatLonPoint.Double

- ‘get’ methods return different values
  - getX(), getY() returns double decimal degrees
  - getRadLat(), getRadLon() returns radians
  - getLatitude(), getLongitude() returns float decimal degrees

- setLocation(X, Y) !!! Just like Point2D
• Projections are now based on Point2D objects
  – Old:
    \[
    \text{Point forward(LatLonPoint);} \\
    \text{LatLonPoint inverse(Point);} \\
    \]
  – New:
    \[
    \text{Point2D forward(Point2D);} \\
    \text{Point2D inverse(Point2D);} \\
    \]
Projection Hierarchy

- Proj/Projection still the top class
- Cartesian projection (x, y)
- GeoProj, new class
  - Generics allow LatLonPoint returns
  - All old projections are GeoProj objects
• ProjectionFactory no longer singleton
• Created like any other Component, should be added to MapHandler
• Available via MapBean
Precision Upgrade

- Projection library calculations all based on doubles
- OMGraphics based on doubles, signatures changed
- Old floats only accurate to 7ft, doubles accurate to centimeters.
OMGeo

• OMGraphic class that internally uses Geo and Extents to represent shapes
• OMGeo abstract, instantiate inner classes
  – OMGeo.Pt
  – OMGeo.Line
  – OMGeo.Polyline
  – OMGeo.Region
• Can be used in ExtentIndex
OMShape

• New OMGraphic that takes java.awt.Shape objects as source geometry
• Intended for use with Cartesian projection, will work with GeoProj projections if coordinate range fits
OMEvents and Time Controls
OMEEvents represent something that happens at a time and place

Time Controls display events in

- List form
- Timeline form
- On the map for a given time

Manage Clock’s current time
<table>
<thead>
<tr>
<th>OMEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Provides text description of event</td>
</tr>
<tr>
<td>• Timestamp</td>
</tr>
<tr>
<td>• Optional location</td>
</tr>
<tr>
<td>• Can contain attributes</td>
</tr>
</tbody>
</table>
OMEventHandler

- Creates OMEvents from data source
- Added to MapHandler
- Can provide filtering
Event List Presenter

• Provides a chronological list of events
• Events are selectable
  – Timeline adjusts
  – Map adjusts of events have location
Timeline

- Provides view to events on time scale
- Selectable
  - Event list responds
  - Map responds if events have location
OpenMap Contact Information: