Creating Dynamic Layers
in a MapObjects Java Application

By Rex Hansen, ESRI Developer Support Group

This article discusses the benefits and drawbacks associated with several of the methods available for creating and managing dynamic layers in an application created using MapObjects—Java Edition (MOJ). It assumes that the reader is familiar with Java programming and has some experience developing MapObjects applications.

When using MOJ, a developer may choose to dynamically create and display graphic features (including text) along with data layers in a map component. In general, the geometry of graphic components in a dynamic layer is created at run time, using either the application code or a data file (i.e., a text file with *x* and *y* coordinates). Different methods can be used to generate a dynamic layer, depending on the extent that the developer needs to interact with the data in the layer. The three primary techniques are

- Override the paintComponent method in AcetateLayer
- Create a GraphicsLayer
- Create a FeatureLayer

Each method will be discussed, and sample code listings are available from the *ArcUser Online* Web site (www.esri.com/arcuser). All listings contain code for generating the same triangle polygon, but each uses a different method for creating a dynamic layer. The triangle's coordinates are specified in decimal degrees, and its extent is located within the continental United States.

Override the paintComponent Method in an AcetateLayer

Subclassing an AcetateLayer and overriding the paintComponent() method is the simplest technique for creating a dynamic layer. An AcetateLayer is essentially a transparent JComponent that contains a com.esri.mo.map.dpy.Layer and is added to the JLayeredPane subcomponent that is internal to a Map. Changes to a Map extent cause the AcetateLayer to be repainted. Note that all features inherent to an AcetateLayer are asynchronously rendered in the paintComponent() to an offscreen buffer, separate from the offscreen buffer used by the Map component. Listing 1, available online, provides an example of this method.

The following characteristics must be considered when using this technique.

- Graphic features must be created using the java.awt.geom.* package.
- Often, graphic geometry must be transformed to display in the proper location on the Map component. In order to transform from world to pixel coordinates, the extent of the Map component must be set to any valid value except null.
- The new AcetateLayer object can be instantiated and added as an AcetateLayer to a Map component. By default, AcetateLayers are visible in the Map and not added to the Map legend. Figure 1 contains some sample code that adds an AcetateLayer to a Map:

```
Figure 1

acetateLayer alayer = new acetateLayer(aMap);
aMap.add(alayer);
```

Use an AcetateLayer when creating simple graphics that do not contain any inherent attribute information. Since it renders to a separate offscreen buffer than the Map component, the redraw() method of the Map component does not have to be called to refresh the contents of the AcetateLayer's paintComponent() method. Therefore, graphics can be refreshed without redrawing the data layers in the Map component, saving valuable processing and performance time in the MOJ application on the client. Common uses for AcetateLayers are displaying temporarily highlighted features or text labels.

Create a GraphicsLayer

The GraphicsLayer interface is implemented by two instantiated subclasses, BaseGraphicsLayer and BaseFeatureGraphicsLayer. Features are added to a GraphicsLayer as Elements or, more specifically, as classes extending from com.esri.mo.map.dpy.BaseElement. Feature elements consist of one or more fields that contain the element's feature geometry and attributes. Feature geometry consists of components from the com.esri.mo.cs.geom.* package. A single GraphicsLayer can contain different element types (e.g., polygons, points, images). In addition, GraphicsLayer feature elements can be searched and selected using spatial or attribute queries. The code in Listing 2, available online, demonstrates the creation of a BaseGraphicsLayer.

The following characteristics must be considered when using this technique.

- Renderer components are set on individual Elements. The Symbol for all the elements in the GraphicsLayer can be set using the GraphicsLayer's setDefaultSymbol() and setDefaultSelectSymbol() methods.
- Associate the GraphicsLayer with an AcetateLayer before adding it to a Map component. No Legend will be added to a Map TOC if a GraphicsLayer is added to a Map as an AcetateLayer. Adding the GraphicsLayer directly to a Map's Layerset will create an unusable legend. The TOC will not be able to recognize the GraphicsLayer component, and an exception will be thrown when it attempts to set the visibility of the GraphicsLayer.
- The GraphicsLayer will use the offscreen buffer used by the Map component even though it has been added as an AcetateLayer. The sample code in Figure 2 adds a GraphicsLayer as an AcetateLayer to a Map.

Use a GraphicsLayer when the graphic features being added to a Map also have underlying attribute data but do not need to be managed as a layer in the Map's TOC. If you need to add map tips, data listeners,

```
Figure 2

graphicsLayer glayer = new graphicsLayer();
AcetateLayer aglayer = new AcetateLayer(glayer);
aMap.add(aglayer);
```

feature labeling, or complex rendering, use a FeatureLayer. On the other hand, if you want to add image data as an acetate layer, use a GraphicsLayer.

Create a FeatureLayer

The FeatureLayer interface is implemented by one instantiated subclass BaseFeatureLayer. A FeatureLayer is only capable of storing vector features of the same type. Creating a new FeatureLayer involves creating a feature class (MemoryFeatureClass), adding features and attributes, and setting the layer capabilities. A dynamic FeatureLayer will have the same properties as FeatureLayers using data located on disk (i.e., shapefile, ArcSDE layer). The code in Listing 3, available online, demonstrates the creation of a BaseFeatureLayer.

The following characteristics must be considered when using this technique.

- All FeatureLayers must set their layer capabilities; otherwise, no features will be visible.
- A dynamic FeatureLayer can be rendered using any applicable label or feature renderer in MOJ. The renderer can be applied in the class itself or where the BaseFeatureLayer is instantiated.
- By default, when a dynamic FeatureLayer is added to a Map component, a Legend is added to the TOC. The renderer defined for the FeatureLayer is used by the Legend. Figure 3 contains sample code that adds a FeatureLayer to a Map.

Use a FeatureLayer when the graphic features being added to a dynamic layer consist of points, lines, or polygons and require

```
Figure 3

featureLayer flayer = new featureLayer();
flayer.setVisible(true);
aMap.getLayerset().addLayer(flayer);
```

properties available only to FeatureLayers such as complex rendering or labeling, map tips, and data listeners. In addition, the Map and TOC components will manage the feature and legend visibility of the FeatureLayer. This method provides the most functional solution for maintaining graphic features and their attributes in an acetate layer.

All techniques listed above use virtual memory allocated to the Java Virtual Machine upon application run time. No caching mechanism has been implemented. As a result, memory usage will increase as additional graphic features are added to an acetate layer. Depending on the number and complexity of graphic features in a dynamic layer, you will need to adjust application code or memory parameters appropriately.

Conclusion

A number of techniques are available to dynamically add graphic features to a Map component. Consider the following scenarios when deciding which technique to use for your MOJ application:

- Use an AcetateLayer for simple features and text that do not contain any inherent attribute information.
- Use a GraphicsLayer when graphic features have underlying attribute data and no legend is required, or the dynamic layer references an image.
- Use a FeatureLayer when the dynamic layer contains vector-based graphic features that have underlying data and a legend is required or when complex rendering, map tips, labeling, and data listeners are needed.

Additional information on MapObjects—Java Edition is available from www.esri.com/mapobjectsjava.

Moving to ArcGIS

Access Online Maps With ArcReader

At version 8.3, ArcReader, the free map viewer application, provides a menu choice that lets users open any map in portable map format (PMF) from a Web site, metadata server, or the Geography Network. ArcReader 8.3 also supports multiple hyperlinks for a feature. Under Options, the hyperlink display graphics can be turned off. ArcReader is available in two versions. It is included as part of the standard ArcGIS Desktop 8.3 installation and a separate stand-alone version, for users who do not have ArcGIS, ships on CD–ROM with ArcGIS and is available as a download from www.esri.com/arcreader.
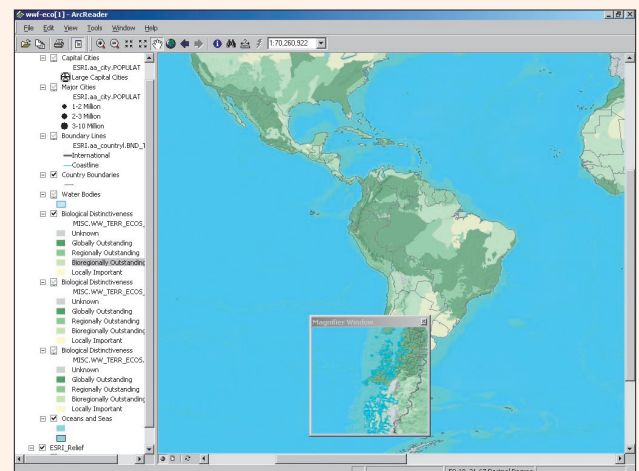
ArcReader 8.3, available as a stand-alone application or as part of ArcGIS, can access map files from Web sites.