Learning Video Series

Extending Map Suite

Lesson 3:
Exploring Layers

Explore what is possible by inheriting from Layer.

Duration: 45 minutes

ThinkGeo

Agenda

- Answer the following questions.
  - What is the difference between a Layer and an Overlay?
  - What does the Layer inheritance hierarchy look like?
  - Why would I inherit from Layer?
  - What is required for a custom Layer?
- Review the MultiGeoRasterLayer.
- Review the WatermarkAdornmentLayer.
- Review the MiniMapAdornmentLayer
- Review the MapShapesLayer
- Answer additional questions.

ThinkGeo

Differences between Layers and Overlays

- Layer
  - A member of the Core name-space
  - Intended to be easily used across Map Suite products
  - Linked one to one to a type of data
- Overlay
  - A member of a specific product
  - Designed to take advantage of platform capabilities
  - Can aggregate Layer
  - Used to facilitate rapid drawing and refreshing of Layers

What does the Layer inheritance hierarchy look like?

- Layer
  - This is what all other Layers inherit from
  - It has the fewest number of overload members
  - It provides the least amount of default functionality
  - Provides the highest degree of customization
- Inherited Classes
  - AdornmentLayer
  - FeatureLayer
  - RasterLayer

ThinkGeo

AdornmentLayer & FeatureLayer

- AdornmentLayer
  - Designed for things drawn in screen coordinates
  - Examples: ScaleBar, Logo, Legend, North Arrow
  - Implemented to not move when panning
  - Very similar to Layer
- FeatureLayer
  - Covered in our first Extending Map Suite video
  - Created to represent vector data
  - Includes query, edit and style capabilities

ThinkGeo

RasterLayer

- RasterLayer
  - Designed for drawing raster imagery
  - Examples: Mr Sid, ECW, GeoTiff, etc.
  - Designed to be drawing system neutral
  - Uses PNG stream in a GeoImage to transport data
  - Beware of some performance problems due to GeoImage
  - RasterLayer will be a topic for an upcoming video

Why inherit from Layer?

- Technically Standpoint
  - Provides the least number of overloads and dependencies
  - Quick to get up and running
- Requirements Standpoint
  - Aggregate raster and feature data
  - Best possible speed by avoiding the overhead of other Layers
  - Special requirements like Map Shapes
  - Produce something simple and straight forward
  - Draw in screen coordinates

What is required for a custom Layer?

- Required Overloads
  - DrawCore – This is where all of the drawing takes place
- Recommended Overloads
  - HasBoundingBoxCore – Indicates this layer has a bounding box
  - GetBoundingBoxCore – Returns the bounding box
  - OpenCore – Initialize expensive resources for the layer
  - CloseCore – Release expensive resource for the layer

Sample Code Overview

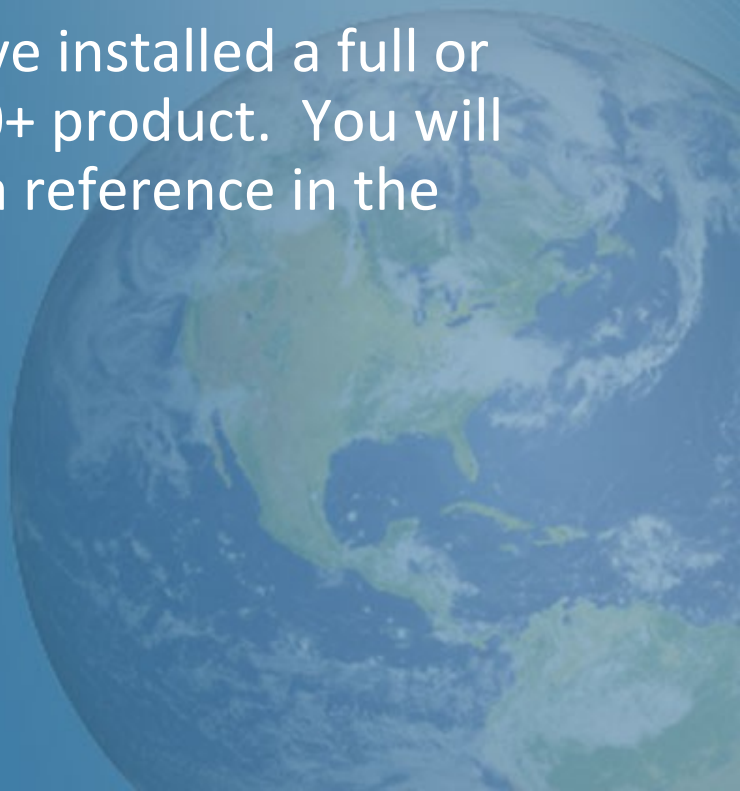
- **MultiGeoRasterLayer** - The first sample shows how we can aggregate a group of raster images into one layer. You can use this to handle hundreds or thousands of images and do not want to create the same number of layers.
- **WatermarkAdornmentLayer** - The second sample shows how we create a layer that draws watermarks on the map. This is handy for providing demos of your software to clients etc.
- **MiniMapAdornmentLayer** - The third example shows how easy it was to create a simple mini map layer to give you a reference of where you are when you zoomed in.
- **MapShapesLayer** - The fourth sample is a flashback to Map Suite 2.x and the concept of Map Shapes. Map Shapes are individual features that have their own styles and zoom levels.

ThinkGeo

Let's Look at Some Code!

You can find the code we will review in a zip file accompanying this video on ThinkGeo.com.

To compile the code it requires you have installed a full or evaluation edition of any Map Suite 3.0+ product. You will need to add the MapSuiteCore.DLL as a reference in the project.



ThinkGeo

Thank You For Watching!

For More Samples and Videos, Visit the Developers Blog Discussion Forum

ThinkGeo