



**Please!**

turn OFF cell phones  
and refrain from using  
flash photography



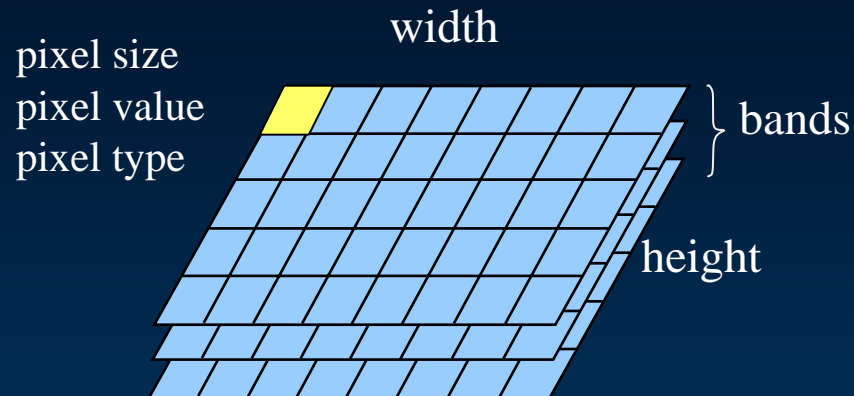
# Developing with the ArcGIS Raster API

*Peng Gao, Hong Xu, and Dan Meeks*

# Presentation Outline

- **Introduction of raster data model**
- **Accessing and displaying raster data**
- **Processing raster data**
- **Creating raster data and writing pixels**
- **Loading raster data to a geodatabase**
- **Questions/Comments**

# Characteristics of Raster Data

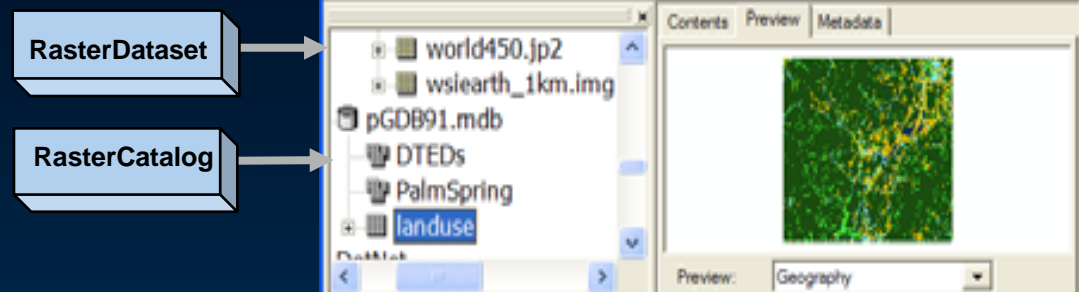


- **Stored in many formats**
  - Files
    - GRID, TIFF, IMG, JPEG, JP2000, BMP, GIF, PNG, BIL/BIP/BSQ, PCI, ECW, USGS DEM, BSB, PCRaster, HDF4, RST
  - Geodatabase
    - ArcSDE, Personal, File
- **Open structure for adding new raster format support**
  - **GDAL driver—Recommended**  
([www.remotesensing.org/gdal/gdal\\_drivertut.html](http://www.remotesensing.org/gdal/gdal_drivertut.html))
  - **Format DLL (edn.esri.com)**

# The ArcGIS Raster Data Model

- **Raster datasets**

- Pyramids
- Colormap
- Statistics
- Raster attribute table
- Spatial reference
- Geodata transformation
- Additional files to supplement the format storage
  - Auxiliary files (\*.aux or \*.aux.xml)
  - Pyramids (\*.rrd)

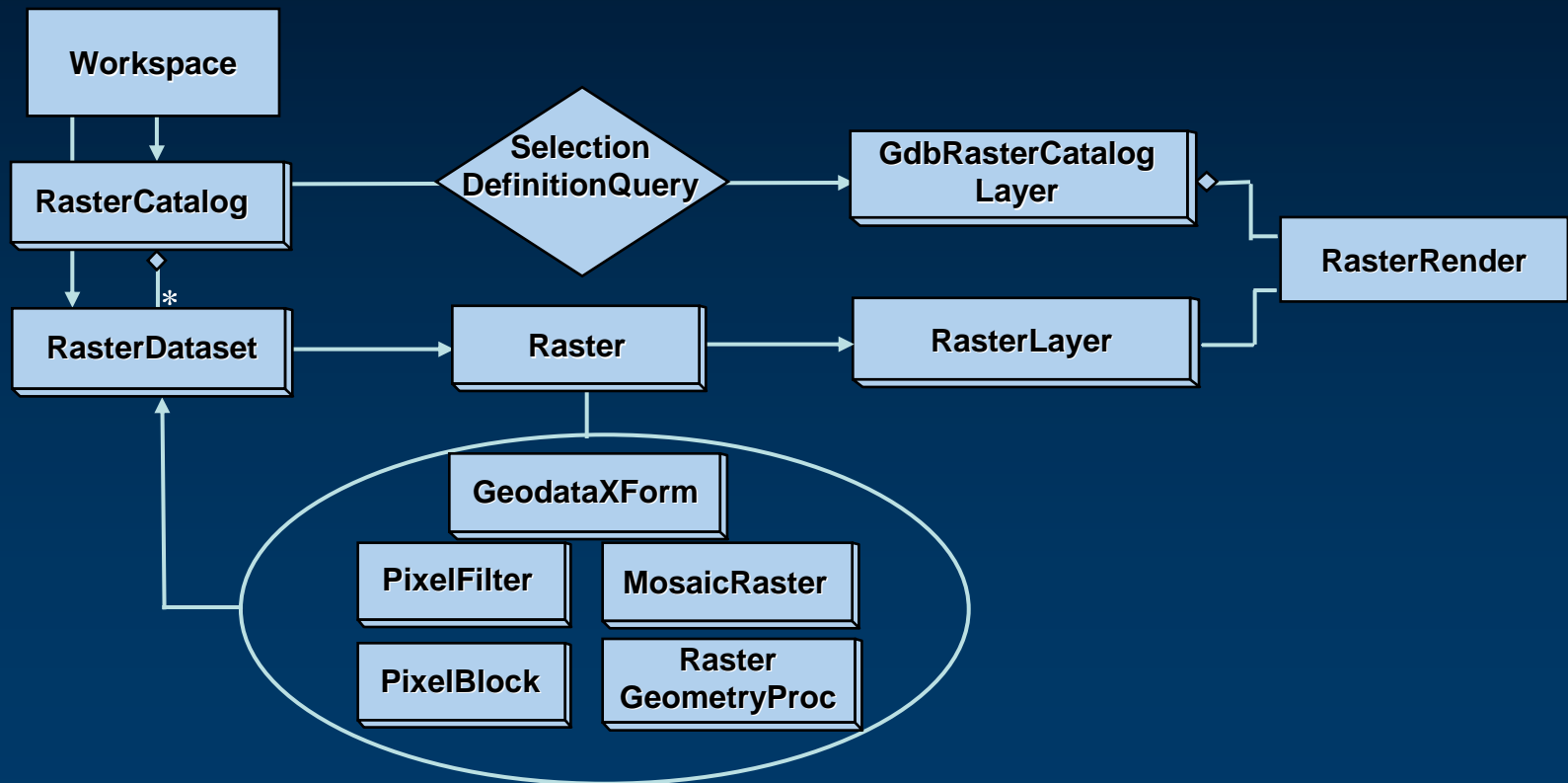


- **Raster catalogs**

- A collection of raster datasets
- Two Types
  - Managed—the Geodatabase controls the storage of the raster datasets
  - Unmanaged (Personal and File Geodatabases only)—the storage of the raster datasets is controlled by the user

# ArcGIS Raster API

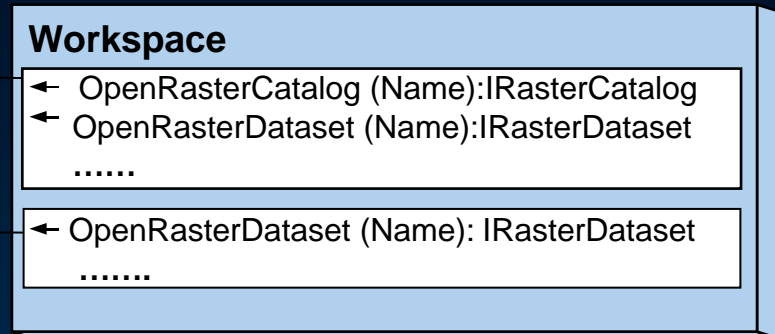
- Common API for accessing raster dataset and raster catalog



# Presentation Outline

- Introduction of raster data model
- **Accessing and displaying raster data**
- Processing raster data
- Creating raster data and writing pixels
- Loading raster data to a geodatabase
- Questions/Comments

# Opening a Raster Dataset or Raster Catalog



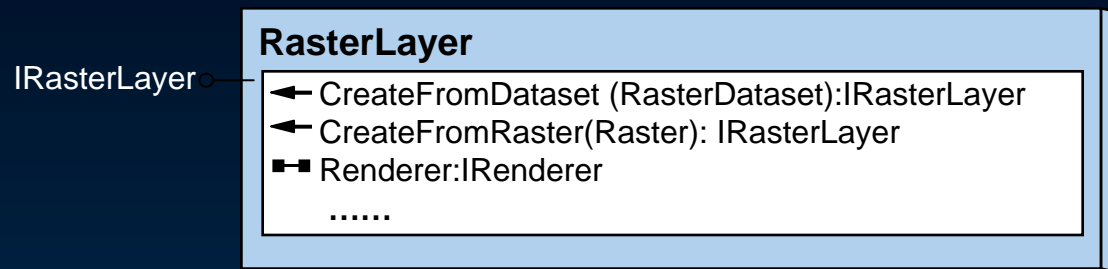
- **Open raster dataset**

```
IRasterWorkspace workspace;  
IWorkspaceFactory workspaceFactory = new RasterWorkspaceFactoryClass ();  
workspace = (IRasterWorkspace) workspaceFactory.OpenFromFile(@"c:\temp", 0);  
IRasterDataset rasterDataset = workspace.OpenRasterDataset("redlands.tif");
```

- **Open GDB raster dataset and raster catalog**

```
IRasterWorkspaceEx workspace;  
IWorkspaceFactory workspaceFactory = new AccessWorkspaceFactoryClass();  
workspace = (IRasterWorkspaceEx)  
workspaceFactory.OpenFromFile(@"c:\temp\data.mdb", 0);  
IRasterDataset rasterDataset = workspace.OpenRasterDataset("airphoto");  
IRasterCatalog rasterCatalog = workspace.OpenRasterCatalog("DRGs");
```

# Displaying a Raster Dataset

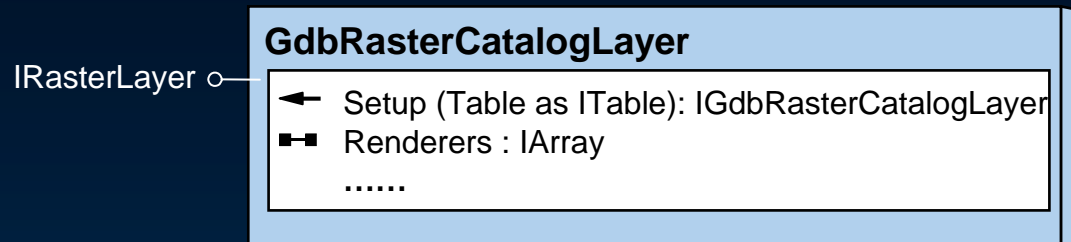


- Set a raster renderer or use the default

```
IRasterLayer rasterlayer = new RasterLayerClass();  
rasterlayer.CreateFromDataset rasterDataset;  
rasterlayer.Renderer = renderer;  
mapControl.FocusMap.AddLayer rasterlayer
```

- Raster Renderers
  - RGB, Stretch, Unique Values, Classify, Colormap
- Custom default renderer (RasterDefaultRendererMaker)

# Displaying a Raster Catalog



- **Set available renderers or use default**

```
IGdbRasterCatalogLayer catalogLayer = new GdbRasterCatalogLayerClass();  
catalogLayer.Setup(RasterCatalog);  
catalogLayer.Renderer = RendererArray;  
mapControl.FocusMap.AddLayer catalogLayer;
```

- **Default renderer is based on the individual raster datasets in the catalog**
- **RasterCatalogLayer is for Image Catalog—uses only one renderer for every raster dataset in the catalog**

# Demo

- **Open and display a raster dataset**

# Presentation Outline

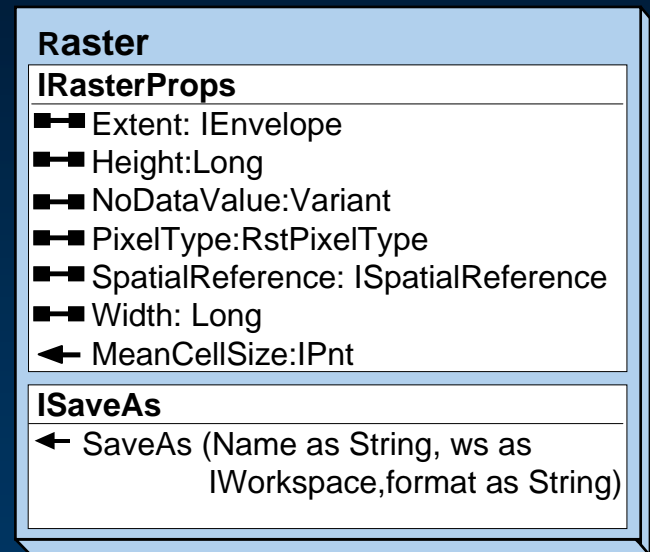
- Introduction of raster data model
- Accessing and displaying raster data
- **Processing raster data**
- Creating raster data and writing pixels
- Loading raster data to a geodatabase
- Questions/Comments

# Raster Datasets vs Rasters

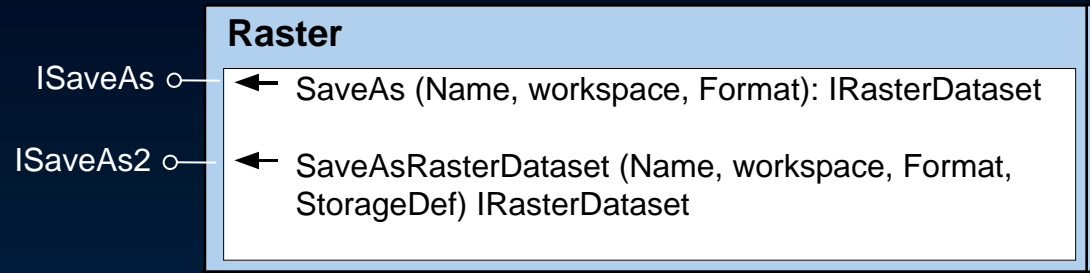
- **Raster Datasets represent files on disk**
  - Have a storage format and defined properties such as:
    - Compression type
    - A path to the dataset
    - Statistics
  - Can be copied and deleted
- **Rasters are in memory representations of raster datasets**
  - Allow access to bands of data
  - Allow access to the pixels
  - Characteristics can be modified

# Raster and its Properties

- **Create a Raster**
  - Interface and method used is determined by the raster dataset being worked with and the intended use
  - `IRasterDataset.CreateDefaultRaster`
  - `IRasterDataset2.CreateFullRaster`
- **Set properties with IRasterProps**
  - `SaveAs` to persist the change
- **Extent, width/height and cell size trilogy**
  - X cell size = extent in X / Width in pixels (columns)
  - Y cell size = extent in Y / Height in pixels (rows)
  - Cell size is derived

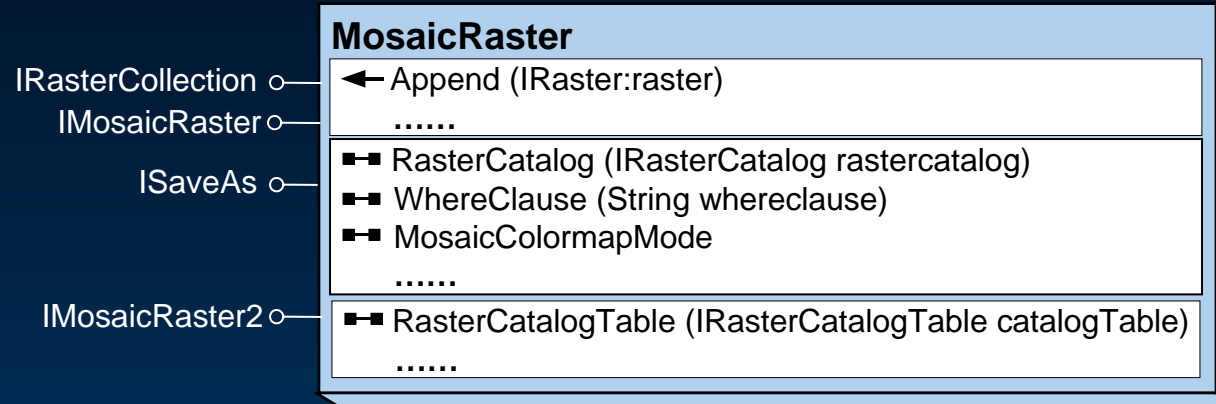


# Raster SaveAs



- **Support SaveAs to**
  - GRID, IMAGINE, TIFF and geodatabases
  - 9.2 added support for JPEG, JP2, PNG, BMP, GIF, PCI, RST, PCRaster, HDF4
- **Control storage using ISaveAs2**
  - Set compression type using `IRasterStorageDef2::CompressionType`
    - JPEG—JPG and GDB Rasters
    - JP2000—JP2 and GDB Rasters
    - LZ77—IMG and GDB Rasters
    - LZW and PackBits—TIFF only
  - Turn off pyramid building using `IRasterStorageDef2::PyramidLevel = 0`
- **ISaveAs is also supported by RasterDataset**
  - Mainly performs format conversion

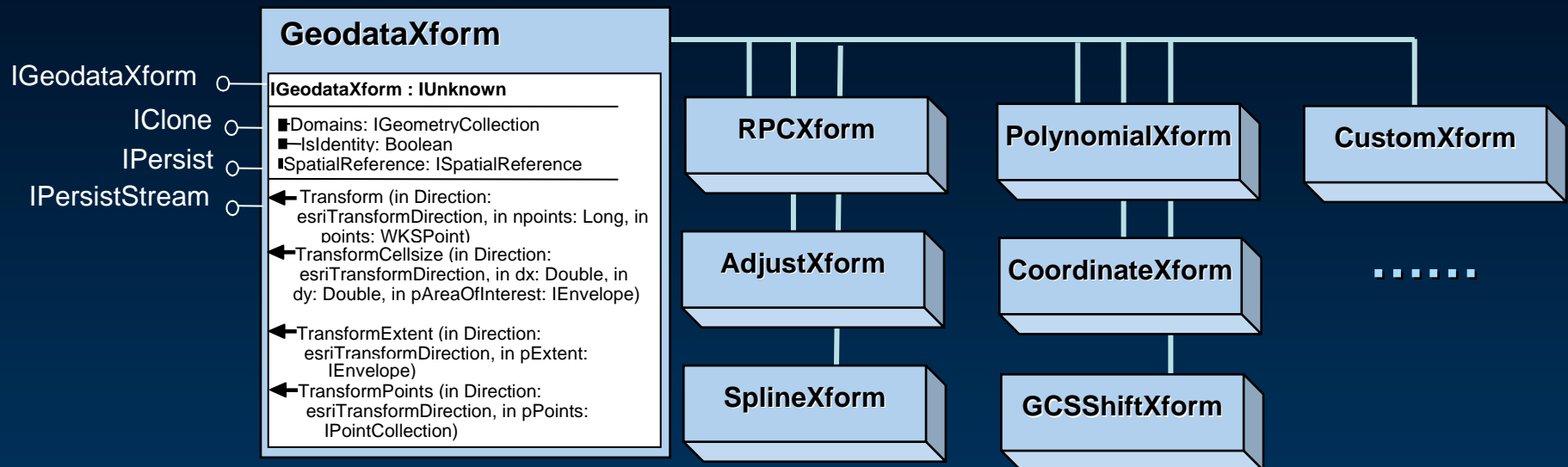
# Mosaicking Raster Data



- Using MosaicRaster to create new dataset
  - Mosaic from many raster datasets
  - Mosaic from RasterCatalog & WhereClause
  - Mosaic from RasterCatalogTable
  - ISaveAs to create output

```
IMosaicRaster mosaic = New MosaicRasterClass();  
Mosaic.RasterCatalog = rasterCatalog;  
ISaveAs saveas = (ISaveAs) Mosaic
```

# Geodata Transformations



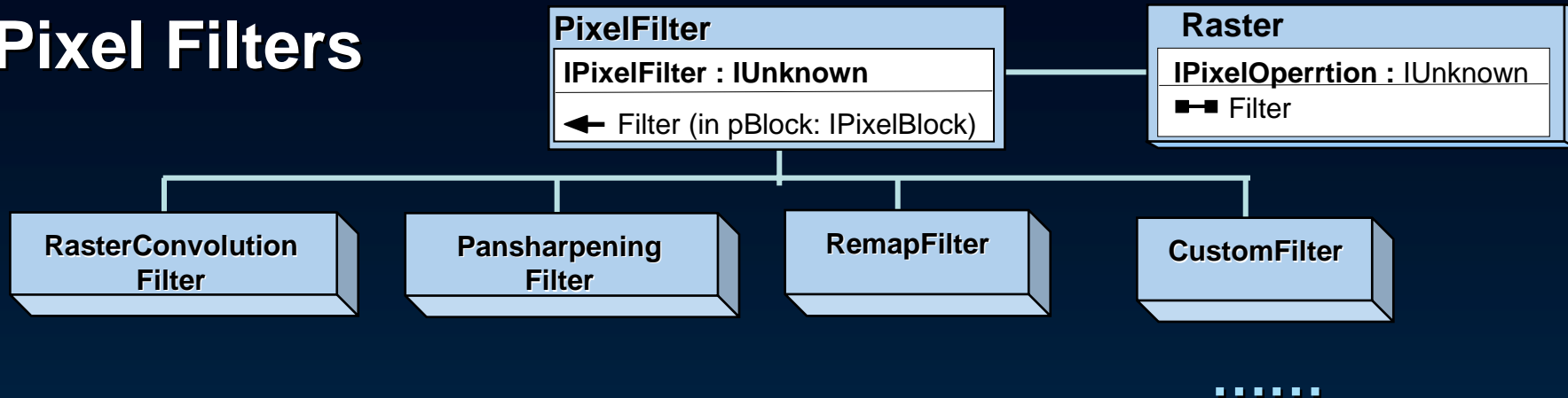
- **Support polynomial, Spline, and Adjust transformations**
  - Used for georeferencing
  - Replaces **IRasterGeometryProc** in ArcGIS 9.1
- **Raster reprojection with CoordinateXForm**
- **Image orthorectification with RPCXform**
- **Create your own Xform class for custom transformations**

# Performing Transformations



- **Persisted with dataset**
  - **IGeoDatasetSchemaEdit2::AlterGeoDataXform**
  - **\*.aux.xml** file for file based raster, internal for GDB raster
- **Apply to a Raster using IRaster2::GeodataXform**
  - Transform extent and cell size
  - Then set width/height
  - SaveAs

# Pixel Filters



- **ConvolutionFilter**
  - 3x3, 5x5, low pass, high pass, and etc.
  - Define your own kernel (PutCoefficients)
- **RemapFilter**
  - Map pixel values to a defined set of values
- **PanSharpeningFilter**
  - Mean, Brovey, IHS, and ESRI methods
- **Create your own PixelFilter class**
- **Apply to Raster using IPixelOperation::Filter**

$$\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$$

```
IPixelOperation pixelOp = (IPixelOperrtion)raster
pixelOp.Filter = myFilter
```

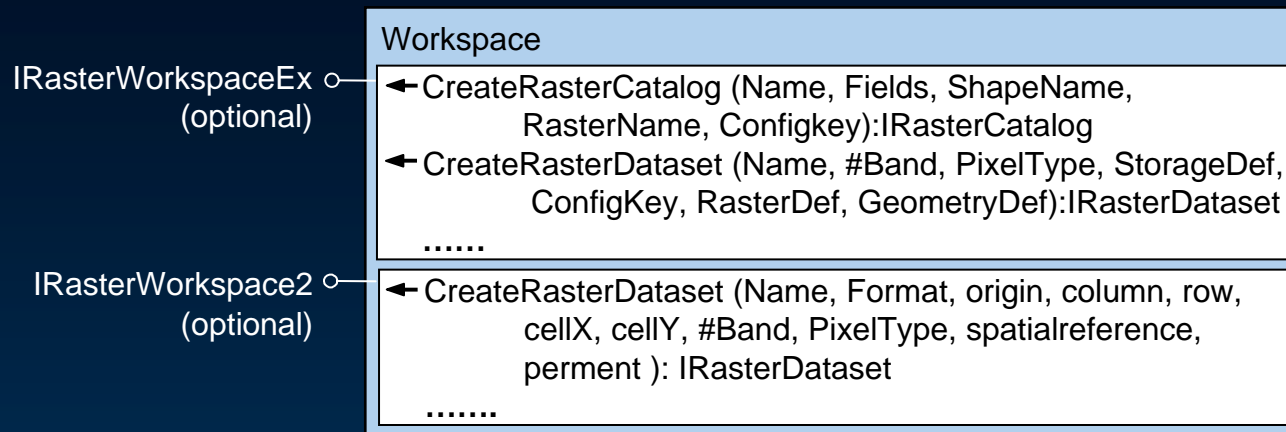
# Demo

- Clipping a raster catalog
- Creating and Applying a Custom Xform

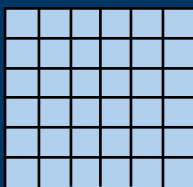
# Presentation Outline

- Introduction of raster data model
- Accessing and displaying raster data
- Processing raster data
- **Creating raster data and writing pixels**
- Loading raster data to a geodatabase
- Questions/Comments

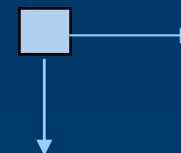
# Creating Raster Dataset



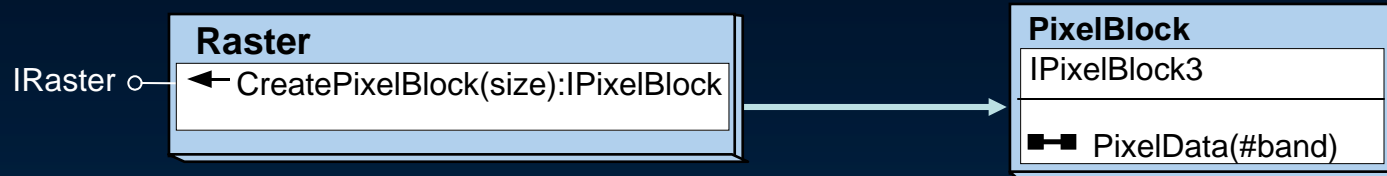
- Create using `IRasterWorkspace2`
  - Used for file based rasters
  - Specify a dimension and extent
  - Created with default pixel values



- Create using `IRasterWorkspaceEx`
  - Apply to GDB rasters
  - Empty, no defined dimensions or extents
  - Origin is defined for the pyramids only
  - Populate pixels through mosaic
  - Can set storage properties



# Working with PixelBlocks



- Allow low level access to pixel data
- Used for applying custom pixel-level processing to raster data accessed initially through ArcObjects or creating new rasters from processes

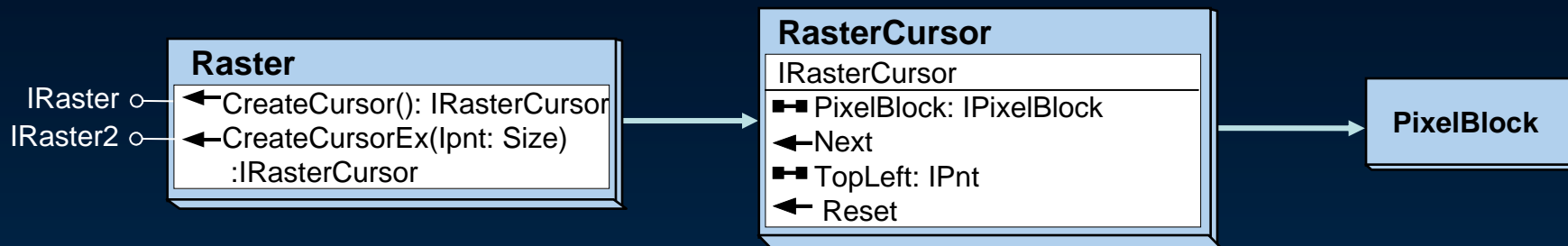
# Accessing PixelBlocks and returning the data to the raster

```
IRaster raster = rasterDataset.CreateDefaultRaster;
IRasterProps rasterProps = (IRasterProps)raster;
rasterProps.Extent = extent;
rasterProps.Width = 512;
rasterProps.Height = 512;

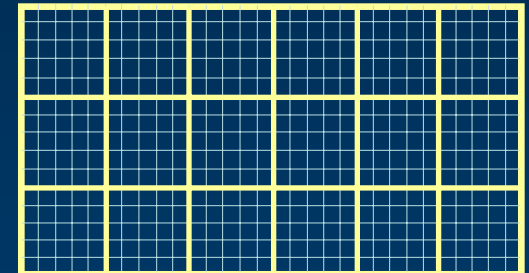
IPixelBlock3 pixelblock= (IPixelBlock3)raster.CreatePixelBlock(blockSize);
//Raster.Read(tlc, pixelblock);
System.Array pixelArray = (System.Array)pixelblock.get_PixelData(0);
pixelArray(i,j) = 10; // edit/modify pixel values

pixelblock.set_PixelData(0, pixelArray);
IRasterEdit rasterEdit = (IRasterEdit)raster;
rasterEdit.Write (tlcorner, pixelblock);
System.Runtime.InteropServices.Marshal.ReleaseComObject(rasterEdit);
```

# Using PixelBlock with RasterCursor



```
IRaster2 raster2 = (IRaster2 ) raster;
IRasterEdit rasterEdit = (IRasterEdit)raster;
IRasterCursor cursor = raster2.CreateCursorEx(blockSize);
IPixelBlock3 pixelBlock;
IPnt tlCorner;
do {
    pixelBlock = (IPixelBlock3)cursor.PixelBlock;
    //do something with the pixel block
    tlCorner = cursor.TopLeft;
    rasterEdit.Write(tlCorner, pixelblock);
} While (cursor.Next == true)
```



# In-memory Raster

- A raster format that stores pixels in memory
- Creating an In-memory raster
  - Use `IRasterWorkspace2::CreateRasterDataset`
  - empty file name and “MEM” as format string

```
IRasterWorkspace2 ws = OpenRasterWorkspace("c:\\temp")
IRasterDataset ds = ws .CreateRasterDataset("", "MEM",
    pOrigin, Width, Height, 30, 30, 1, PT_UCHAR, Nothing,
    True)
```

- Work as any other raster dataset
  - Can create layer and display
  - Used in GP tools as a raster layer

# Accessing and Exporting a Byte Array

- **Used to communicate with other applications**
- **Display a raster dataset in a window's form**
  - Export raster dataset to bytes
  - `IRasterExporter::ExportToBytes (....) as Byte()`
    - Supports BMP, PNG, GIF, or TIFF
- **Open a raster dataset from bytes directly**
  - `IRasterWorkspace3::OpenRasterDatasetFromBytes`

# HDF and JPEG access

- Access HDF4 subdatasets
  - IRasterDatasetJukebox

```
IRasterDatasetJukebox hdfDataset= IRasterDatasetJukebox)rasterDataset;  
hdfDataset.Subdataset = 2;  
IRasterDataset subDataset = (IRasterDataset)hdfDataset;
```

- Reading JPEG's Exif header
  - EXIF\_GPSLatitude, EXIF\_GPSLongitude, etc.
  - Use IDataset::PropertySet to get all the tags and the associated values

```
IDataset dataset = (IDataset)rasterDataset;  
  
//Get the EXIF tags and the associated values  
IPropertySet propertySet = dataset.PropertySet;  
System.Object tag_names;  
System.Object tag_values;  
propertySet.GetAllProperties(out tag_names, out tag_values);
```

# Demo

- **Creating a raster dataset**

# Presentation Outline

- Introduction of raster data model
- Accessing and displaying raster data
- Processing raster data
- Creating raster data and writing pixels
- **Loading raster data to a geodatabase**
- Questions/Comments

# Load Raster Dataset (Mosaic)

IRasterWorkspaceEx

← CreateRasterDataset (Name, #Band, PixelType, StorageDef, ConfigKey, RasterDef, GeometryDef):IRasterDataset

- Create an empty raster dataset
  - Define spatial reference with IRasterDef
  - Storage parameter with IRasterStorageDef
    - Pyramid reference point
    - Pyramid level and resampling method
    - Compression

```
IRasterStorageDef rasterStorageDef = new RasterStorageDefClass();  
//set pyramid origin  
IPoint origin = new PointClass();  
origin.PutCoords(-180, 90);  
rasterStorageDef.Origin = origin;  
  
//set compression to JPEG 25  
rasterStorageDef.CompressionType = esriRasterCompressionType.esriRasterCompressionJPEG;  
rasterStorageDef.CompressionQuality = 25;  
  
//set spatial reference to GCS_WGS_1984  
IRasterDef rasterDef = new RasterDefClass();  
rasterDef.SpatialReference = CreateSpatialReference(4326);
```

# Load Raster Dataset (continued)

- Mosaic raster datasets
  - Resolve overlapping area
  - Process background
  - Shift/resample if cells are not aligned

## IRasterLoader

- □ Background::Variant
- □ Foreground::Variant
- □ MosaicColormap::rstMosaicColormapMode
- □ PixelAlignmentTolerance::Double
- ← Load (out rasterDataset, in rasterArray)
- ← LoadRasters (out rasterDataset, in raster)

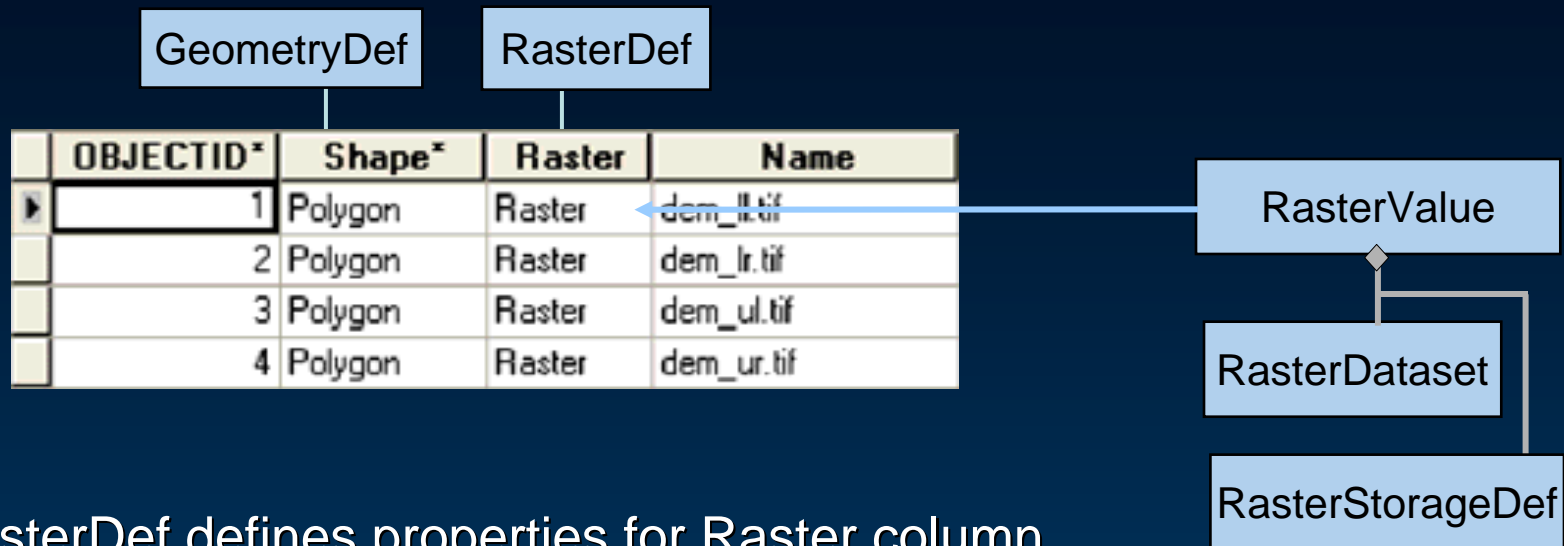
```
void MosaicRaster(IRasterDataset dataset, IRaster inputRaster)
{
    IRasterLoader rasterLoader = new RasterLoaderClass();

    //set background to ignore
    rasterLoader.Background = 0;

    //set tolerance so that it shifts if there is a cell misalignment
    rasterLoader.PixelAlignmentTolerance = 0.5;

    //mosaic
    rasterLoader.Load(dataset, inputRaster);
}
```

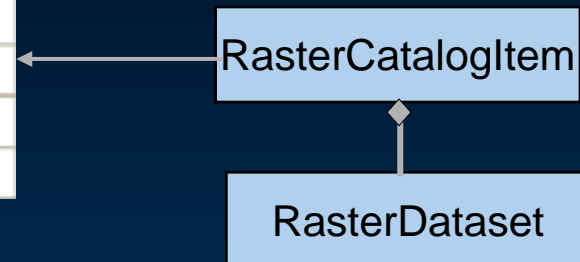
# Raster Catalog



- IRasterDef defines properties for Raster column
  - Spatial reference, management type
  - raster values can persist its own spatial reference
- IGeometryDef defines properties for Shape field
  - Spatial reference, spatial index grids
- IRasterStorageDef is used to set storage properties
  - Compression, pyramid building, pyramid origin, tile size, etc.

# Access Raster Dataset in Raster Catalog

	OBJECTID*	Shape*	Raster	Name
▶	1	Polygon	Raster	dem_ll.tif
	2	Polygon	Raster	dem_lr.tif
	3	Polygon	Raster	dem_ul.tif
	4	Polygon	Raster	dem_ur.tif



```
IRasterDataset GetRasterCatalogItem(IRasterCatalog catalog, int oid)
{
    IFeatureClass featureClass = (IFeatureClass)catalog;
    IRasterCatalogItem rasterCatalogItem = (IRasterCatalogItem) _
        featureClass.GetFeature(oid);
    return rasterCatalogItem.RasterDataset;
}
```

# Create Raster Catalog

IRasterWorkspaceEx

← CreateRasterCatalog (Name, Fields, ShapeName, RasterName, Configkey):IRasterCatalog

- IRasterWorkspaceEx::CreateRasterCatalog
  - Specify properties for raster field
  - Specify properties for geometry field
  - Supports adding any other fields

```
//shape field
IGeometryDef geometryDef = new GeometryDefClass();
IGeometryDefEdit geometryDefEdit = (IGeometryDefEdit)geometryDef;
geometryDefEdit.SpatialReference_2 = CreateSpatialReference(4326);

fieldEdit.Name_2 = "SHAPE";
fieldEdit.Type_2 = esriFieldType.esriFieldTypeGeometry;
fieldEdit.GeometryDef_2 = geometryDef;

//raster field
IField2 field2 = new FieldClass();
IFieldEdit2 field2Edit = (IFieldEdit2)field2;

IRasterDef rasterDef = new RasterDefClass();
rasterDef.SpatialReference = CreateSpatialReference(29714);
rasterDef.IsManaged = true;

field2Edit.Name_2 = "RASTER";
field2Edit.Type_2 = esriFieldType.esriFieldTypeRaster;
field2Edit.RasterDef = rasterDef;
```

# Load Raster Catalog

- Basic loading
  - Similar to inserting feature to a featureclass
  - Create a Feature
  - Set raster field with IRasterValue
  - Specify storage parameter with IRasterStorageDef

```
void LoadRasterCatalog(IRasterCatalog catalog, IRasterDataset dataset)
{
    IFeatureClass featureClass = (IFeatureClass)catalog;
    IFeature feature = featureClass.CreateFeature();

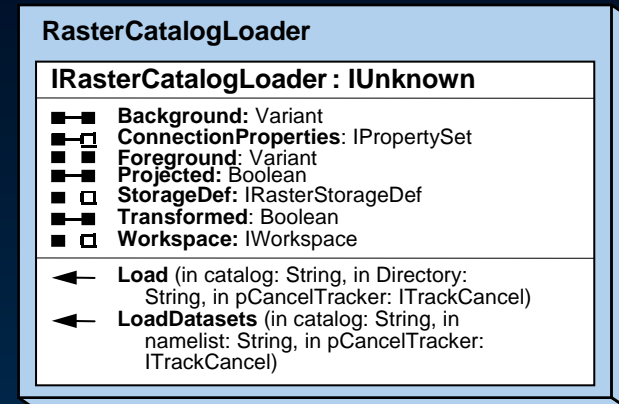
    //create and set raster value
    IRasterValue rasterValue = new RasterValueClass();
    rasterValue.RasterDataset = dataset;
    feature.set_Value(catalog.RasterFieldIndex, rasterValue);

    feature.Store();
}
```

# Load Raster Catalog (continued)

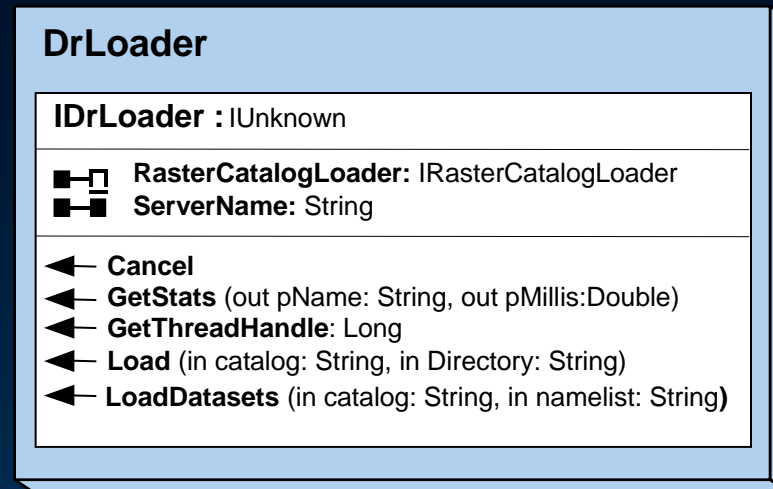
- Batch loading

- Load from a directory
- Control projection on-the-fly
- Control geodata transformation
- Specify storage parameters for all the rasters



```
IRasterCatalogLoader catalogLoader = new RasterCatalogLoaderClass();  
//set workspace where raster catalog resides  
catalogLoader.Workspace = workspace;  
  
//maintain spatial reference of the input raster  
catalogLoader.Projected = false;  
  
//set storage parameter  
catalogLoader.StorageDef = storageDef;  
  
//load  
catalogLoader.Load(catalogName, folder, null);
```

# Load Raster Catalog (continued)



- Distributed loading
  - Designed for an ArcGIS server environment
  - Parallel loading using available server containers
  - Load a directory or a list of rasters
  - Only for Raster Catalogs, Raster Datasets cannot be populated with parallel processes

# Demo

- **Load raster with a background**
- **Load raster and apply Remap filter**

# Resources

- DataSourcesRaster library overview
  - A lot of new content in 9.2
- Samples/Raster
  - <http://edndoc.esri.com/arcobjects>
  - Many DotNet samples available at 9.2 release

# Session Evaluations Reminder

Session Attendees:

Please turn in your session evaluations.

*. . . Thank you*

**Questions/Comments ?**

# Tech Talk Map

