



# Using the ArcGIS Engine Controls in .NET

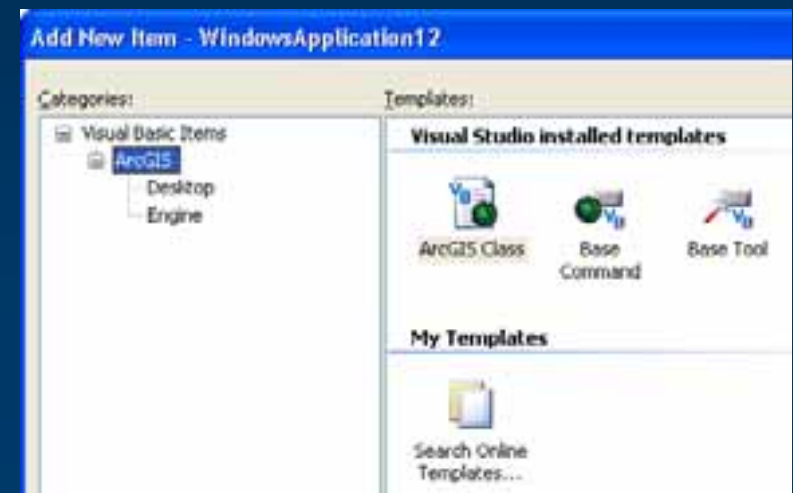
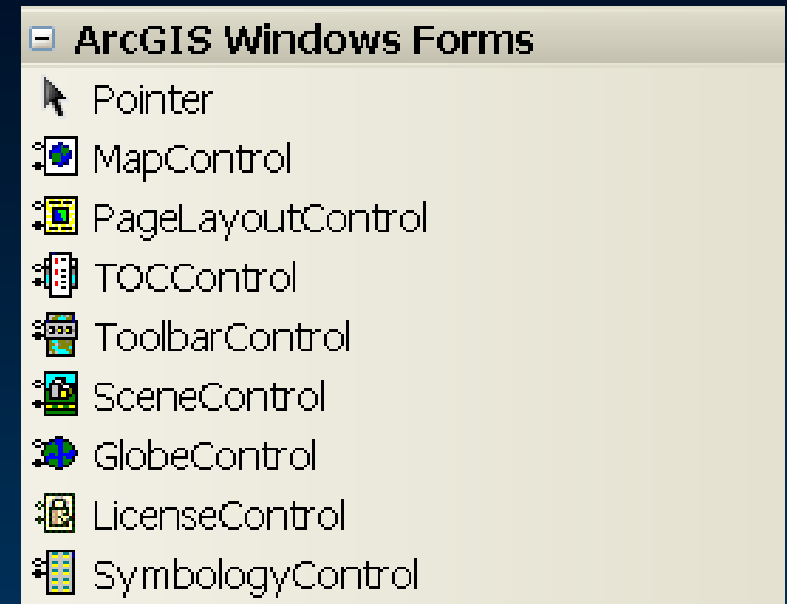
*Mike Rudden and Mary Harvey*

# Overview

- **Extend with custom components**
- **Persisting Application State**
- **Tips & Tricks**

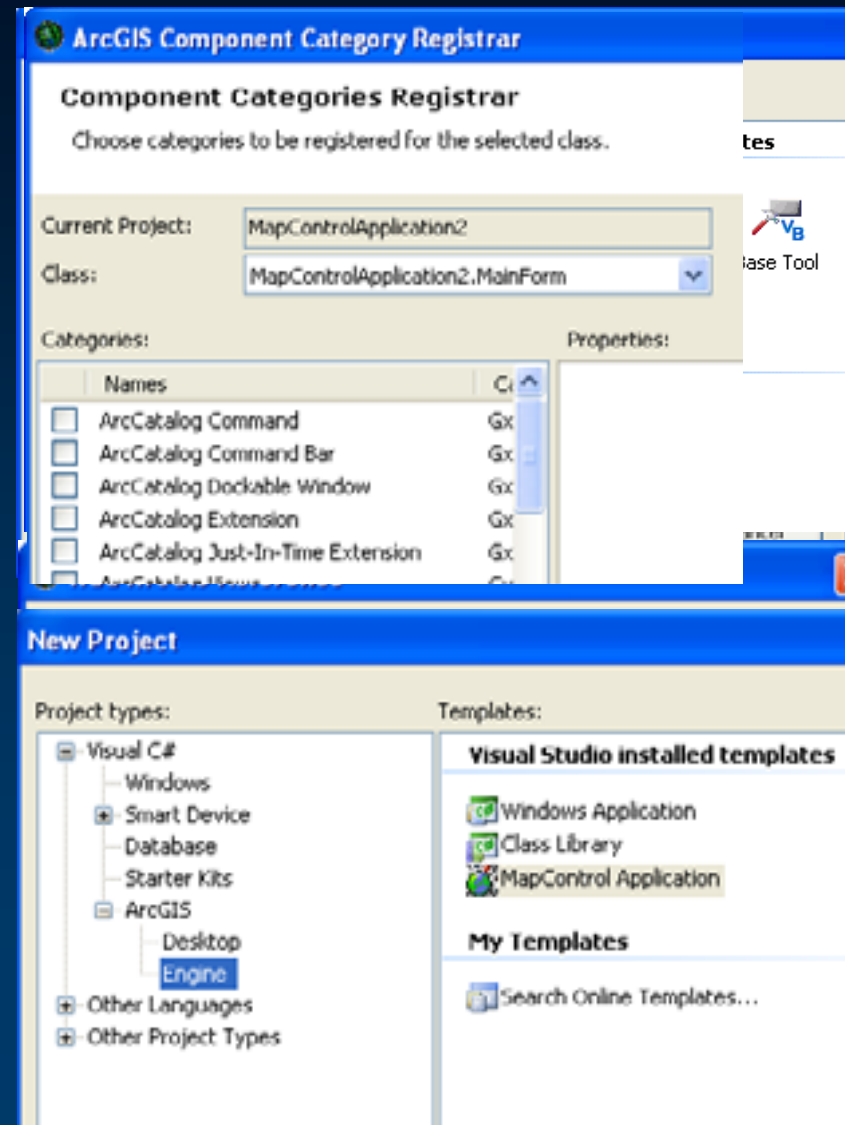
# ArcGIS Engine SDK Overview

- APIs
  - COM, **.NET**, Java and C++
- Components
  - Controls library: **2D Controls**, **3D Controls**, **Framework Controls**, **ControlsCommands**
  - ArcObjects libraries: **Carto**, **Geodatabase**, **Geometry...**
- Tools
  - Documentation
  - **VS2005 Integration**



# ArcGIS Engine 9.2 Visual Studio 2005 Integration

- **Solution / Project Level**
  - ArcGIS Code Converter
  - Project Templates
- **Project Level**
  - Add Class Wizard
  - Add ArcGIS Reference
  - ArcGIS Toolbox Reference
- **Class Level**
  - Component Category Registrar
  - ArcGIS License Initializer
  - Snippets



# ArcGIS Engine 9.2 Licensing

- You must ‘explicitly’ license an application:
  - Using the “ArcGIS License Initializer” wizard
  - Using the LicenseControl
  - N.B. ‘Shutdown the application’ checkbox



Demo

# Extending ArcGIS Engine

- **The framework can be extended by writing custom COM components:**
  - **Commands / Tools**
  - **Menus / Palettes / Pop-up Menus**
  - **Extensions**

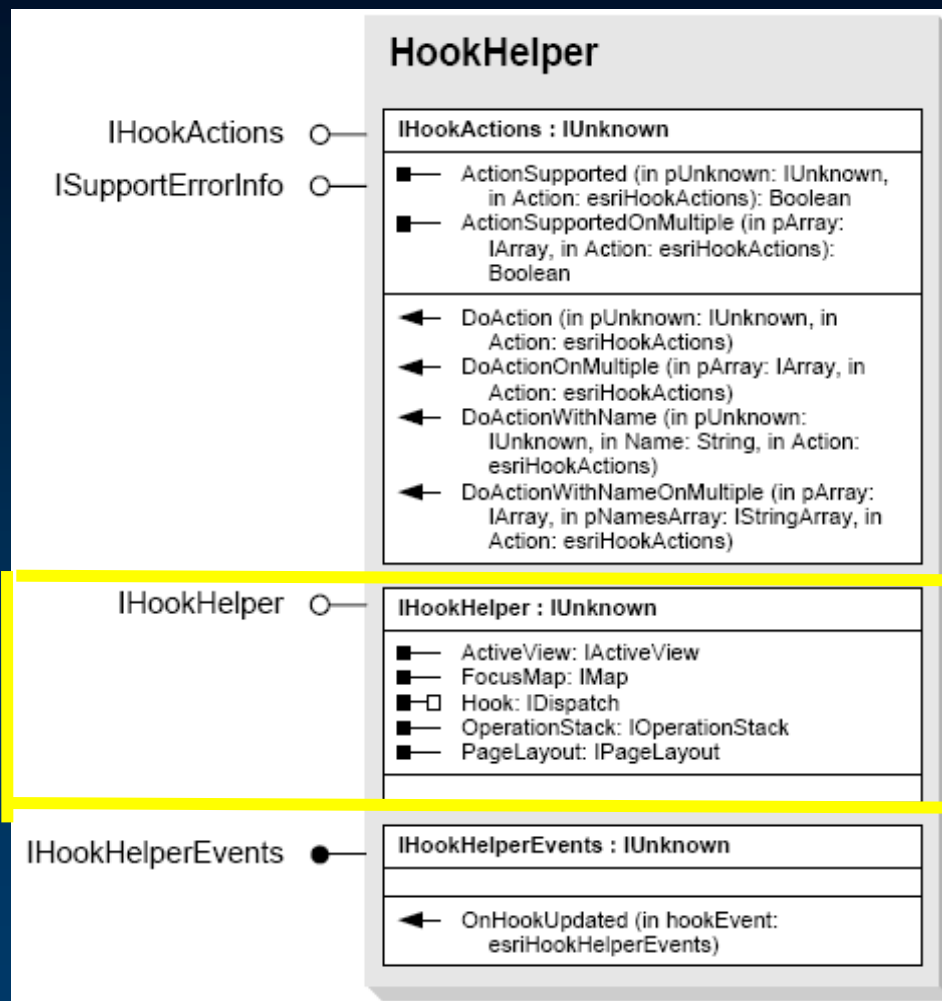
# Implementing Custom COM Components

1. Create a .NET project
  2. Create a class
  3. Reference the appropriate assemblies
  4. Implement an interface
  5. Compile and register DLL
  6. Register with a component category
- Base classes provided in `ESRI.ArcGIS.ADF.BaseClasses`
    - `BaseCommand`, `BaseTool`, `BaseMenu...`

```
public override void OnCreate(object hook)
{
    if (hook == null)
        return;
```

# The HookHelper Class

- The hook may be many different objects:
  - MapControl, ToolbarControl, MxApplication...
- HookHelper takes care of the type of hook
- IHookHelper provides useful properties..



# HookHelperClass

```
public override void OnCreate(object hook)
{
    if (m_hookHelper == null)
        m_hookHelper = new HookHelperClass();

    m_hookHelper.Hook = hook;

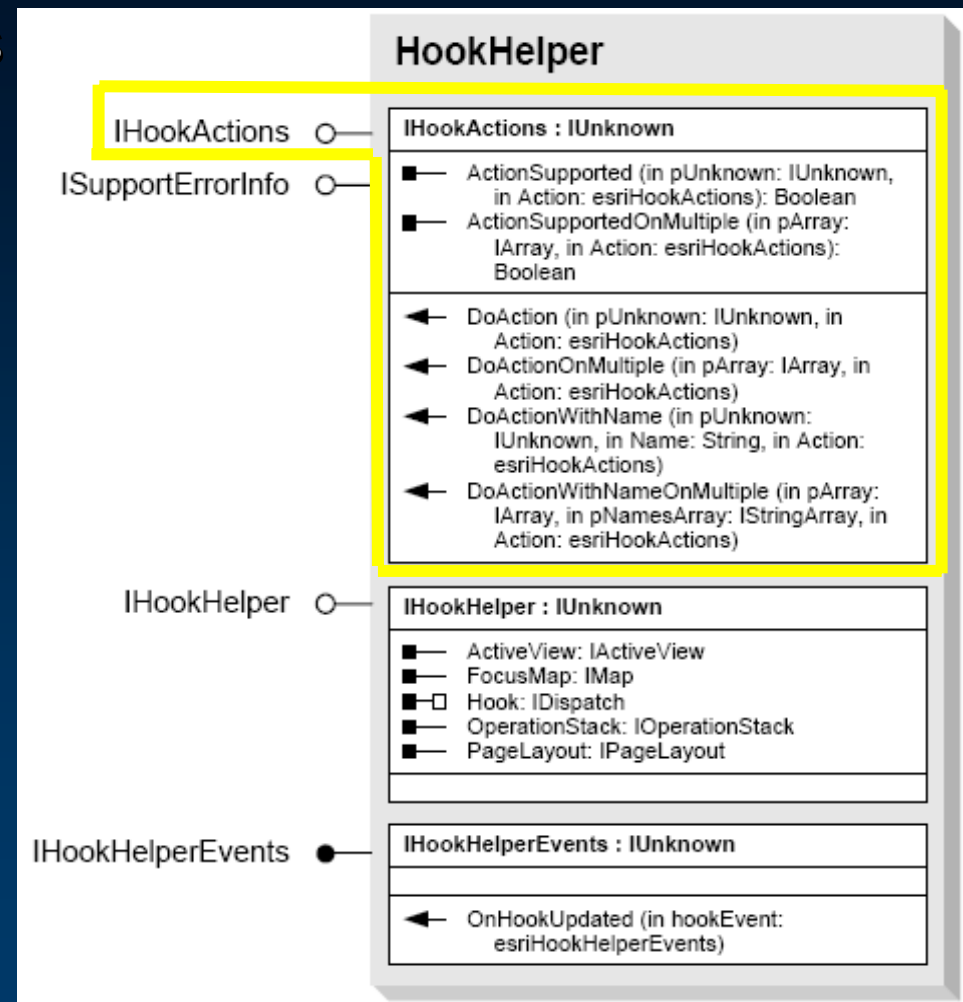
    m_hookHelper.ActiveView.....
```

- Also GlobeHookHelper...

**Demo**

# HookActions

- Performs common actions on simple geometries.
  - Is action supported on geometry
  - Do action on geometry using esriHookActions constants:
    - esriHookActionsFlash
    - esriHookActionsPan
    - esriHookActionsZoom ...
- GlobeHookHelper also implements IHookActions



Demo

# Creating ToolbarMenus

- **Types**

- Add to **ToolbarControl**
- **Popup menus**
- **Context menu to work with TOCControl**
- **Sub menus**

- **Steps**

- **Create ToolbarMenu**
- **Add Commands or Tools or IMenuDef**
- **Share CommandPool**

# Creating ToolbarPalettes

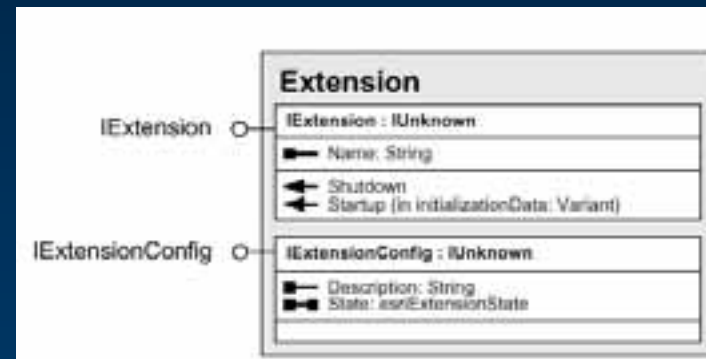
- **Types**
  - Add to ToolbarControl
  - Popup palette
- **Steps**
  - Create ToolbarPalette
  - Add Commands or Tools or IPaletteDef
  - Share CommandPool
- **3 out-of-the-box palettes provided:**
  - Eraser
  - Pen
  - Highlighters



**Demo**

# Implementing COM extensions

- Extensions can be used to share data between software components
- Interfaces: IExtension, IExtensionConfig
- Type Library: esriSystem
- State of commands is controlled by state of extension



```
Public Class ZoomExtension
    Implements IExtension
    Implements IExtensionConfig
```

**Demo**

# Runtime Customization & Persistence

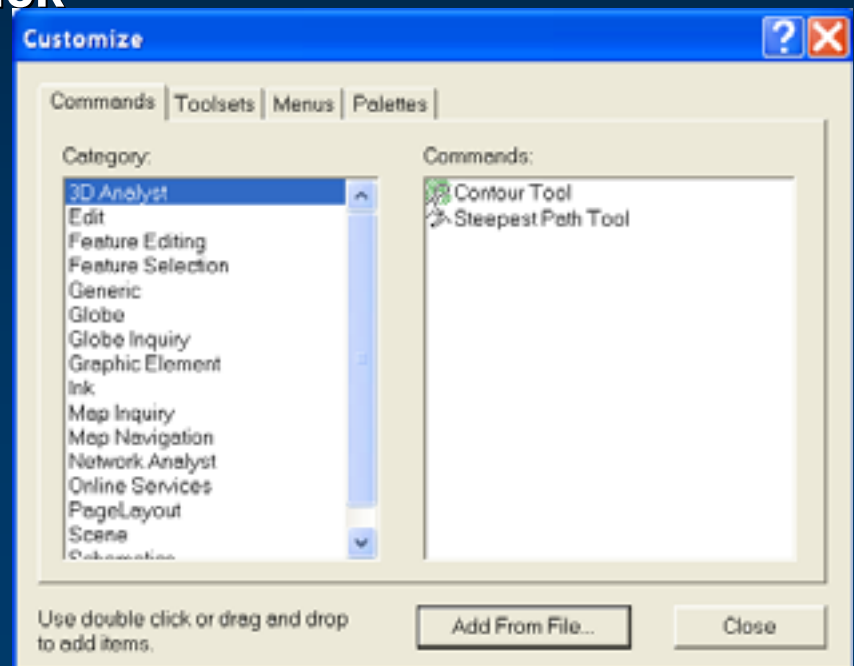
- **ToolbarControl CustomizeDialog**
- **Persistence**
  - Saving the ToolbarControl state
  - Saving MapDocuments
  - Saving LayerFiles

# The CustomizeDialog

- **Modeless dialog that allows you to:**
  - Add and remove commands from the **ToolBarControl** at runtime
  - Browse existing **ESRI** commands, tools, menus, palettes and toolbars
  - Drag and drop items or double-click

- **Need to enable programmatically**
- **Set ToolBarControl into Customize mode**

**Demo**



# Persisting Customizations

- Saving and loading ToolbarControl contents

- Writing map documents & layerfiles

– Open Document:

```
m_MapDocument = new MapDocumentClass();  
m_MapDocument.Open(sFilePath, "");
```

– Save Document:

```
m_MapDocument.Save(m_MapDocument.UsesRelativePaths, true)
```

**Demo**

# Tips and Tricks

- **Synchronizing MapControl and PageLayoutControl**
- **DPI Issue – 96 v 120 DPI**
- **Dynamic Display**
- **Singletons**
- **Geoprocessing**
- **Migrating Existing applications to 9.2**



# DPI Considerations

**Workaround: Controls display issue in 96 vs. 120 dpi**

**Summary**  
ArcGIS Engine Controls do not resize correctly when the dots per inch (DPI) of the screen display is changed. This effects both the design time and run time appearance of the ArcGIS Engine Controls. For example, an ArcGIS Engine Controls application developed on a machine at 96 dpi, when deployed to a machine at 120 dpi will have incorrectly resized controls and vice versa. This topic discusses the problem and provides two possible workarounds to help mitigate the issue.

Development licensing	Deployment licensing
Engine Developer Kit	Engine Runtime
	ArcView
	ArcEditor
	ArcInfo

The Microsoft .NET Framework 2.0 provides [Automatic Scaling](#) for Windows Forms to ensure that all embedded controls are correctly resized irrespective of the screen display resolution or system font.

There is a bug in the .NET Framework Automatic Scaling functionality that does not correctly resize ActiveX controls embedded on .NET Windows Forms when the resolution is changed. All of the ArcGIS Engine Controls are affected by this.

The illustration below shows a .NET application containing a form designed at 96 dpi and embedded with 2 PictureBox controls and 2 simple ActiveX controls all sized at 192\*192 pixels.

# Dynamic Display

- **Supports High Performance Drawing in 2D**
  - Leverages OpenGL ( No experience required )
  - ArcObjects API for drawing to Dynamic Display
- **Developers Create ‘Dynamic’ Layers**
- **Map handles the drawing of Dynamic Layers**
- **Why use Dynamic Display?**
  - Useful for drawing many objects on the display quickly
  - Improved performance

**Demo**

# Singletons

- Only ever 1 instance of a singleton per thread
- CommandsEnvironment
- EngineEditor
- EngineInkEnvironment
- EngineNetworkAnalystEnvironment
- MyPlaceCollection

**Demo**

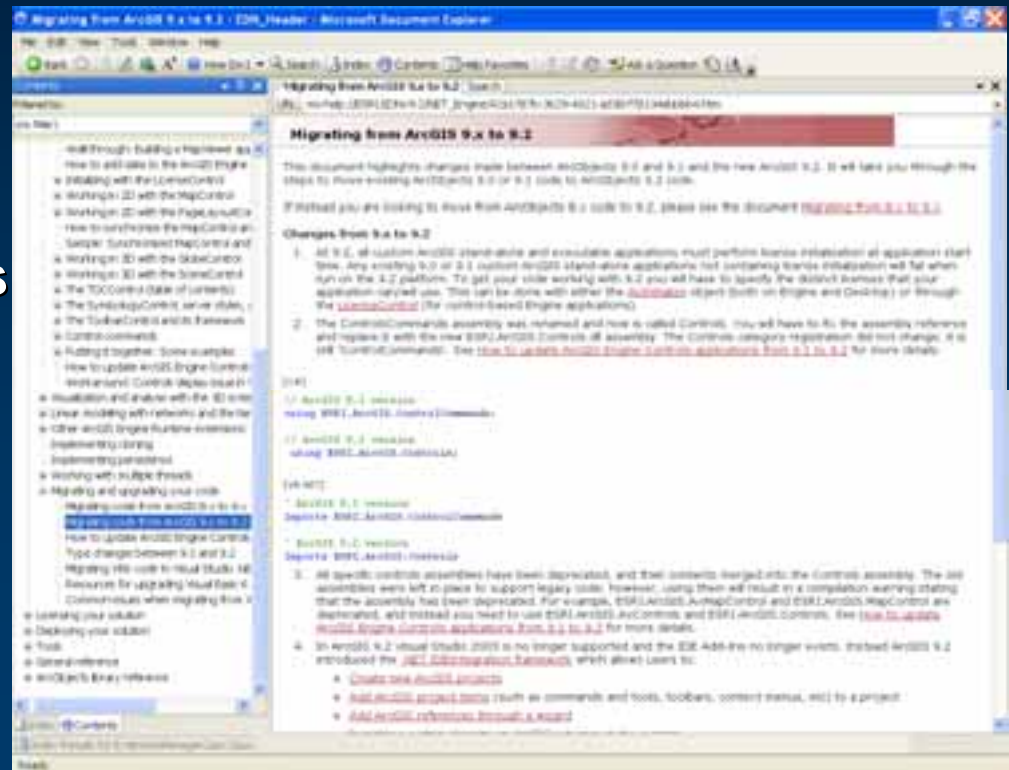
# Geoprocessing

- **ArcGIS 9.2 provides a ESRI.ArcGIS.Geoprocessor assembly**
  - Contains a managed Geoprocessor class
  - Used to execute GP tools
- **Each system Geoprocessing toolbox has its own managed assembly**
  - Contains classes representing each GP tool in the toolbox
- **Use the Geoprocessor class to set up and execute the GP tools**

**Demo**

# Migrating Existing Applications to 9.2

- **New Controls library**
  - esriControls.olb
- **New Primary Interop Assemblies (PIA's)**
  - ESRI.ArcGIS.AxControls
  - ESRI.ArcGIS.Controls
- **Contains**
  - All ArcGIS Engine Controls
  - All out of box commands
  - menus
  - palettes
  - multi-items



# Summary

- **Extend with custom components**
- **Persistence**
- **Tips & Tricks**



# Questions?

*Mary Harvey and Mike Rudden*

# Other Sessions

**4:30pm – 5.45pm Tuesday**

- **Leveraging the Geoprocessing Framework in ArcGIS Engine in .NET**

**1:00pm – 2:15pm Wednesday**

- **Creating Editing Applications with ArcGIS Engine in .NET**

**2:45pm – 4.00pm Wednesday**

- **ArcGIS Engine and ArcGIS Desktop Panel Discussion**

**4:30pm – 5:45pm Wednesday**

- **Network Analysis in ArcGIS Engine and ArcGIS Desktop**