Agenda

• Introduction to Engine
  • Controls
  • SDK Requirements
• Runtime Binding
• Authoring content for ArcGIS Engine
• Adding functionality to ArcGIS Engine
• ArcGIS Engine and ArcGIS Runtime
What is ArcGIS Engine?
What is ArcGIS Engine?

• Product used to build custom standalone GIS solutions
  - ArcObjects
  - ArcGIS Engine Controls
    - Components that can be added to windows forms
    - Add mapping and supporting GIS functionality to your custom application
    - 8 Controls
    - Over 200 built in commands

• Leverage the power of ArcGIS Desktop and Server
  - Use Desktop to author content
  - Consume Maps, Models, Geodatabases, Layers, Layer Packages, Query Layers, Services, and more...
What is needed to Develop .NET Engine Applications?

With 10.1

- Visual Studio 2010
- VS Express 2010
- .NET Framework – 3.5 sp1

- ArcGIS Requirements
  - ArcGIS Engine Runtime
  - ArcObjects .NET SDK
  - ArcGIS Engine Developer Kit License
What is needed to Develop Engine Applications?

With 10.2

- Visual Studio 2012
- Visual Studio 2010
- Visual Studio 2010 Express
- .NET Framework – 3.5 sp1

- ArcGIS Requirements
  - ArcGIS Engine Runtime
  - ArcObjects .NET SDK
  - ArcGIS Engine Developer Kit License
.NET Framework

- .NET Framework 3.5 is min requirement
  - Guaranteed to be installed with Engine or Desktop

- .NET Framework 4.0 is fully supported
  - Need to handle distribution
    - Desktop developer Add-ins install without Admin requirements?

- Do NOT embed iterop types
64 Bit Support

• ArcGIS Engine is a 32 bit application
  - Run as a 32 bit application on 64 bit OS
  - Set platform to x86 in Visual Studio Configuration Manager
  - Default is “Any CPU”

• At Version 10 Engine applications are Large Address Aware
  - On 64 Bit OS 32 Bit processes can take up to 4 GB of RAM (if available)

• When compiling VS 2010 applications on a x64 machine
  - Follow KB 37879
Runtime Binding and Licensing
Runtime Binding

What is Runtime Binding?

- **Starting at ArcGIS 10 each product has it’s own runtime**
  - Products have separate install locations
  - Service pack products separately
  - Uninstall service packs
- **ArcObjects must be pointed to a runtime to work**
  - Before any other ArcObjects calls
  - Required for all standalone applications
  - Engine applications can bind to either Desktop or Engine Runtimes
Runtime Binding

How to bind to a runtime

• Add reference to:
  ESRI.ArcGIS.Version

• Bind using the RuntimeManager static class

ESRI.ArcGIS.RuntimeManager.Bind(ESRI.ArcGIS.ProductCode.Engine)
Runtime Binding

When to bind to a runtime?

- **Before any other ArcObjects calls**
  - Preferably in your main method (C#)
  - Preferably in your application events StartUp method (VB)

- **Tip:**
  - The Assembly is called ESRI.ArcGIS.Version
  - The Namespace is ESRI.ArcGIS
  - Bind method returns a Boolean that you can use to handle binding errors
Runtime Binding

Additional Functionality and uses

- **RuntimeManager.BindLicense**
  - Bind and License with one method

- **RuntimeCollection**
  - Identify Installed Runtimes

- **RuntimeInfo**
  - Identify the Path, Product, & Version

- **ActiveRuntime**
  - Currently bound runtime
Licensing

- Engine applications must check out a license at runtime
  - Either Desktop or Engine License
  - Product Licenses are checked out for the life of the application
  - Extensions can be checked out and returned as needed
- At ArcGIS 10 Engine concurrent Engine licenses are available
- Licensing is not the same as binding
  - Binding specifies the runtime
  - Licensing specifies product functionality
Licensing

- Engine
- Geodatabase Update
- ArcView / Basic
- ArcEditor / Standard
- ArcInfo / Advanced

Single Use  Concurrent Use
Controls and Binding Demo
Binding and Deployment

- ArcGIS Engine is not required on the target machine
  - Engine applications work with either a Desktop or Engine Runtime

- Can mix and match Licensing and runtime for flexible Engine applications
  - Leverage existing Desktop licenses and runtime on clients machine
Content for ArcGIS Engine
Basemap and RasterBasemap Layers

What are Basemap Layers

- Layers that draw with a high-performance multi-threaded drawing engine
  - Improves display speed and responsiveness of your map
  - Seamless display

- Limitations
  - Layers in a Basemap Layer have less functionality
    - Cannot modify layer properties
  - Some data sources, layer types, and symbols are not supported
Using Basemap Layers in Engine

- Best to author through ArcMap
  - Make use of the analyzers for optimization
- For optimal performance
  - Test to see if you can enable hardware acceleration
    - IGlobalScreenDisplaysettings.CanEnableHardwareAcceleration
  - Enable hardware acceleration
    - IGlobalScreenDisplaysettings.EnableHardwareAcceleration
Working with Packages

What are packages?

- Packages are a single file that contains a map or layer[s] and supporting data
  - Also can contain references to SDE data
- Easy to share
  - Single file
- ArcGIS 10.0 supports
  - Layer Packages (*.lpk)
  - Map Packages (*.mpk)
- ArcGIS 10.1 adds additional support for
  - Geoprocessing Packages (*.gpk)
  - Locator Packages (*.apk)
  - Tile Packages (*.tpk)
Working with Packages

**Using Packages in Engine**

- At ArcGIS 10 sp1
  - Programmatically consume packages
- IMapDocument.Open
  - Map Packages
  - Layer Packages
  - Web Maps
- ILayerFile.Open
  - Layer Packages
- Just point to the path of the package
Working with Packages

Using Packages in Engine

• Opening Layer Packages

```vbnet
Dim layerFile As ILayerFile = New LayerFileClass
layerFile.Open("c:\Data\LayerPackages\USCities.lpk")
Dim layer As ILayer = layerFile.Layer
axMapControl1.AddLayer(layer)
```

• Opening Map Packages

```vbnet
Dim mapDocument As IMapDocument = New MapDocumentClass
mapDocument.Open("c:\Data\LayerPackages\MyMapPackage.mpk", "")
axMapControl1.Map = mapDocument.get_Map(0)
```
Working with Packages

Using Online Content

• Consume data on ArcGIS Online
  - Pass in a URL with the id as the filename
    http://www.arcgis.com/home/item.html?id=a50c645f83bb4f5dbddd457df29d639
Working with Packages

Using IPackageFile at 10.1

- Add reference to:
  `ESRI.ArcGIS.EngineCore`

- Unpack using `IPackageFile.Unpack`

```csharp
private void unPack(string packagePath, string unpackToLocation)
{
    IPackageFile packageFile = new PackageFile();
    IStringArray stringArray = packageFile.Unpack(packagePath, ref unpackToLocation);
    axMapControl1.AddLayerFromFile(stringArray.GetStringElement(0));
}
```
Working with Packages

Why use Packages in Engine?

• Easily deploy maps, data, and functionality with your Engine solution
  - Simple deployment of a single file
  - Easy to update off cycle

• Packages can be uploaded to ArcGIS Online
  - Once a package is downloaded it can be used locally
  - Use ArcGIS Online groups to manages access
Adding Functionality to ArcGIS Engine
Adding Functionality to Engine

- Leverage existing commands and tools included with the Engine SDK
  - On a ToolbarControl
  - Programmatically
- Build your own components
  - Such as Commands, Tools, Extensions, Custom Layers, etc…
  - Specific to your Engine Application
  - Generic for all Engine applications
  - Work in both Engine and Desktop applications
Using Geoprocessing in Engine

- Why use geoprocessing in Engine?
  - Why reinvent the wheel?
  - ArcGIS comes with hundreds of Geoprocessing tools
  - Developed by specialists in their field

- Provides a framework to author Model and Script tools in ArcGIS Desktop
  - Consume these tools in Engine
Using Geoprocessing in Engine

- Using system tools

```vbnet
Imports ESRI.ArcGIS.Geoprocessor
‘System Toolboxes have their own Assembly
Imports ESRI.ArcGIS.AnalysisTools

Dim gp As Geoprocessor = New Geoprocessor

‘Create the clip tool
Dim clipTool As Clip = New Clip
clipTool.in_features = “C:\Data\Test.gdb\InFeatures”
clipTool.clip_features = “C:\Data\Test.gdb\ClipFeatures”
clipTool.out_feature_class = “C:\Data\Test.gdb\ResultFeatures”

‘Execute the Tool
gp.Execute(clipTool, Nothing)
```
Using Geoprocessing in Engine

- Using custom tools

```vbnet
Imports ESRI.ArcGIS.Geoprocessor

Dim gp As Geoprocessor = New Geoprocessor

' Add the toolbox
gp.AddToolbox("C:\Data\MyToolbox.tbx")

' Populate the parameters
Dim parameters As IVariantArray = New VarArray
parameters.Add("C:\Data\Test.gdb\InFeatures")
parameters.Add("C:\Data\Test.gdb\ProcessFeatures")
parameters.Add("C:\Data\Test.gdb\ResultsFeatures")

' Execute the Tool
gp.Execute("MyTool", parameters, Nothing)
```
Using Background Geoprocessing in Engine

What is Background Geoprocessing

- Framework to allow Geoprocessing tools to execute in a separate process
- Great alternative to multi-threaded applications
- Allows User Interface to remain responsive while processing
Background Geoprocessing

What is Background Geoprocessing

• Execute tool using ExecuteAsync method on the Geoprocessor object
• Wire in the events
  - ToolExecuted
    - Must handle to know when tools complete
  - ProgressChanged
  - MessagesCreated
  - ToolExecuting
• Run system tools, models, and script tools
Using Geoprocessing in Engine
Background Geoprocessing

Tips for working with the Geoprocessor

- **Tips:**
  - Set OverwriteOutput = True
  - Tools require different license levels or extensions
  - Desktop help is your friend
  - Understand GP Messaging
  - Learn about the Result Object
New Runtime for ArcGIS
New Runtime for ArcGIS

- New Architecture
- Easy to Deploy Solutions
  - Software, maps and data
- 64 bit Native Support
- Modern API
  - WPF and Java
- Reduced Memory and Disk Footprint
New Runtime for ArcGIS

- Not a functional equivalent to Engine
- Not an ArcObjects replacement
- New APIs
  - Code will need to be re-written
  - Should leverage general patterns for data sharing and distribution
Engine/Runtime

Please don’t forget to fill out the surveys
Questions?

Please don’t forget to fill out the surveys