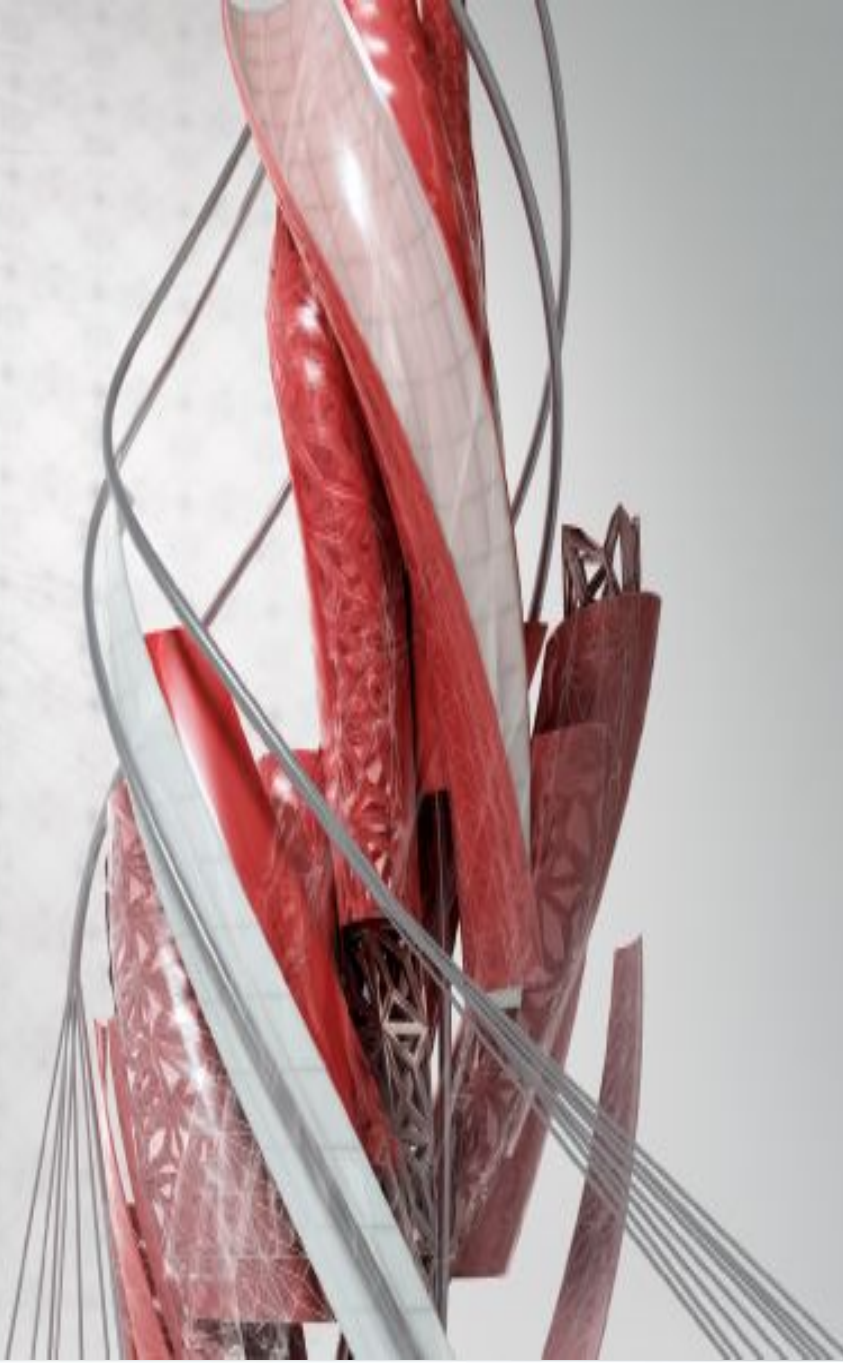
AutoCAD 2014 .NET API

Developer Technical Services



AUTODESK.

Getting Acquainted

- The Developer Technical Services Group

- About You?

Developer Technical Services

- Worldwide Workgroup
  - Over 25 Specialists World Wide
  - Virtually 24 hour support, 5 days a week
- Americas Team
  - US (CA, AZ, WA), Canada, Brazil
- European Team
  - Switzerland, United Kingdom, France, Czech Republic, Russia
- APac Team
  - China, Japan, India

Getting Support

- http://www.autodesk.com/adn-devhelp
  - Provides access to
    - On-line knowledge base
    - Call submission
    - Newsgroups
  - Calls are logged automatically
    - 1-3 day turnaround
    - Callbacks as needed
  - Answers to frequently asked questions are posted in our on-line knowledge base

Course Objective

- It is to understand:
  - the fundamentals of the AutoCAD .NET API
  - how to teach yourself the AutoCAD .NET API
  - where to get help afterwards

- What it is not:
  - Teach you .NET framework or C# , VB programming language
  - Give you complete coverage of all API functions

Class Agenda

- Lectures with Labs
  - Slides give an abstract overview
  - Labs and discussion give a practical perspective

- Lectures
  - Overview of .NET.
  - AutoCAD .NET Visual Studio project settings – Hello World!
  - User Interaction - User Input and Entity Selection
  - Database Fundamentals – Symbol tables, Transactions
  - User Interface design – Win Form Dialogs and Palettes
  - Event handling – Reacting to AutoCAD Events in .NET. Database
  - Dictionaries, XRecords, Table Traversal
  - Point Monitor
  - Jigs
  - Additional User Interface Elements (Non CUI related)

Class Schedule

Time 9:30 - 5:30

- Lunch 12:00 - 1:00

Day 1

- Overview of .NET
- Visual Studio Project Settings – Lab 1
- User Input – Lab 2
- Creating Entity, Block definition and Block References – Lab 3
- UI Design – Win form Dialogs, Palettes and Event Handling – Lab 4

Day 2

- Dictionaries – Lab 5
- Point Monitor – Lab 6
- Jigs - Lab 7
- Non-CUI related UI elements – Lab 8

.NET Overview

- What is .NET?
- Benefits of programming in .NET
- Important Concepts

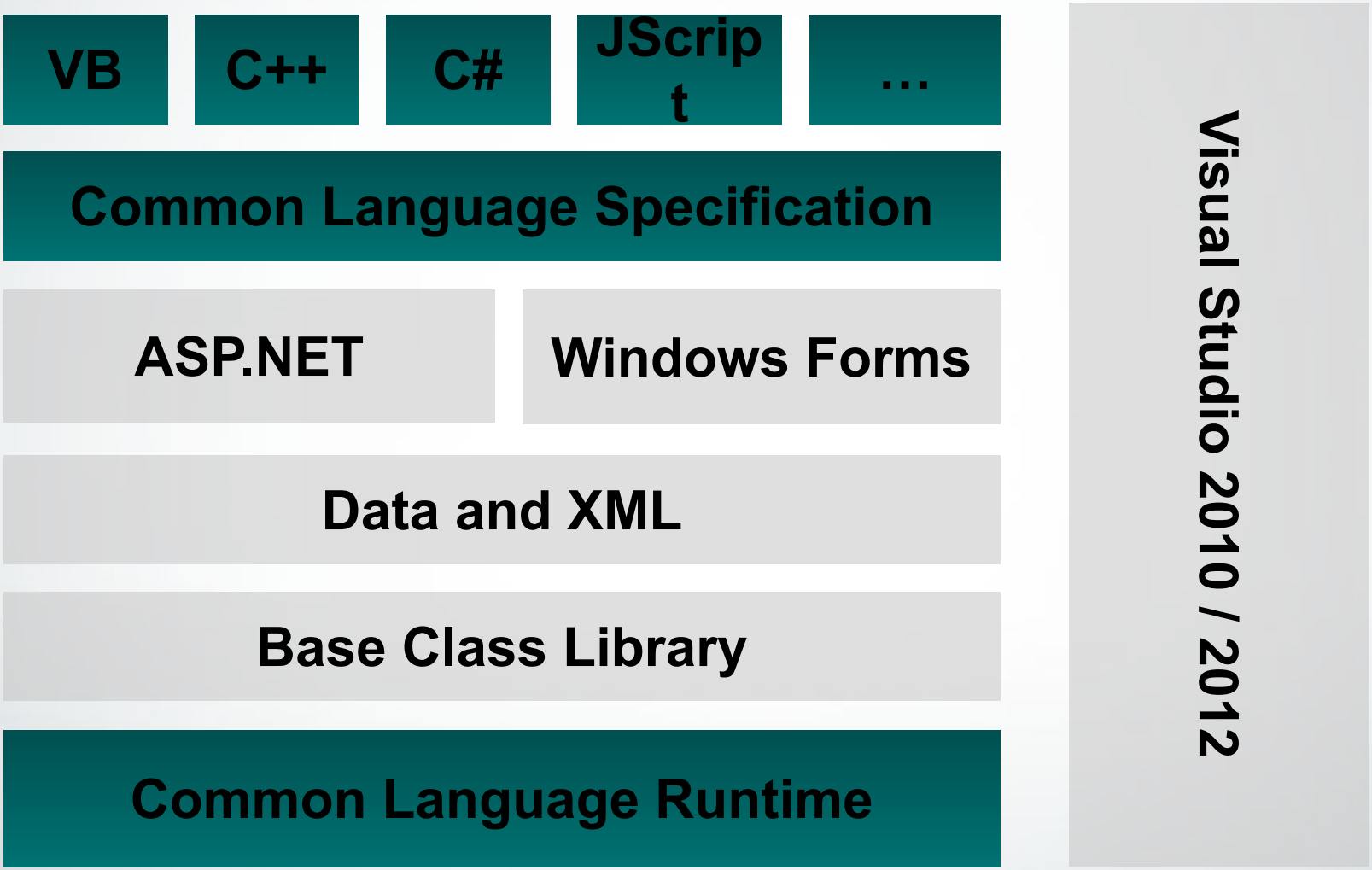
.NET Overview

- What is .NET?
  - Microsoft's Technology of a Web based infrastructure
    - Seamless interaction between applications and the Internet
    - Access information across anytime, anywhere from any device

.NET Overview

- What is .NET?
- Components of .NET
  - The .NET Framework used for building and running all kinds of software, including Web-based applications, smart client applications, and XML Web Services
  - Developer tools such as Microsoft Visual Studio 2008 / 2010 / 2012
  - A set of servers that integrate, run, operate, and manage Web services and Web-based applications
  - Client software that helps developers deliver a deep and compelling user experience across a family of devices and existing products.

.NET Framework



.NET Overview

- What is .NET?

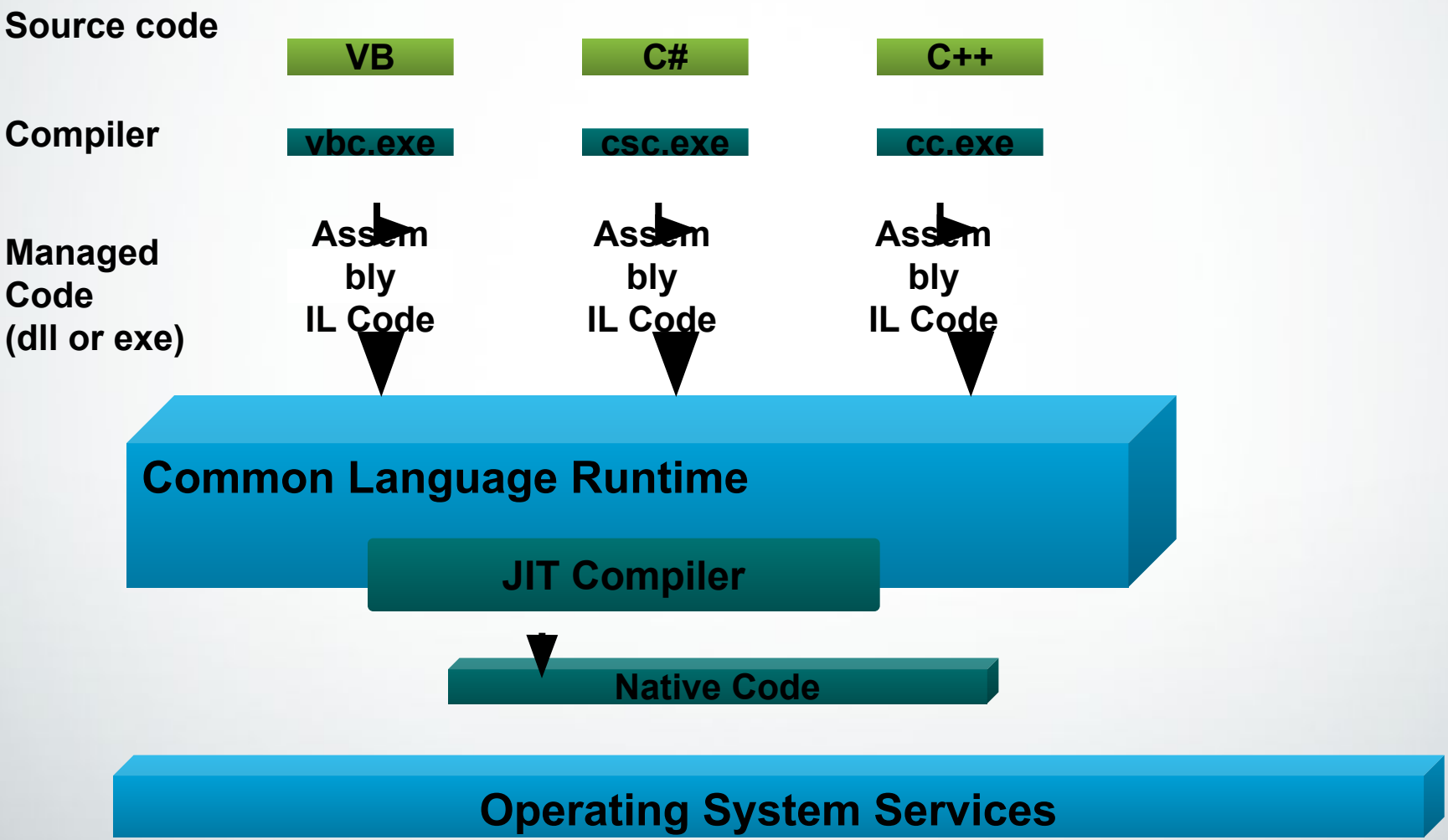
- NET Framework
  - Common Language Runtime (CLR)
    - Object-Oriented programming environment
    - Common execution environment for .NET applications

    -Similar to Java VM – but with much stronger interoperability

  - Framework Class Library (FCL)
    - Object Oriented Collection of re-usable types

    (Source: MSDN)

CLR Execution Model



What is Microsoft .NET?

What we know from our experience so far…

- Intelligent symbolic representation
  - Mature language constructs
    - Collections, Events, Delegates
  - Common programming pitfalls addressed
    - Memory management, consistent Exception handling, unified strings
- Source and binary inter-module communication
  - goes beyond C++ and COM
    - Meta data allows design- and run-time object usage and extension
- Programming style
  - Multiple supported languages – Choose your weapons

.NET Overview

- What is .NET?
- Benefits of programming in .NET
- Important Concepts

.NET Overview

- Benefits of programming in .NET
  - Consistent Object Oriented Development platform
  - Automatic memory management – Garbage collection
  - Support for multiple languages

(Source: MSDN)

.NET Overview

Consistent Object Oriented Development platform

Everything you see can be treated as an Object!

```
Dim myLine As New Line()
myLine.StartPoint = New Point3d(0, 0, 0)
myLine.EndPoint = New Point3d(10, 10, 0)
myLine.GetClosestPointTo(New Point3d(5, 5.1, 0), False)

Dim x as Interger = 7
Dim s as String = x.ToString()
```

- Objects are instances of a ***Type*** or ***Class*** (for example **myLine** is an object and **Line** is a type)
- Objects have properties such as **StartPoint**, and methods such as **GetClosestPointTo()**

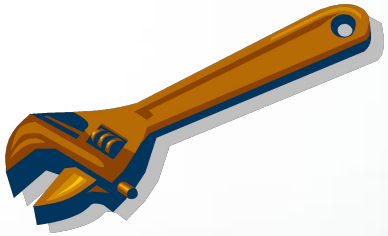
.NET Overview

Consistent Object Oriented Development platform

Mature API Constructs

What's wrong with this function?

```
int acedSSGet(const char * str,
   const void * pt1,
   const void * pt2,
   const struct resbuf * filter,
   ads_name ss);
```



.NET Overview

Consistent Object Oriented Development platform

Mature API Constructs

Some 6 new classes defined to encapsulate acedSSGet()

```
Dim values(2) As TypedValue

‘Define the selection criteria
values(0) = New TypedValue(DxfCode.Start, “Circle”)
values(1) = New TypedValue(DxfCode.Color, 1)

Dim selFilter As New SelectionFilter(values)
Dim selOpts As New PromptSelectionOptions()
selOpts.AllowDuplicates = True

‘Run the selection
Dim res As PromptSelectionResult = Editor.GetSelection(selOpts, selFilter)
```

.NET Overview

Benefits of programming in .NET

- Consistent Object Oriented Development platform
- Automatic memory management (Garbage collection)

  and consistent exception handling
- Support for multiple languages

(Source: MSDN)

.NET Overview

Benefits of programming in .NET

Automatic memory management

- Old Way (C++) - Potential for memory leaks!

```
char *pName=(char*)malloc(128);
strcpy(pName,"Hello");
//...
free(pName);
```

- New Way - .NET
  - C++ - String *pName=new String("Hello")
  - VB - Dim Name As String = "Hello"
  - C# - String Name="Hello";
  - // Garbage collection handles deallocation; no 'delete'!0

.NET Overview

Benefits of programming in .NET

Consistent exception handling

- Old Way – VB: Can be very confusing and problematic!

```
On Error GoTo UnexpectedError
Dim x As Double=10/0 '…error!
UnexpectedError:
MsgBox Str$(Err.Number)
```

- New – VB .NET

```
Try
        Dim x As Double=10/0 '…error which throws exception
Catch
        '…what happened?  Division by Zero!
Finally
        '…cleanup - do this either way
End Try
```

.NET Overview

Benefits of programming in .NET

- Consistent Object Oriented Development platform
- Automatic memory management (Garbage collection) and consistent exception handling
- Support for multiple languages
  (Source: MSDN)

.NET Overview

Benefits of programming in .NET

Support for multiple languages

- C#, VB most commonly used
- Can interop between code written in different languages. For example, a class written in C# can be inherited from a class written in VB! In fact, AutoCAD's managed assemblies are written using managed C++ which you will access from VB.NET.
- No significant difference in performance as all languages compile to IL (Intermediate Language) executed by the CLR

.NET Overview

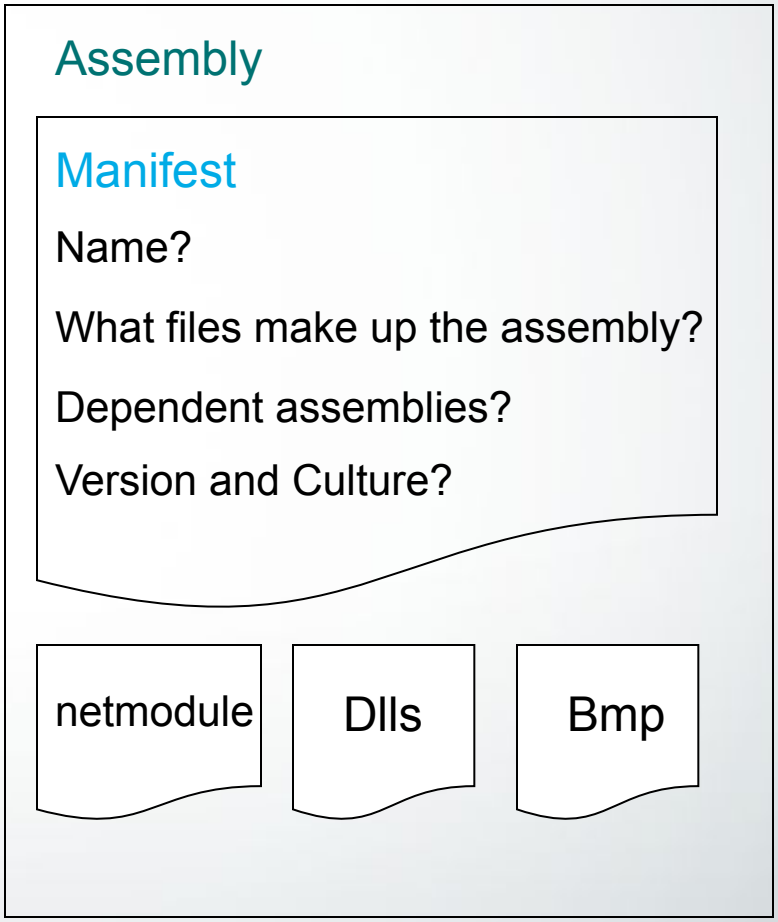
- What is .NET?
- Benefits of programming in .NET
- Important Concepts

.NET Overview

Important Concepts

Assemblies

- Fundamental unit of deployment and execution in .NET
- Contains a manifest that describes the assembly
- Boundary for code execution and access permission



(Source: MSDN)

Class Agenda

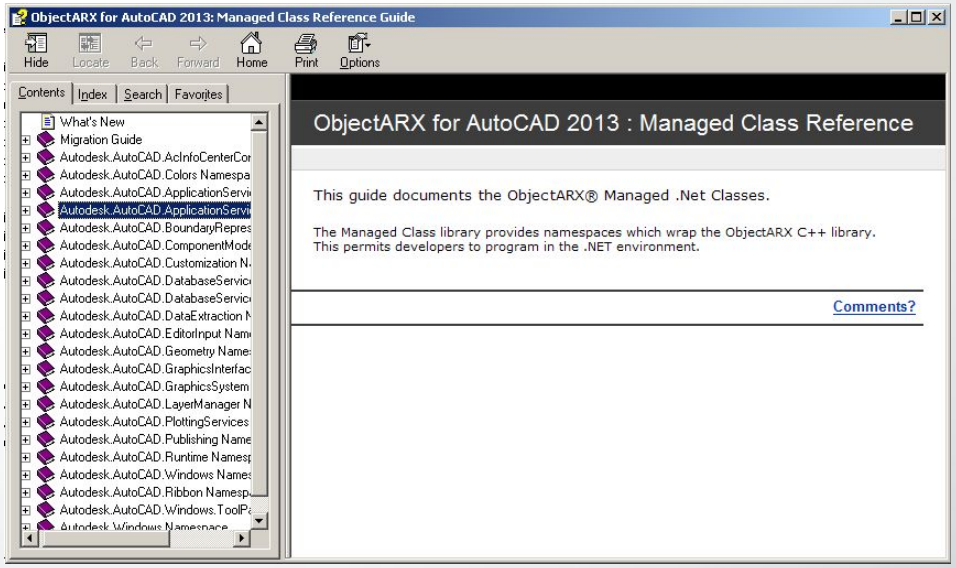
Lectures and Labs

- Overview of .NET.
- **AutoCAD .NET Visual Studio project settings – Hello World!**
- User Interaction - Simple User Input and Entity Selection
- Database Fundamentals – Symbol tables, Transactions
- Database Fundamentals – Dictionaries, XRecords, Table Traversal
- More User Interaction – Advanced Prompts
- User Interface design - WinForm Dialogs and Palettes
- Event handling – Reacting to AutoCAD Events in .NET.

AutoCAD .NET API Documentation

- *How do I get started?*

- *ObjectARX SDK Includes*:
  - SDK Samples!
  - ObjectARX Developer's Guide
  - Managed Reference Guide
    - Arxmgd.chm



- *ADN website*
  - DevNotes
  - DevHelp Online

- *Visual Studio Class Browser*

Development Environment

- Microsoft Visual Studio 2010 (SP1) or
  Microsoft Visual Studio 2012

- AutoCAD 2014

- Microsoft Windows 8
- Microsoft Windows 7
- Microsoft Windows XP

.NET Debugging Tools

- Reflector
  - Browse .NET assemblies, disassemble, decompile
    - http://sharptoolbox.madgeek.com
- Ildasm
  - Disassemble .NET assemblies
    - Visual Studio Tools
- Fuslogv
  - Diagnose load time problems
    - Visual Studio Tools
- FxCop
  - Check conformance with Design Guidelines
    - http://www.gotdotnet.com/team/fxcop/

Snoop Tools (for AutoCAD's database)

- ArxDbg (C++) ObjectARX SDK
- MgdDbg(C#) ADN

Visual Studio project settings– Hello World!

- Start with a Class Library application type with DLL output.

- Add references to AutoCAD's managed assemblies
  - acdbmgd.dll
    - Database services and DWG file manipulation (like ObjectDBX)
  - acmgd.dll
    - AutoCAD Application specific
  - accoremgd.dll
    - AutoCAD core logic
  - Find them in the AutoCAD install folder (set COPY LOCAL = FALSE)

    C:\Program Files\AutoCAD 2014

    C:\ObjectARX 2014\inc

How does a plugin for AutoCAD work ?

```
Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.Interop

Public Class Commands
    Implements IExtensionApplication

    Private WithEvents docs As DocumentCollection

    Private Sub docs_DocumentActivated(ByVal sender As Object, ByVal e As Autodesk.AutoCAD.Ap
        ThisDrawing = e.Document.AcadDocument
    End Sub

    Private Sub ThisDrawing_BeginDocClose(ByRef Cancel As Boolean) Handles ThisDrawing.BeginI
        ThisDrawing = Nothing
    End Sub

    Private Sub ThisDrawing_EndSave(ByVal FileName As String) Handles ThisDrawing.EndSave
        MsgBox("EndSave called")
    End Sub
End Class
```

Code witten in Visual Basic .NET

Project VB.NET

Reference to AutoCAD DLLs. Use it from ObjectARX INC folder

Compile

Assembly (.dll)

Load inside AutoCAD with NETLOAD

Visual Studio project settings– Hello World!

- Reference namespaces you will use in your project
- In VB.NET Use Imports keyword:

Imports Autodesk.AutoCAD.ApplicationServices

Access to the AutoCAD application

Imports Autodesk.AutoCAD.EditorInput

Access to the AutoCAD editor

Imports Autodesk.AutoCAD.Runtime

Command registration

Imports Autodesk.AutoCAD.DatabaseServices

Access to the AutoCAD Database and Entities

Visual Studio project settings– Hello World!

Add a simple command – HelloWorld

- Make a function an AutoCAD command by adding an *attribute*

```
Public Class Class1
    <CommandMethod("HelloWorld")> _
    Public Function HelloWorld()
    End Function
End Class
```

- The attribute is added to the metadata for that function

- CommandMethod or CommandMethodAttribute type accepts several parameters in its constructor such as group name, global and local names, command flags and more (Use the object browser)

Visual Studio project settings– Hello World!

To print a string to command line

- Get the editor *object* for the active document

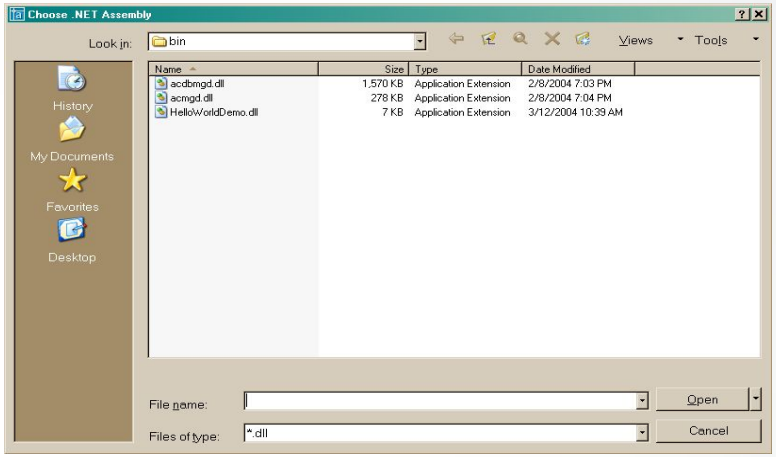
```
Dim ed As Editor =
Application.DocumentManager.MdiActiveDocument.Editor
```

- Call the editor's WriteMessage method

```
Public Class Class1
    <CommandMethod("HelloWorld")> _
    Public Function HelloWorld()
        ed.WriteMessage("Hello World")
    End Function
End Class
```

Loading .NET assembly

- NETLOAD command
- AUTOLOADER
  - Startup
  - On command invocation
- Demand Load (Registry)
  - Startup
  - On command invocation
  - On request
    - **From another application**
  - On proxy detection




```
[HKEY_LOCAL_MACHINE\SOFTWARE\Autodesk\AutoCAD\R19.1\ACAD-D001:409\Applications\AcLayer]
        "DESCRIPTION"="AutoCAD Layer Manager"
        "LOADER"="C:\\Program Files\\AutoCAD 2014\\aclayer.dll"
        "LOADCTRLS"=dword:0000000e
        "MANAGED"=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\Autodesk\AutoCAD\R19.1\ACAD-D001:409\Applications\AcLayer\Commands]
        "LAYER"="LAYER“

[HKEY_LOCAL_MACHINE\SOFTWARE\Autodesk\AutoCAD\R19.1\ACAD-D001:409\Applications\AcLayer\Groups]
        "ACLAYER_CMDS"="ACLAYER_CMDS“
```

Use Installers to set these keys!

AutoLoader

- AutoCAD loads “bundles” from %appdata%\Autodesk\ApplicationPlugins
- Each bundle has “PackageContents.xml”

```xml
<?xml version="1.0" encoding="utf-8"?>

<ApplicationPackage SchemaVersion="1.0" AutodeskProduct="AutoCAD" ProductType="Application" Name="MyApp"
                    AppVersion="1.0" Description="MyTestApp"
                    Author="Autodesk" Icon="./Contents/Help/Resource/TDIcon.jpg"
                    OnlineDocumentation="http://www.autodesk.com"
                    HelpFile="./Contents/Help/MyApp.chm" ProductCode="{DF51A41E-DC4F-4ad8-8F8A-B6CB7130840F}">

  <RuntimeRequirements OS="Win32|Win64" Platform="AutoCAD*" SeriesMin="R19.0" SeriesMax="R19.1" />

  <CompanyDetails Name="Autodesk" Phone=" " Url="http://www.autodesk.com" Email="Support@autodesk.com" />

  <Components>
    <RuntimeRequirements SupportPath="./Contents/Support" OS="Win32|Win64" SeriesMin="R19.0" />

    <ComponentEntry AppName="MyApp" ModuleName="./Contents/Windows/MyApp.fas" AppDescription="MyTestApp"
                    LoadOnAutoCADStartup="True" LoadOnCommandInvocation="True" />

    <ComponentEntry AppName="MyApp" ModuleName="./Contents/Support/MyApp.cuix" />
  </Components>

</ApplicationPackage>
```

NETLOAD or Registry Keys

HKEY_CURRENT_USER — For all users

HKEY_LOCAL_MACHINE — For a specific user only

- SOFTWARE
  - Autodesk
    - AutoCAD
      - R19.0
        - ACAD-B001:409
          - Applications
            - YourAppName

R18.0: 2010
.1: 2011
.2:2012

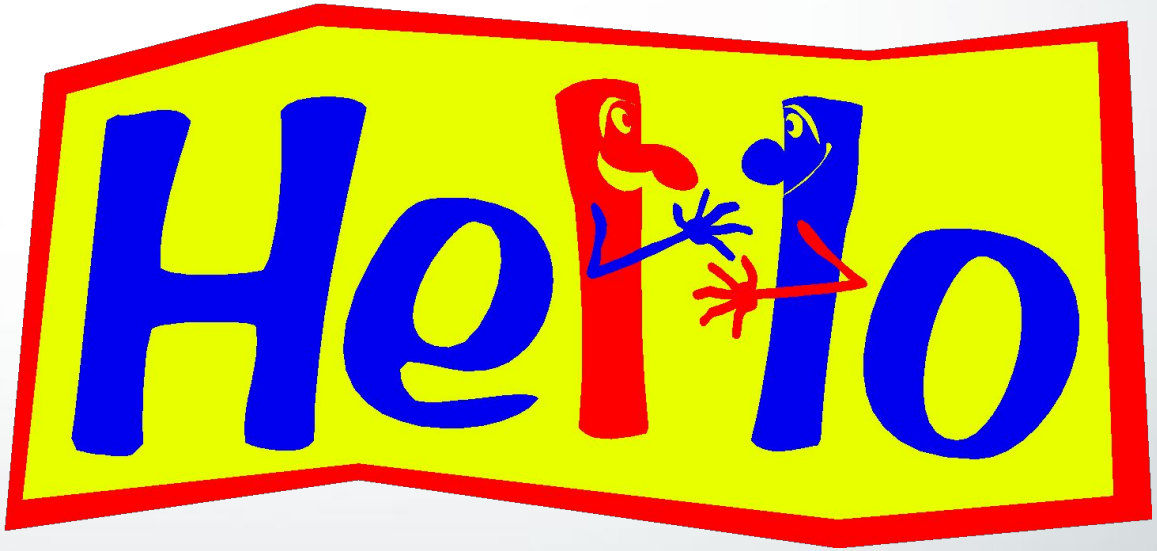
R19.0: 2013
.1: 2014

X000: Civil3D
X001: AutoCAD

409: English
416: Portuguese
040A: Spanish

```
"DESCRIPTION"="Custom App Name"
"LOADER"="C:\\folder\\appName.dll"
"LOADCTRLS"=dword:0000000e
"MANAGED"=dword:00000001
```

Lab 1 – Hello World!



Class Agenda

Lectures and Labs

- Overview of .NET.
- AutoCAD .NET Visual Studio project settings – Hello World!
- User Interaction - User Input and Entity Selection
- Database Fundamentals – Symbol tables, Transactions
- User Interface design – Win Form Dialogs and Palettes
- Event handling – Reacting to AutoCAD Events in .NET. Database
- Dictionaries, XRecords, Table Traversal
- Point Monitor
- Jigs
- Additional User Interface Elements

Prompting for User Input

- Use Prompt*XXX*Options to set the parameters for prompting
  - *XXX* is the value type we want to prompt, such as Angle, String, Distance, Corner etc.
  - Use Message and Keywords properties to set the prompt string and list of keywords
  - Use Allow*YYY* to set conditions for prompting. *For e.g., AllowNegative*
- To prompt, use Editor's GetXXX functions
  - *Examples - GetAngle, GetString, GetDistance, GetCorner etc*
  - *Pass PromptXXXOptions into GetXXX*
- Result of prompting stored in PromptResult or derived types
  - *Examples – PromptDoubleResult, PromptIntegerResult etc.*

Prompt for a point on screen

- Config the options to select a point on screen

```
Dim pointOptions As New PromptPointOptions("Select a point: ")
```

- Ask the user to select the point and store the selection result

```
Dim pointResult As PromptPointResult = ed.GetPoint(pointOptions)
```

- Create a Point3d variable to store the selected point
  - Requires an additional *imports* for Point3d: **Autodesk.AutoCAD.Geometry**

```
Dim selectedPoint As Point3d = pointResult.Value
```

- Write the point coordinates (XYZ)

```
ed.WriteMessage(selectedPoint.ToString())
```

More User Interaction

Prompts**XXX**Options is used to control prompting such as

- Set the Message
  - Enter Number of Sides:
- Set Keywords
  - Enter Number of Sides [Triangle/Square/Pentagon] :
- Set Defaults
  - Enter Number of Sides [Triangle/Square/Pentagon] <3>:
- Set Allowed values
  - Enter Number of Sides [Triangle/Square/Pentagon] <3>: -5
    - Value must be positive and nonzero.

Prompt**XXX**Result is used to obtain result of prompting

Additional prompts

Types:

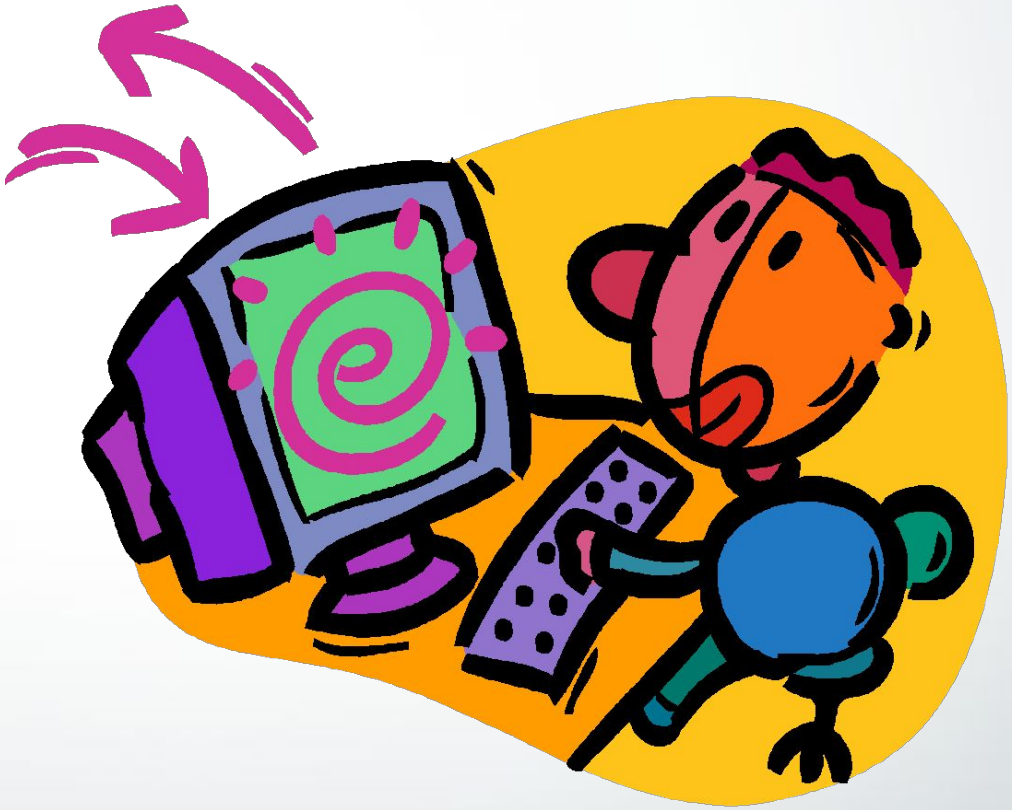
- PromptPointOptions
- PromptStringOptions
- PromptDoubleOptions
- PromptAngleOptions
- PromptCornerOptions
- PromptDistanceOptions
- PromptEntityOptions
- PromptIntegerOptions
- PromptKeywordOptions
- PromptNestedEntityOptions
- PromptSelectionOptions
- Etc.

Help file for the rescue!

Dotnet 2014 Wizards

AppWizard – Templates for a VB.NET or C# application

Lab 2 –User Input



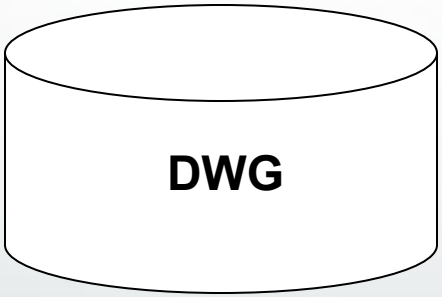
Class Agenda

Lectures and Labs

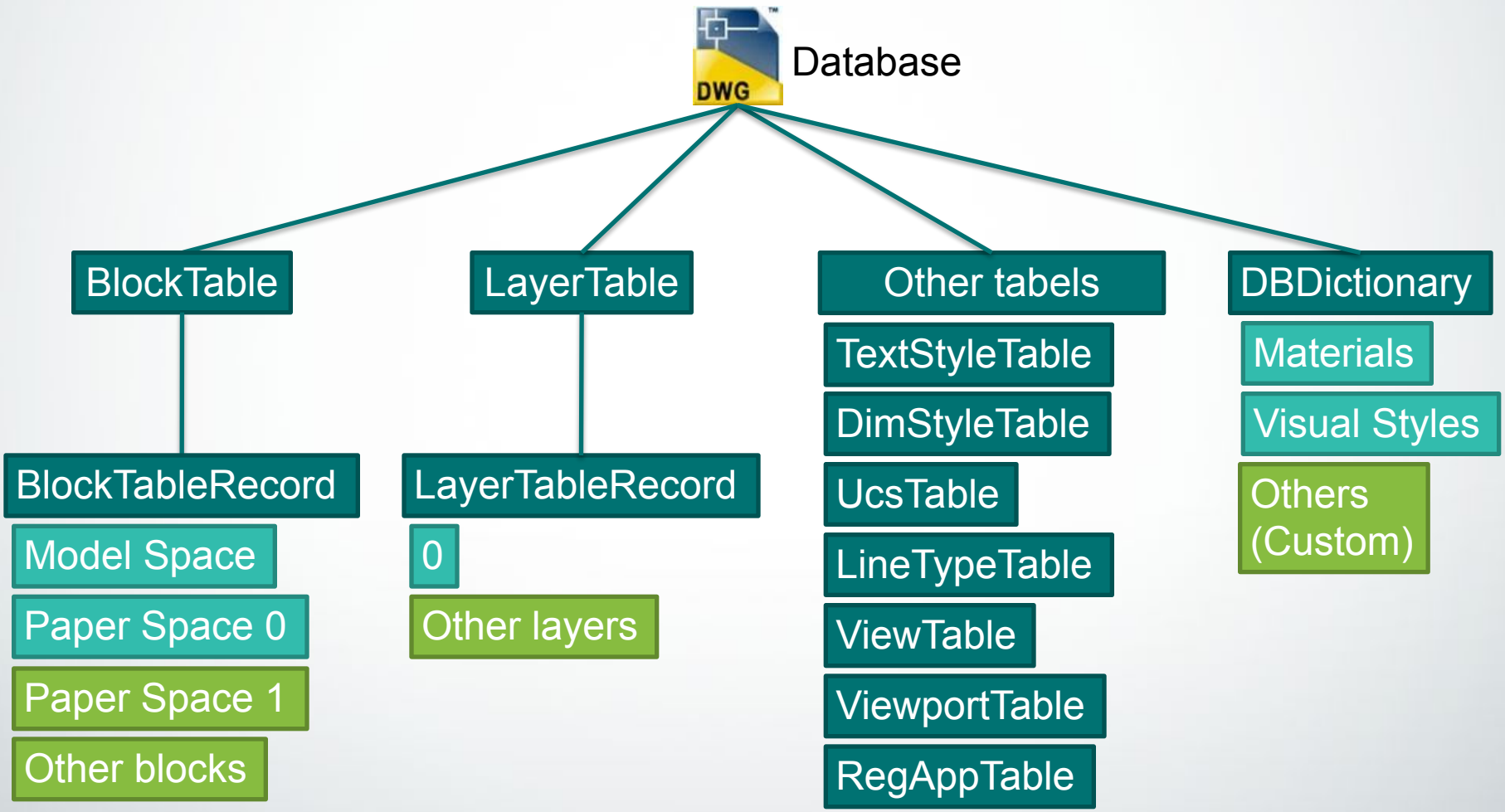
- Overview of .NET.
- AutoCAD .NET Visual Studio project settings – Hello World!
- User Interaction - User Input and Entity Selection
- Database Fundamentals – Symbol tables, Transactions
- User Interface design – Win Form Dialogs and Palettes
- Event handling – Reacting to AutoCAD Events in .NET. Database
- Dictionaries, XRecords, Table Traversal
- Point Monitor
- Jigs
- Additional User Interface Elements

AutoCAD Drawing Database

- In-Memory representation of the Dwg File
  - Objects are stored hierarchically in the database - Db Structure
  - All objects have identities – ObjectId, like *Primary Key* in a relational database
  - Objects are always accessed in a transaction
    - The transaction defines the boundary of database operations
    - Objects have to be opened first in a transaction before they can be used
  - Objects can refer to other objects – such as a line having a reference to a layer



Database Structure: Overview



Database Components

- Symbol Tables
  - Examples Layer Table, Linetype Table, Textstyle Table etc.
  - Containers to store Symbol Table *Records*
    - Example LayerTableRecord, LinetypeTableRecord etc
  - All Symbol Tables have common methods of a container such as
    - Add – to add a record
    - Item – to lookup an entry with a search string
    - Has – To know if an entry exists
  - Is enumerable
  - Each symbol table can hold only records of a specific type
    - For example, a LayerTable can hold only LayerTableRecords

Getting a Database Object

- Construct One
  - In Memory
- Get the one currently active in AutoCAD
  - HostApplicationServices.WorkingDatabase()

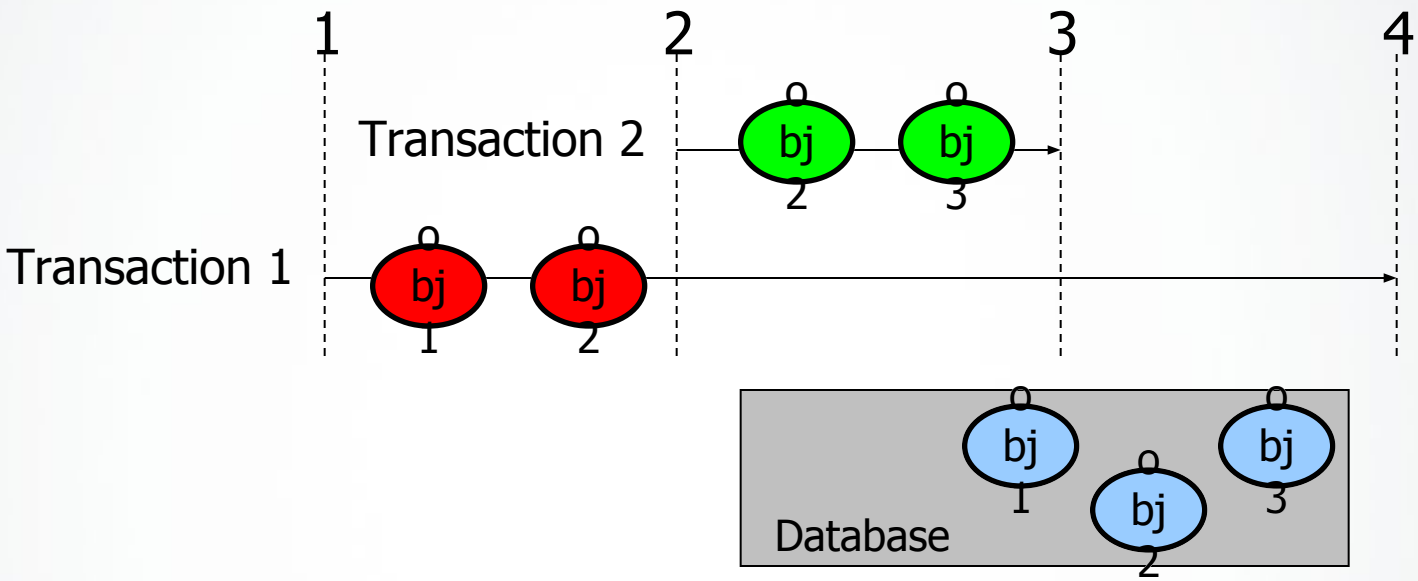
Object Identity - ObjectID

- All Objects that exist in the database have an ObjectId
  - Is *unique* per session (or instance) of AutoCAD
  - Is generated automatically for an object when it is added to the database
  - Non-database resident objects do not have an ObjectId set
  - Can be cached to open an object later
- Get it using ObjectId property

Transactions

- Transactions
  - Sets the boundary for database operations
  - Handles exception cleanly
  - Operates with a single Undo filer
  - Can be
    - committed – All database operations are saved
    - rolled back – All database operations are aborted
  - Can be nested

Nesting Transactions



1. Client starts Trans1 and gets Obj1 & Obj2
2. Client starts Trans2 and gets Obj2 & Obj3
3. Client commits Trans2

Trans2 changes are committed

4a. Client commits Trans1

- **Trans1 changes are committed**

4b. Client aborts Trans1 instead

- **Trans1 (and Trans2) changes are rolled back**

Transactions

- Standard db access with Transactions

```
Public Function MyFunc()

    'Get the database current in AutoCAD
    Dim db As Database = HostApplicationServices.WorkingDatabase()

    'Start a transaction using the database transaction manager
    Dim trans As Transaction = db.TransactionManager.StartTransaction()

    Try
        'Do all database operations here
        'Lets get the block table from the database
         'Drill into the database and obtain a reference to the BlockTable
        Dim bt As BlockTable = trans.GetObject(db.BlockTableId,
    OpenMode.ForWrite)
        'Everything successful, so commit the transaction
        trans.commit()
    Catch
        trans.Abort()
    Finally
        'All ok. Call Dispose explicitly before exiting
        trans.dispose()
    End Try
End Function
```

Recommended Transaction Use

- Standard DB access with Transactions

```
Public Function MyFunc()
    'Get the database current in AutoCAD
    Dim db As Database = HostApplicationServices.WorkingDatabase()

    'Start a transaction using the database transaction manager
    Using trans As Transaction =
    db.TransactionManager.StartTransaction()

        'Do all database operations here
        'Lets get the block table from the database
         'Drill into the database and obtain a reference to the BlockTable

        Dim bt As BlockTable = trans.GetObject(db.BlockTableId,
    OpenMode.ForWrite)
        'Everything successful, so commit the transaction
        trans.commit()
    End Using
End Function
```

Transaction - Opening an Object

- Use transaction's GetObject to open an object
  - The first parameter is the ObjectId
  - Second parameter is the open mode
    - ForRead – Access but not Modify
    - ForWrite – Modify and Access
    - ForNotify – When the object is notifying

```
Dim bt As BlockTable = trans.GetObject( _
                db.BlockTableId, _
                OpenMode.ForWrite _
                )
```

Adding an Object to the database

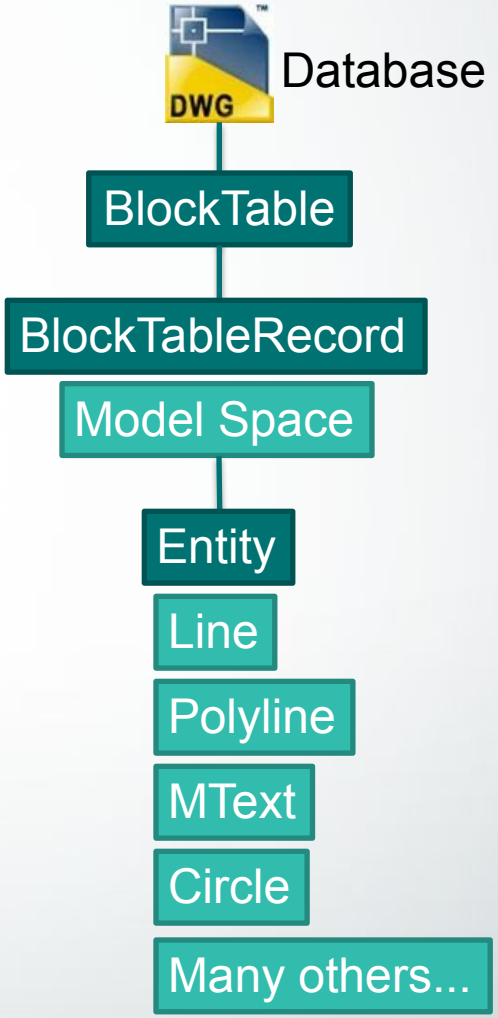
- Find the right owner to add a newly created object to
  - All objects have exactly *one* owner
  - For example, newly created LayerTableRecord can *only* be added to the Layer Table, its owner, or a newly created entity can be *only* added to a block table record
- Use Add method for adding Symbol Table Records to add to Symbol Table
- Use AppendXXX to add add other kinds of objects to its owners For example
  - AppendEntity to add to BlockTableRecord
- Once an object is added to an owner, always let the transaction know!

For example :

```
newBtr.AppendEntity(circle) 'Add our circle to its owner the BTR
trans.AddNewlyCreatedDBObject(circle, True)
```

Database Structure: Model Space

- Under BlockTable
- Model Space is a BlockTableRecord (BTR)
  - This concept also applies to Paper Spaces and other internal and user-defined blocks
- Each BTR contains Entities
- One type of entity for each geometric type
- Is enumerable – Iterate with 'For Each'



Append an Entity to Model Space

```
Dim db As Database = Application.DocumentManager.MdiActiveDocument.Database
Using trans As Transaction = db.TransactionManager.StartTransaction

    'open the current space (can be any BTR, usually model space)
    Dim mSpace As BlockTableRecord = trans.GetObject(db.CurrentSpaceId, _
                                                     OpenMode.ForWrite)

    'Create and configure a new line
    Dim newLine As New Line
    newLine.StartPoint = New Point3d(0, 0, 0)
    newLine.EndPoint = New Point3d(10, 10, 0)

    'Append to model space
    mSpace.AppendEntity(newLine)

    'Inform the transaction
    trans.AddNewlyCreatedDBObject(newLine, True)
    trans.commit()
End Using 'Dispose the transaction
```

Open the current space for write. Can be any other BTR.

Create and configure a new line (in-memory)

Append to the current space, now the line it is database resident

Object memory management

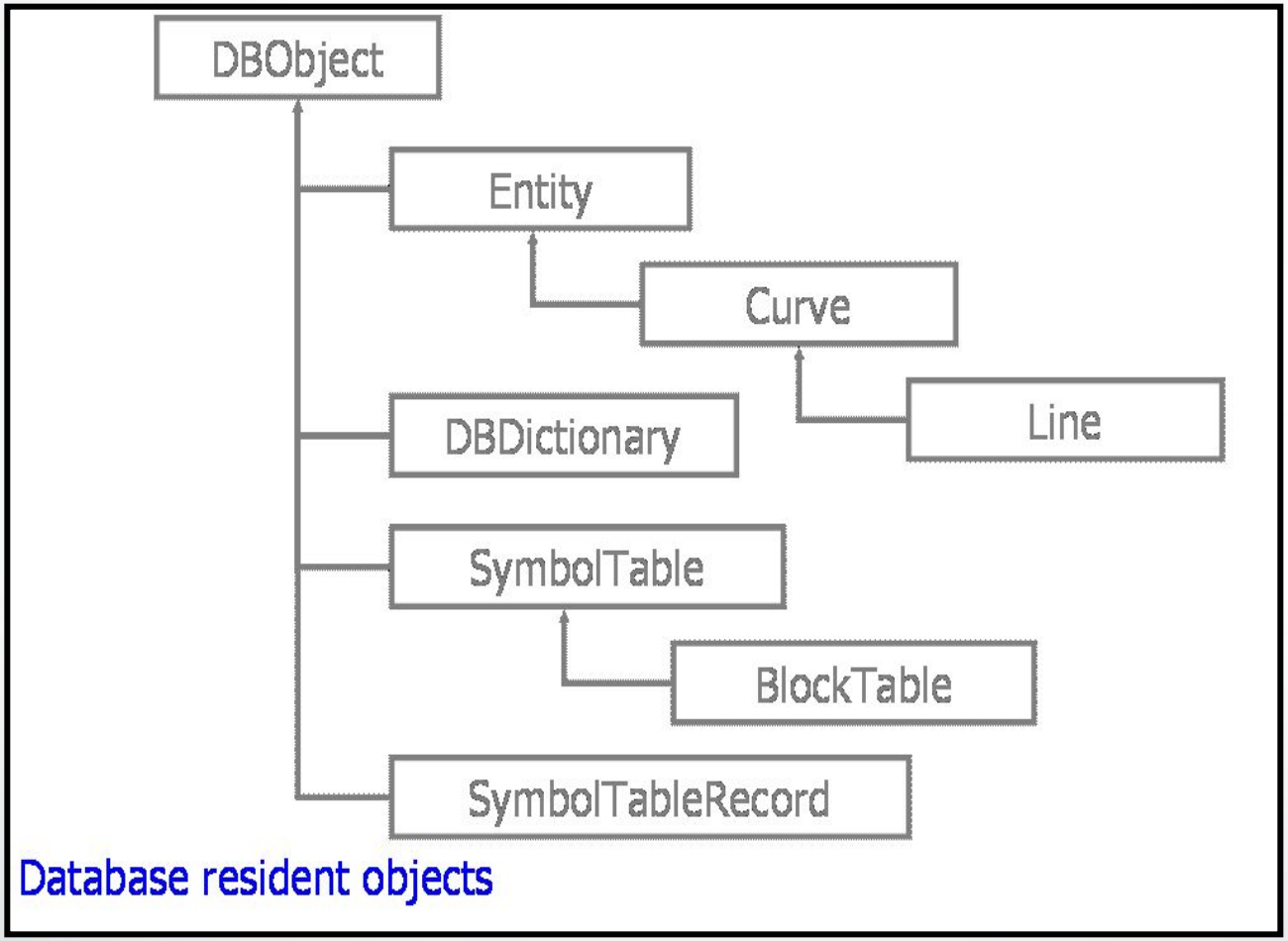
- Note managed objects *wrap* an unmanaged C++ object!
- So we create them with New, Do they need to be disposed?
  - No - garbage collection disposes the object when it wants to reclaim memory
    - If the object is not in the database — this *deletes* the underlying unmanaged object
    - If the object is in the database – this *Closes* the underlying unmanaged object
- If opened in a transaction, disposing or committing the transaction closes it automatically! – Clean memory management.
- Manual Dispose is required in a few cases, e.g. Transaction (Using)

Object Model Overview

classmap.dwg

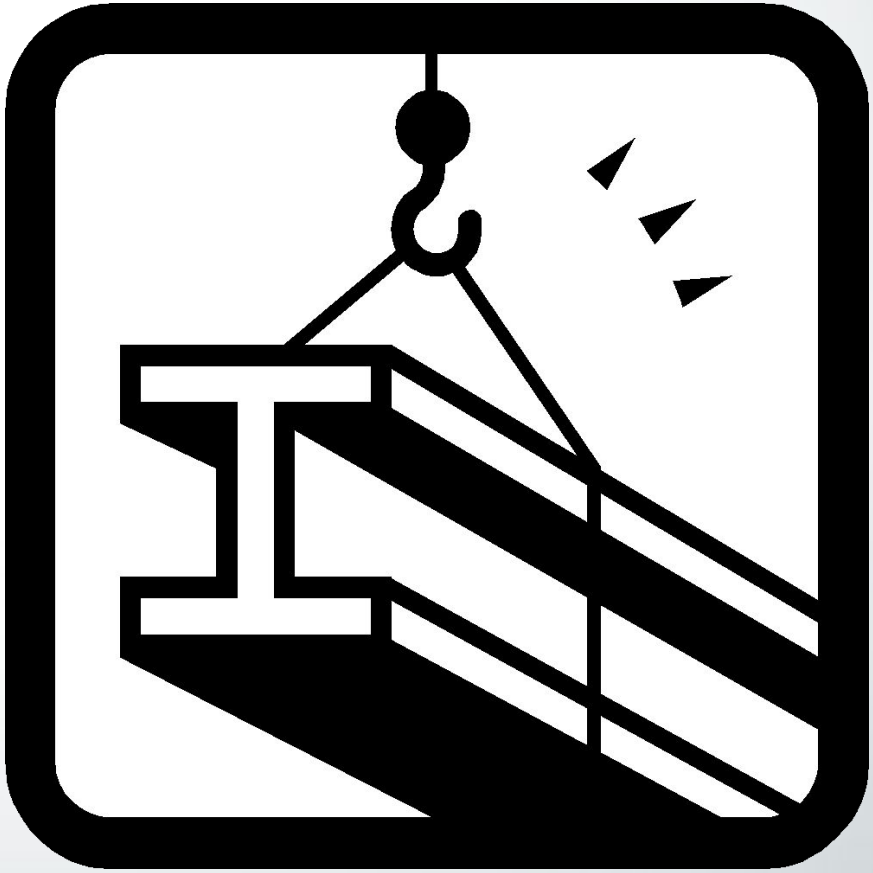
- in ObjectARX distribution

Important Managed Classes



Lab 3

Create Entity, Block and Block Reference



Class Agenda

- Lectures and Labs
  - Overview of .NET.
  - AutoCAD .NET Visual Studio project settings – Hello World!
  - User Interaction - User Input and Entity Selection
  - Database Fundamentals – Symbol tables, Transactions
  - User Interface design – Win Form Dialogs and Palettes
  - Event handling – Reacting to AutoCAD Events in .NET. Database
  - Dictionaries, XRecords, Table Traversal
  - Point Monitor
  - Jigs
  - Additional User Interface Elements

Forms and UI Basics

- Win Form API basics
- How create a form
- Common used controls
- Respond to user actions
- Using the form

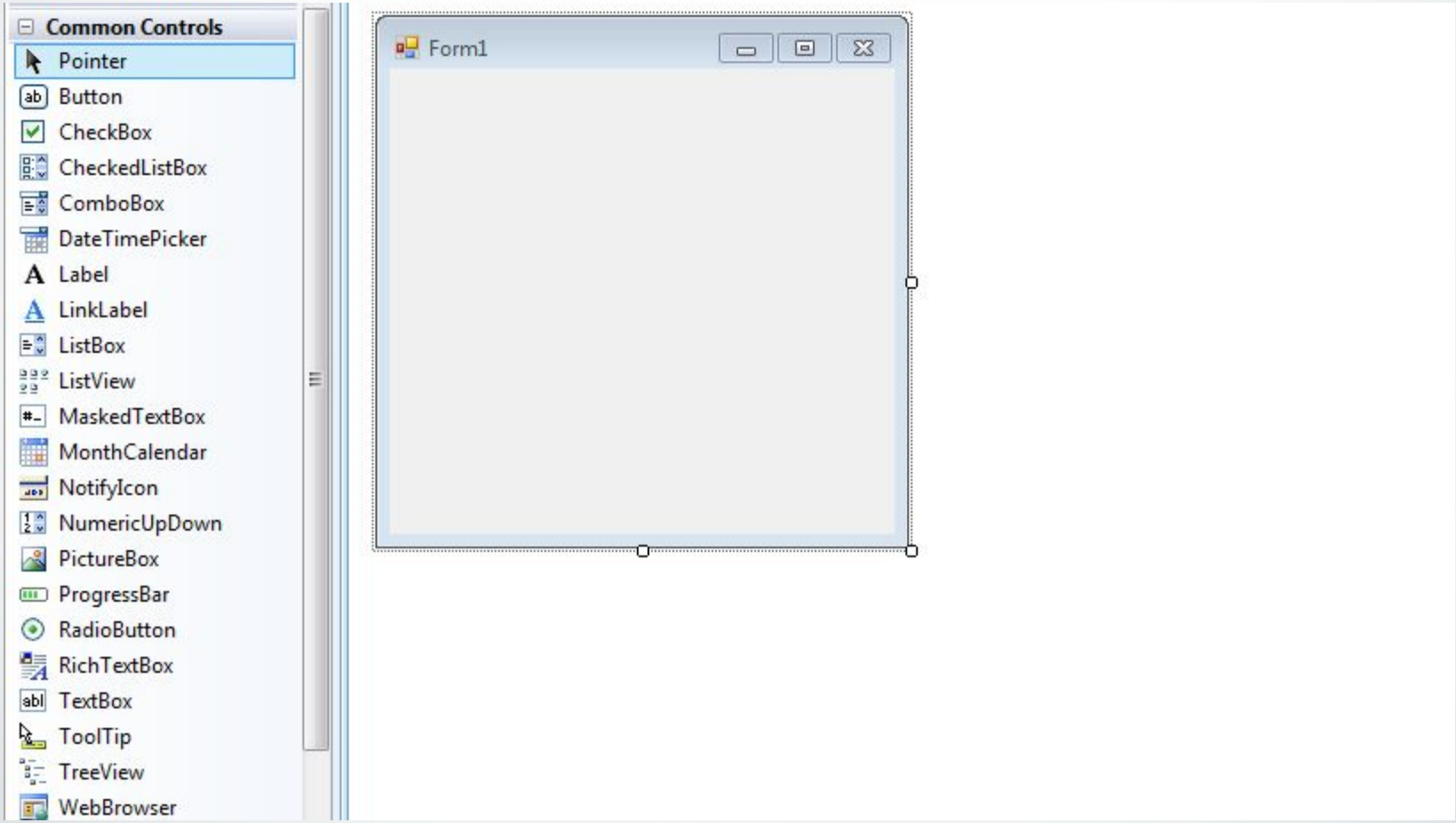
Windows® OS is based on windows

- Everything on screen is some type of window
- You move focus across windows
  - The window with focus is usually “highlighted” and will receive keyboard input
- Forms are specialized windows
  - Can host other windows, in this case controls

Forms with WinForms API

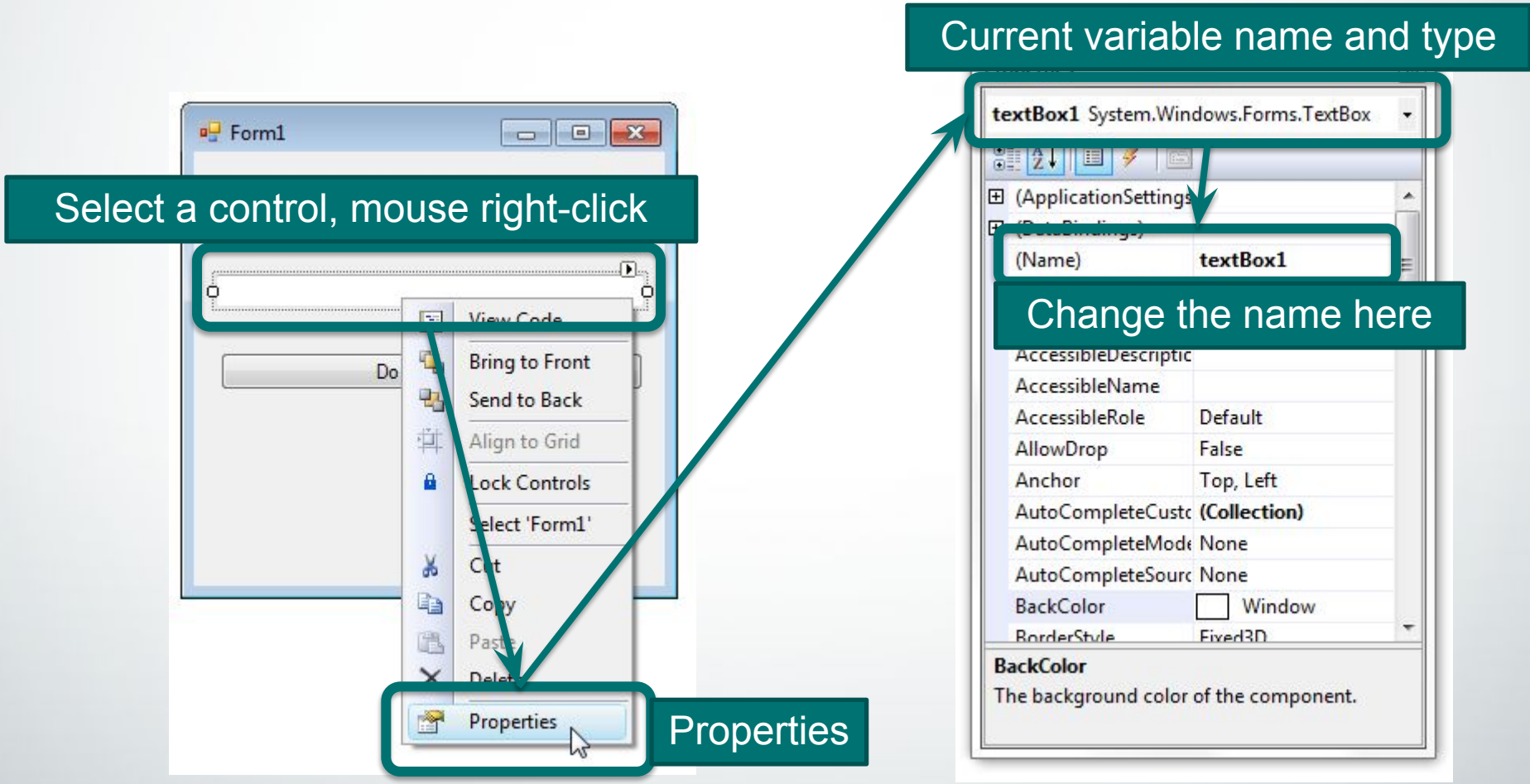
- Require reference to System.Windows.dll
- Namespace **System.Windows.Forms**
- Main features
  - Forms
  - Controls for forms (button, textbox, combobox)
  - Ready to use forms (dialogs to open, save, folder)
  - Others (menus, ribbon, tooltip)

Creating my first Form



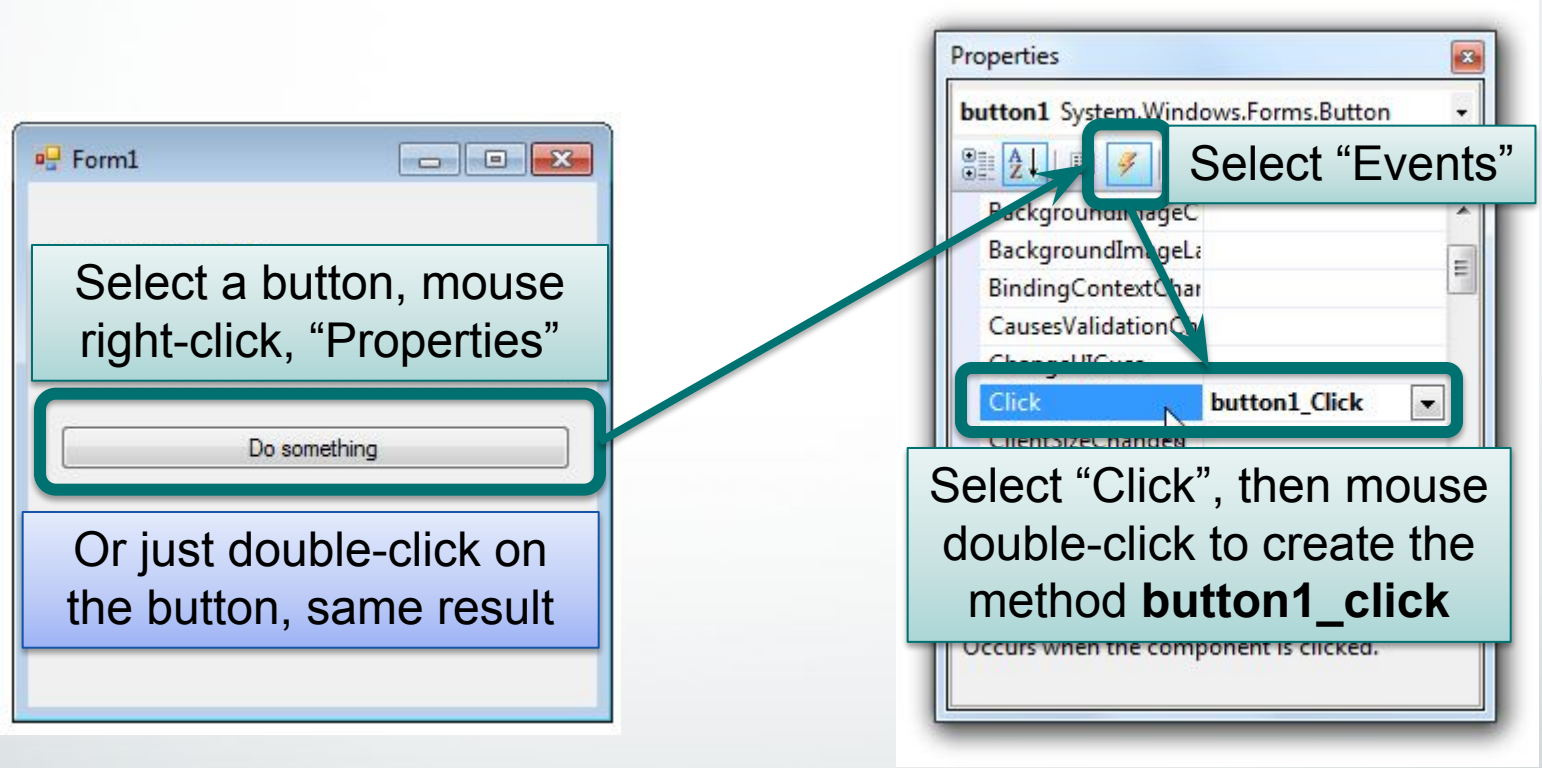
Controls are variable too

- Each control is a variable – rename for further use

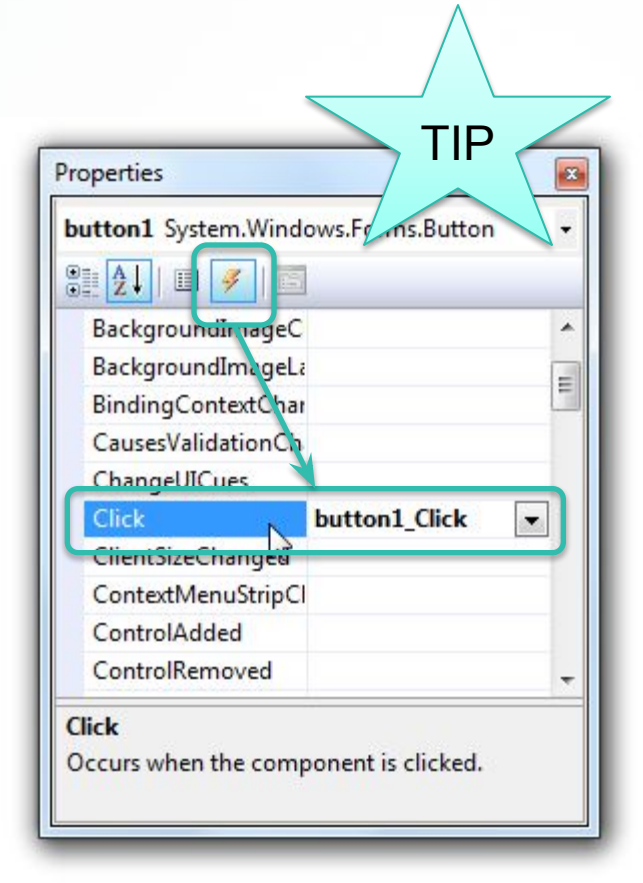


Do something when the user click!

- For some controls we need execute something when the user interacts with it
  - Example: when the user clicks on a button



Create the control events



User Interface Design

- AutoCAD Defined
  - Menus – Application level menu, Context menu
  - Dialogs
    - AutoCAD's Enhanced Secondary Windows (Palettes)
    - Color, Linetype, Lineweight, OpenFile dialogs
  - Tabbed Dialog Extensions (to Options Dialog)
  - Status Bar
  - Tray
  - Drag-Drop
  - And more. Explore Autodesk.AutoCAD.Windows namespace

- Windows Defined
  - Windows Forms (Winform)
  - Host of other controls defined in CLR

Using a form inside AutoCAD

- Modal forms
  - Application.ShowModalDialog
- Modeless forms (consider Palette instead)
  - Application.ShowModelessDialog
- Persists size and position automatically

Palette in AutoCAD

- Create a user control
- Create a PaletteSet
- Add the user control to the palette using Add method.
- Set the Visible property of the paletteset

Handling Events

- Event
  - message sent by an object to notify something has happened
  - message is received in a function call by one or more listeners
  - event sender only requires function pointer to dispatch a message
  - any interested party can implement the function and receive the event
  - function must have a specific signature that the sender requires
  - Use .NET delegates to '*wire*' sender and receiver
- Delegates
  - Like a class (can be instantiated) but with a signature
  - Holds references to functions having same signature
  - Like 'Type-Safe' function pointer
  - Can encapsulate any method which matches the specific signature

Using Delegates

- Delegates in AutoCAD's .NET API usually have 'EventHandler' suffix
  - Lookup the signature of the delegate in the object browser

- Implement a function with same signature
- Instantiate the delegate passing address of the function into its constructor
- Add the delegate instance to sender's list of listeners
  - C#, use += operator
  - VB, use AddHandler

```
Delegate myDelegate = new Delegate(address of myFunction);
EventSender.Event += myDelegate;

myFunction(delegate signature)
{
}
'Don't forget to remove the listener!
```

Event Handling - Example

- Create the event handler (callback)

```
Sub objAppended(ByVal o As Object, ByVal e As ObjectEventArgs)
    MessageBox.Show("ObjectAppended!")
    ‘Do something here
    ‘Do something else, etc.
End Sub
```

- Associate the event handler with an event

```
Dim db As Database
db = HostApplicationServices.WorkingDatabase()
AddHandler db.ObjectAppended, New
ObjectEventHandler(AddressOf objAppended)
```

- Disconnect the event handler

```
RemoveHandler db.ObjectAppended, AddressOf objAppended
```

Lab 4

- PaletteSet and DB Events

Class Agenda

Lectures and Labs

- Overview of .NET.
- AutoCAD .NET Visual Studio project settings – Hello World!
- User Interaction - User Input and Entity Selection
- Database Fundamentals – Symbol tables, Transactions
- User Interface design – Win Form Dialogs and Palettes
- Event handling – Reacting to AutoCAD Events in .NET. Database
- Dictionaries, XRecords, Table Traversal
- Point Monitor
- Jigs
- Additional User Interface Elements

Dictionaries and XRecords

- Dictionaries (Type DbDictionary)
  - Containers to hold data only
  - Holds other Dictionaries
  - Holds non-graphical Objects (derived from DbObject but not DbEntity!)
  - Is enumerable
    - Each item has a string key
    - Items searchable with a string key using GetAt() or Item
- Two *Root* dictionaries
  - Named Objects Dictionary (NOD)
    - Owned by the database
    - Available by default
    - Used to store database level data
  - Extension Dictionary
    - Owned by an Entity
    - Created by the user only when needed
    - Used to store entity level data
- Use SnoopDb to look where the dictionaries are stored

Dictionaries and XRecords

- XRecord
  - Data containers
  - Holds data in a Resbuf chain (Result Buffer)
    - Resbuf – Linked List of TypedValues (DataType–Value pair)
  - No “Key” to search values within a Resbuf.

    Should know the order data is stored in the list
  - XRecords can be added to Dictionaries
    - If stored in NOD –database level data
    - If stored in Extension Dictionary – entity-level data

Get NOD

- To *get* the NOD for the database

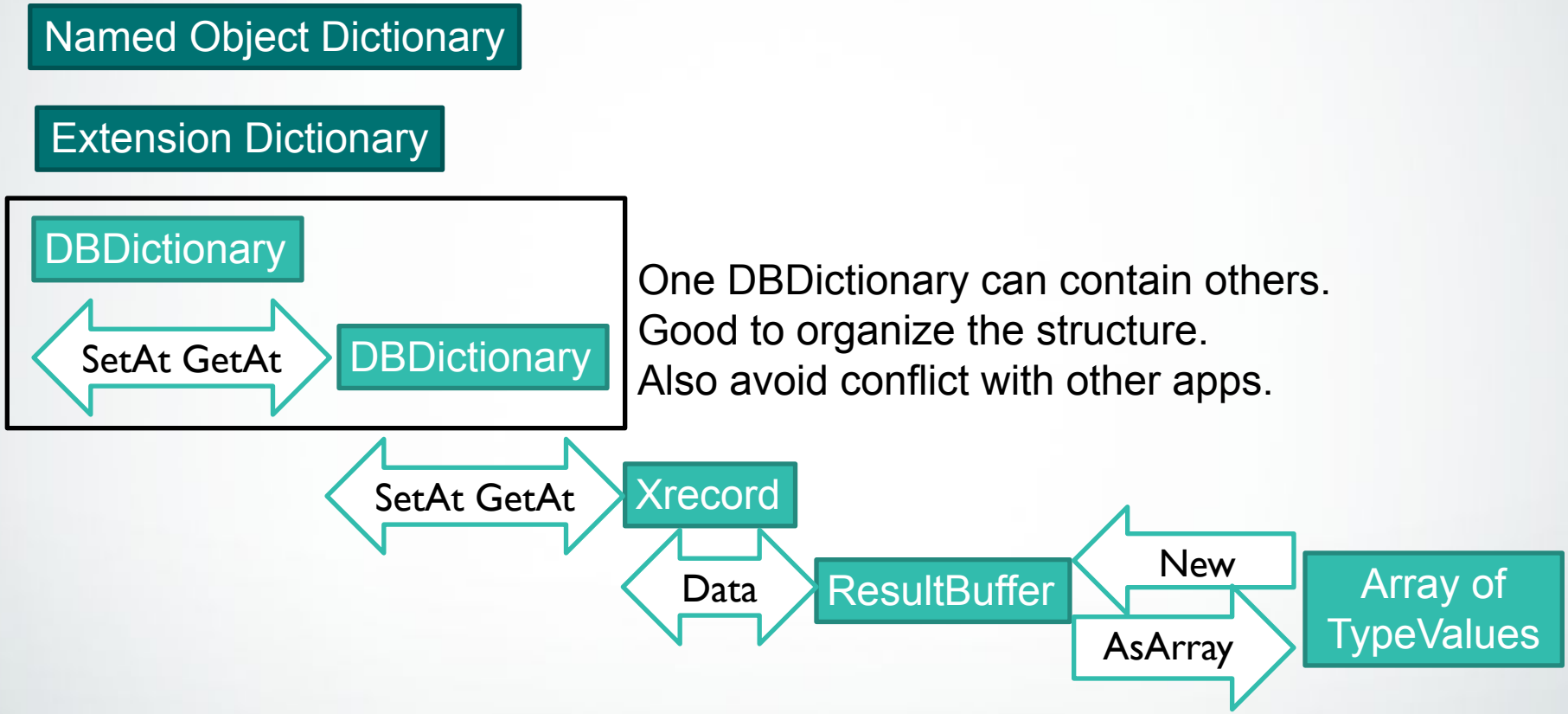
```
Dim db = HostApplicationServices.WorkingDatabase

Dim NOD As DBDictionary =
trans.GetObject(db.NamedObjectsDictionaryId,
                OpenMode.ForWrite, False)
```

- To *create* an ExtensionDictionary for an entity

```
myEntity.CreateExtensionDictionary()
```

Dictionary Hierarchy



Iterating Through Containers

- Objects that are enumerable
  - Symbol Tables
  - Block Table Records
  - Dictionaries
  - Polylines
  - PolyFaceMesh & PolygonMesh
  - ACIS Solids
    - Called traversers
  - BlockReferences (Inserts)
    - Only useful when attributes are present

Symbol Table Traversal

- Start with a database pointer
  - HostApplicationServices.WorkingDatabase – Get used to this one!

- Iterate down into each sub-table from there…

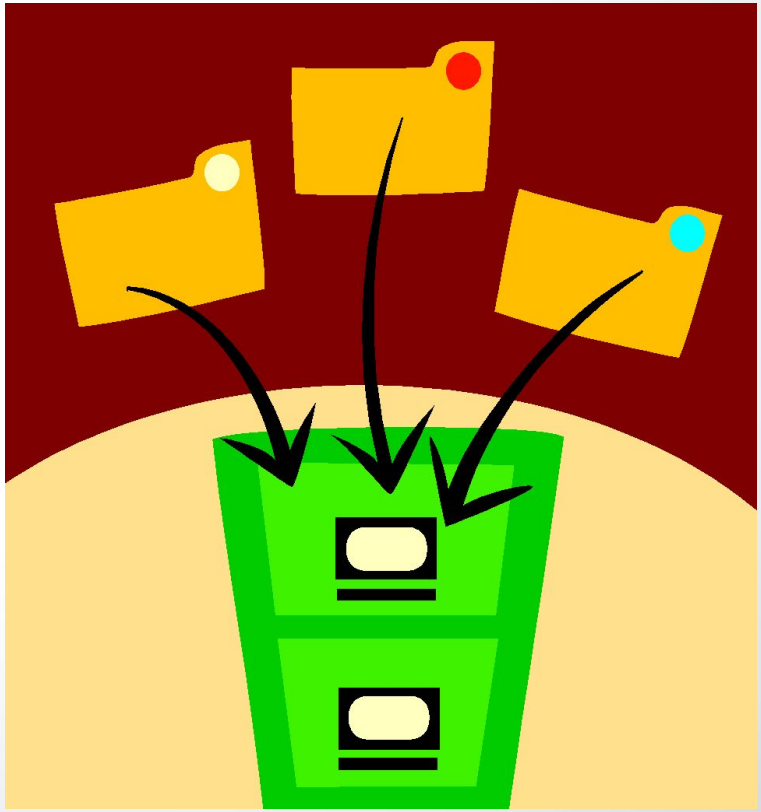
```
Dim db As Database = HostApplicationServices.WorkingDatabase()

Dim bt As BlockTable = trans.GetObject(db.BlockTableId, OpenMode.ForWrite)

Dim id As ObjectId
 For Each id In bt
    Dim btr As BlockTableRecord = trans.GetObject(id, OpenMode.ForRead)
 Next
```

Lab 5

Adding Custom Data



Class Agenda

Lectures and Labs

- Overview of .NET.
- AutoCAD .NET Visual Studio project settings – Hello World!
- User Interaction - User Input and Entity Selection
- Database Fundamentals – Symbol tables, Transactions
- User Interface design – Win Form Dialogs and Palettes
- Event handling – Reacting to AutoCAD Events in .NET. Database
- Dictionaries, XRecords, Table Traversal
- Point Monitor
- Jigs
- Additional User Interface Elements

Input Point Monitor

- Allows us to monitor relevant input in AutoCAD
- Provides relevant data to the input received – Osnap, Draw context, various computed points, Entities underneath aperture etc.
- Allows you to draw temporary graphics, and to easily implement tooltips.
- Created with the PointMonitor event of the editor
  The delegate is a PointMonitorEventHandler

```
<CommandMethod("addPointmonitor")> _
Public Sub startMonitor()
    Dim ed As Editor = Application.DocumentManager.MdiActiveDocument.Editor
    AddHandler ed.PointMonitor, New PointMonitorEventHandler(AddressOf MyPointMonitor)

End Sub

Public Sub MyPointMonitor(ByVal sender As Object, ByVal e As PointMonitorEventArgs)
```

Input Point Monitor

- AppendToolTipText
- Context
  - GetPickedEntities
  - FullSubentityPath

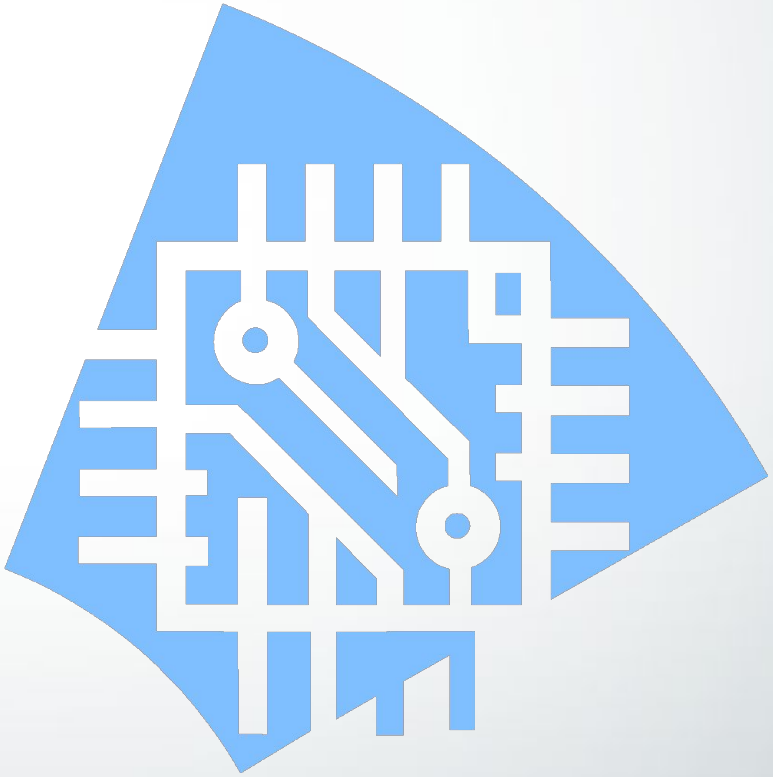
```
Public Sub MyPointMonitor(ByVal sender As Object, ByVal e As PointMonitorEventArgs)
    Dim fullEntPath() As FullSubentityPath = e.Context.GetPickedEntities()
```

  - DrawContext
    - Geometry.Draw

```
e.Context.DrawContext.Geometry.Draw(circle)
```

Lab 6

PointMonitor



Class Agenda

Lectures and Labs

- Overview of .NET.
- AutoCAD .NET Visual Studio project settings – Hello World!
- User Interaction - User Input and Entity Selection
- Database Fundamentals – Symbol tables, Transactions
- User Interface design – Win Form Dialogs and Palettes
- Event handling – Reacting to AutoCAD Events in .NET. Database
- Dictionaries, XRecords, Table Traversal
- Point Monitor
- Jigs
- Additional User Interface Elements

Jigs

- Allows you to graphically manipulate and form an Entity in real time.

- Two types of Jig available
  - EntityJig – Controls only one entity
  - DrawJig – Controls one or more.

  - Need to use a Class that inherits from EntityJig or DrawJig

Jigs

- The constructor for this class takes the entity being jigged.
- Use the Editor Drag function to start the Jig
  - Pass in the Jig

```
Dim circle As Circle = New Circle(Point3d.Origin, Vector3d.ZAxis, 10)
Dim jig As New MyCircleJig(circle)
Dim ed As Autodesk.AutoCAD.EditorInput.Editor = _
                    Application.DocumentManager.MdiActiveDocument.Editor
Dim promptResult As PromptResult = ed.Drag(jig)
```

Jig Functions

- Two functions that must be overridden
- Sampler
  - Used to get input from the user
- Update
  - Used to update the entity that is being jigged.

Jig Function - Sampler

- One Argument Passed into this function
  - JigPrompts
    - AcquirePoint, AcquireDistance
- Returns SamplerStatus
  - NoChange
  - OK

```
Protected Overrides Function Sampler _
(ByVal prompts As Autodesk.AutoCAD.EditorInput.JigPrompts) _
    As Autodesk.AutoCAD.EditorInput.SamplerStatus
```

Jig Function - Update

- Change the Properties of the Entity
- Use a Select Case
  - To Get multiple inputs

```
Protected Overrides Function Update() As Boolean
    Select Case (currentInputValue)
        Case 0
            CType(Me.Entity, Circle).Center = centerPoint
        Case 1
            CType(Me.Entity, Circle).Radius = radius
    End Select
End Function
```

Lab 7

Jigs

Class Agenda

Lectures and Labs

- Overview of .NET.
- AutoCAD .NET Visual Studio project settings – Hello World!
- User Interaction - User Input and Entity Selection
- Database Fundamentals – Symbol tables, Transactions
- User Interface design – Win Form Dialogs and Palettes
- Event handling – Reacting to AutoCAD Events in .NET. Database
- Dictionaries, XRecords, Table Traversal
- Point Monitor
- Jigs
- Additional User Interface Elements

Context menu

- Application Level
  - Application.AddDefaultContextMenuExtension
- Object Level
  - Application. AddObjectContextMenuExtension –per RXClass

Tabbed Dialog Extensions

- Create a new tab inside Options dialog

- Create a user control
- Hook to Application.DisplayingOptionDialog event
- Add the tab (user control) in the event handler

```
private void TabHandler(object sender, TabbedDialogEventArgs e)
 {
  myCustomTab myCustomTab = new myCustomTab();

  TabbedDialogAction tabbedDialogAct = new TabbedDialogAction(myCustomTab.OnOk);

        TabbedDialogExtension tabbedDialogExt = new
TabbedDialogExtension(myCustomTab, tabbedDialogAct);

        e.AddTab("My Custom Tab", tabbedDialogExt);
 }
```

Drag and Drop

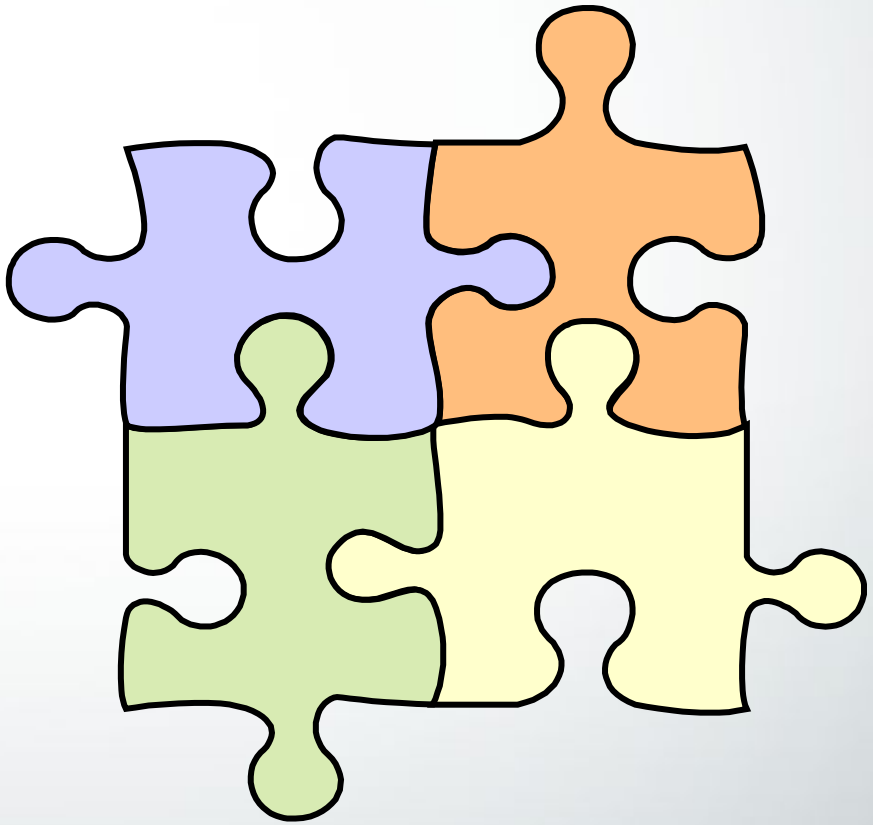
- Handle MouseMove
- Create an instance of your DropTarget class
- Call Application.DoDragDrop

```
Application.DoDragDrop ( this, this, Forms.DragDropEffects.All,
                                     new MyDropTarget() );

public class MyDropTarget : Autodesk.AutoCAD.Windows.DropTarget
 {
      public override void OnDrop(System.Windows.Forms.DragEventArgs e)
      {
      }
}
```

Lab 8

Additional User Interface Elements
(Non CUI based)



More API Resources

Blogs

Through the Interface (AutoCAD.NET)
http://through-the-interface.typepad.com/through_the_interface/

AutoCAD DevBlog
http://adndevblog.typepad.com/autocad/

Developer Center : http://www.autodesk.com/developautocad

API Training Classes : www.autodesk.com/apitraining

DevTVs, Recorded Webcasts, Code Samples - ADN website
http://adn.autodesk.com

Discussion Groups : http://discussion.autodesk.com

Thank You !