SD6174-L - Sparring with Autodesk® ObjectARX®—Round 2 Stepping into the Ring

Lee Ambrosius – Autodesk, Inc.

Principal Learning Content Developer – IPG

AutoCAD Products – Learning Experience

Where Am I and Who Should Be Here

You are in session:
SD6174-L - Sparring with Autodesk® ObjectARX®—Round 2
Stepping into the Ring

You should know:

- AutoCAD 2015 (or AutoCAD 2013 and later)
- Previous programming experience, C++ would be a plus (no pun intended)

Who Am I?

My name is Lee Ambrosius

- Principal Learning Content Developer at Autodesk
- AutoCAD user for over 20 years
- Work on the Customization, Developer, and CAD Administration documentation
  - Customizing and programming for over 18 years
  - AutoLISP/DCL, VBA, Managed .NET, and ObjectARX/MFC
- Author and Technical Editor
  - AutoCAD and AutoCAD LT Bible Series
  - AutoCAD Customization Platform Series

Who Are the Lab Assistants?

The lab assistants for this session are:

<Placeholders>
<Placeholders>
<Placeholders>

Their roles are to

- Help out when you get stuck
- Ensure no one gets left behind

Session Rules

A few rules for this session:

- Silent your mobile phone and any other device
- If you have to leave at anytime, please do so quietly
- Hold all questions the end
- If you get stuck, raise your hand and one of the lab assistants will help you out

Thanks for Your Cooperation

Getting Started

Session Handouts

The handouts are broken into two parts:

- Supplemental – Content covered in the lecture and for the flight back
- Exercises – What will be covered in the lab

What You Will Learn Today

By the end of the lab, you will know how to:

- Create a project in Microsoft® Visual Studio®
- Create a command and work with objects in a drawing
- Display messages to the user and request user input
- Load an ObjectARX application into AutoCAD
- Perform basic debugging of an ObjectARX project

Creating a New ObjectARX Project

Accessing ObjectARX Libraries

The libraries and headers are installed as part of the ObjectARX SDK. By default, the latest release is installed to:

- C:\ObjectARX 2015

The libraries are broken up into shared, and 32-bit and 64-bit specific files:

- inc
- inc-win32 and lib-win32
- inc-x64 and lib-x64

Creating a New ObjectARX Project

To define a new ObjectARX project you must:

1. Create a new Win32 project in Microsoft Visual Studio
2. Set the build settings
3. Reference the include and library files
4. Create or reference a definition file that exports the *acrxEntryPoint* and *acrxGetApiVersion* functions
5. Add a source code file that defines the *acrxEntryPoint* function

Basic *acrxEntryPoint* Function Implementation

```
AcRx::AppRetCode acrxEntryPoint(AcRx::AppMsgCode msg, void* appId)
{
     switch (msg) {
     case AcRx::kInitAppMsg:
          acrxDynamicLinker->unlockApplication(appId);
          acrxDynamicLinker->registerAppMDIAware(appId);

          // Add tasks here that should happen when loading the ARX file
          break;
     case AcRx::kUnloadAppMsg:
          // Add tasks here that should happen when unloading the ARX file
          break;
     }

     return AcRx::kRetOK;
}
```

acrxGetApiVersion Function Implementation

Implemented by referencing *rxapi.lib* as part of the Win32 Visual Studio project file

Creating an ObjectARX Project

Do exercise “E1 - Create a New Project from Scratch”

This exercise has you

- Creating a new Win32 project in Microsoft Visual Studio
- Configuring a new project to build an ObjectARX application
- Defining the *acrxEntryPoint* function and the minimalist code

Compiling and Loading a Project

Compiling and Loading a Project

ObjectARX projects must be compiled before they can be loaded into AutoCAD.

When an ObjectARX project is compiled, it has an ARX file extension.

Microsoft Visual Studio allows you to compile a project for:

- Debug – Used for testing purposes
- Release – Final build of the project for deployment to users

Compiling and Loading a Project

An ObjectARX file can be loaded into AutoCAD using one of the following methods:

- APPLOAD command
- AutoLISP arxload function
- Drag and drop into the drawing window (Windows only)
- /ld command line switch used for Desktop shortcuts (Windows only)
- Plug-in bundle

Compiling and Loading a Project

Do exercise “E2 - Compile and Load an ObjectARX Project”

This exercise has you

- Building a debug version of the ObjectARX application
- Loading an ObjectARX application
- Unloading an ObjectARX application

Defining a New Command

Defining a New Command

The *acedRegCmds* macro is used to register a new command.

Commands require:

- Command group name
- Global or international command name
- Local command name
- Command flag(s)
- Name of the function to execute

Defining a New Command

```
// Registers a command named Uno
acedRegCmds->addCommand(_T("AU2014App"), _T("First"),
                        _T("Uno"), ACRX_CMD_TRANSPARENT, Greetings);

// Callback function to start when the Uno command is used
static void Greetings()
{
    acutPrintf(_T("\nHello AU2014!!!"));
}

// Removes the command group AUCommands
acedRegCmds->removeGroup(_T("AUCommnds"));
```

Defining a New Command

Do exercise “E3 - Define a Custom Command”

This exercise has you

- Defining a function that displays a message at the Command prompt
- Registering a command
- Removing a command group
- Testing a global and local command name

Accessing Drawing Objects

Accessing Drawing Objects

Before you can working with drawing objects, you must obtain a database from the:

- Current drawing
- Document Manager

The following gets the database of the current drawing:

```
AcDbDatabase *pDb = acdbHostApplicationServices()->workingDatabase();
```

Graphical Objects

Objects you add and interact with in the drawing area:

- Lines
- Circles
- Polylines
- Helixes
- 3D solids
- Among others…

Graphical Objects

Are added to:

- Model space (*MODEL_SPACE)
- Paper space (*PAPER_SPACE0, *PAPER_SPACE1, …)
- Block definition

appendAcDbEntity method is used to add a new object to a block

Opening and Closing Objects

```
// Opens the Block table for read-only
AcDbBlockTable *pBlockTable;
acdbHostApplicationServices()->workingDatabase()->
    getBlockTable(pBlockTable, AcDb::kForRead);

// Closes the block table
pBlockTable->close();
```

Opening and Closing Objects

```
// Creates a new Line object in memory
AcDbLine *pLine = new AcDbLine(startPt, endPt);

// Adds the new Line object to Model space
pBlockTableRecord->appendAcDbEntity(pLine);
```

Defining a New Command

Do exercise “E4 - Add a Line to Model Space”

This exercise has you

- Defining a function that adds a new graphical object to model space
- Getting a referenfce to model space
- Creating a new line object
- Appending the new line object to model space

Accessing Non-graphical Objects

Non-Graphical Objects

Non-graphical objects are:

- Text, dimension, multileader, and table styles
- Groups
- Blocks
- Named viewports, views, and UCSs
- Layouts
- Among others…

Non-Graphical Objects

Non-graphical objects are stored in:

- Symbol tables
- Dictionaries

Non-Graphical Objects

Check for the existence of an non-graphical object before

- Trying to work with an existing object
- Creating a new object

When checking for the existence of an object, you should open its parent for read.

The read/write state of an object can be changed by using:

- *upgradeOpen* method – Read to write
- *downgradeOpen* method – Write to read

Adding a New Non-graphical Object

Use the *add* method to add a new non-graphical object

```
// Add the new layer to the Layer table
pLayerTable->add(pLayerTableRecord);
```

Adding a New Non-graphical Object

Do exercise “E5 - Create a New Layer”

This exercise has you

- Defining a function that adds a new non-graphical object; a LayerTableRecord to the LayerTable
- Checking for the existence of an existing layer
- Creating and modifying a new layer
- Appending the new LayerTableRecord to the LayerTable

Stepping through Objects in Model Space

Iterating a Block Table Record

An iterator allows you to step through a symbol table.

```
AcDbBlockTableRecordIterator *pItr = pBlockTableRecord->newIterator(pItr);
AcDbEntity *pEnt;

for (pItr->start(); !pItr->done(); pItr->step())
{
    pItr->getEntity(pEnt, AcDb::kForRead);
    acutPrintf(_T("\nClass name: %s"), pEnt->isA()->name());
    pEnt->close();
}

delete pItr;
```

Iterating a Block Table Record

Do exercise “E6 - Step through All Objects in the Database”

This exercise has you

- Defining a function that steps through the block of the current space
- Outting the class name of each entity in the current space
- Displaying the AutoCAD Text Window or expanding the Command Line window

Requesting a Point

Requesting a Point

The *acedGetPoint* function allows the user to specify a point.

```
ads_point pt1;
acedGetPoint(NULL, _T("\nSpecify first point: "), pt1);

// Defines a new point data type
AcGePoint3d startPt(pt1[0], pt1[1], pt1[2]);
```

Requesting a Point

Do exercise “E7 - Add a Line Using User Input”

This exercise has you

- Defining a function that prompts for two points
- Adding a line based on the two points

Prompting for a Keyword and Selecting Objects

Prompting for a Keyword

acedInitGet allows you define the keywords that are supported by the *acedGetKword* method.

```
// Prompt the user for a keyword/option
acedInitGet(0, _T("1 2 3 Red Yellow Green Bylayer") );
int retVal = acedGetKword(
    _T("\nEnter a color [Red/Yellow/Green/Bylayer] <Red>: "),
    kWord);
```

Selecting Objects

acedSSGet allows the user to select objects on the drawing area.

```
// Prompts for a selection set
ads_name sset;
acedSSGet(NULL, NULL, NULL, NULL, sset);
```

Selecting Objects

```
// Gets the number of entities in a selection set
long lSSCnt = 0;
acedSSLength(sset, &lSSCnt);

// Returns the first object in a selection set
ads_name eName;
acedSSName(sset, 0, eName);

// Release a selection set
acedSSFree(sset);
```

Prompting for a Keyword and Selecting Objects

Do exercise “E8 - Select Objects and Request a Keyword”

This exercise has you

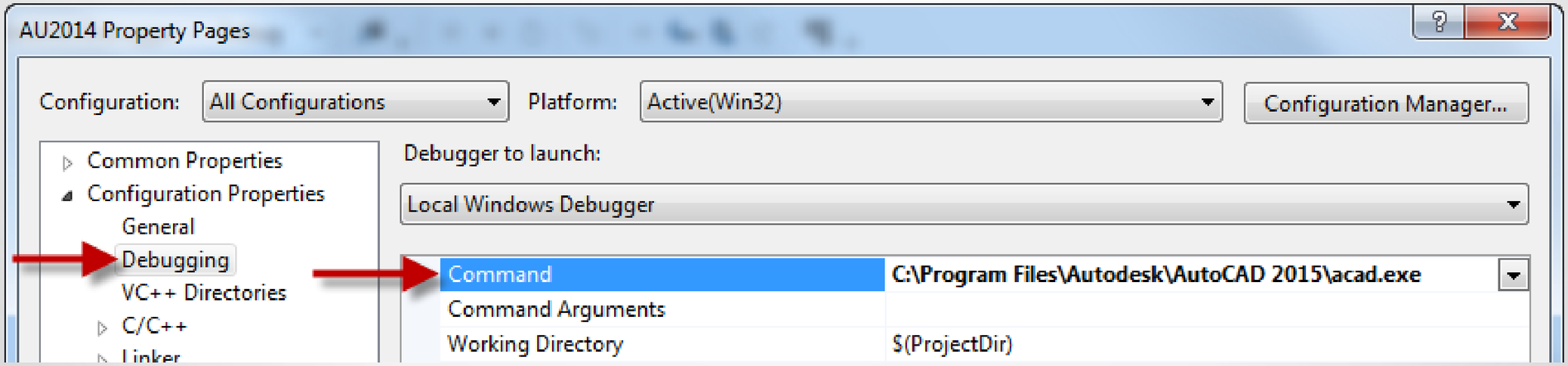
- Defining a function that prompts for objects
- Prompts for a keyword
- Stepping through the objects in the selection set
- Modifying the color of the selected objects based on keyword chosen

Debugging a Project

Debugging a Project

Microsoft Visual Studio allows you to debug your application in AutoCAD.

- Project must be compiled for debug
- AutoCAD must be the application started when debugging begins



Debugging a Project

You can step through each line by setting a breakpoint in Visual Studio.

```
static void addLine()
{
    // Get the current database
    AcDbDatabase *pDb = acdbHostApplicationServices()->workingDatabase();

    // Open the Block Table for read-only
    AcDbBlockTable *pBlockTable;
    pDb->getSymbolTable(pBlockTable, AcDb::kForRead);

    // Get the Model Space block
    AcDbBlockTableRecord *pBlockTableRecord;
    pBlockTable->getAt(ACDB_MODEL_SPACE,
                       pBlockTableRecord, AcDb::kForWrite);
    pBlockTable->close();
```

Debugging a Project

Do exercise “E9 - Debug an ObjectARX Application”

This exercise has you

- Debugging an ObjectARX application
- Setting breakpoints
- Stepping through an ObjectARX application

Building a Project for Release

Building a Project for Release

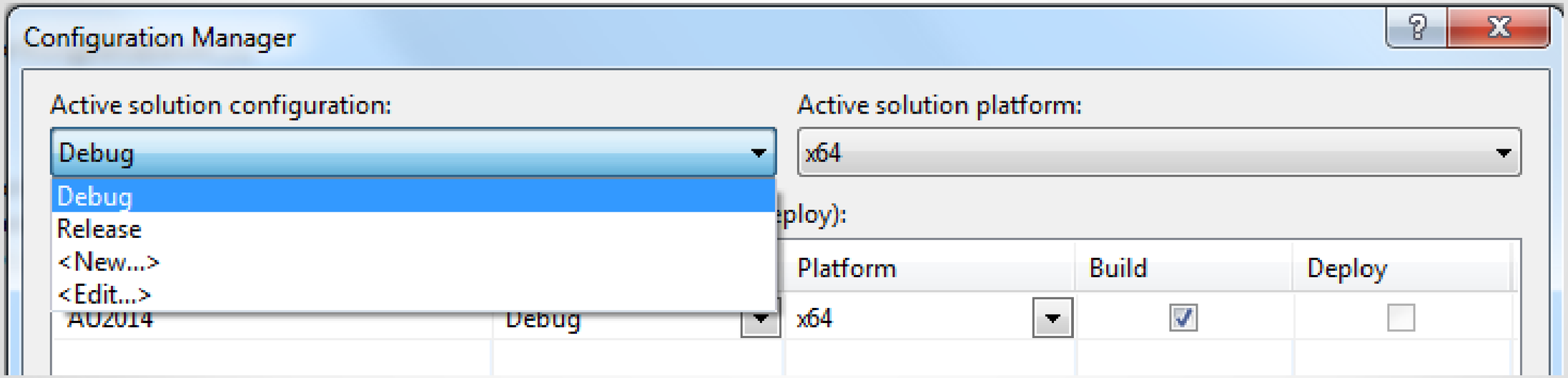
ObjectARX applications must be compiled for Release for others to use them.

Compiling an ObjectARX application for Release:

- Removes debugging information and references release DLLs
- Create a smaller file because of the missing debug information

Building a Project for Release

Use the Configuration Manager to switch between Debug and Release project settings.



Building a Project for Release

Do exercise “E10 - Build an ObjectARX Application for Release”

This exercise has you

- Changing project configurations
- Building an ObjectARX application for release

Extra Exercises

Extra Exercises

Do exercise “Extra E1 - Work with System Variables and Use Commands”

This exercise has you

- Working with system variables
- Executing a command

Extra Exercises

Do exercise “Extra E2 - Work with Input and Single Line Text Objects”

This exercise has you

- Prompting the user for an integer, string, and point value
- Adding single-line text objects to the current space

Extra Exercises

Do exercise “Extra E3 - Defining AutoLISP Functions”

This exercise has you

- Defining and registering functions that can be executed with AutoLISP
- Executing the custom AutoLISP functions

Final Thoughts and Questions

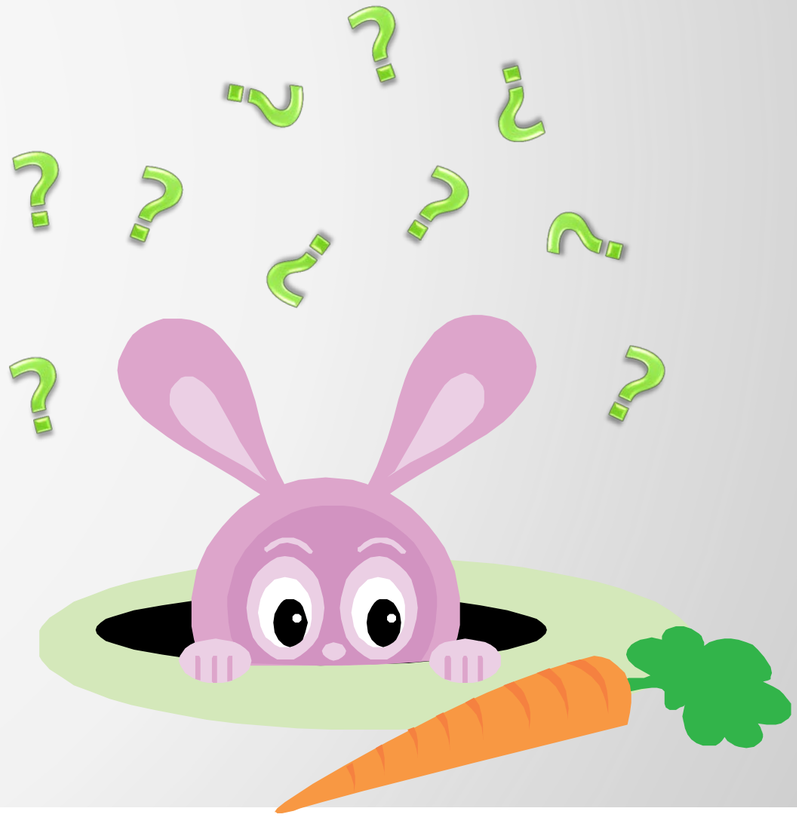
Final Thoughts and Questions

Scripting and programming can

- enhance productivity
- improve or introduce new workflows

Programming has many similarities to the rabbit hole in *Alice's Adventures in Wonderland* by Lewis Caroll. Both

- are virtually endless, and
- hold many mysteries that are waiting to be discovered



Closing Remarks

Thanks for choosing this session and hope you got something out of it.

Do not forget to complete the online evaluation.

If you have any further questions contact me via:

email: lee.ambrosius@autodesk.com
twitter: http://twitter.com/leeambrosius

Enjoy the rest of the conference.