SD21033

Get More from 3ds Max with Custom Tool Development

Kevin Vandecar
Autodesk Inc.

Learning Objectives

- Learn how the 3ds Max Python and .NET API environments differ
- Learn how to get started in 3ds Max .NET API developments
- Learn how to get started in 3ds Max Python development
- Learn how to create a UI for your custom tools in .NET API or Python

Description

This class will discuss using the 3ds Max software development kit (SDK) for automating common tasks within 3ds Max software. We will spend time discussing the .NET API and the Python environments, and we'll compare them both. The Python SDK is using a newer API called MaxPlus, but it is not fully exposing the C++ API. While .NET API is exposing most of the C++ API, it can sometimes be more difficult to work with. We will compare them and show techniques for automating common tasks, and creating user interfaces to control your tools. This session features 3ds Max. AIA Approved

Your AU Expert…?

Kevin Vandecar has worked for Autodesk, Inc., for 20 years in various customization-related capacities, ranging from developer support and testing to software development and consulting. He has also worked for a large reseller/consulting firm, and he has a customer-centric sensibility. Kevin currently is a principal developer advocate and also the manager for the Media & Entertainment Autodesk Developer Network Workgroup. His current specialty is 3ds Max software customization areas, and also the Forge APIs.

3ds Max customization environments

Before we get started, let’s take a quick look at all the various ways of customization. 3ds Max has a long history, and customization has been part of it since its start. To begin, a short history lesson. In the beginning… the product started out as 3d Studio for DOS. Wow, long time ago. Many readers may not even have knowledge of DOS, and for that I am glad for you. ☺ But even then, there was a plugin environment called IPAS. Programming environments for desktop software richly improves the usability, and for 3ds Max it is no different. If interested in the history of 3ds Max, there are a few articles here:

http://cgpress.org/archives/cgarticles/the_history_of_3d_studio
http://area.autodesk.com/blogs/the-3ds-max-blog/experience-matters---tom-hudson

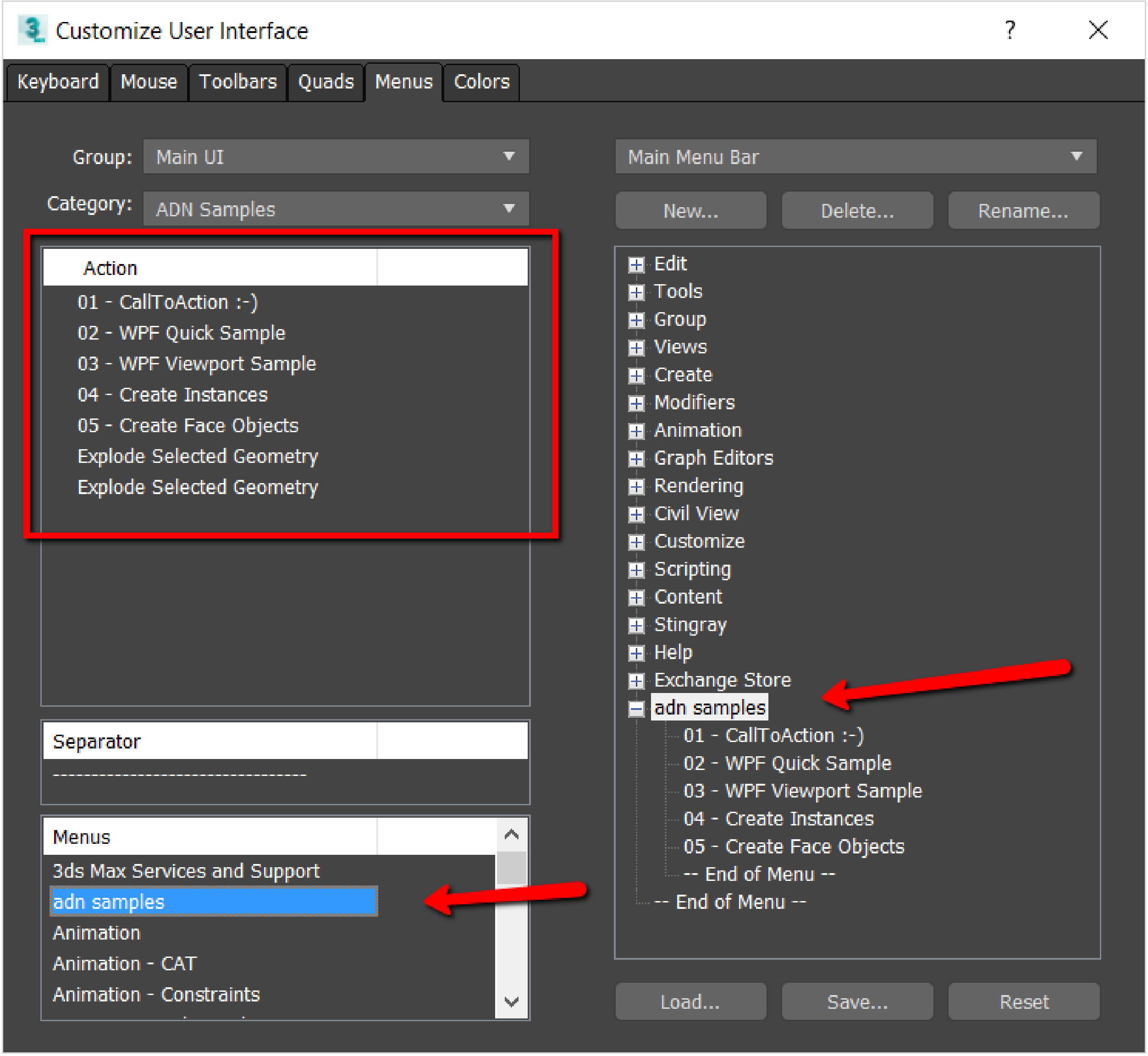
Looking at the history, shows that customization was an important aspect even from the beginning, and has led 3ds Max into the rich software it is today. In fact, many of the “out-of-box” features are provided as plugins. Not convinced? See the <install>\3ds Max <version>\StdPlugs folder for proof. Those are all plugins for the standard install. Additionally, now that 3ds Max is provided only as one version (previously there was also a 3ds Max Design flavor), those UI experiences are all part of the customization aspects of core 3ds Max.

Because this class is focusing on the newest programming tools, .NET API, and Python, I will not go into great detail. However, I think it is important to have the bigger picture, so in this handout I will spend a little more time pointing you to resources to help you understand the vast landscape of customization tools available in 3ds Max.

UI Customization

Let’s start with UI customization. You probably have already discovered this if you have used 3ds Max for more than a couple of hours. Basically the “Customize” menu is your entry point to various UI and runtime customizations (ie. paths to support files). There you will find the “Customize User Interface” dialog. This will provide you with the ability to create custom menus, toolbars, keyboard short cuts, mouse actions, and color schemes. All of these settings can be saved in a CUI file, and shared between users if you want to provide custom workflows and actions to a user base. For example, a modeling artist may need different tools to be present than an animation artist or rigger.

See below for a screen capture of the menu tab, that has a custom menu showing:



Group: Main UI

Category: ADN Samples

Main Menu Bar

Action
01 - CallToAction :-)
02 - WPF Quick Sample
03 - WPF Viewport Sample
04 - Create Instances
05 - Create Face Objects
Explode Selected Geometry
Explode Selected Geometry

Separator

Menus
3ds Max Services and Support
adn samples
Animation
Animation - CAT
Animation - Constraints

Edit
Tools
Group
Views
Create
Modifiers
Animation
Graph Editors
Rendering
Civil View
Customize
Scripting
Content
Stingray
Help
Exchange Store
adn samples
01 - CallToAction :-)
02 - WPF Quick Sample
03 - WPF Viewport Sample
04 - Create Instances
05 - Create Face Objects
-- End of Menu --
-- End of Menu --

The tabs are not always obvious on how to use them, but it is a good idea to take the time to understand the workflow, because it can be very powerful without having to write any code. In the above example, I have created a new menu called “adn samples” (lower arrow), and then you can drag that menu over to the right into the live “Main Menu Bar”. Once the menu is there, you can then drag actions. As you can see, the steps do not align with the layout of the dialog, but it works very well once you understand it. For more information, check out these sections in the help:

Customizing 3ds Max (top level customization topic in user docs)

Customize User Interface Dialog (specifics about Customize User Interface dialog)

Max Creation Graph

One of the newest and hottest features of 3ds Max customization, is the Max Creation Graph feature. It was introduced a few releases back, and has many great examples available already. The feature is basically visual programming, by providing a node-based editor to build graphs that produce 3ds Max commands and other functionality. Most nodes have inputs and outputs that you connect by dragging between them.

The feature is documented here: Max Creation Graph

The Autodesk MCG samples are provided on GitHub here:
https://github.com/ADN-DevTech/3dsMax-MCG-Samples

MAXScript

MAXScript has been a part of 3ds Max for a very long time. As result it is a vary mature and complete environment. This facility is ideal for automating tasks and can be enough for the most complex tasks. The challenge is that this environment is unique to 3ds Max, and has it's own syntax. It's certainly worth the investment if you are a power 3ds Max user. It is also the entry point for the Python environment.

Because of its rich history, there are many standard routines out there, and people are often sharing things in various forums. If you have a complex task outside your MAXScript skill set, always do a search for something already, because it's likely someone has already solved it (or come close. ☺) See here for details: MAXScript Introduction

C++ SDK

Before we touch on the newer environments of .NET API and Python, I also want to discuss the C++ 3ds Max SDK. This is really the root of all evil if you are not an experience programmer, or the interface to enlightenment if you are an experience C++ person. ☺ This environment is what the core 3ds Max engineering team will often use to develop new features, and it is also the place where almost anything can be changed or added to 3ds Max. Commercial developers will use this environment to provide rich new functionality. For example, the V-Ray renderer is a good example of a very mature and sophisticated C++ plugin. The particle system plugin is also a great example of a 3rd party developer providing rich functionality, that was attractive to Autodesk and became acquired and brought into 3ds Max as a core feature. Because of the complexity of C++ (3ds Max itself is developed in C++) there is great deal of knowledge required to be successful here, but it is also very powerful and provides the most flexibility. It is also considered a "native" language so it compiles into a pure binary that also provides protection of your code. Certain feature aspects can ONLY be created from this environment, too (for example renderers). The good news is that this is also a very mature environment and there are many, many resources available if you chose to dive-in.

See this link for specifics: 3ds Max SDK

Additional API help links:

http://www.autodesk.com/develop3dsmax	(3ds Max Developer Center page)
http://www.autodesk.com/me-sdk-docs	(Entry for all current M&E API docs)

.NET API

The .NET API is relatively new to 3ds Max and provides several managed assemblies. It's first showing was to provide interop with MAXScript.

Python

Python is also relatively new, and has two modules that help you utilize 3ds Max functionality. The older one is called MaxPlus and is a wrapper for a sub-set of the C++ native functionality. In 3ds Max 2017, there is new module, pymsx that allows you to call any of the MAXScript functionality directly from the Python language environment.

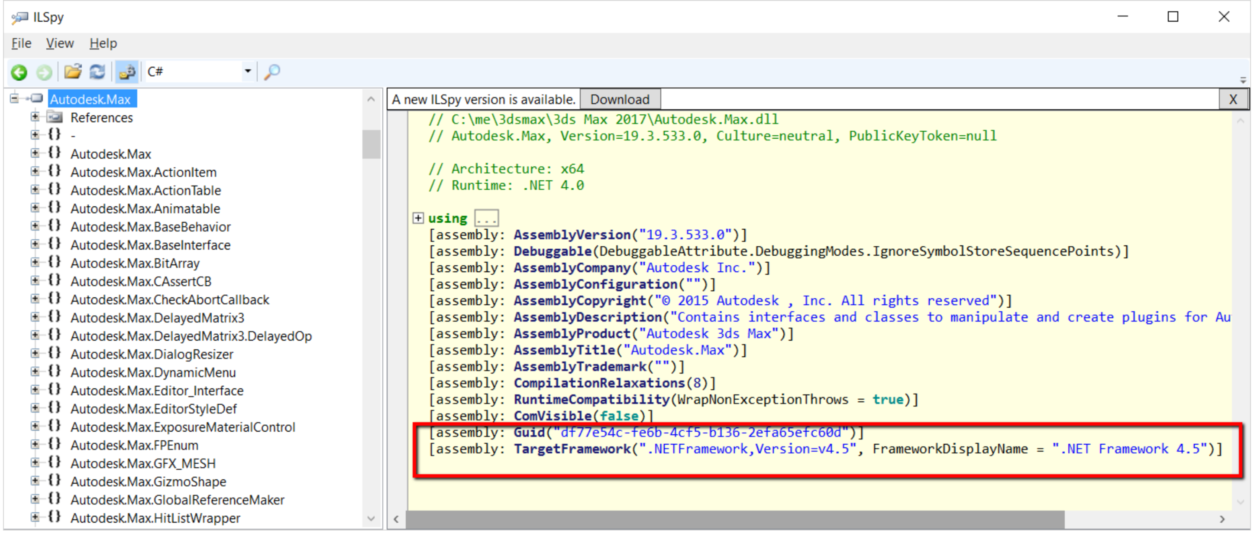
.NET vs. Python

First of all, your main consideration between choosing .NET API or Python is more about your personal preference than anything else. This presentation is meant to help you decide, based on key factors and differences with the 3ds Max environment. However, if you already have many tools in one or other language, you may decide to stick there, and deal with the caveats as you find them.

Some key differences for the environments

For the .NET environment, you will need Visual Studio. This gives you a nice editor and debugging environment. The community (free) edition should work fine, but you can also use other editions. The version is not as important, but recent versions are best. VS 2012, or VS 2015 are ideal candidates.

The .NET framework version needs to be matching what 3ds Max is using or later.
Later versions of framework are limited by 3ds Max assemblies features. Other libraries should work fine, though as long as frameworks are compatible. You can find the required versions by checking the assembly's requirement in a reflection tool (for example, below you can see it in ILSpy, a free open source reflector tool you can get from here: http://ilspy.net/)


ILSpy
File
View
Help
C#
Autodesk.Max
References
-
Autodesk.Max
Autodesk.Max.ActionItem
Autodesk.Max.ActionTable
Autodesk.Max.Animatable
Autodesk.Max.BaseBehavior
Autodesk.Max.BaseInterface
Autodesk.Max.BitArray
Autodesk.Max.CAssertCB
Autodesk.Max.CheckAbortCallback
Autodesk.Max.DelayedMatrix3
Autodesk.Max.DelayedMatrix3.DelayedOp
Autodesk.Max.DialogResizer
Autodesk.Max.DynamicMenu
Autodesk.Max.Editor_Interface
Autodesk.Max.EditorStyleDef
Autodesk.Max.ExposureMaterialControl
Autodesk.Max.FPEnum
Autodesk.Max.GFX_MESH
Autodesk.Max.GizmoShape
Autodesk.Max.GlobalReferenceMaker
Autodesk.Max.HitListWrapper
Download
X

A new ILSpy version is available.

```
// C:\me\3dsmax\3ds Max 2017\Autodesk.Max.dll
// Autodesk.Max, Version=19.3.533.0, Culture=neutral, PublicKeyToken=null

// Architecture: x64
// Runtime: .NET 4.0

using ...
[assembly: AssemblyVersion("19.3.533.0")]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
[assembly: AssemblyCompany("Autodesk Inc.")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCopyright("© 2015 Autodesk , Inc. All rights reserved")]
[assembly: AssemblyDescription("Contains interfaces and classes to manipulate and create plugins for Au
[assembly: AssemblyProduct("Autodesk 3ds Max")]
[assembly: AssemblyTitle("Autodesk.Max")]
[assembly: AssemblyTrademark("")]
[assembly: CompilationRelaxations(8)]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: ComVisible(false)]
[assembly: Guid("df77e54c-fe6b-4cf5-b136-2efa65efc60d")]
[assembly: TargetFramework(".NETFramework,Version=v4.5", FrameworkDisplayName = ".NET Framework 4.5")]
```

The C# compiler will also tell you, after referencing the assemblies, if you have not targeted the minimum required framework.

The Python environment has matured over past 3-4 releases. In 3ds Max 2017, we added support to allow the entire MAXScript API to be used directly from Python, and also the MAXScript editor now supports python. You can also use the PyCharm or other IDEs, (and was only choice prior to 2017). You can also install Python libraries, but need to be careful about the compatibilities with the 3ds Max distribution. You can find the required version of python, by simply running the version that comes with 3ds Max. For example, in 3ds Max 2016, you will find the python runtime here: <install path>\3ds Max 2016\python. Then executing the python.exe will give you this output: `Python 2.7.3 (default, Apr 10 2012, 23:24:47) [MSC v.1500 64 bit (AMD64)] on win32`. That tells you that you need python 2.7.3 compatible modules in order to install them to 3ds Max python instance. Note that in 3ds Max 2017, there is a special version of python that is called 3dsmaxpy.exe that is used instead. This custom version is required for the MAXScript API interop. Here, the idea is the same, running 3dsmaxpy gives this output: `Python 2.7.6 (default, Dec 12 2015, 01:09:31) [MSC v.1900 64 bit (AMD64)] on win32`. This shows that we upgraded to python 2.7.6, and your environment of custom libraries will also need to be matching.

Runtime Inspection

Both .NET and Python offer runtime inspection. However, the .NET one is really the only useful one due to strongly typed language support. In .NET it is called reflection, and allows you to inspect and execute APIs at runtime. In Python it is called Introspection, and allows the same, but more difficult because of loosely typed language. For further information, see these links:

.NET Reflection:
https://msdn.microsoft.com/en-us/library/f7ykdhsy(v=vs.110).aspx
http://www.codeproject.com/Articles/55710/Reflection-in-NET

Python Introspection:
http://zetcode.com/lang/python/introspection/
http://www.ibm.com/developerworks/library/l-pyint/index.html

Evolution

The evolution of both environments has taken place in the past few releases.

.NET API was first introduced to MAXScript without any true APIs. This allows you to call into other .NET assemblies from MAXScript. The .NET API fully matured with the Autodesk.Max, in the 2012 SAP release. Because the Autodesk.Max assembly is a wrapper for the C++ native functionality, we still have some APIs that are not well formed. Occasionally you may need a workaround or become blocked. For example, the `IntPtr` type can be found, and these are a result of the API generation tool not knowing what type to wrap the manage type as. Some of these types have marshaler tools (to convert the `IntPtr` native pointer to a managed type), so usually you can get around it, but be aware and plan ahead before spending a lot of time.

Because .NET API is a wrapper for the C++ APIs, it's documentation is included there. See this link as the entry point to the .NET API docs: 3ds Max .NET SDK

Remember that the C++ reference docs, as well as the C++ SDK samples can be help to understand the necessary techniques in .NET API.

Python is more recent and came first in the 2014 Extension release and included only the MaxPlus API. This API was a small subset of the C++ APIs and provided only the most important APIs. For example, the tri-mesh (Mesh) was supported, but quad-mesh (MNMesh) was not. In the 2017 release a significant change came to include the MAXScript APIs into Python. Also the MaxPlus API was expanded (including support for quad-mesh).

Python has its own section in the docs. See this link as the entry point to the Python docs:
About the 3ds Max Python API

.NET Specifics - Why use it?

.NET API is very mature and in other products. Based on Autodesk products, it is considered the CAD/Design choice. For example, Revit, AutoCAD, and Inventor all have mature managed environments.

.NET Specifics – What Can You Do?

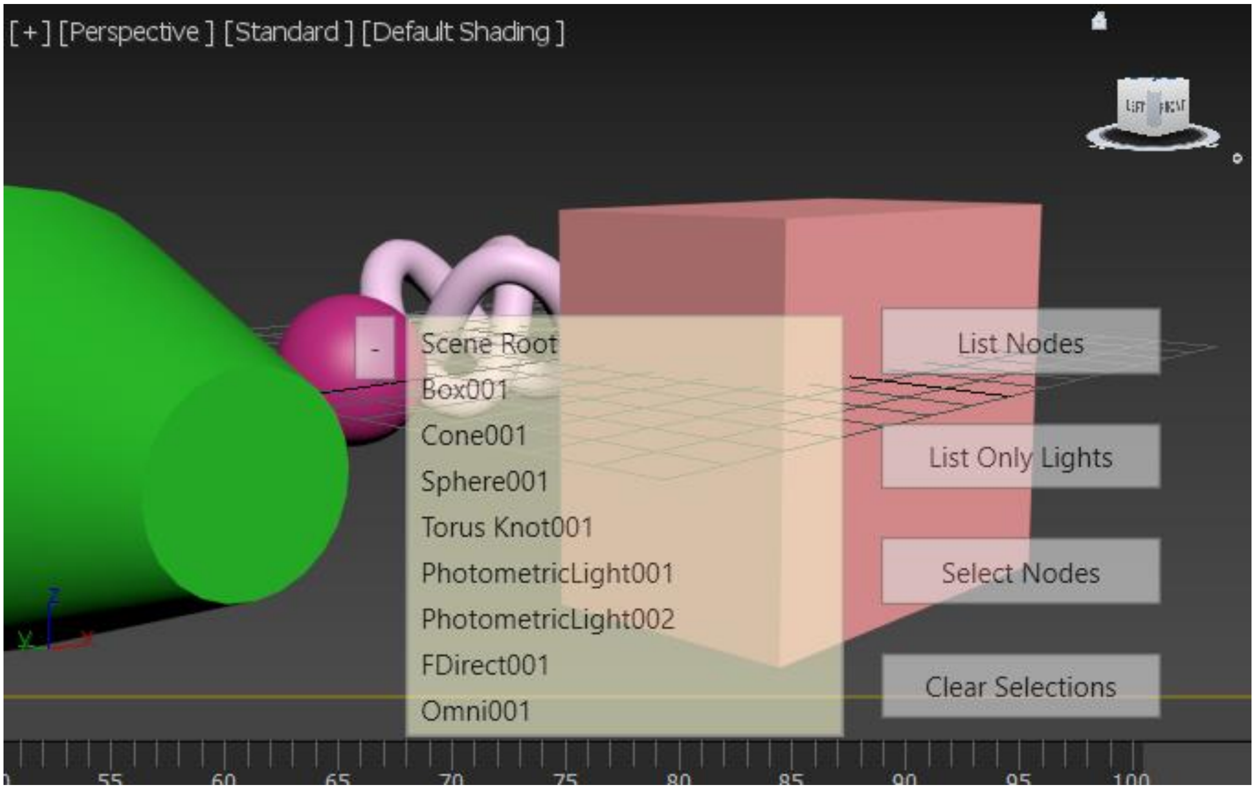
One of the biggest benefits to using .NET API in 3ds Max would be to automate tasks that you may perform manually, or with a script and have performance issues. One of the samples we will provide with this class demonstrates this idea. The Explode Geometry sample will take the selected node's geometry, and iterate its underlying mesh, and then allow you to separate the geometry into individual faces. You can also control several other options:

- Add Shell Modifier -- will add the modifier to the resulting face object, and apply the given shell offset.
- Add Edit Mesh Modifier -- will add the modifier to the stack resulting in edit mesh operation being immediately available.
- Collapse Modifier Stack -- will collapse the stack down to become an editable mesh as the end result.
- Center Pivot -- will center the pivot on each resulting face.
- Delete Original -- will remove each original node that was used to create the new faces.

All of those operations and options could be done manually, but would take several minutes of user interaction, and also require much experience on how. By developing a tool, it saves much time, and also provides specific workflow so that if the operations need to be done in many times, and by different people, then it becomes consistent and precise.

The .NET environment provides interop with both MAXScript and also the C++ SDK trhough the C++/CLI environment.

You can also create nice UI elements with your tools. For example, looking at the AdnCuiSamples example code, you can see the viewport overlay, that could not be easily done in MAXScript, or even in C++.



This is possible by using the WPF (Windows Presentation Foundation) system. Notice how the UI has some transparency and can be placed over the viewport.

When using .NET API, you can also use components that are part of the .NET environment. One example is the Linq feature. This allows you to build and iterate lists of information very easily. An example can be seen in the code block below:

```
// Use LINQ to filter for lights!
var sceneLights = from nodeLight in m_sceneNodes
             where nodeLight.ObjectRef is ILightObject
             select nodeLight;
```

In this simple code block, we can filter all the lights in the scene from the list of all nodes with a single statement of code.

How to Setup .NET plugin setup for 3ds Max

1. Create new managed "class library" project using Visual Studio.

2. Add references to the necessary 3ds Max assemblies.

   TIP: The 3ds Max assemblies are not registered with the GAC, so you need to use the “browse” tab of the “Add Reference” dialog. You will then browse to the 3ds Max installation folder, and choose the assemblies. Because there is no “managed assembly” naming convention for the 3ds Max assemblies, you have to find each one in the directory of many DLLs. The browse tab is a file-style dialog and supports multi-select, but because there are so many general DLLs it is hard to use it. It is best just to select them one at a time. The best long-term solution is to create one blank project, hand edit into the environment variables (see below) and then use that blank project as a template for new projects.

3. Set the "Copy Local" property to false on all 3ds Max Assemblies.

   WARNING: After selecting the assemblies, you MUST change the **“Copy Local”** flag to **False**. To do this, multi-select all the references in the Solution Explorer window, then in the Properties Window change Copy Local to False. This setting is used to tell the compiler whether it should copy the assembly to your project’s local directory, rather than reference it from its home location. Because these files should always be present with any 3ds Max install, it can cause 3ds Max to error or even crash if you try to package these with a distribution. Basically 3ds Max may try to load them again depending on the location where they are copied. Only the assemblies included with a 3ds Max installation should be loaded, and 3ds Max handles that aspect.

4. Choose and setup how to get your assembly into the 3ds Max loading directory. The special auto-load directory is <3dsMax installation>\bin\assemblies>. The best way for this during development, is to add a post build copy operation. The sample code uses this approach as it is safer than building directly to this location.

   Tip: Although the UI aspects of a Visual Studio managed project does not support it, you can use environment variables by hand editing the project file and inserting the variables into the appropriate locations. These variables will become expanded in the UI and cannot be edited there, but this is very useful for locating the 3ds Max installation directory and allowing your projects to be independent of fixed paths. The variables that are present after an installation of the 3ds Max products are in the form of: ADSK_3DSMAX_<platform>_<version>. For example, the sample code for this class uses ADSK_3DSMAX_x64_2017. Although the 3ds Max SDK is not required for .NET API programming, it can be helpful to have installed for reviewing some of the C++ samples for techniques. The SDK also creates an environment variable in the form of: ADSK_3DSMAX_SDK_<version>.

   WARNING: Note that 3ds Max will attempt to load any DLLs in the <3dsMax installation>\bin\assemblies> directory and its subdirectories, so make sure to place only managed assemblies meant to be loaded into 3ds Max. Any other assemblies or supplemental DLL models should be located in another place.

Assemblies

Autodesk.Max.dll - contains the "enhanced functionality" introduced in 2012 Subscription Advantage Pack (SAP) in September of 2011. Autodesk.Max is the root namespace. Use the IGlobal Interface, and Interface14 as the primary entry points. From the C++ SDK, functions with zero arguments and names beginning with "Get" are mapped to properties. IGlobal contains Create method for types that can be instanced. Class IDs are required to identify object types.

UiViewModels.dll - Contains the Cui functionality and is now part of 2013 going forward. This same Cui functionality for 2012 is in MaxCustomControls.dll.

AssemblyLoader.dll - Contains the Loader class providing functionality to load other assemblies from another location.

CSharpUtilities.dll - Utility assembly. Various utilities for 3ds Max including classes for file manipulation, strings handling, etc. Also contains WPF utilities, including the ability to get 3ds Max window and viewport information.

ManagedServices.dll - Utility assembly. Provides some basic notifications that are also used in the 3ds Max UI. These are better formed than the ones in the Autodesk.Max.dll wrappers, however, they are also older and complete notification functionality is not provided.

MaxCustomControls.dll - Utility assembly. Contains some older Windows forms functionality. Windows forms are not ideal to use any longer, unless you absolutely need them (for example you want to reuse some older UI code). It is better to use WPF when starting from new. However, there is a 3ds Max specific form class that allows a windows form to behave well within the 3ds Max main window.

NOTE: Each of the utility assemblies mentioned above are worth exploring when needing 3ds Max utilities including things like Windows handling and notifications.

ExplorerFramework.dll - Abstract explorer framework classes upon which the Scene Explorer is built. It can be used for creating a new node traversal for arbitrary scene explorers.

SceneExplorer.dll - Specification of the ExplorerFramework components for the SceneExplorer with bindings to 3ds Max data.

How to be loaded and executed

There are three ways that your assembly can be loaded and executed. The first is automatic by the simple placement of your assembly in the <3dsmax.exe folder>\bin\assemblies folder as discussed in project setup. The second way involves using the APIs from the AssemblyLoader module. With these APIs you still need at least one assembly to be placed in the bin\assemblies location to setup the APIs to load other modules from other locations. Also note that the CuiActionCommandAdaptor initialization is not supported through this loader. The third way uses the Autodesk App store package XML to specify the module to load and is placed in the %programdata%\Autodesk\ApplicationPlugins folder (see app store information for details). The Explode Geometry includes a sample package that shows this technique.

The Cui Action facility using the CuiActionCommandAdaptor class is one way to setup an “action” during the loading process. Actions can then be added to the UI through the Customize User Interface feature, or through other API access. These UI customizations are stored in a CUI file. For more information on how the Customize User Interface dialog works with Actions, see the 3ds Max documentation: Customize User Interface.

This Cui Action facility allows you to easily hook into the UI and allows for flexible options as to where the command ends up (for example, keyboard shortcut, menu, toolbars, ribbon panel, etc.)

The assembly loading entry point allows execution into your code during the load process. This option allows loading and operations similar to the C++ Global Utility Plug-in (GUP).

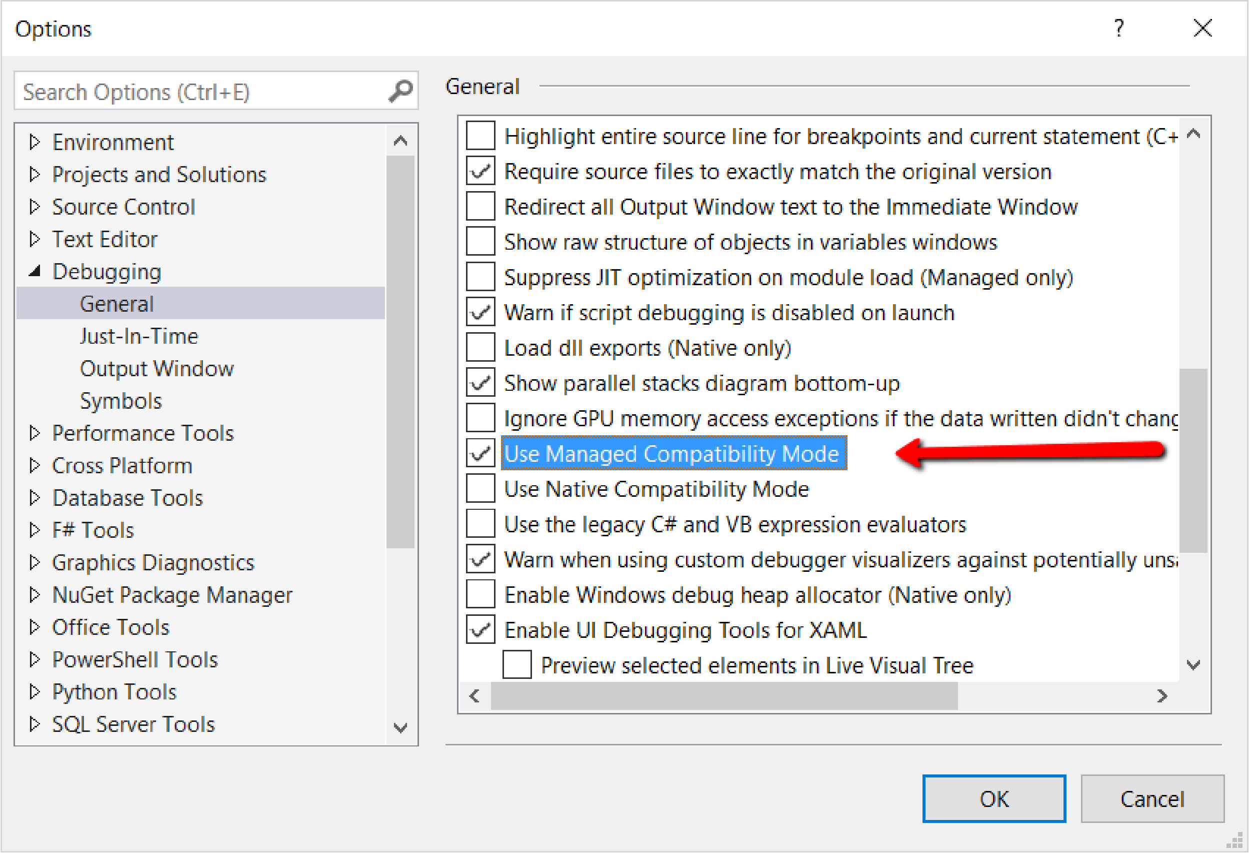
Interop Specifics

The .NET environment can also interop with MAXScript, and so this allows even further customization. By default, classes/functions are exported if public. For details, see this help topic in the MAXScript documentation.

Additionally, there is interop via C++/CLI to full SDK plugin. Is it not for the faint of heart, but a C++ based plugin can be fitted with C++/CLI wrappers, that would expand that native functionality into the .NET environment. For details, see this help topic in the C++ SDK documentation.

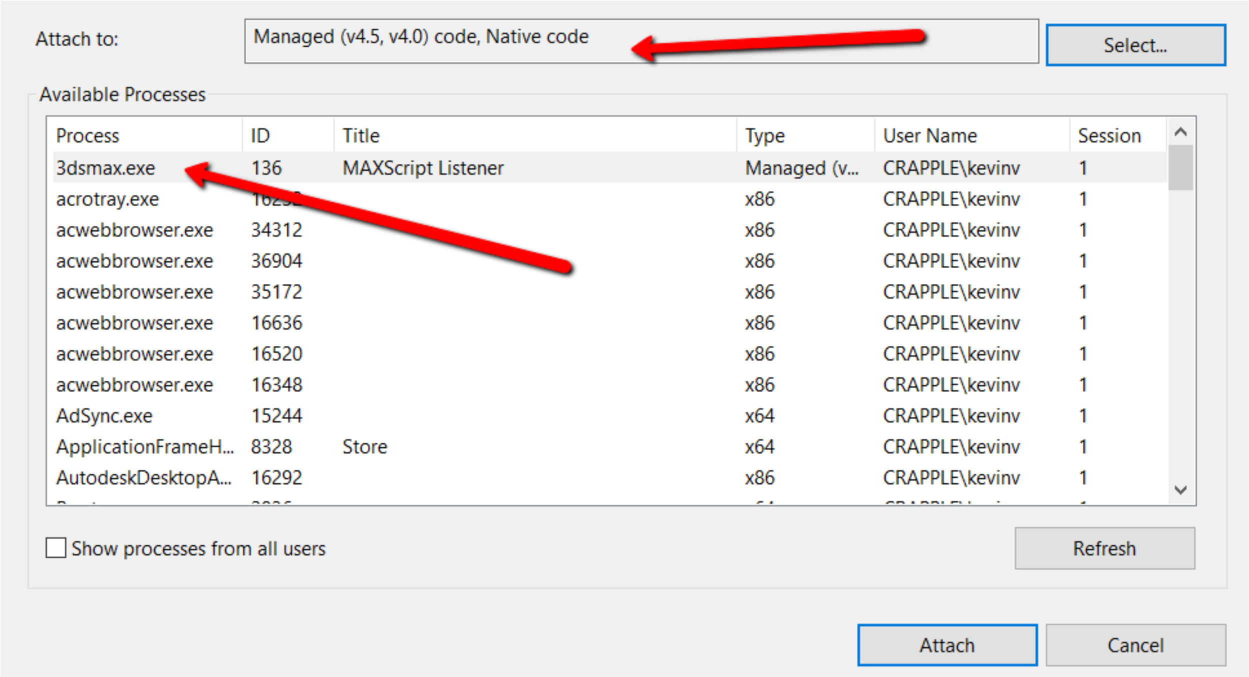
Debugging

To use direct debugger, you will need to point the debug target to the 3dsmax.exe. If you are using later versions of Visual Studio, you will also need to set the “Use Managed Compatibility Mode”. See below screen shot:



- □ Highlight entire source line for breakpoints and current statement (C+
- ☑ Require source files to exactly match the original version
- □ Redirect all Output Window text to the Immediate Window
- □ Show raw structure of objects in variables windows
- □ Suppress JIT optimization on module load (Managed only)
- ☑ Warn if script debugging is disabled on launch
- □ Load dll exports (Native only)
- ☑ Show parallel stacks diagram bottom-up
- □ Ignore GPU memory access exceptions if the data written didn't chang
- ☑ Use Managed Compatibility Mode
- □ Use Native Compatibility Mode
- □ Use the legacy C# and VB expression evaluators
- ☑ Warn when using custom debugger visualizers against potentially uns
- □ Enable Windows debug heap allocator (Native only)
- ☑ Enable UI Debugging Tools for XAML
  - □ Preview selected elements in Live Visual Tree

To Attach, it is best to use the “Managed, Native Code” mode. For example:



Attach to: Managed (v4.5, v4.0) code, Native code

Available Processes

Process	ID	Title	Type	User Name	Session
3dsmax.exe	136	MAXScript Listener	Managed (v...	CRAPPLE\kevinv	1
acrotray.exe	1625[illegible]		x86	CRAPPLE\kevinv	1
acwebbrowser.exe	34312		x86	CRAPPLE\kevinv	1
acwebbrowser.exe	36904		x86	CRAPPLE\kevinv	1
acwebbrowser.exe	35172		x86	CRAPPLE\kevinv	1
acwebbrowser.exe	16636		x86	CRAPPLE\kevinv	1
acwebbrowser.exe	16520		x86	CRAPPLE\kevinv	1
acwebbrowser.exe	16348		x86	CRAPPLE\kevinv	1
AdSync.exe	15244		x64	CRAPPLE\kevinv	1
ApplicationFrameH...	8328	Store	x64	CRAPPLE\kevinv	1
AutodeskDesktopA...	16292		x86	CRAPPLE\kevinv	1

☐ Show processes from all users

Samples

The provided sample code is in C# and is setup to use the 3ds Max 2017 x64 version. It should work easily with 3ds Max 2017, and also can be used with previous versions of 3ds Max with a few simple modifications to the project.

Explode Geometry sample is available as full App from App Store: explode geometry
The full sample code is in our GitHub repo here: https://github.com/ADN-DevTech/3dsMax-Explode-Geometry.

Cui Sample code NEED TO ADD THIS TO REPRO AS WELL

Tips/Tricks

Remember to set the CopyLocal flag setting to false when referencing 3ds Max assemblies in a new project!

When using CuiActionCommandAdapter make a custom abstract base class, then derive from it. That way, the category and other normally duplicated requirements can be implemented with defaults. For new commands/actions, derive from your abstract class and simply implement Execute() and ActionText(), plus any other overrides you want.

Make sure to use VS tools to your advantage. For example, when deriving from an abstract class such as CuiCommandAdaptor, use the “Implement Abstract Class” that will stub in all required implementation. This will save you coding time and is one good reason to use .NET API.

Remember to use the Object Browser as necessary. Even if you have another tool, the nice thing with Object Browser is that it is built in to the Visual Studio tool. It provides basic searching, and also has the “Custom Component Set” to allow you to have a group of components to search and browse. For example, this can isolate only the 3ds Max assemblies when you are searching and prevents false finds in System and other components not directly related to 3ds Max. This helps immensely during learning.

Make sure to do appropriate error checking. Always use the try/catch methodology to catch exceptions because this is the standard .NET API way of error handling. But you also have to be aware of error codes and return values. This is especially important with the Autodesk.Max assembly because it is a simple wrapper around the C++ SDK.

Do not forget to use all the 3ds Max assemblies as necessary. It’s easy to become stuck using only the Autodesk.Max because it seemingly provides almost everything you would need. However, the other assemblies are also very useful and provide additional functionality that can help as shown within the sample code.

What to avoid

Even though the Autodesk.Max assembly exposes what appears to be full plugin capability, it is advised by engineering to NOT create a full plugin type. This means you should not create a full plugin derived from Autodesk.Max.Plugins.<types>. Instead, use C++ SDK for full plugins.

Python

Python scripting language has been around since the late 1980's. See Python history from Wikipedia… https://en.wikipedia.org/wiki/Python_%28programming_language%29

Very early it became popular because of the philosophy that it should be code that can be easily read and understood. There are claims that the speed of development is several times faster than using languages like C++ or C# and there are even anecdotes of Python started to be used for extending the functionality of some C++ applications and ultimately replacing the C++ code completely.

Its success is due to a simplified syntax, with fewer "syntactical punctuations" compared with other languages like C++ and C#. However, this syntactical simplification (in my opinion, anyway) could create some confusion. For example, a tab, and 4 spaces can have different behavior in Python if you are not careful.

Because it is an open source implementation of a scripting language, it quickly became the natural choice for technology consumption as well. Autodesk Maya for example, has been using it for many years now, and is one of the primary customization languages. For 3ds Max, it is relatively new, and was brought into 3ds Max to support the workflows that were already in place for many customers using other products.

CPython is indicated to be the reference implementation of Python. The CPython implementation is the distribution mainly used by Autodesk. It is a free and open-source version and has a community-based development model. The CPython project is managed by the non-profit Python Software Foundation. CPython is named this way due to its internal implementation being written in the C programming language. This is one reason it is such a popular implementation, because it can easily be integrated into other C and C++ based technologies like 3ds Max.

See https://www.python.org for the full details about python as a language.

Why would you use Python in 3ds Max?

The first obvious reason would be because you have other tools already implemented in Python and have an experience pool. It is also very useful for automating certain tasks and building tools. Without experience in Python, it may not be your first choice for customization in 3ds Max. But it does have many general benefits that has drawn people to it. There are many benefits for using Python. For example:

- Large pool of people and knowledge. Ie. 250,000 questions and answers on StackOverflow.com
- Easy to re-use scripts
- Able to extend. ie. CPython API, SIP, SWIG, Boost, etc.
- Easy to learn
- Good for exploratory programming
- Many reusable libraries

Python is also used extensively in other M&E products, including Maya, MotionBuilder, and FBX SDK. The Shotgun toolkit is also Python oriented.

Although MAXScript is very powerful on its own, it is proprietary and has its own syntax.

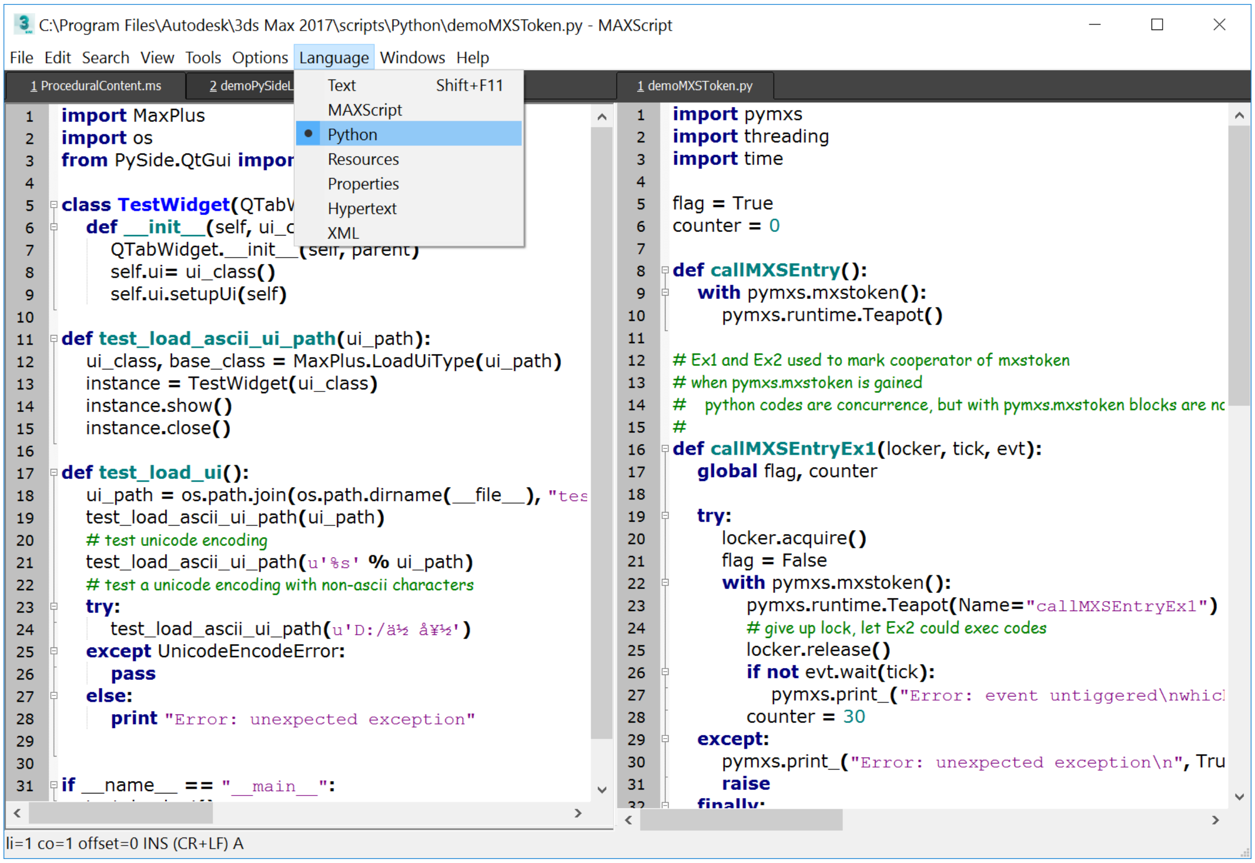
What can you do with Python?

Automating tasks is one of the main reason to use it. Python provides interop with MAXScript so you can plug it into existing MAXScripts very easily and of course into other python pipelines you may also be supporting.

Using the Qt library (typically as PySide), you can create nice UI. Other Python libraries are also supported, including numpy, Flask, PyQtGraph, etc.

How to Work with 3ds Max Python

To start writing scripts, you only need an editor. In 3ds Max 2017 the MAXscript editor can be used, but you can also use external IDEs like PyCharm. The modules to import are MaxPlus and/or pymsx. The MAXScript listener can be selected to be a Python console.



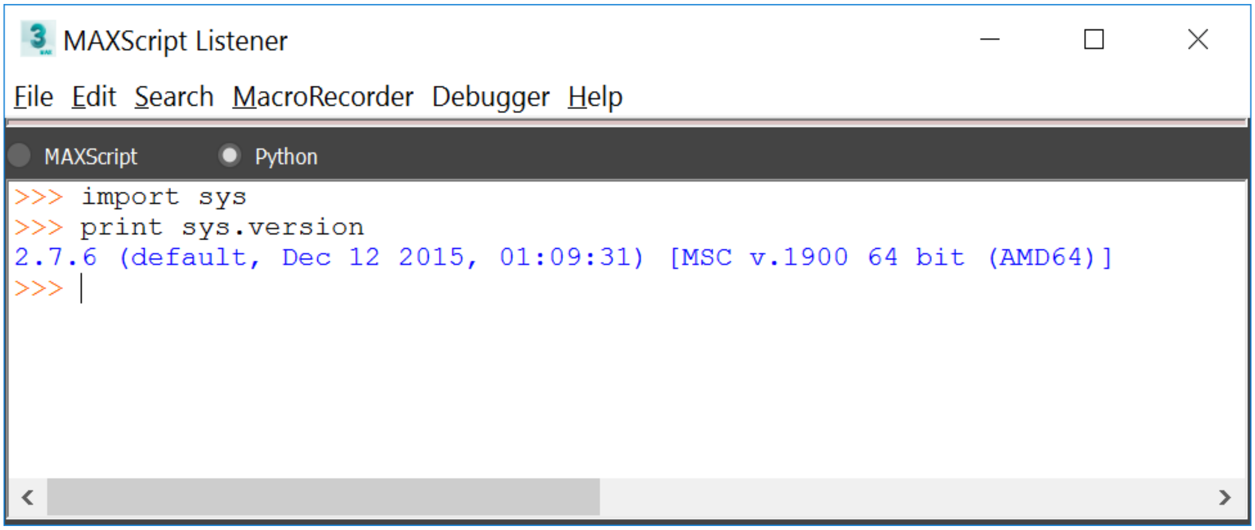
C:\Program Files\Autodesk\3ds Max 2017\scripts\Python\demoMXSToken.py - MAXScript

```
import MaxPlus
import os
from PySide.QtGui impor

class TestWidget(QTabV
    def __init__(self, ui_c
        QTabWidget.__init__(self, parent)
        self.ui= ui_class()
        self.ui.setupUi(self)

def test_load_ascii_ui_path(ui_path):
    ui_class, base_class = MaxPlus.LoadUiType(ui_path)
    instance = TestWidget(ui_class)
    instance.show()
    instance.close()

def test_load_ui():
    ui_path = os.path.join(os.path.dirname(__file__), "tes
    test_load_ascii_ui_path(ui_path)
    # test unicode encoding
    test_load_ascii_ui_path(u'%s' % ui_path)
    # test a unicode encoding with non-ascii characters
    try:
        test_load_ascii_ui_path(u'D:/ä½ å¥½')
    except UnicodeEncodeError:
        pass
    else:
        print "Error: unexpected exception"


if __name__ == "__main__":
```

li=1 co=1 offset=0 INS (CR+LF) A

```
import pymxs
import threading
import time

flag = True
counter = 0

def callMXSEntry():
    with pymxs.mxstoken():
        pymxs.runtime.Teapot()

# Ex1 and Ex2 used to mark cooperator of mxstoken
# when pymxs.mxstoken is gained
#   python codes are concurrence, but with pymxs.mxstoken blocks are nc
#
def callMXSEntryEx1(locker, tick, evt):
    global flag, counter

    try:
        locker.acquire()
        flag = False
        with pymxs.mxstoken():
            pymxs.runtime.Teapot(Name="callMXSEntryEx1")
            # give up lock, let Ex2 could exec codes
            locker.release()
            if not evt.wait(tick):
                pymxs.print_("Error: event untiggered\nwhicl
            counter = 30
    except:
        pymxs.print_("Error: unexpected exception\n", Tru
        raise
    finally:
```

MAXScript editor for Python



```
>>> import sys
>>> print sys.version
2.7.6 (default, Dec 12 2015, 01:09:31) [MSC v.1900 64 bit (AMD64)]
>>> |
```

MAXScript listener as Python console

How is your Code Loaded?

3ds Max looks in the following directories (in order) to find script for loading:

- User Scripts directory
- User Startup Scripts directory
- Scripts directory
- Startup Scripts
- Windows paths specified by the Path environment variable
- app store folder

How is your Code Executed?

You can execute Python code directly from within the MAXScript listener window. In 2017, you can select Python and will enter a python command-line environment for direct execution as well. In version prior to 2017, you can use the MAXScript Python class to execute python code and files. For example:

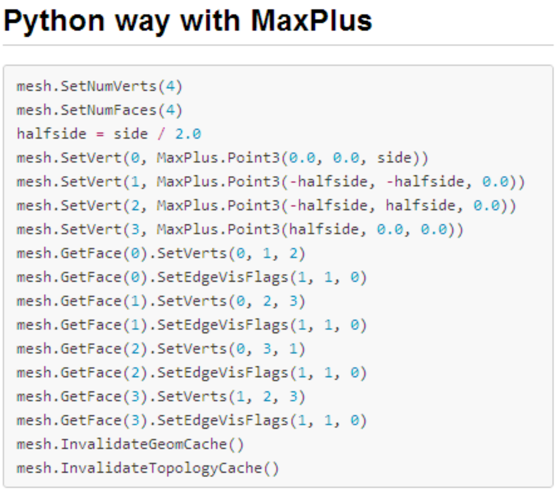
```
python.executeFile "demoHelloWorld.py"
```

You can also pass-in python files to execute at the command-line. For example:

```
3dsmax.exe –U PythonHost "demoHelloWorld.py"
```

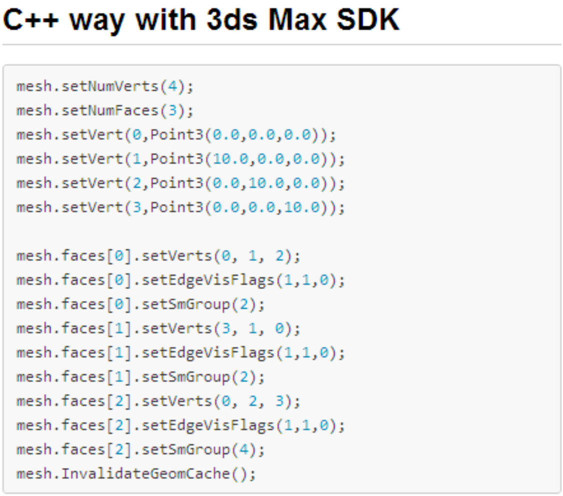
Why two different modules? MaxPlus and pymxs

MaxPlus is wrapper over a sub-set of the C++ SDK. This exposes some of the low level parts of the 3ds Max. For example, the class ids are needed here. It is also used for integrating PySide UI into the 3ds Max window system. AS comparison to see the mirror to the C++ see below:



Python way with MaxPlus

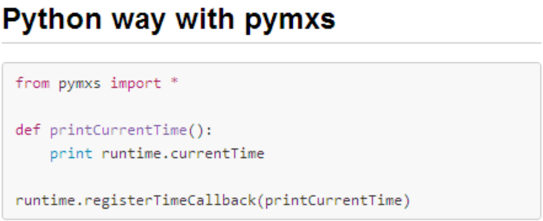
```
mesh.SetNumVerts(4)
mesh.SetNumFaces(4)
halfside = side / 2.0
mesh.SetVert(0, MaxPlus.Point3(0.0, 0.0, side))
mesh.SetVert(1, MaxPlus.Point3(-halfside, -halfside, 0.0))
mesh.SetVert(2, MaxPlus.Point3(-halfside, halfside, 0.0))
mesh.SetVert(3, MaxPlus.Point3(halfside, 0.0, 0.0))
mesh.GetFace(0).SetVerts(0, 1, 2)
mesh.GetFace(0).SetEdgeVisFlags(1, 1, 0)
mesh.GetFace(1).SetVerts(0, 2, 3)
mesh.GetFace(1).SetEdgeVisFlags(1, 1, 0)
mesh.GetFace(2).SetVerts(0, 3, 1)
mesh.GetFace(2).SetEdgeVisFlags(1, 1, 0)
mesh.GetFace(3).SetVerts(1, 2, 3)
mesh.GetFace(3).SetEdgeVisFlags(1, 1, 0)
mesh.InvalidateGeomCache()
mesh.InvalidateTopologyCache()
```



C++ way with 3ds Max SDK

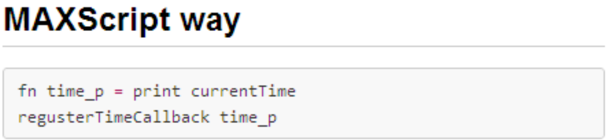
```
mesh.setNumVerts(4);
mesh.setNumFaces(3);
mesh.setVert(0,Point3(0.0,0.0,0.0));
mesh.setVert(1,Point3(10.0,0.0,0.0));
mesh.setVert(2,Point3(0.0,10.0,0.0));
mesh.setVert(3,Point3(0.0,0.0,10.0));

mesh.faces[0].setVerts(0, 1, 2);
mesh.faces[0].setEdgeVisFlags(1,1,0);
mesh.faces[0].setSmGroup(2);
mesh.faces[1].setVerts(3, 1, 0);
mesh.faces[1].setEdgeVisFlags(1,1,0);
mesh.faces[1].setSmGroup(2);
mesh.faces[2].setVerts(0, 2, 3);
mesh.faces[2].setEdgeVisFlags(1,1,0);
mesh.faces[2].setSmGroup(4);
mesh.InvalidateGeomCache();
```

The pymxs module exploits the MAXScript runtime itself. Anything that is possible in MAXScript, is possible in Python with this module. For example:



Python way with pymxs

```
from pymxs import *

def printCurrentTime():
    print runtime.currentTime

runtime.registerTimeCallback(printCurrentTime)
```



MAXScript way

```
fn time_p = print currentTime
regusterTimeCallback time_p
```

Interop

Interop is provided mainly with MAXscript. MAXScript and Python can be executed from each other. MAXScript can import and use Python libraries. Data can be shared between MAXScript and Python.

Debugging

Debugging can be done with PyCharm, but you must purchase the full version.
PyCharm is available here: http://www.jetbrains.com/pycharm
Setup can be tricky… See blog post. Note that it is not directly suorted by Autodesk, but does seem to work ok (just no guarantees in future.)

Sample code

There are many samples included with 3ds Max. Find them located here in the install:
<Install location>\ 3ds Max 2017\scripts\Python

We also took the .NET API sample, Explode Geometry, and rewrote it using Python. You can find it in the GitHub repo here: https://github.com/ADN-DevTech/3dsMax-Explode-Geometry
See Python specific folder.

Tips & Tricks

- Don't pollute global namespace
  ie. from <lib> import <sub-lib>
- Use: if __name__ == '__main__' idiom.
  Remember every file is potentially a library
- Use MAXScript, .NET, or C++ where appropriate
  Python is not always "the answer" ☺

Autodesk App Store

The Autodesk App Store has been online for several years now and we are approaching 2 million downloads this year. That shows its importance in the Autodesk eco-system, and how much visibility it has.

Autodesk App Store – Consumer

As user of 3ds Max, it's important to know there is a store dedicated to you. You can find it here: https://apps.autodesk.com/3DSMAX/en/Home/Index There are many apps there for 3ds Max already, so we encourage you to check it out and support other partners. As an example, the Explode Geometry plugin discussed in the earlier in the .NET API portion is published there for free.

Autodesk App Store – Publisher

If you are interested to distribute the tools you are developing, the App Store is a great place to visibly. You can publish your app there for Free, Trial, or for Fee. We use either PayPal or BlueSnap as a means of payment to transfer funds from the customer to you directly. Autodesk does not take a fee, and this is a free service. We also have an entitlement API to help you secure any apps that you may want to prevent being shared to other than the paid customer.

To learn more, check out http://www.autodesk.com/developapps There is an overview and general information about the App Store publishing. Additionally, look for the 3ds Max specific page for details about publishing an app for 3ds Max.

Resources for further learning and support

3ds Max Dev Center: http://www.autodesk.com/develop3dsmax

Blogs:

- http://getcoreinterface.typepad.com
- http://area.autodesk.com/blogs/chris

Autodesk resources:

- http://area.autodesk.com/
- 3ds Max Programming forum

Others:

- http://www.gamedev.net
- http://www.cgsociety.org

Thank You!