Introduction to Tools Scripting

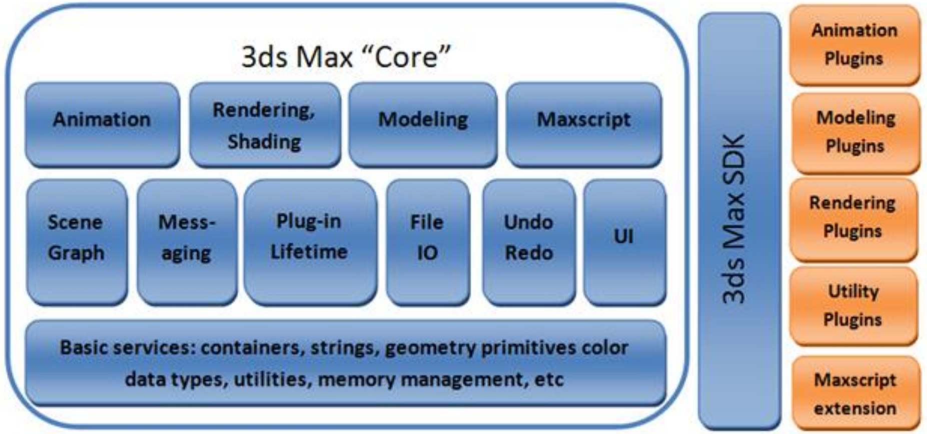
MaxScript

3DSMax Architecture

- Extend functionality by third party developers.

- Different layers below it that allow us to customize the software

  - SDK C++
  - .NET API
  - Python API
  - Maxscript



3DSMax: SDK

What is 3DSMax SDK:

- Allows us to develop plugins for 3DSMax.
- Mainly focused in c++ libraries (original 3DSMax core code)
- Requires high programming skills and serious OOP knowledge

Why 3DSMax SDK:

- Very flexible and provides resources to create and modify almost every functionality of 3DSMax
- Very fast vs other languages supported by the software.
- It is the standard for serious plugins developed by companies (e.g: vray)

3DSMax: SDK

- Why to use the SDK:
    - It’s the standard for commercial plugins
    - It’s more time consuming to develop than other options available for coding in 3DSMax.
    - Maintenance in comparison with maxscript is way more difficult.
    - Not much sources from where to learn from, only a few given by the sdk itself.
    - It’s the most powerful tool to be used in 3DSMax.

- You can reload and delay plugins, but is very tedious and slow to be used.

3DSMax: .NET API

What is 3DSMax .NET API:

- Extension of the C++ SDK libraries from 3DSMax.
- More flexible than SDK, allow us to code in higher level languages like C#.
- It’s mostly based on wrappers code that has been added during the past years through the software updates. Not much resources to learn from.

Why 3DSMax .NET API:

- Easier to understand in comparison with the C++ SDK.
- Very easy to extend the UI with, together with WPF design tools.

MaxScript

What is maxscript:

- The scripting language for 3DSMax.
- It’s an interpreted language embedded in Max.
- Very easy to use in comparison with the previous mentioned languages.
- Does not have full access to modify or create new functionality in comparison with previous mentioned languages.

Why maxscript:

- Easier to understand in comparison with the C++ SDK.
- Faster to code and implement new functionality.

MaxScript: Features

Maxscript allows us to develop scripts for the following 3DSMax sections:

User Interface	Splines/Nurbs	Render
Lights	Animation	Import/Export
Camera	Controllers	Batch processes
Geometries	Particles	...
Modifiers	Helpers	...

3DS Max: Plugins

Rendering

- VRAY: https://www.chaosgroup.com/
- Renderman: https://renderman.pixar.com

Utility tools

- RailClone: https://www.itoosoft.com/es/railclone
- Bones pro: https://www.bonespro.com/
- Unwrella: http://www.unwrella.com/

MaxScript: Key Learning

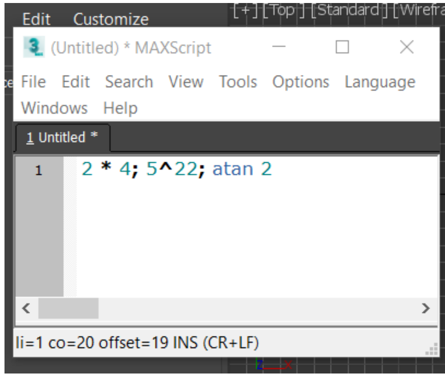
- How to access maxscript
- Maxscript programming fundamentals
- Maxscript advanced programming
- Maxscript deployment
- Existing samples
- Exercises

Today contents

- Maxscript tools: listener & editor basics
- Maxscript fundamentals
- Variables, blocks, and functions
- Data structures
- UI Scripting
- OOP, Classes
- Debugging
- Security
- Deployment
- Samples

Listener & editor

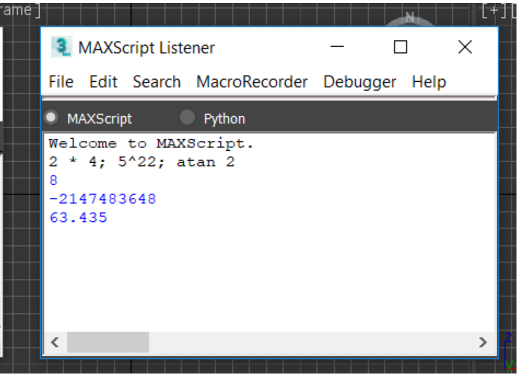
- Tools fully integrated within the software
- No need to install third party libraries.



```
2 * 4; 5^22; atan 2
```

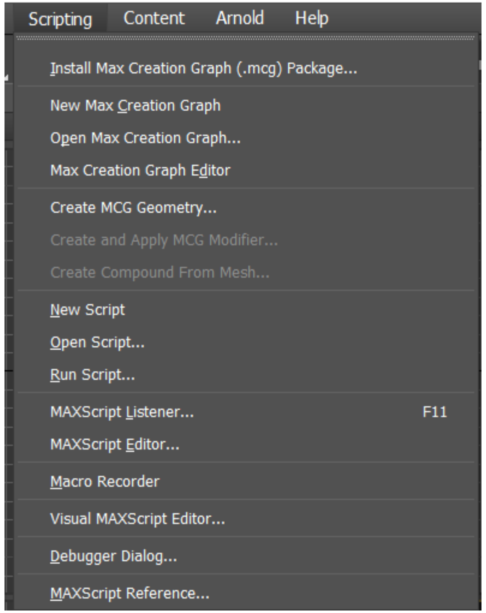
li=1 co=20 offset=19 INS (CR+LF)

MaxScript Editor



```
Welcome to MAXScript.
2 * 4; 5^22; atan 2
8
-2147483648
63.435
```

MaxScript Listener


Scripting
Content
Arnold
Help
Install Max Creation Graph (.mcg) Package...
New Max Creation Graph
Open Max Creation Graph...
Max Creation Graph Editor
Create MCG Geometry...
Create and Apply MCG Modifier...
Create Compound From Mesh...
New Script
Open Script...
Run Script...
MAXScript Listener... F11
MAXScript Editor...
Macro Recorder
Visual MAXScript Editor...
Debugger Dialog...
MAXScript Reference...

MaxScript Tools Content

MaxScript fundamentals

- Similar to any other scripting language with its own characteristics
- Similar reserved words: if, else, for, while, do…
- We don’t use brackets!
- Weak typing language

Tip: Use $ to use the current selected object

```
fn exportMap map alias = (

    local map_filename = "default_texture"
    if map != undefined then (
         map_filename = map.filename
    )

    local base_name = getFilenameFile map_filename
    local json_filename = "data/textures/" + base_name + ".dds"
    local ofull_path = project_path + json_filename

    -- Check if ofull_path exists
    if not doesFileExist ofull_path then (
        copyFile map_filename ofull_path
    )

    fs.writeKeyValue alias json_filename
),
```

Note: Use the help reference on the editor to access maxscript reference and semantics.

MaxScript fundamentals

- Constructor: Sphere creation or box
  sphere radius:20 segs:30 pos:[0,1,0] name:"blabla"
  (Class / Primitive name) +params
- They can see other classes in the reference maxscript web page: https://help.autodesk.com/view/3DSMAX/2018/ENU/
- Right click after word to see properties.
- Current max selection, get object by name
- Transformations (move, rotate, scale)
- Copy reserved word
- Create a material
- Execute code:
  - Selection + enter
  - Drag into rollout
  - Evaluate from UI
  - Evaluate from ctrl + e
- Save code into file and test it so they can check results

```
box name:"test" wirecolor:(color 255 0 0)
rotate $test (eulerAngles 0 0 45)
scale $test [1,1,2]
```

```
box name:"test" wirecolor:(color 255 0 0)
rotate $test (eulerAngles 0 0 45)
rotate $test -35 z_axis
scale $test [1,1,2]
```

MaxScript fundamentals

Types of words

- Reserved words
  - Reserved list: https://help.autodesk.com/view/3DSMAX/2018/ENU/?guid=__files_GUID_874741B6_FE4B_496F_856D_1C66541F5DBC_htm
- Quoted text
- Names
  - Classes
  - Objects
  - Functions

MaxScript fundamentals

For statement

```
for i = 1 to 20 do
    sphere name:("itr" + i as string) pos:[i^2,0,0] radius:(i*1.05)
```

If statement

```
for i = 1 to 20 do (
    if (mod i 2) == 1 do
        sphere name:("itr" + i as string) pos:[i^2,0,0] radius:(i*1.05)
)
```

Tip: Clear the listener with maxscript command

```
        delete $objects  or resetMaxFile #noprompt
```

MaxScript: Code layout

- Comments
  - /* container comment */
  - Inline comment --
- Use indenting
- Use C++ bracket style
- Code blocks
- Local variables vs global variables
- Showclass command
- Classoff command
- Select command
- `group (GetCurrentSelection() as array) name:"myGroup"`
- name* , selects everything that contains that string as name on it’s left.

MaxScript fundamentals

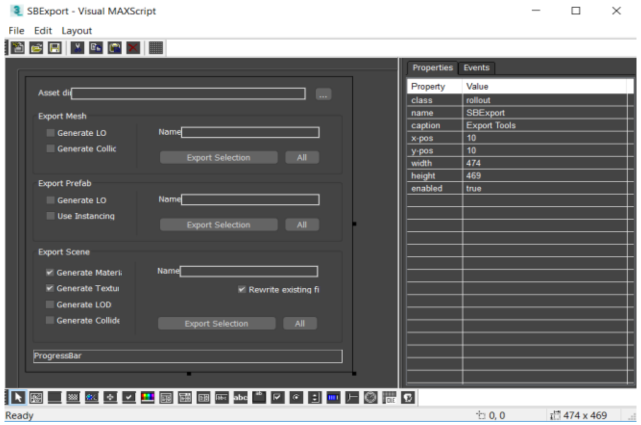
- Similar to any other scripting language with its own characteristics

```
resetMaxFile #noprompt --reset the scene
mybox = box length:10 width:10 height:10 wirecolor:blue --new box

for i = 1 to 5 do --repeat five times, for each iteration do:
(
    box_copy = copy mybox
    box_copy.pos = [i*20, 0, 0]
    box_copy.wirecolor = [i*25,i*50,(5-i)*50]
) --end of the for loop
```

```
-- Skyscraper example
d = plane widthsegs:25 lengthsegs:25
p = convertToPoly(d) --create a plane, convert to Epoly

for i = 1 to (polyop.getNumFaces p) do --repeat 50 times
(
    polyop.setFaceSelection p i
    p.extrudeFaces (random 1 16) --extrude the selection
)
```

UI Scripting

- Design editor (rollout) included in the tools



Property	Value
class	rollout
name	SBExport
caption	Export Tools
x-pos	10
y-pos	10
width	474
height	469
enabled	true

```
clearListener()


rollout baseRollout "Untitled" width:307 height:88
(
    pickbutton 'btn1' "Replace [Object Name]" pos:[27,24] width:117 height:31
    button 'btn4' "Replace" pos:[163,24] width:118 height:31 align:#left

   on btn1 picked obj do (
       if isValidNode obj do (
           btn1 .tooltip = "You picked " + obj.name
           btn1.text = obj.name
           nodeBase = getnodebyname obj.name
        )
   )

   on btn4 pressed do (

       replaceItem nodeBase
   )
)

createDialog baseRollout
```

MaxScript: Functions

```
fn drawLineBetweenTwoPoints pointA pointB =
(
    ss = SplineShape pos:pointA
    addNewSpline ss
    addKnot ss 1 #corner #line PointA
    addKnot ss 1 #corner #line PointB
    updateShape ss
    ss
)

newSpline = drawLineBetweenTwoPoints [10,20,30] [100,200,10]
```

```
for h in ($Bip001_R*) do print h.name

for h in ($Bip001_R*) do print (biped.getTransform h #pos)

rotate $Bip001_R* -35 x_axis
```

MaxScript fundamentals

Spiral exercise

```
gc()
delete $objects
r =40
step = 5
total_amount = 360 / step

for i = 1 to total_amount do (

   local out_angle = i *step;
   x = r * cos(out_angle);
   y = r * sin(out_angle);
   sphere name:("itr" + i as string) pos:[x,y,0] radius:1
)
```

Recursive function

```
delete $objects
global max_depth =5

function createChildren childAmount depth = (

b = box name:(depth as string) width:1 height:1 length:1
depth = depth + 1

if depth > max_depth do ( return b)

initial = - (childAmount^depth) / 2
step = ((childAmount^depth) / childAmount)*2

….
```

MaxScript: animations

```
with animate on
(
  at time 0 selection.pos.z=10
  at time 100 selection.pos.z=199
)
```

```
Local current = $
Animate on for t = 1 to 100 by 5 do
At time t
(
    Current.position = current.position + [0,1,0]
)
```

MaxScript: modifiers

Basic exercise

```
B = bend angle:45 direction:90
Addmodifier $box01 b
Addmodifier $box01 (twist angle:90)

$box01.modifiers
$box01.modifiers[|].angle = 45
```

Extended ball angle rotation

```
addModifier $ (bend())
Animate on for t iin 0 to 100 by  5 do (
At time t
 (
    Local dv = $ball.pos - %column.pos
    $ciolumn.bend.angle = atan2 dv.z dv.x
    %column.bend.direction = -(atan2 dv.y dv.x)
)
)
```

Data Structures

- Common data structures: arrays, lists,
- Collect within a loop inside array
- Max 2018: Dictionaries

- We can use .NET utilities with maxscript!!
- Very useful depending on the situation
- Structs: primitive way of defining a class

```
/* Arrays examples
#(<value>, <value>, ...)
#() -- an empty array */

local a = #(1,2,3,4) -- declares the array
join a #(5,6,7,8) -- concatenates another array into a
append a 9 -- adds a new number to the array

Dictionary() -- empty dictionary of type #name
Dictionary (#integer | #name | #string) -- empty dictionary of the
specified type
Dictionary {#(key, value)}+ -- one or more two-value arrays
Dictionary {key:value}+ -- one or more explicit key:value pairs
Dictionary {(DataPair key value)} -- one or more DataPair objects

getDictValue dictName “key”
putDictValue dictName “value”
SetDictValue dictName key value


-- .NET Usage example
hsh = dotNetObject "System.Collections.Hashtable"
hsh.Add "1" "test"
hsh.Add "foo" "bar"
hsh.Item["foo"]
```

Maxscript: OOP

Properties : Accessible parameters for objects of the class. Examples of properties are height, width, and length for boxes, and radius for spheres.

Methods : Defines all the functions you can call for objects of the class. Examples of methods are moving or rotating a 3ds Max object, adding a modifier to a 3ds Max object, and accessing the position of vertices in a 3ds Max object. The terms *method* and *function* are synonymous in this document.

Operators : Defines the math and other symbolic operators that are defined on values in the class. An example of an operator is the '-' operator, which will perform a mathematical operation on numbers, colors, vectors, and matrices, but will perform a Boolean subtraction when used with 3ds Max objects.

Constructors : The various ways you can create objects of the class. For example Point3 0 0 0 and <color> as Point3 are constructors for the Point3 class. Executing either of these constructors will create a new Point3 object.

Maxscript: OOP

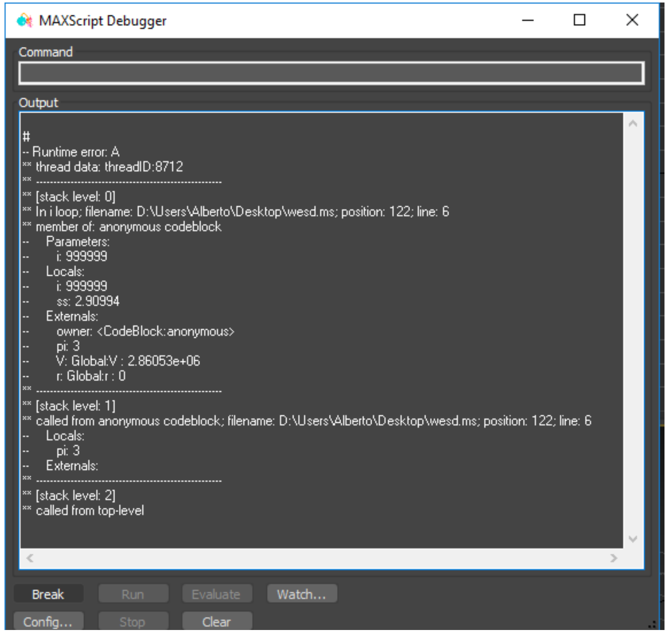
```
struct MyClass
(
    public
    -- The constructor function that gets everything started.
    fn Constructor =
    (
         print ("The Constructor has been run")
         return true
    ),

    -- An example public function
    fn MyFunction =  (   ),

    initalized = Constructor(),

    private
    fn MyPrivateFunction =  (    )

)
```

TO-DO

Debugging

- By using the debugger dialog, we are able to debug Our code by using breakpoints and watches.

```
v = 0
r = 0

(
    local pi = 3
    for i = 1 to 1000000 do (
        ss = random e pi;
        v += ss;
        v += r;
        if i == 999999 do throw "A")
)
```



```
#
-- Runtime error: A
** thread data: threadID:8712
** ------------------------------------------------------
** [stack level: 0]
** In i loop; filename: D:\Users\Alberto\Desktop\wesd.ms; position: 122; line: 6
** member of: anonymous codeblock
--    Parameters:
--       i: 999999
--    Locals:
--       i: 999999
--       ss: 2.90994
--    Externals:
--       owner: <CodeBlock:anonymous>
--       pi: 3
--       V: Global:V : 2.86053e+06
--       r: Global:r : 0
** ------------------------------------------------------
** [stack level: 1]
** called from anonymous codeblock; filename: D:\Users\Alberto\Desktop\wesd.ms; position: 122; line: 6
--    Locals:
--       pi: 3
--    Externals:
** ------------------------------------------------------
** [stack level: 2]
** called from top-level
```

MaxScript: Security

Security:

- We can simply distribute our code by sharing the .ms file.
- This is not safe, everybody can see our code and use it.
- MaxScript allow us to encrypt our code if needed.

To encrypt the code do the following:

```
encryptScript "script_name.ms"
```

Raw source code from .ms file is converted into .mse file which can also be launched from Max.

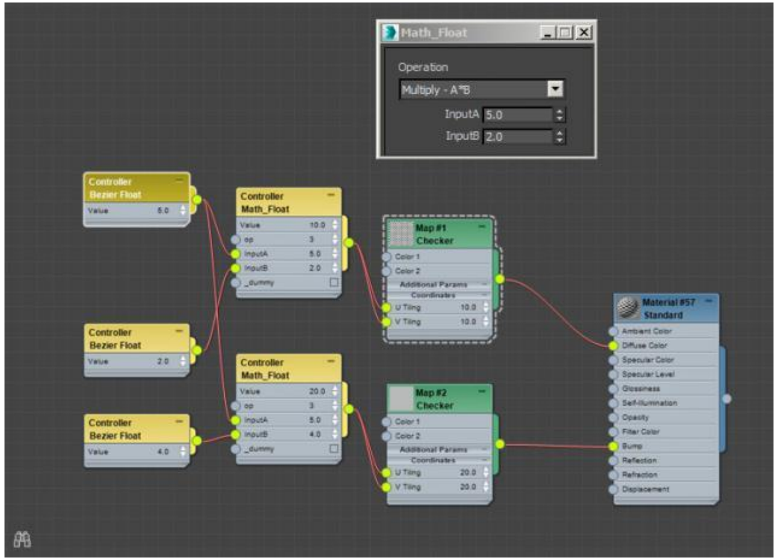
MaxScript: Deployment

- The code can be used by users in many different ways:
  - Drag and drop the maxscript file into max.
  - Directly execute the source code within the maxscript editor
  - Drop the script into the toolbar, creates a shortcut

- Sometimes is very useful to launch our script when max is loaded. Artist won't need to worry about code locations and other problems.
  - Max searches for startup.ms file.
  - Here we should include our directory were the script to be launched on start should be.
- Tweak the UI with a dark theme
- Macroscript

Controllers

- Works like blueprints (e.g Unreal)
- Acts the same way like scripts.
- It’s more visual and widely used by artists nowadays.
- Doesn’t need any kind of programming knowledge.


Math_Float

Operation: Multiply - A*B

InputA 5.0

InputB 2.0

Samples

Sample 1: Create elements within an spline. / Make camera move through spline

```
Full code: https://github.com/AlbertoMVD/MVD_ToolScripting
```

```
Full code: https://github.com/AlbertoMVD/MVD_ToolScripting
```

Samples

Sample 2: Place objects in surface/ teapot example

```
Full code: https://github.com/AlbertoMVD/MVD_ToolScripting
```

```
Full code: https://github.com/AlbertoMVD/MVD_ToolScripting
```

Samples

Sample 3: Create elements around an spline: practical example road and trees

```
Full code: https://github.com/AlbertoMVD/MVD_ToolScripting
```

Resources:

- http://getcoreinterface.typepad.com/blog/
- https://doc.lagout.org/Others/Game%20Development/Designing/3ds%20Max%206%20Bible.pdf
- http://www.scriptspot.com/3ds-max
- https://area.autodesk.com/blogs/the-3ds-max-blog/max-creation-graph-samples/
- https://help.autodesk.com/view/3DSMAX/2018/ENU/?guid=__files_GUID_E6FD6664_B41B_4FF4_9086_D0EAAC6BD6A8_htm