Tools Programming

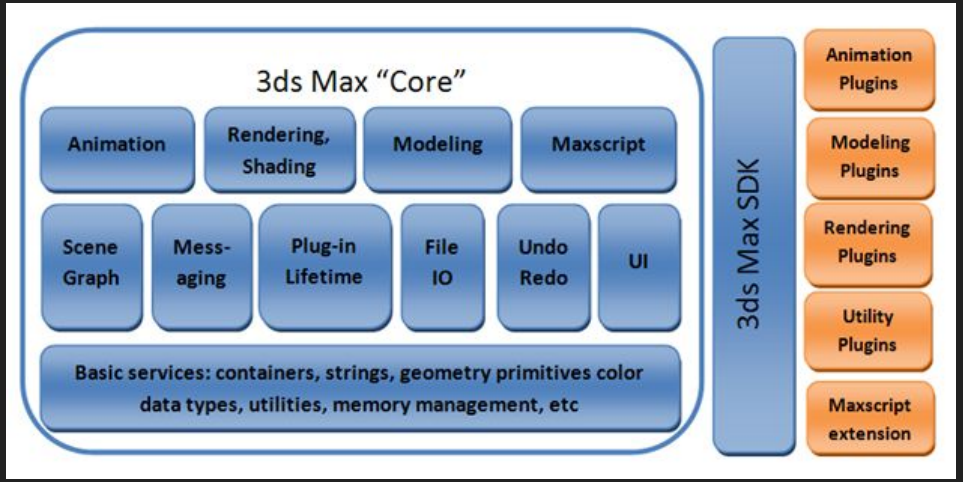
Tools scripting (Basics)

Class 2

Index

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: SDK

- Demo: Show how to compile and load one of the projects.
- Show examples of commercial SDK plugins that are widely used by the community.

Note1: http://docs.autodesk.com/3DSMAX/16/ENU/3ds-Max-SDK-Programmer-Guide/index.html
Note2: https://github.com/ADN-DevTech/3dsMax-Bake-Radiosity

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.

3DSMax: .NET API

- Demo: Show how to compile and load one of the projects.
- Show examples of commercial SDK plugins that are widely used by the community.

Note: https://github.com/ADN-DevTech/3dsMax-Explode-Geometry

MaxScript

What is maxscript:

- The scripting language for 3DSMax.
- It’s an interpreted language.
- 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:

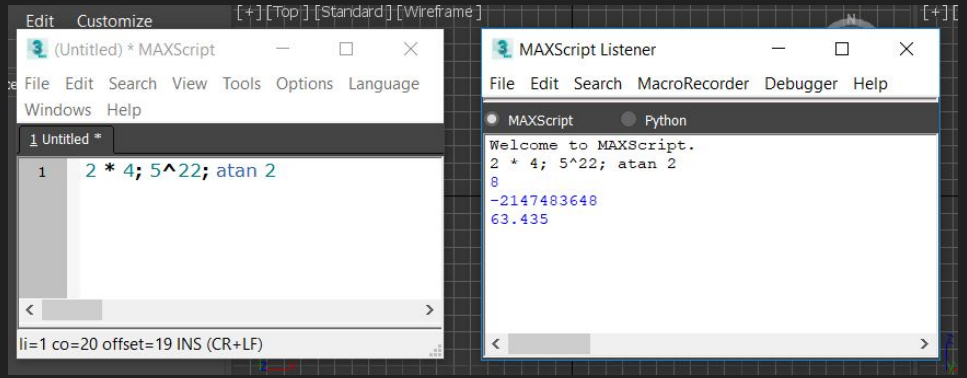
User Interface	Splines/Nurbs	Render
Lights	Animation	Import/Export
Camera	Controllers	Batch processes
Geometries	Particles	
Modifiers	Helpers	

MaxScript contents

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

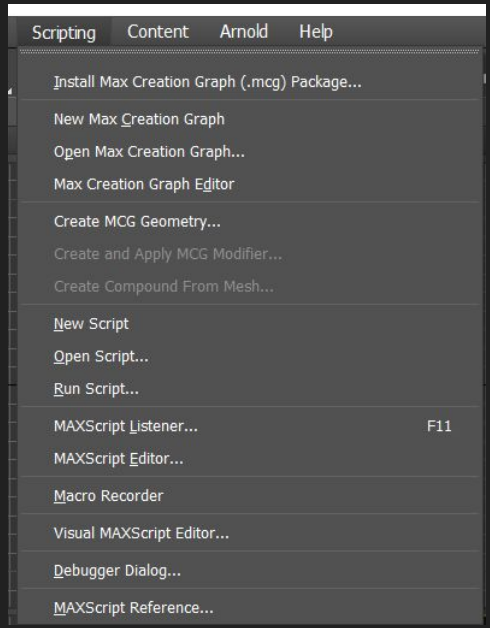
Listener & editor

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



MaxScript Editor

MaxScript Listener



MaxScript Tools Content

MaxScript fundamentals

- Similar to any other scripting language with its own characteristics
- Similar reserved words: if, else, for, while, do…
- We use round brackets!
- Weak typing language

Tips: Use $ to use the current selected object
Evaluation is assigned to the ? symbol

```
-- Maxscript grammar rules

[...] -- items inside the brackets are optional
(...|...|...) -- choose one of the items separated by the bars
{...} -- you can specify the braced item ZERO or more times
{...}+ -- you can specify the braced item ONE or more times
::= -- define a name for a syntax rule
<rule> -- you can insert what is defined by the named rule

<assignment> ::= <destination>=<expr>
<destination> += <expr>
<destination> -= <expr>
<destination> *= <expr>
<destination> /= <expr>
```

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

MaxScript fundamentals

- Variables are weak typed
- Can be used as global or local depending on the scope.
  - Globals can be used widely around the file
  - Locals belong to the scope where they are defined.
- Comments are created by adding --
- We can print on console by using ‘print’ reserved keyword
- There are other reserved keywords such as
  - Geometry names: box, sphere
  - Transforms: rotate, scale,
  - Delete
- Variables are not cleared after execution!

```
-- This is a comment
-- This is a global variable, visible from the whole snippet
Global name = “globalvar”

-- Creating a dummy scope
(
   local localvar = “localvar”

   print names
   print localvar
)

print localvar -- This gives an error
print globalvar -- this works
```

MaxScript Transforms

- Reserved transform keywords:
  - Translate: move objname [x,y,z]
  - Rotate: rotate objname (eulerangles x y z)
  - Scale: scale objname [x,y,z]

- Other transform keywords:
  - Position: obj.pos [x,y,z]
  - Rotation: obj.rotation (eulerangles x y z)
  - Scale: obj.scale [x,y,z]

```
delete $*
box name:"box_create"  pos:[0,0,0]

box name:"box_move" pos:[0,0,0]
move $box_move [50,0,0]

box name:"box_rotate"  pos:[100,0,0]
rotate $box_rotate (eulerangles 0 0 45)

box name:"box_scale" pos:[160,0,0]
scale $box_scale [2,2,2]

t = TextPlus name:"debug_text" size:25 pos:[90,-70,0] wirecolor:(color 255 255 255)
t.SetPlaintextString("Transformations")
```

```
delete $*
obj = box name:"box_transforms"  pos:[0,0,0]
obj.rotation = eulerangles 90 0 0
obj.scale = [2,2,2]
obj.pos = [10,10,10]
```

Note: First method keeps changing on iteration, second not.

MaxScript Sequencing/Iteration

- We can create loops in maxscript by:
  - For loop: for i = n to m do ( )
  - Do While: while x > n do ( )
  - Do while 2: do (while var)
- Nested loops and other iterator compositions can also be used.

```
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

for i = 1 to 15 do --repeat five times, for each iteration do:
(
	sphere name:(i as string) position:[0,i^2,0] radius:i
) -- end of the for loop
```

```
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:
(
	for j = 1 to 5 do
	(
		box_copy = copy mybox
		box_copy.pos = [i*20, j*20, 0]
		box_copy.wirecolor = [i*25,j*25,0]
	)

) -- end of the for loop
```

MaxScript fundamentals

- Branching is also present in maxscript
  - `if <expr> then <expr> [else <expr> ]`
  - `if <expr> do <expr>`

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

for i = 1 to 15 do --repeat five times, for each iteration do:
(
	for j = 1 to 15 do
	(
		box_copy = copy mybox
		box_copy.pos = [i*12, j*12, 0]
		if mod i 2 == 0 then (
			box_copy.wirecolor = [0,0,255]
		)
		else (
			box_copy.wirecolor = [255,0,0]
		)
	)
) -- end of the for loop
```

- We can also work with 3dsmax geometry
  - E.g: editable poly, editable mesh

```
-- Create a skyscraper-cube like structure through random extrusions

delete $*
myplane = Plane lengthsegs:15 widthsegs:15
p = convertToPoly(myplane) --converts any shape to editpoly

for i = 1 to 15 do -- repeat length
(
	for j = 1 to 15 do -- repeat width
	(
		if (random 1 3) == 1 then (
		polyop.setFaceSelection p #{15*(i-1) + j}
		p.extrudeFaces (random 1 15) --extrude the selection
		)
	)
)
```

Maxscript geometry

- We can also create geometry in maxscript from scratch:
  - We can create shapes out of primitives: primitives, splines, nurbs..
  - We can create shapes out of splines
  - We can edit meshes with editable poly, editable mesh

- Many plugins are geometry based
  - Procedural building generation: Ghost town lite
  - Procedural detail generation: bump map high poly

MaxScript fundamentals

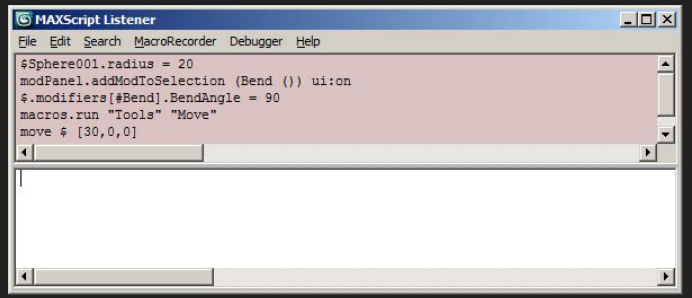
- Mesh modification methods
  - Remove backface method
  - Increase performance on static meshes
- Modify script to work with selection
  - Use $selection command

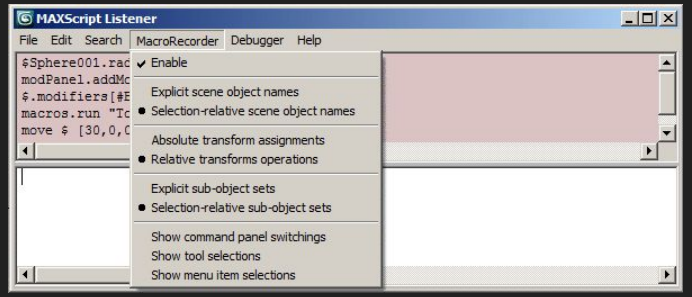
```
/*
Snippet used to remove mesh backface
*/

delete $*
global obj = sphere() -- Create an sphere
convertToMesh obj -- Convert to editable mesh

-- Determine camera front vector
viewDir = Inverse(getViewTM())
viewDir = viewDir.row3

for v = obj.numFaces to 1 by -1 do
(
	local faceNormal = getFaceNormal obj v
	if dot faceNormal viewDir < 0 then
	(
		deleteFace obj v
	)
)

-- Update on mesh needed after editing its geometry!
update obj
```

Note: http://help.autodesk.com/view/3DSMAX/2015/ENU/?guid=__files_GUID_582C310B_0875_4CDA_B113_B81449CAE157_htm

MaxScript MacroRecorder

- Captures most actions done within the 3DSMax editor
- MacroRecorder settings can be overridden at preferences section panel
- Fastest way to learn maxscript fundamentals





Data Structures

- Common data structures: arrays, lists,
- 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”


-- .NET Usage example
from System.Collections import Hashtable as hsh
hsh = hsh()
hsh.Add(1, "test")
hsh.Add("foo", "bar")
for o in hsh: print o.Key
```

Sample 1

- Create a circle made out of spheres
- User can determine (in code)
  - Radius of the circle
  - Radius of the spheres
  - Density of the circle (amount of spheres that define it)

- Generate a circled mesh instead of spheres
  - Use arrays to generate the vertices needed
  - Use mesh operations to create the final mesh

MaxScript fundamentals

- Functions are defined by the following structure:
  - function functioname arg1 arg2… = ( )
  - Can be abbreviated to fn functioname arg1 arg2 …. = ( )

```
function createCircle =
(
	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
	)
)
```

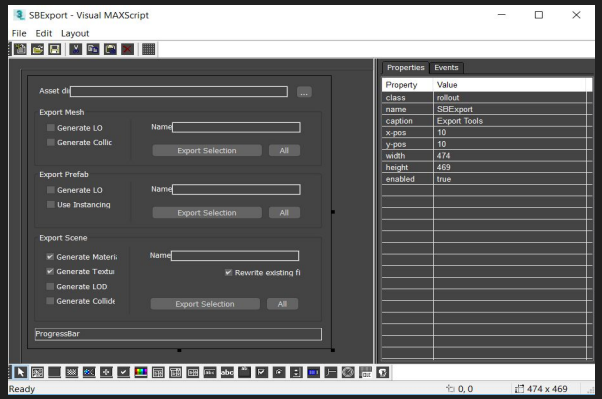
```
fn exportMap map alias =
(
	local map_filename = "default_texture"
	if map != undefined then (
		map_filename = map.filename
	)
	local base_name = getFilenameFile map_filename
	local json_filename = "data/textures/" + base_name + ".dds"
	local ofull_path = project_path + json_filename
	print ofull_path

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

	fs.writeKeyValue alias json_filename
),
```

Sample 2

- Create a children based recursive function
- User can determine
  - The number of children per branch
  - The total depth of the tree
  - Process must be done within a function
- Childrens must be added by order to an array

UI Scripting

- Design editor (rollout) included in the tools
  - Design done within the editor
  - Functionality added with maxscript




```
-- Selection rollout
rollout SBSelection "Selection Tools" width:477 height:252
(
	GroupBox 'grp1' "Selection" pos:[11,12] width:450 height:222 align:#left
	edittext 'mesh_filter' "Name:" pos:[37,47] width:184 height:21 fieldwidth:150
	button 'btn_select' "Select" pos:[34,175] width:188 height:33 toolTip:"Tip"

	dropdownList 'sel_layers' "" pos:[73,85] width:149 height:21 align:#left

	listbox 'lbx1' "Selection Hierarchy" pos:[274,31] width:161 height:12
	label 'lbl1' "Layers:" pos:[35,87] width:39 height:14 align:#left
	button 'btn13' "Button" pos:[248,36] width:2 height:181 align:#left

	-- On window open
	on SBSelection open do (
		sel_layers.items = man_selection.layers
		sel_cat.items = man_selection.categories
		var2 = sel_layers.selected
		var3 = sel_cat.selected
		lbx1.items = man_selection.updateHierarchy var1 var2 var3
	)

),
```

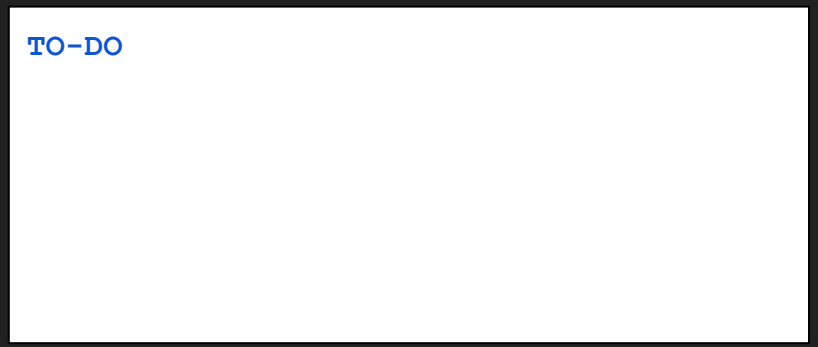
Control mechanisms

- Control mechanism from other languages are also present in maxscript
- Some of these mechanism are:
  - Conditionals
  - Switch statements (very useful)
  - Try expressions

```
a= random 1 20
b = random 1 20
c = random 1 20
d = random 1 20

case of
(
    (a > b): print "a is big"
    (b < c): print "b is little"
    (c <= d*3): print "c is less than 3 times d"
    default: print "none of the above"
)
```

Sample 3

- Create a tool with UI that can create circles, spirals and ellipse
- User can determine
  - By dropdown the type of shape he wants to create
  - The radius (width and height) of the shape depending the type
  - Density of the shape that you want to create



Sample 4

- Create a tool that can
  - Replace the selected items with another item
  - Tool must have an UI to select which element is used as replacement

Resources:

- https://vimeo.com/showcase/1514565?page=1
- www.maxplugins.de
- 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