

# MAXScript Cheat Sheet 1/2

## How and where to enter commands

<b>MAXScript Listener</b>	MAXScript>MAXScript Listener or F11
<b>MAXScript Editor</b>	MAXScript>New Script evaluate with CTRL-E or Tools>Evaluate All

## Selecting objects, \$ and wildcards

<b>\$</b>	current selection
<b>\$Box01</b>	object with the name "Box01"
<b>select \$Box01</b>	select object with the name "Box01"
<b>\$Tile*</b>	objects whose names start with "Tile"
<b>\$'Tile A'*</b>	objects whose names start with "Tile A" (use this for objects with whitespace characters)
<b>\$*</b>	all objects

## Access and Modification of Object Properties

<b>\$.name = "table"</b>	set name of current object to "table"
<b>\$Box01.width = 10</b>	set width of object "Box01"
<b>\$Sphere01.radius = 5</b>	set radius of object "Sphere01"
<b>\$.renderable = false</b>	turn off "renderable" property of current object
<b>\$.&lt;property&gt; = value</b>	general syntax for modifying properties

## Transforming Objects

<b>move \$ [10, 0, 0]</b>	move relatively by 10 units in x
<b>rotate \$ (eulerangles 90 0 0)</b>	rotate relatively by 90 degrees around x
<b>scale \$ [2, 2, 1]</b>	scale twice in x and y, z is unchanged
<b>\$.position = [10, 10, 0]</b>	set new absolute position
<b>\$.position.z = \$.position.z + 10</b>	move object up by 10 units
<b>\$.rotation = (eulerangles 0 0 90)</b>	rotate by 90 degrees around z
<b>\$.scale = [1, 1, 2]</b>	set new scale value: twice as high in z

## Creating Objects, Modifiers and Materials

<b>Box ()</b>	create regular box at the origin
<b>Box pos:[-5, 5, 0] wirecolor:red</b>	create a red box at [-5, -5, 0]
<b>addModifier \$ ( Bend() )</b>	apply bend modifier
<b>addModifier \$ ( Taper amount:1 primaryaxis: 1 )</b>	apply taper modifier with amount 1 and y as primary axis (enter command as one line)
<b>\$.material = StandardMaterial()</b>	apply standard material
<b>\$.material = VRayMtl()</b>	apply VRay material
<b>\$.material = VRayMtl diffuse:green reflection:gray reflection_glossiness:0.7</b>	apply green glossy reflective VRay material
<b>\$.material.diffuse = color 255 0 0</b>	change diffuse color to red (for VRay the name of this property is diffuse, for standard materials it is diffuseColor !)

## Variables and data types, strings, arrays

<b>SCALAR VARIABLES</b> (= numbers, standard mathematical operations: +, -, *, /, ^)	
<b>w = 10</b> <b>h = 10</b> <b>a = w * h</b>	assign the value 10 to w assign the value 10 to h calculate the area a
<b>STRINGS</b> (= characters/text/symbols)	
<b>strPrefix = "LIGHT_"</b> <b>\$.name = strPrefix + \$.name</b>	assign string to a variable apply prefix to object name

<b>ARRAYS</b> (= collection of values and objects of arbitrary types)	
<b>a = #( \$Box01, \$Box02, \$Box03 )</b>	array definition with three objects
<b>a.count</b>	returns number of elements in array
<b>a [1].renderable = false</b>	access elements of array with brackets [ ]

## Random Values

<b>\$.pos.x = random 0 10</b>	random value between 0 and 10 (integer)
<b>\$Box01.height = random 2.0 5.0</b>	random height between 2 and 5 (float)
<b>\$.rotation = eulerangles 0 0 ( random 0.0 360 )</b>	random z rotation (enter as one line)
<b>\$.material.diffuseColor = color 0 0 ( random 0 360 )</b>	random blue part of diffuse color

## Outputting Values to the MAXScript Listener

<b>print \$.position</b>	print position of current object
<b>format "% objects selected." selection.count</b>	output number of selected objects

## Hierarchies and Animation

<b>\$.Tile.parent = \$Wall</b>	link tile to wall object
<b>\$.children [1].name = "Tile_01"</b>	set name of first child object
<b>animate on ( at time 50f ( move \$ [0, 0, 5] ) )</b>	with Autokey on, move object at frame 50
<b>animate on ( at time 100f ( \$.radius = 10 ) )</b>	with Autokey on, set radius at frame 100

## ObjectSets

<b>ObjectSets are collections of the current scene objects divided into the main 3ds max object type categories / classes.</b>	
<b>objects</b>	all objects in the scene
<b>geometry</b>	primitives, meshes and other geometry objects
<b>lights</b>	lights (includes VRay and other third party lights)
<b>cameras</b>	cameras
<b>helpers</b>	helpers
<b>shapes</b>	lines, circles, rectangles, splines
<b>systems</b>	systems (e.g. bones)
<b>spacewarps</b>	forces, spacewarps, deflectors etc.
<b>selection</b>	current selection (this is the same as \$)

## Nifty "one-liners"

<b>Examples for using ObjectSets to apply a command to a bigger group of objects</b>	
<b>delete cameras</b>	deletes all cameras in current scene
<b>delete \$lights/Fill*</b>	delete all lights starting with "Fill"
<b>hide shapes</b>	hides linework
<b>freeze geometry</b>	freeze all geometric objects
<b>\$*.material = undefined</b>	removes all materials of scene
<b>\$Tile*.material = medit.getCurMtl()</b>	apply currently selected material in editor to all objects whose name starts with "Tile"

# MAXScript Cheat Sheet 2/2



## MAXScript control structures

<b>Loops</b> For-Loops are useful for parts of code that need to be executed more than once or, for example, when you would like to change a certain property for many objects.	
<b>for i = 1 to 10 do</b> ( Box pos: [i * 30, 0, 0] )	create 10 boxes at different x positions
<b>for obj in (selection as array)</b> <b>where</b> classOf obj == Cylinder <b>do</b> ( obj.scale.z = 2 )	set scale value for cylinders in current selection
<b>Conditional Statements</b> Conditional statements are used to make a decisions based on comparisons at runtime. They allow you to create different branches and to control the program flow.	
<b>if ( \$.pos.z &lt; 0 ) then</b> \$.wirecolor = red <b>else</b> \$.wirecolor = green	set wirecolor to red if object is below 0, otherwise green
<b>if ( superclassof \$ == Light ) then delete \$</b>	delete current object if it is a light
<b>Functions</b> This is an example function to calculate the volume value from width, length and height. It will return the result, in this case the volume calculation. Generally, functions can be used to encapsulate code of any type or purpose to prevent repetition and clutter within bigger scripts.	
function calculateVolume w l h = ( w * l * h )	Function definition: What are the expected parameters (e.g. width, length, height) and what does the function do with it (e.g. multiply them to calculate volume) ? The result of the last expression in the function body will be returned as a value to the caller of the function.
calculateVolume 10 20 5	Function call with concrete values: return result will be 1000.

## Mass changing properties

<b>for obj in ( selection as array ) do</b> ( obj.primaryVisibility = false )	turn off "visible to camera" property for selected objects
<b>for obj in ( selection as array )</b> <b>where hasProperty obj "radius" do</b> ( obj.radius = 10 )	set the radius property for objects which expose it (applies to spheres and cylinders but not to a box, for example)
<b>for obj in ( selection as array ) do</b> ( <b>for m in obj.modifiers where</b> <b>( classof m == TurboSmooth do</b> ( m.enabled = false ) )	turn off turbosmooth modifiers on selected objects by iterating all objects and also all their modifiers. The classof command checks if m is a TurboSmooth modifier and then disables it.

## Environment Settings

<b>backgroundColor</b> = color 0 0 0	set environment background color
<b>useEnvironmentMap</b> = false	toggle environment map (true/false)
<b>environmentMap</b> = Checker ()	apply environment map (e.g. a checker map)

## Basic Render Settings

<b>rendTimeType</b> = 1	set render type (1=single frame, 2=active time segment, 3= user range, 4= frame string)
<b>rendStart</b> = 100	set render start frame
<b>rendEnd</b> = 250	set render end frame
<b>renderWidth</b> = 1920	set render width (horizontal resolution)

<b>renderHeight</b> = 1080	set render height (vertical resolution)
<b>rendSaveFile</b> = true	activate writing render to disk
<b>rendOutputFilename</b> = "P:\viz01\beauty_v01_.exr"	set render output file path
...	for a full list of render parameters, please consult the MAXScript reference
<b>max quick render</b>	equivalent to pushing the "render" button
<b>render camera:\$Camera01 frame:100 outputfile:</b> "P:\viz01\beauty_v01_.exr"	additional function to kick off renders independent of the current render dialog settings

## Renderer-Specific Settings (e.g. V-Ray)

<b>renderers.current = VRay()</b>	set current renderer to V-Ray
<b>renderers.current.adaptive-Subdivision_minRate</b> = 1	set adaptive sampling min rate
<b>renderers.current.adaptive-Subdivision_maxRate</b> = 2	set adaptive sampling max rate
<b>renderers.current.gi_on</b> = true	turn on V-Ray's GI calculation
...	for a full list of your renderer's parameters, please consult its user manual

## Rollout Floater Basic Framework

The following code is a basic framework for building your own tools with graphical user interfaces (such as rollouts, buttons, sliders etc.) to increase the degree of interactivity of your scripts and provide more advanced functionality.

( <b>global rofTool</b>  <b>Rollout</b> rofTool "Tool" ( <b>spinner</b> spSpinner "Value:" <b>button</b> btnButton "Do it !"  <b>on</b> btnButton <b>pressed do</b> ( format "value: %\n" spSpinner.value ) )  try ( <b>closeRolloutFloater</b> rofTool ) <b>catch</b> ()  rofTool = <b>newRolloutFloater</b> "Tool Floater" 300 100  <b>addRollout</b> rofTool rofTool )	-start a new local variable scope here and declare a global variable for the rollout floater -start a new rollout definition -add a spinner and a button control to the rollout -event handler for the button -output current spinner value -mechanism to prevent opening more than one floater window. If the user executes the script without closing the old floater, it will automatically close it before bringing up the new floater. -add rollout to the floater
---	---

The MAXScript reference provides a lot of examples, FAQs and scripts to help you build better tools and improve your workflow. Furthermore, there are useful websites such as scriptspot.com, cgarchitect.com, cgtalk.com and many more with a big collection of existing scripts and friendly experts to ask. Have fun with MAXScript !