FUSION 8

GENERATION SCRIPTING MANUAL

February 2016

Fusion 8

Fusion 8

Fusion 8

Fusion 8

Introduction

1

Introduction

Scripting in Generation

Scripting can be used for tasks like automation of repetitive processes, extending Generation's functionality or connecting and syncing Generation with third party applications and databases.

Generation supports two programming languages:

- eyeonScript
- Python

eyeonScript is eyeon's native scripting language based on the lightweight Lua language. Python is a popular scripting language adopted by many third party products for visual effects. Consider the official documentations for more information on its features and differences.

- Lua documentation
- Python documentation

Choice of language

The decision on which scripting language to use depends on what other APIs are used, what libraries are needed to support a script or simply on syntax preference. Both languages access the same API and have similar functionality.

Since eyeonScript is shipped with Generation a general recommendation is to use eyeonScript if there are no dependencies on other products or libraries and use Python if requirements exist.

Installation

2

Installation

eyeonScript

eyeonScript does not require an installation prior its use. A script can either run from within Generation or by using the console application called eyeonScript.exe. To start eyeonScript as an interactive console use the following command line option:

```
eyeonScript.exe -i
```

In order to run a script file with eyeonScript drop it on eyeonScript.exe or pass it as argument.

Python

Generation uses the local Python installation. Install the 64 bit version of CPython 2.7 or 3.1 from python.org. Generation will run python scripts from the “Scripts” subfolder or by connecting through PeyeonScript in an external python.exe instance. Make sure you have eyeon’s latest PeyeonScript library installed from eyeon- line.com.

Principles

3

Principle

Internal and external scripting

If a script is being executed from an external eyeonScript or Python instance it needs to establish a connection to Generation.

→ Python

```
# Establishes a connection to Generation if not already set

import PeyeonScript as eyeon

gen = eyeon.scriptapp(“Generation”)
```

→ eyeonScript

```
-- Establishes a connection to Generation if not already set

gen = gen or Generation()
```

Using the user interface to execute scripts doesn’t require any import for eyeonScript. However you should make sure the the handle is defined.

Types of scripts

Generation, being a highly customizable application, makes use of scripting in various ways:

2.1 Interface scripts

Interface scripts are scripts that create functionality in Generations interface by showing up as a command button.

Toolbar scripts

Toolbar scripts create menu items on the toolbar that contain context menus with scripts as buttons. Clicking a script causes the script to be run. Toolbar Scripts are created based on the folder structure and naming in the Generation Script\generation\toolbar subfolder. Each folder represents a menu and each contained script name will be used as label.

Userbar scripts

Userbar scripts are similar to Toolbar scripts but instead the floating window called Userbar will receive but- tons that execute a script.

2.2 Event Scripts

Event scripts are scripts that are executed automatically when certain events are triggered. They have to follow a special format and naming convention in order to run.

Metadata scripts

Scripts that are created in the Scripts\events\metadata subfolder act as metadata scripts. They act as callback for each file that is loaded in. If a function called GetMetaData is present this function will make use of three arguments which hold the item, metadata and schema. The item has access to the item in Generation and therefor its file name etc. The metadata handle points to the items metadata and may be used to store arbitrary data. schema is reserved for various additional information. A global variable called SubKey defines the parent name of the metadata handle. The following example stores creation date, modification date and the filesize on each item at a metadata key called "FileAttributes":

→ Python

```
import os.path, time

SubKey = “FileAttributes”

def GetMetaData(item, data, schema):

    filename = item[“File”][“Path”]

    data[“Created”] = time.ctime(os.path.getctime(filename))

    data[“Modified”] = time.ctime(os.path.getmtime(filename))

    data[“Size”] = os.path.getsize(filename)
```

Place this in a python file inside the Scriptsmetadata subfolder.

DropFile scripts

Scripts that are created in the Scripts\events\events subfolder act as events scripts. If a function called Project.DropFiles is present this function will be called whenever a file is dropped into Generation. The obj argument points to the project in question while the args have a Known attribute that is true if Generation is able to load the file on its own. The args.Path and args.Filename point to the location of the file that has been dropped. args.Count will be more then one if a file sequence was dropped.

→ eyeonScript

```
function Project.DropFiles(obj, args)
    if args.Known then
        print("Generation will handle known images, video etc. formats")
    else
        print("Do something with the file at " .. args.Path .. args.Filename)
    end
end
```

→ Python

```
def dropfiles(obj, args):
    if args["Known"]:
        print("Generation will handle known images, video etc. formats")
    else:
        print("Do something with the file at " + args["Path"] + args["Filename"])
Project.DropFiles = dropfiles # Monkeypatch the dropfile event with custom function
```

Object Model

Generation scripts address data by objects. These are organized in a very straight forward hierarchy. The hierarchy important for scripting is:

```
Generation - The application handle. Generation itself.
    Project - A list of projects in the application. This can hold multiple Subs.
        Sub - A list of Sub-projects/Subs in the project. This can hold multiple Tracks.
            TrackM - A list of tracks in the Sub. This can hold multiple Clips.
                ClipM - This can hold multiple ClipVs for versioning.
                    ClipV - A clip version. Has reference to its loader. Loader
                     - A loader is the reference to the physical media.
```

So to establish a handle to a specific file on a clip use:

```
loader = App():ProjectGet():SubGet():TrackGet():ClipGet():VersionGet():GetLoader()
```

It is recommended to make use of variables for easier access. A Project handle could look like this:

```
proj = App():Project()
track = proj:SubGet():TrackGet()
```

Getting started

4

Getting started

This example explores Generation scripting with eyeonScript/Lua. Use the python version of this document if you prefer to work with python.

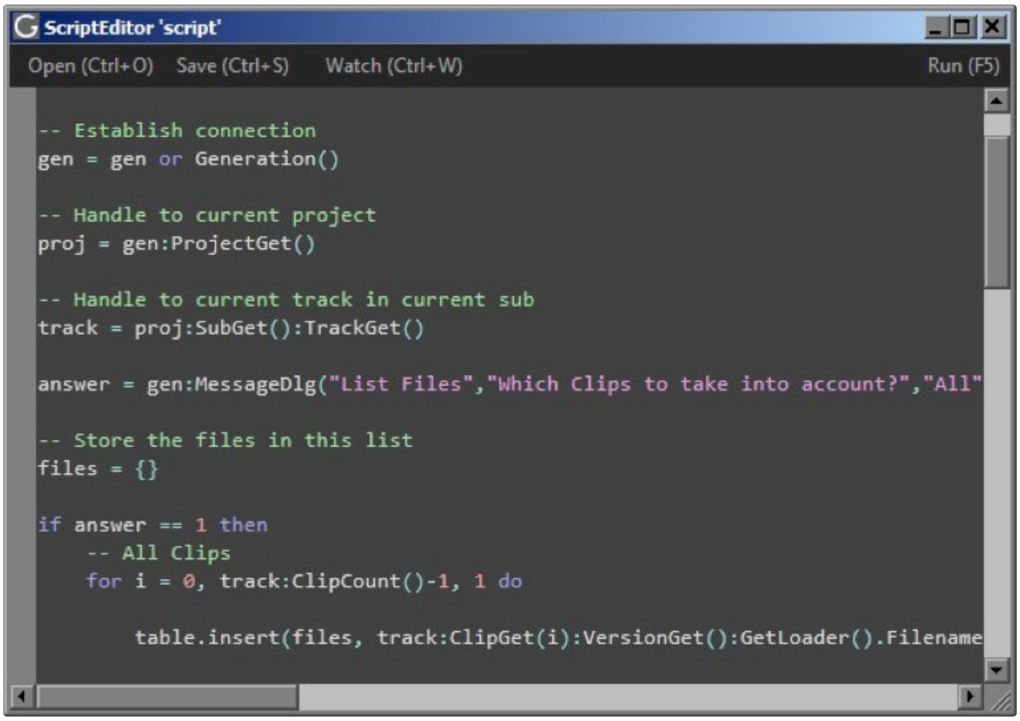
Scripting Editor

5

Scripting Editor

General Usage

The script editor allows you to write and execute Lua scripts or snippets directly within Generation. The script editor opens with a Log window that shows the output of the script operations. The windows are opened by pressing Ctrl+Shift+L.



```
-- Establish connection
gen = gen or Generation()

-- Handle to current project
proj = gen:ProjectGet()

-- Handle to current track in current sub
track = proj:SubGet():TrackGet()

answer = gen:MessageDlg("List Files","Which Clips to take into account?","All"

-- Store the files in this list
files = {}

if answer == 1 then
    -- All Clips
    for i = 0, track:ClipCount()-1, 1 do

        table.insert(files, track:ClipGet(i):VersionGet():GetLoader().Filename
```

The most common operations are:

- Directly write Lua code
- Open existing script with the Open button or by pressing Ctrl+O
- Run the scripts in the buffer with the Run button or by pressing F5
- Save the buffer into a script file with the Save button or by pressing Ctrl+S

The text input, also referred to as buffer, supports syntax highlighting. This visually distinguishes different types of Lua code with colors.

Debugging

Additionally, to easy the development of scripts, the ScriptEditor includes debugging features. This includes common functions like breakpoint, stepping through code line by line as well as a Watch window.

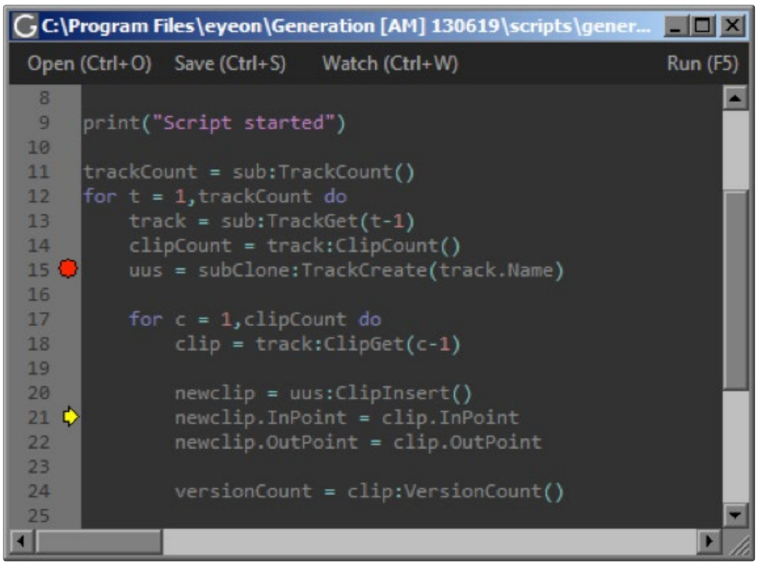
The Shortcuts for debugging are:

F9	Toggle breakpoint on current line
F5	Run script under debugger
Ctrl-F5	Run script (no debug)
F10	Step-Over (single step, stepping over function calls)
F11	Step-Into (single step, stepping into function calls)
Shift-F11	Step-Out (run until return from current function)
Ctrl-W	Open watch window

2.1 Breakpoints

Breakpoints allow to pause the execution of the code in the buffer before executing the code. You can toggle the breakpoint on the line of the text cursor by pressing F9.

2.2 Stepping of code


C:\Program Files\eyeon\Generation [AM] 130619\scripts\gener...
Open (Ctrl+O)
Save (Ctrl+S)
Watch (Ctrl+W)
Run (F5)

```

print("Script started")

trackCount = sub:TrackCount()
for t = 1,trackCount do
    track = sub:TrackGet(t-1)
    clipCount = track:ClipCount()
    uus = subClone:TrackCreate(track.Name)

    for c = 1,clipCount do
        clip = track:ClipGet(c-1)

        newclip = uus:ClipInsert()
        newclip.InPoint = clip.InPoint
        newclip.OutPoint = clip.OutPoint

        versionCount = clip:VersionCount()

```

With F10 only one line of the buffer is executed and then the execution is paused until you either run the rest of the script or make another step.

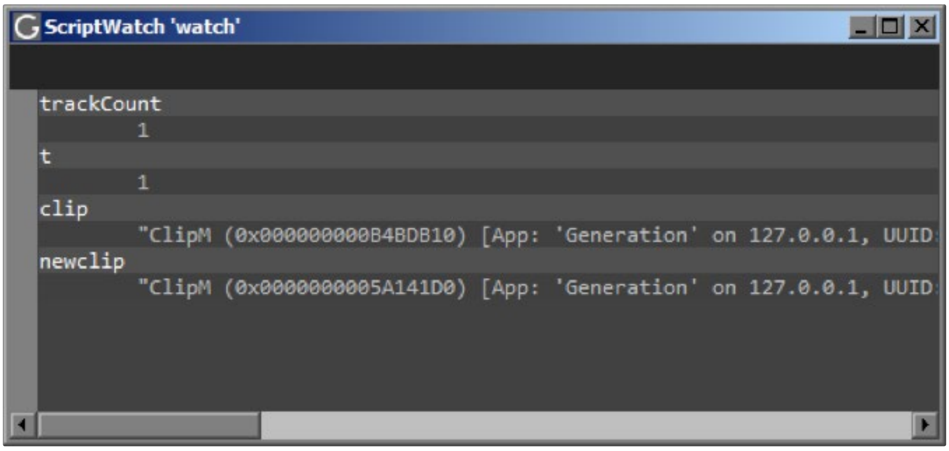
F10 steps over function calls. This means that if a function is triggered it will be completed and the execution is stepped to the next line.

F11 Steps-Into function calls. The stepping is therefore resumed inside the function.

You can leave the function call into the outer function by pressing Shift-F11 to Step-Out.

2.3 Watch Window

The watch window is opened by the Watch button or by pressing Ctrl+W. This view allows you to monitor variables of the buffer by showing at any point what values they store in memory.



```
trackCount
        1
t
        1
clip
        "ClipM (0x000000000B4BDB10) [App: 'Generation' on 127.0.0.1, UUID
newclip
        "ClipM (0x0000000005A141D0) [App: 'Generation' on 127.0.0.1, UUID
```

To watch a variable simply write the variable name in the watch window. On the next debug execution the variable value is displayed beneath the variable in the watch window. This is only true as long as the variable is valid. After the script has finished execution you won't see the variable value anymore.

To watch multiple variables simply use one line per variable.

Debugging Tutorial

In this tutorial the debugging functions will be explained in practice. Open the Script Editor and Log window by pressing Ctrl+Shift+L. Copy and Paste the following Lua code in the Script Editor:

```
function SafeSum(a, b)
    return a + b
end

j = 10
k = nil
for i = 1, 20, 2 do
    print ( SafeSum(i, j) )
end
```

Now simply run the code by hitting F5.

The Log window will show a similar output to this:

```
------run script-------
[string “Main”]:2: attempt to perform arithmetic on local ‘b’ (a nil value)
```

This tells us that there is a problem on line 2 of the code.

3.1 Watching variables

Open the ScriptWatch by pressing the Watch button. In the WatchWindow write the following lines:

```
a
b
```

Now go to line number two of the ScriptEditor and press F9 to create a breakpoint. A red dot next to the line number marks the breakpoint. Press F5 to run the script.

A yellow arrow indicates the position where the execution stopped. This should be exactly on the red break- point on line 2. Note that the ScriptEditor is greyed out. You can not alter code while it is being executed.

In the ScriptWatch the values of a and b should be visible.

```
a
    1
b
    nil
```

3.2 Stepping over and breakpoints

Note that the execution on a breakpoint is paused before the line was executed. So the next action on line 2 will be:

```
return a+b
```

Obviously Lua does not know how to compute the result of 1 and nil. Press F10 to step over that line and see that this line generates the error message that was displayed earlier in the log window. Since an exception like this stops the execution of the script the yellow arrow disapears, while the code becomes editable again.

That was pretty fast. Let's go over the whole script line by line and see where the problem starts. First disable the breakpoint by moving the text cursor to line 2 and hitting F9 again. The red dot disappeared.

Now press F10 to start the execution of the whole script step by step. You will notice by the arrow that the function SafeSum is evaluated first, which means that is is stored in memory but not yet executed. Press F10 again until the arrow is at line 5.

```
k = nil
```

To watch the variable k simply add k as a new line to the script watch. Now press F10 to step over to line 6, the for loop. Also add the i variable as new line into the ScriptWatch.

Hit F10 again. The arrow pauses at line 7, the print statement. In the ScriptWatch k is nil while i is evaluated as

1. In order to Step into the SafeSum function press F11 this time. The arrow jumps to line 2 inside the SafeSub function.

As we can see in the ScriptWatch a is 1 and all the other variables are nil. Why is i also nil? The ScriptWatch only shows the variables in the current scope. This means that it only shows the values of the variables in the current part of the execution.

Inside the SafeSum function there is no access to the i or k variables similar as inside the for loop there is no access to the a and b variables of the SafeSum function's scope.

3.3 Fixing the bug

As we know the next execution will trigger the exception. But now we know where to look for the error. The k variable is nil and is passed to the SafeSum.

Let's make sure the Sum function checks for this first. After all it is called Safe Sum. Change the function accordingly:

```
function SafeSum(a, b)

if a == nil and b == nil then return nil end

if a == nil then return b end

if b == nil then return a end

return a + b

end
```

This will check first:

- If either of the variables is nil.
- If both are it will return nil.
- If a is nil it will return b.
- If b is nil it will return a.
- If neither a or b is nil it will return the sum.

Let's go over the script once again. Go to the print statement at line 10. Hit F9 to mark it as breakpoint. Hit F5 to execute the buffer.

It will pause at the breakpoint. Hit F11 to Step into the SafeSum function.

Hit F10 a few times and you will see that the last if statement triggers the return of a, since b is nil.

The execution will jump to the for loop. Use F10 to run through the loop by stepping over the SafeSum function.

Now that we are happy we can either toggle the breakpoint of or use Shift-F5 to resume the execution without the debugging. This means that the script will be executed and breakpoint will be ignored.

Reference

6

Reference

This is a complete reference to all available objects in Generation.

PeyeonScript -- eyeon module

The PeyeonScript is the main Python module that connects to the eyeon products. It is available in eyeon- Script by default as eyeon. Therefore it is recommended to import it in Python as eyeon.

```
# Get basic connection to generation.
import PeyeonScript as eyeon
gen = eyeon.scriptapp(“Generation”)
```

In eyeonScript/Lua you do not need PeyeonScript. All you need is to check if a handle to Generation is already available.

```
-- Get basic connection to generation.
gen = gen or Generation()
```

_____LuaScript

class_____LuaScript

Parent class: Object

_____LuaScript

____Fuse

class____Fuse

Parent class: Object

____Fuse

_TimeSpeed

class_TimeSpeed

Parent class: ThreadedOperator

_TimeSpeed

_TimeStretcher

class_TimeStretcher

Parent class: `ThreadedOperator`

_TimeStretcher

AbcImport

class `AbcImport`

Parent class: `Utility`

AbcImport

AbcKeyframe

class `AbcKeyframe`

Parent class: `Parameter`

AbcKeyframe

AccumBuffer3D

class `AccumBuffer3D`

Parent class: `Object`

AccumBuffer3D

AccumBufferCpu3D

class `AccumBufferCpu3D`

Parent class: `AccumBuffer3D`

AccumBufferCpu3D

AccumBufferGpu3D

class `AccumBufferGpu3D`

Parent class: `AccumBuffer3D`

AccumBufferGpu3D

AccumSSFramebuffer

class `AccumSSFramebuffer`

Parent class: `SSFramebuffer`

AccumSSFramebuffer

ActionManager

class `ActionManager`

Parent class: `LockableObject`

ActionManager

AirBrush

class `AirBrush`

Parent class: `PaintTool`

AirBrush

ALUT3Format

class `ALUT3Format`

Parent class: `LUTFormat`

ALUT3Format

ALUTFormat

class `ALUTFormat`

Parent class: `LUTFormat`

ALUTFormat

ALUTLookUpTable

class `ALUTLookUpTable`

Parent class: `LookUpTable`

ALUTLookUpTable

Anaglyph

`class Anaglyph`

Parent class: `ThreadedOperator`

Anaglyph

AngleControl

`class AngleControl`

Parent class: `PreviewControl`

AngleControl

ArriRawFormat

`class ArriRawFormat`

Parent class: `ImageFormat`

ArriRawFormat

Audio

`class Audio`

Parent class: `Parameter`

Audio

AudioDisplay

`class AudioDisplay`

Parent class: `ThreadedOperator`

AudioDisplay

AudioPreview

`class AudioPreview`

Parent class: `Preview`

AudioPreview

AutoDomain

class `AutoDomain`

Parent class: `ThreadedOperator`

AutoDomain

AutoGain

class `AutoGain`

Parent class: `ThreadedOperator`

AutoGain

Background

class `Background`

Parent class: `SourceOperator`

Background

Bender3D

class `Bender3D`

Parent class: `SceneGeometryOperator3D`

Bender3D

BetterResize

class `BetterResize`

Parent class: `ThreadedOperator`

BetterResize

BezierSpline

class `BezierSpline`

Parent class: `Spline`

BezierSpline

Methods

`BezierSpline.AdjustKeyFrames(`*`start, end, x, y, operation`*`[, `*`pivotx`*` ][, `*`pivoty`*` ])`

AdjustKeyFrames

→ Parameters

start (*number*) – start

end (*number*) – end

x (*number*) – x

y (*number*) – y

operation (*string*) – operation

pivotx (*number*) – pivotx

pivoty (*number*) – pivoty

`BezierSpline.DeleteKeyFrames(`*`start`*`[, `*`end`*` ])`

DeleteKeyFrames

→ Parameters

start (*number*) – start

end (*number*) – end

`BezierSpline.GetKeyFrames()`

GetKeyFrames

→ **Returns**: keyframes

→ **Return type**: table

`BezierSpline.SetKeyFrames(`*`keyframes`*`[, `*`replace`*` ])`

SetKeyFrames

→ Parameters

keyframes (*table*) – keyframes

replace (*boolean*) – replace

Bin

class `Bin`

Parent class: `Object`

Bin

BinClip

class `BinClip`

Parent class: `BinItem`

BinClip

Methods

`BinClip.CreateStamp()`

CreateStamp

`BinClip.Defragment()`

Defragment

`BinClip.DeleteStamp()`

DeleteStamp

BinComp

class `BinComp`

Parent class: `BinItem`

BinComp

BinFile

class `BinFile`

Parent class: `BinItem`

BinFile

BinFolder

class `BinFolder`

Parent class: `BinItem`

BinFolder

BinGeneric

class `BinGeneric`

Parent class: `BinItem`

BinGeneric

BinItem

class `BinItem`

Parent class: `Object`

BinItem

Methods

`BinItem.Delete()`

Delete

`BinItem.GetData([`*`name`*` ])`

GetData

- **Parameters name**: (*string*) – name
- **Returns**: Value
- **Return type**: (*number|string|boolean|table*)

`BinItem.SetData(`*`name, value`*`)`

SetData

- **Parameters**

name (*string*) – name

value ((*number|string|boolean|table*)) – value

BinManager

class `BinManager`

Parent class: Object

BinManager Methods

Bins

`class Bins`

Parent class: `Utility`

Bins

BinSetting

`class BinSetting`

Parent class: `BinTool`

BinSetting

BinStill

`class BinStill`

Parent class: `BinItem`

BinStill

Methods

`BinStill.Defragment()`

Defragment

BinTool

`class BinTool`

Parent class: `BinItem`

BinTool

BinTree

`class BinTree`

Parent class: `Object`

BinTree

BitmapMask

`class BitmapMask`

Parent class: `MaskOperator`

BitmapMask

Blackmagic_Preview

class `Blackmagic_Preview`

Parent class: `Preview`

lackmagic_Preview

BlendModeDataGL

class `BlendModeDataGL`

Parent class: `Data3D`

BlendModeDataGL

BlendModeDataSW

class `BlendModeDataSW`

Parent class: `Data3D`

BlendModeDataSW

BlendModeInputsGL

class `BlendModeInputsGL`

Parent class: `Inputs3D`

BlendModeInputsGL

BlendModeInputsSW

class `BlendModeInputsSW`

Parent class: `Inputs3D`

BlendModeInputsSW

Blur

class `Blur`

Parent class: `ThreadedOperator`

Blur

BMPFormat

class `BMPFormat`

Parent class: `ImageFormat`

BMPFormat

BrightnessContrast

class `BrightnessContrast`

Parent class: ThreadedOperator

BrightnessContrast

BSpline

class `BSpline`

Parent class: `Spline`

BSpline

BSplineMask

class `BSplineMask`

Parent class: `PolylineMask`

BSplineMask

BSplinePolyline

class `BSplinePolyline`

Parent class: `Polyline`

BSplinePolyline

BumpMap

class `BumpMap`

Parent class: `MtlOperator3D`

BumpMap

BumpMapData

class `BumpMapData`

Parent class: `MtlData`

BumpMapData

BumpMapGL

class `BumpMapGL`

Parent class: `MtlGL`

BumpMapGL

BumpMapInputs

class `BumpMapInputs`

Parent class: `MtlInputs`

BumpMapInputs

BumpMapSW

class `BumpMapSW`

Parent class: `MtlSW`

BumpMapSW

ButtonControl

class `ButtonControl`

Parent class: `InputControl`

ButtonControl

Calculation

class `Calculation`

Parent class: `ThreadedOperator`

Calculation

Camera3D

class `Camera3D`

Parent class: `CameraOperator`

Camera3D

CameraControl

class `CameraControl`

Parent class: `TransformControl`

CameraControl

CameraData

class `CameraData`

Parent class: `Data3D`

CameraData

CameraOperator

class `CameraOperator`

Parent class: `Transform3DOperator`

CameraOperator

CameraShake

class `CameraShake`

Parent class: `ThreadedOperator`

CameraShake

CameraStdData3D

class `CameraStdData3D`

Parent class: `CameraData`

CameraStdData3D

CameraViewData

class `CameraViewData`

Parent class: `CameraData`

CameraViewData

CanonRMFFormat

class `CanonRMFFormat`

Parent class: `ImageFormat`

CanonRMFFormat

CardinalSplinePolyline

class `CardinalSplinePolyline`

Parent class: `Polyline`

CardinalSplinePolyline

ChangeDepth

class `ChangeDepth`

Parent class: `ThreadedOperator`

ChangeDepth

ChannelBoolean

class `ChannelBoolean`

Parent class: `ThreadedOperator`

ChannelBoolean

ChannelInputs3D

class `ChannelInputs3D`

Parent class: `Inputs3D`

ChannelInputs3D

ChatView

class `ChatView`

Parent class: `FuView`

ChatView

CheckboxControl

class `CheckboxControl`

Parent class: `InputControl`

CheckboxControl

CheckListControl

class `CheckListControl`

Parent class: `InputControl`

CheckListControl

ChildFrame

class `ChildFrame`

Parent class: `FuFrame`

ChildFrame

Methods

`ChildFrame.ActivateFrame()`

ActivateFrame

`ChildFrame.ActivateNextFrame()`

ActivateNextFrame

`ChildFrame.ActivatePrevFrame()`

ActivatePrevFrame

`ChildFrame.GetControlViewList()`

GetControlViewList

→ **Returns**: views

→ **Return type**: table

`ChildFrame.GetMainViewList()`

GetMainViewList

- **Returns**: views
- **Return type**: table

`ChildFrame.GetViewLayout()`

GetViewLayout

- **Returns**: layout
- **Return type**: table

`ChildFrame.SetViewLayout(`*`layout`*`)`

SetViewLayout

- **Parameters**:

  layout (table) – layout
- **Returns**: success
- **Return type**: boolean

`ChildFrame.SwitchControlView(`*`id`*`)`

SwitchControlView

- **Parameters**:

  id (*string*) – id

`ChildFrame.SwitchMainView(`*`id`*`)`

SwitchMainView

- **Parameters**:

  id (*string*) – id

ChildGroup

`class ChildGroup`

Parent class: `Object`

ChildGroup

Methods

`ChildGroup.GetID()`

GetID

`ChildGroup.GetOwner()`

GetOwner

ChromaKeyer

`class ChromaKeyer`

Parent class: `ThreadedOperator`

ChromaKeyer

CineFormat

`class CineFormat`

Parent class: `ImageFormat`

CineFormat

CineonFormat

`class CineonFormat`

Parent class: `ImageFormat`

CineonFormat

CineonInputs

`class CineonInputs`

Parent class: `SubInputs`

CineonInputs

CineonLog

`class CineonLog`

Parent class: `ThreadedOperator`

CineonLog

Circle

`class Circle`

Parent class: `PaintTool`

Circle

CircleBrush

class `CircleBrush`

Parent class: `PaintBrush`

CircleBrush

Clip

class `Clip`

Parent class: `Parameter`

Base class of ClassM and ClassV.

ClipControl

class `ClipControl`

Parent class: `InputControl`

ClipControl

ColorControl

class `ColorControl`

Parent class: `InputControl`

ColorControl

ColorCorrector

class `ColorCorrector`

Parent class: `ThreadedOperator`

ColorCorrector

ColorCurves

class `ColorCurves`

Parent class: `Parameter`

ColorCurves

ColorGain

class `ColorGain`

Parent class: `ThreadedOperator`

ColorGain

ColorGamutControl

class `ColorGamutControl`

Parent class: `InputControl`

ColorGamutControl

ColorGamutList

class `ColorGamutList`

Parent class: `HashList`

ColorGamutList

ColorRangesControl

class `ColorRangesControl`

Parent class: `InputControl`

ColorRangesControl

ColorSpace

class `ColorSpace`

Parent class: `ThreadedOperator`

ColorSpace

ColorSuppressionControl

class `ColorSuppressionControl`

Parent class: `InputControl`

ColorSuppressionControl

ColorWheelControl

class `ColorWheelControl`

Parent class: `InputControl`

ColorWheelControl

Combiner

class `Combiner`

Parent class: `ThreadedOperator`

Combiner

ComboControl

class `ComboControl`

Parent class: `InputControl`

ComboControl

ComboIDControl

class `ComboIDControl`

Parent class: `InputControl`

ComboIDControl

CompName

class `CompName`

Parent class: `ThreadedOperator`

CompName

Composition

class `Composition`

Parent class: `Object`

Composition

Attributes

`Composition.ActiveTool`

→ Getting:

tool = Composition.ActiveTool (*Tool*)

`Composition.AutoPos`

→ Getting:

val = Composition.AutoPos (*boolean*)

→ Setting:

Composition.AutoPos = val (*boolean*)

`Composition.CurrentFrame`

→ Getting:

frame = Composition.CurrentFrame (*FuFrame*)

`Composition.CurrentTime`

→ Getting:

val = Composition.CurrentTime (*number*)

→ Setting:

Composition.CurrentTime = val (*number*)

`Composition.UpdateMode`

→

→

`Composition.XPos`

→ Getting:

val = Composition.XPos (*number*)

→ Setting:

Composition.XPos = val (*number*)

`Composition.YPos`

→ Getting:

val = Composition.YPos (*number*)

→ Setting:

Composition.YPos = val (*number*)

Methods

`Composition.AddTool(id[, defsettings ][, xpos ][, ypos ])`

AddTool

→ **Parameters**:

id (*string*) – id

defsettings (*boolean*) – defsettings

xpos (*number*) – xpos

ypos (*number*) – ypos

Returns: tool

Return type: Tool

`Composition.AddToolAction(id[, xpos ][, ypos ])`

AddToolAction

→ **Parameters**:

id (*string*) – id

xpos (*number*) – xpos

ypos (*number*) – ypos

Returns: tool

Return type: Tool

`Composition.AskUser(title, controls)`

AskUser

→ **Parameters**:

title (*string*) – title

controls (*table*) – controls

Returns: results

Return type: table

`Composition.ChooseTool(path)`

ChooseTool

→ **Parameters**:

path (*string*) – path

Returns: ID

Return type: string

`Composition.ClearUndo()`

ClearUndo

`Composition.Close()`

Close

`Composition.Copy()`

Note: This method is overloaded and has alternative parameters. See additional reference.

Copy

Returns: success

Return type: boolean

`Composition.Copy(`*`tool`*`)`

Note: This method is overloaded and has alternative parameters. See additional reference.

Copy

→ Parameters

tool (*Tool*) – tool

Returns: success

Return type: boolean

`Composition.Copy(`*`toollist`*`)`

Note: This method is overloaded and has alternative parameters. See additional reference.

Copy

→ Parameters

toollist (*table*) – toollist

Returns: success

Return type: boolean

`Composition.CopySettings()`

Note: This method is overloaded and has alternative parameters. See additional reference.

CopySettings

Returns: settings

Return type: table

`Composition.CopySettings(`*`tool`*`)`

Note: This method is overloaded and has alternative parameters. See additional reference.

CopySettings

→ **Parameters**:

tool (*Tool*) – tool

Returns: settings

Return type: table

`Composition.CopySettings(`*`toollist`*`)`

Note: This method is overloaded and has alternative parameters. See additional reference.

CopySettings

→ **Parameters**

toollist (*table*) – toollist

Returns: settings

Return type: table

`Composition.DisableSelectedTools()`

DisableSelectedTools

`Composition.EndUndo(`*`keep`*`)`

EndUndo

→ **Parameters**:

keep (*boolean*) – keep

`Composition.FindTool(`*`name`*`)`

FindTool

→ **Parameters**:

name (*string*) – name

Returns: tool

Return type: Tool

`Composition.FindToolByID(`*`id`*`[, `*`prev`*` ])`

FindToolByID

→ **Parameters**:

id (*string*) – id

prev (*Tool*) – prev

Returns: tool

Return type: Tool

`Composition.GetCompPathMap([`*`built_ins`*` ][, `*`defaults`*` ])`

GetCompPathMap

→ **Parameters**:

built_ins (*boolean*) – built_ins

defaults (*boolean*) – defaults

Returns: map

Return type: table

`Composition.GetData([`*`name`*` ])`

GetData

→ **Parameters**:

name (*string*) – name

Returns: value

Return type: (*number|string|boolean|table*)

`Composition.GetNextKeyTime([`*`time`*` ][, `*`tool`*` ])`

GetNextKeyTime

→ **Parameters**:

time (*number*) – time

tool (*Tool*) – tool

Returns: time

Return type: number

`Composition.GetPrefs([`*`prefname`*` ][, `*`exclude-defaults`*` ])`

GetPrefs

→ **Parameters**:

prefname (*string*) – prefname

exclude-defaults (*boolean*) – exclude-defaults

Returns: prefs

Return type: table

`Composition.GetPrevKeyTime([`*`time`*` ][, `*`tool`*` ])`

GetPrevKeyTime

→ **Parameters**:

time (*number*) – time

tool (*Tool*) – tool

Returns: time

Return type: number

`Composition.GetPreviewList([include_globals ])`

GetPreviewList

→ **Parameters**:

include_globals (*boolean*) – include_globals

Returns: previews

Return type: table

`Composition.GetToolList([selected ][, regid ])`

GetToolList

→ **Parameters**:

selected (*boolean*) – selected

regid (*string*) – regid

Returns: tools

Return type: table

`Composition.IsLocked()`

IsLocked

Returns: locked

Return type: boolean

`Composition.IsPlaying()`

IsPlaying

Returns: playing

Return type: boolean

`Composition.IsRendering()`

IsRendering

Returns: rendering

Return type: boolean

`Composition.Lock()`

Lock

Indices and tables

7

Indices and tables

- genindex
- modindex
- search

Index

8

Python Module Index

Index