installPseudoEffects
checkPresetEffectsVersion

Duik.uiStrings
Attributes

Duik.settings
Attributes
Methods
save
load
restoreDefaults

Duik.utils
Methods
prepareProperty
getPropertyDimensions
getLength
getAverageSpeed
addPseudoEffect
getPuppetPins
getDistance
rigProperty
deselectLayers
checkNames
getItem
getKey
getPropertyAnims
getPropertyAnim
setPropertyAnim
addKey
getFirstKeyTime
hasSelectedKeys
convertCollectionToArray
prepIK
getControllers
getAverageSpeeds
replaceInExpressions
replaceInLayersExpressions
renameLayer
renameItem

Introduction

libDuik is a complete library of objects, attributes and methods from Duik – Duduf IK & Animation Tools for After Effects. It allows to easily include Duik functions into other scripts.

Including libDuik in your scripts

There are three ways to use libDuik in your scripts:

• ***#include «libDuik.jsxinc»***
Adding this line at the beginning of the script automatically loads libDuik at first run of the script. *libDuik.jsxinc* must be in the same folder as your script.
This is the recommended way of including libDuik.

• **Copying all content of libDuik.jsxinc in the beginning of your script**
Copying the whole library inside your script allows you to deploy only one file.

• **Renaming *libDuik.jsxinc* to *libDuik.jsx* and move it to Scripts/Startup/**
libDuik will be loaded during After Effects startup, and will then be available to all scripts. This is a good way to use Duik functions in severeal scripts without having to include libDuik in all scripts.

Installing libDuik

• **Using pseudo effects**
This is the default behaviour, and you should prefer to use libDuik this way.

At first launch, libDuik will automatically check if the pseudo effects it needs are already installed, and, if not, it will attempt to install them, by writing them in the file called *presetEffects.xml* inside the installation folder of After effects.
To achieve this, **libDuik needs to be allowed to write files** by After Effects. The only way to do this is for the user to check the box called « Allow scripts to write files... » in the general preferences of After Effects.

Note: You can open the preferences dialog in your scripts with:

```
app.executeCommand(2359);
```

but the user will have to check the box itself.

After the very first run of libDuik, if the pseudo effects were not already available, the user will have to restart After Effects for the pseudo effects to be loaded by After Effects.

If you want to use libDuik without allowing the scripts to write files, you can manually add the pseudo effects to *presetEffects.xml*: Copy/paste the content of the file Duik_presetEffects.xml distributed with libDuik, in *presetEffects.xml*, just before the last line « </effects> ».
Note that on Mac OS you will have to change the file permissions to be able to modify it.

• **Using presets**
If you cannot modify *presetEffects.xml*, or for any other reason, you can use *.ffx* presets.
You just have to set *Duik.usePresets* to *true*.

Note: if libDuik was not able to update *presetEffects.xml*, it will default *Duik.usePresets* to

true. If *presetEffects.xml* is up-to-date, ***Duik****.usePresets* will be ***false*** by default.

By default, libDuik will look for *.ffx* files inside its own folder. You can specify another folder by setting the path to *Duik.presetPath* with an ending « / ».

The *.ffx* files must be named by the corresponding pseudo effects matchNames plus the extension (*.ffx*). A complete list of those matchNames is available in this document.

Note: if *presetEffects.xml* is not updated with libDuik pseudo effects, when using presets After Effects may warn for missing effects. libDuik will work well anyway.

Note: the presets distributed with libDuik are CC2014 versions (for this alpha version of libDuik. Later versions may be distributed with CS6, or even CS4 versions of presets). Sadly, After Effects presets cannot be used with older versions of After Effects than the one used to create them. If you need to use presets with older versions, you will have to create your own.

Using libDuik

Once libDuik has been loaded, all its classes, attributes and methods are available in the javascript object *Duik*, for all scripts run by After Effects.

libDuik is loaded only once; this allows a faster launch of your scripts.

Modifying libDuik

If you're modifying libDuik and need to test it without having to reboot After Effects to reload it, you can un-comment the first line:

*if (typeof **Duik** === 'object') delete **Duik**;*

inside libDuik itself, or you can include this line in your own script **before** *#include libDuik;*

Pseudo Effects List

libDuik uses pseudo effects instead of expression controls. Those effects must be added to *presetEffects.xml* (see *Introduction, Installing libDuik* for more details).

The XML code used to create those effects is *Duik_presetEffects.xml*

Here is a list of the effects available.

Those effects can be added on any layer with:

```
layer.effect.addProperty(matchName)
```

Example:

```
app.project.activeItem.layer(1).effect.addProperty('DUIK_One_Layer_IK');
```

matchName	Description	Screenshot
DUIK_One_Layer_IK	Used by one layer IK	
DUIK_Two_Layer_IK	Used by two layer IK	
DUIK_3D_Wiggle	Used for wiggle on 3D properties	
DUIK_2D_Wiggle	Used by wiggle on 2D properties	
DUIK_1D_Wiggle	Used by wiggle on 1D properties	
DUIK_Exposure	Used by exposure, in fixed mode	
DUIK_RotMorph	Used by Rotation Morph	
DUIK_Swing	Used by Swing (oscillation)	
DUIK_Wheel	Used by Wheel	
DUIK_LensFlare	Used by Lens Flare on the layer of the center to control size and intensity	
DUIK_LensFlareDistance	Used by Lens Flare on flare layers to control their distance from the center	
DUIK_DistanceLink	Used by Distance Link	
DUIK_Spring	Used by Spring on 2D and 3D properties	
DUIK_Spring_Bounce	Used by spring on 1D properties, includes a checkbox called 'bounce'.	
DUIK_Paint_Rig	Used by the paint rig tool to control the end, begin and diameter properties of the paint brushes	
DUIK_Blink_1D	Used by blink on 1D properties	

DUIK_Blink_2D	Used by blink on 2D properties	
DUIK_Blink_3D	Used by blink on 3D properties	

Objects

libDuik creates new javascript instantiable javascript objects, which can be very helpful when working with After Effects, and are needed by Duik.

Name	Description
KeySpatialProperties	Describes all spatial properties of a KeyFrame.
KeyFrame	Represents an animation keyframe of After Effects
PropertyAnim	Describes the keyframe animation of a given property
MaskAnim	Describes all the keyframe animations of the properties of a given Mask
EffectAnim	Describes all the keyframe animations of the properties of a given Effect
LayerAnim	Describes all the keyframe animations of the transformation, masks, and effects of a layer
IKRig	Describes an IK created by Duik (layers needed, type, goal, controller...)
PropertyDescription	Describes any property (useful to retrieve a property if the selection changes in the effects)
Controller	A controller created by Duik

KeySpatialProperties object attributes

Describes all spatial properties of a KeyFrame.

KeySpatialProperties.inTangent
KeySpatialProperties.outTangent
KeySpatialProperties.continuous
KeySpatialProperties.autoBezier
KeySpatialProperties.roving

Name	Type	Description
inTangent	float or Array of float	In spatial tangent of the keyframe
outTangent	float or Array of float	Out spatial tangent of the keyframe
continuous	boolean	Spatial interpolation set to continuous
autoBezier	boolean	Spatial interpolation set to auto Bezier
roving	boolean	Keyframe set to roving

KeyFrame object attributes

Represents an animation keyframe of After Effects

See *Duik.utils.getKey()* and *Duik.utils.addKey()*

KeyFrame.time
KeyFrame.value
KeyFrame.inInterpolationType
KeyFrame.outInterpolationType
KeyFrame.spatial
KeyFrame.spatialProperties
KeyFrame.inEase
KeyFrame.outEase
KeyFrame.continuous
KeyFrame.autoBezier

Name	Type	Description
time	float	Time of the keyframe in the comp
value	Any AFX propertyValueType	Value of the keyframe
inInterpolationType	Enumerated value; one of: KeyframeInterpolationType.LINEAR KeyframeInterpolationType.BEZIER KeyframeInterpolationType.HOLD	In interpolation type of the keyframe
outInterpolationType	Enumerated value; one of: KeyframeInterpolationType.LINEAR KeyframeInterpolationType.BEZIER KeyframeInterpolationType.HOLD	Out interpolation type of the keyframe
spatial	boolean	True if the keyframe is on a spatial property, one of: PropertyValueType.ThreeD_SPATIAL PropertyValueType.TwoD_SPATIAL
spatialProperties	KeySpatialProperties	All spatial properties of the keyframe. See *KeySpatialProperties object attributes*
inEase	Array of AFX KeyframeEase objects	Incoming temporal ease of the keyframe
outEase	Array of AFX KeyframeEase objects	Outgoing temporal ease of the keyframe
continuous	boolean	Temporal interpolation set to continuous

PropertyAnim object attributes

Describes the keyframe animation of a given property
See *Duik.utils.getPropertyAnim()* and See *Duik.utils.setPropertyAnim()*

PropertyAnim.name

PropertyAnim.keys
PropertyAnim.startValue

Name	Type	Description
name	string	Name of the animated Property
keys	Array of KeyFrames	Keyframes of the animation, see *KeyFrame object attributes*
startValue	Any AFX propertyValueType	First value of the animation. If there's no keyframe PropertyAnim.keys.length == 0, the value of the property.

MaskAnim object attributes

Describes all the keyframe animations of the properties of a given Mask
See *Duik.utils.getPropertyAnims()*

MaskAnim.name
MaskAnim.anims

Name	Type	Description
name	string	Name of the animated Mask
anims	Array of PropertyAnim	Animations of the properties of the mask, see *PropertyAnim object attributes*

EffectAnim object attributes

Describes all the keyframe animations of the properties of a given Effect
See Duik.utils.getPropertyAnims()

EffectAnim.name
EffectAnim.matchName
EffectAnim.anims

Name	Type	Description
name	string	Name of the animated Effect
matchName	string	matchName of the animated Effect
anims	Array of PropertyAnim	Animations of the properties of the effect, see *PropertyAnim object attributes*

LayerAnim object attributes

Describes all the keyframe animations of the transformation, masks, and effects of a layer
See *Duik.copyAnim()* and *Duik.pasteAnim()*

LayerAnim.name
LayerAnim.index
LayerAnim.transformAnims
LayerAnim.effectsAnims
LayerAnim.masksAnims

Name	Type	Description
name	string	Name of the animated layer
index	string	Index of the animated layer
transformAnims	Array of PropertyAnim	Animations of the transformations, see *PropertyAnim object attributes*
effectsAnims	Array of EffectAnim	Animations of the effects, see *EffectAnim object attributes*
masksAnims	Array of MaskAnim	Animations of the masks, see Mask*Anim object attributes*

IKRig object attributes

Describe an IK created by Duik.

IKRig.type
IKRig.layer1
IKRig.layer2
IKRig.layer3
IKRig.goal
IKRig.controller

Name	Type	Description
type	int	Type of the IK, either 1, 2, or 3. 0 if the IK is not valid.
layer1	AVLayer	First layer of the IK (the root, the top parent)
layer2	AVLayer or null	The second layer of the IK, if type is 2 or 3, or null if type is 1.
layer3	AVLayer or null	The third layer of the IK, if type is 3, or null if type is 1 or 2.
goal	AVLayer or null	A goal layer attached to the IK, or null
controller	AVLayer	The controller layer of the IK
threeD	boolean	true if this is a 3D IK (used for type 2 only)
frontFacing	boolean	true if the 3D layers face the front/back views, false if they face the right/left views.
clockWise	boolean	true if the IK bends clockwise. Used with type 2 and 3 only.

created	boolean	true if the IK has already been successfully created and exists in the comp.

IKRig object methods

IKRig.create()

Name	Description	Return
create()	Creates the rig in the comp	AVLayer, the zero created (if any) or null

IKRig.create()

Creates the IK Rig in the comp. Sets the created attribute to true if sucessful.

returns

AVLayer, the zero created (if any) or null.

PropertyDescription object attributes

Describes any property (useful to retrieve a property if the selection changes in the effects)

PropertyDescription.isEffect
PropertyDescription.index
PropertyDescription.depth
PropertyDescription.parentName
PropertyDescription.dimensions
PropertyDescription.canSetExpression

Name	Type	Description
isEffect	boolean	Property.parentProperty.isEffect
index	integer	Property.propertyIndex
depth	integer	Property.propertyDepth
parentName	string	Property.parentProperty.name
dimensions	integer	1, 2 or 3
canSetExpression	boolean	Property.canSetExpression

Controller object attributes

A controller created by Duik

Controller.locked
Controller.xPosition
Controller.yPosition
Controller.rotation
Controller.scale

Controller.arc
Controller.eye
Controller.layer
Controller.size
Controller.type
Controller.color

Name	Type	Description
locked	boolean	If true, transformation properties not controlled by the controller are locked with a simple expression, to prevent inadvertantly changing them
xPosition	boolean	If true, the X Position of the controller may be animated
yPosition	boolean	If true, the Y Position of the controller may be animated
rotation	boolean	If true, the Rotation of the controller may be animated
scale	boolean	If true, the Scale of the controller may be animated
arc	boolean	If true, the Rotation of the controller may be animated. The controller is displayed differently than with Controller.rotation, because its anchor point may be moved.
eye	boolean	If true, the Position of the controller may be animated. The icon is an eye.
layer	ShapeLayer	The controller layer
size	float	The size of the controller (in % if type is *VECTOR*, pixels if type is *NULL*) Set to 0 to use Duik.settings.controllerSize
type	integer	Enumerated value, one of: Duik.layerTypes.NULL Duik.layerTypes.VECTOR
color	Array of floats [R,V,B,A]	The color of the controller

Controller object methods

Controller.lock()
Controller.unlock()
Controller.update()

Name	Description	Return
lock()	Locks the transformation properties not controlled by the controller, to prevent inadvertantly changing them	void

unlock()	Unlocks the previously locked transformation properties. Note that before parenting a controller, it should be unlocked.	void
update()	Updates the shape of the controller, if its properties have changed	void

Duik

Duik Attributes

string *Duik.version*
float *Duik.versionNumber*
boolean *Duik.forceReload*
boolean *Duik.usePresets*
string *Duik.presetPath*
float *Duik.presetEffectsInstalledVersion*

Name	Type	Description
version	string, read-only	Version string of libDuik
versionNumber	float, read-only	Version number of libDuik
usePresets	*boolean*	true to use presets instead of pseudo effects.
presetPath	*string*	Path where presets are located; By default, the path of *libDuik.jsxinc* itself.
presetEffectsInstalledVersion	float, read-only	Version number of installed pseudo effects. Should be the same of *Duik.versionNumber*
copiedAnim	Array of LayerAnim	The layer animations copied with *Duik.copyAnim()* method.

Duik Objects

Duik.uiString
Duik.settings
Duik.utils
Duik.setup

Name	Description
uiStrings	Contains all string names used by effects created by Duik. You can set these strings to translate libDuik at runtime. Default values are English names.
settings	Access to settings used by Duik.
utils	Some useful tools
setup	Methods and attributes to correctly install libDuik & pseudo effects.

Duik Methods

//TODO tri par ordre alphabétique

Low-level methods are listed below (greyed) but they are not documented.
If you do not understand what low-level methods do by reading them in *libDuik.jsxinc*, you shouldn't need them.

Duik.autoIK(layers, clockWise, frontFacing)
Duik.goal(layer, controller)
Duik.addController(layer,color,rotation,xPosition,yPosition,scale,arc)
Duik.addControllers(layers,color,rotation,xPosition,yPosition,scale,arc)
Duik.oneLayerIK(controller,layer)
Duik.twoLayerIK(controller,root,end,clockWise,frontFacing)
Duik.threeLayerIK(controller,root,middle,end,clockWise)
Duik.wiggle(layer,property,separateDimensions)
Duik.threeDWiggle(layer,property,)
Duik.twoDWiggle(layer,property)
Duik.oneDWiggle(layer,property)
Duik.adaptativeExposure(layer,property,precision,minExp,maxExp)
Duik.fixedExposure(layer,property)
Duik.addBones(layers)
Duik.addZero(layer)
Duik.addZeros(layers)
Duik.rotationMorph(layer,prop)
Duik.swing(layer,prop)
Duik.wheel(layer,radius,curved)
Duik.morpher(layers)
Duik.lensFlare(layers)
Duik.distanceLink(layer, property, parentLayer)
Duik.spring(property, layer, simulated)
Duik.copyAnim(layers, selectedKeysOnly, startTime, endTime)
Duik.pasteAnim(layers, layerAnims, startTime, getLayerMethod)
Duik.rigPaint(layers)
Duik.blink(layer,prop)
Duik.lockProperty(layer, prop)
Duik.scaleZLink(layers)
Duik.timeRemap(layers)
Duik.onionSkin(layer,activate,duration)
Duik.importRigInComp(comp,rigComp,rigName, progressBar, progressText,containingWindow)

Name	Description	Return
autoIK(layers, clockWise, frontFacing)	Adds IK on the layers	true if successful, false if anything went wrong
goal(layer, controller)	Adds a goal effect to the layer, which may be controlled by a controller	true if successful, false if anything went wrong
addController(layer, color, autoLock, rotation, xPosition,	Creates a null object (controller) at layer position and named by	Controller object

yPosition, scale, arc)	layer.name	
addControllers(layers, color, autoLock, rotation, xPosition, yPosition, scale, arc)	For each layer, Creates a null object (controller) at layer position and named by layer.name	Array of Controller objects
wiggle(layer, property, separateDimensions)	Adds a wiggle effect to given property	true if successful, false if anything went wrong
adaptativeExposure(layers, precision, minExp, maxExp, sync, layerSync)	Adds exposure controls to the animation of the property.	true if successful, false if anything went wrong
fixedExposure(layer, prop)	Adds exposure controls to the animation of the property.	true if successful, false if anything went wrong
addBones(layers)	Adds bones to the layers	Array of AVLayer; bones
addZero(layer)	Adds zero to the layer	AVLayer; zero
addZeros(layers)	Adds zeros to the layers	Array of AVLayer; zeros
rotationMorph(layer, prop)	Creates a rotation morph on the given property	true if successful, false if anything went wrong
swing(layer,prop)	Creates a swing on the given property	true if successful, false if anything went wrong
wheel(layer, radius, curved)	Automates the rotation of the given layer using its position	true if successful, false if anything went wrong
morpher(layers)	Adds a slider to easily control interpolations of selected properties of the given layers.	true if successful, false if anything went wrong
lensFlare(layers)	Rigs the layers to move like a lens flare.	true if successful, false if anything went wrong
distanceLink(layer, property, parentLayer)	Links the property to the distance of parentLayer	true if successful, false if anything went wrong
spring(property, layer, simulated)	Adds a spring effect on the properties	true if successful, false if anything went wrong
copyAnim(layers, selectedKeysOnly, startTime, endTime)	Copies the animation of the layers	Array of LayerAnim
pasteAnim(layers, layerAnims, startTime, getLayerMethod)	Pastes the animations on the layers	int, the number of the layers on whichh an animtion was pasted
rigPaint(layers)	Rigs the paint effects to be able to animate all strokes as if there was only one.	Void
blink(layer, prop)	Adds a blink effect to the property.	true if successful, false if anything went wrong
lockProperty(layer, prop)	Locks the property with a simple	void

	expression.	
scaleZLink(layers)	Links the distance of the layer from the camera to its scale so its apparent size won't change.	void
timeRemap(layers)	Activates the time remapping of the layers, extending them to the length of the comp and adjusting the last keyframe.	Void
onionSkin(layer, activate, duration)	Activates or deactivates an onion skin on the paint effects of the layer.	void
importRigInComp(comp,rigComp, rigName, progressBar, progressText,containingWindow)	Imports a rig in the current comp (taking care of duplicates, expressions, controllers and adding a Master Controller to move, scale & flip the rig.	void

Duik.autoIK(layers, clockWise, frontFacing)

Adds IK on the layers. Duik will attempt to autodetect each layer role, using *Duik.utils.prepIK()*. If it can't (wrong parenting, wrong placement...) it will use the order of the layers in the Array or LayerCollection: first the layers, from end to root (from child to parent), last the controller.

parameters:

layers | Array of AVLayers or LayerCollection
clockWise | boolean, used only with two-layer and three-layer IK, default: false
frontFacing | boolean, default: false

returns

IKRig object created

Duik.goal(layer, controller)

Adds a goal effect to the layer, which may be controlled by a controller

parameters:

layer | AVLayer
controller | AVLayer or undefined

returns

true if successful, false if anything went wrong

Duik.addController(layer, color, autoLock, rotation, xPosition, yPosition, scale, arc)

Creates a null object (controller) at layer position and named by layer.name

If *Duik.settings.controllerType* is *Duik.layerTypes.VECTOR*, the parameters are used to draw a nice icon instead of using a null object.

If autoLock is true, the transformations which should not be changed are locked with a simple expression.

See *Controller object*.

parameters

layer | AVLayer
color | Array of 4 floats : [R,V,B,A], default [1,1,1,1]
autoLock | boolean, default false
rotation | boolean, default true
xPosition | boolean, default true
yPosition | boolean, default true
scale | boolean, default false
arc | boolean, default false

returns

Controller object

Duik.addControllers(layers, color, autoLock, rotation, xPosition, yPosition, scale, arc)

This is a convenience method, which runs Duik.addController() on each layer of the given array of layers.

parameters

layers | Array of AVLayer or LayerCollection
color | Array of 4 floats : [R,V,B,A], default [1,1,1,1]
autoLock | boolean, default false
rotation | boolean, default true
xPosition | boolean, default true
yPosition | boolean, default true
scale | boolean, default false
arc | boolean, default false

returns

Array of Controller objects

Duik.wiggle(layer, property, separateDimensions)

Adds a wiggle effect to given property.

parameters

layer | AVLayer of the property
property | Property
separateDimensions | boolean, false to apply the same wiggle to all dimensions, default: false

returns

true if successful, false if anything went wrong

Duik.fixedExposure(layer,prop)

Adds exposure controls to the animation of the property.

parameters

layer | AVLayer
prop | Property

returns

true if successful, false if anything went wrong

Duik.adaptativeExposure(layers,precision,minExp,maxExp,sync,layerSync)

Adds exposure controls to the animation of the property. The exposure adapts automatically to the speed, according to the given precision, of the properties between a minimum and a maximum exposure (in frames).

parameters

layers | Array of AVLayer or LayerCollection
precision | integer, default: 100
minExp | integer, default: 1
maxExp | integer, default: 4
sync | boolean, wether to sync all properties, default: true
layerSync | boolean, wether to sync all layers, if sync == true, default: false

returns

true if successful, false if anything went wrong

Duik.addBones(layers)

Adds bones to the layers, only on selected pins if any, or else on all puppet pins found on those layers.

parameters

layers | Array of AVLayers

returns

Array of AVLayers, the bones created

Duik.addZero(layer)

Adds a null object for the layer, at the same place and orientation, and then parents the layer to it, parenting the null object (the zero) to the former parent of the layer.

parameters

layers | Array of AVLayers

returns

Array of AVLayers, the zeros created

Duik.addZeros(layers)

This is a convenience method, which runs Duik.addZero() on each layer of the given array of layers.

parameters

layers | Array of AVLayers or LayerCollection

returns

Array of AVLayers, the zeros created

Duik.rotationMorph(layer,prop)

Creates a rotation morph on the given property.

Parameters

layer | AVLayer
prop | Property

returns

true if successful, false if anything went wrong

Duik.swing(layer,prop)

Creates a swing on the given property

parameters

layer | AVLayer
prop | Property

returns

true if successful, false if anything went wrong

Duik.wheel(layer, radius, curved)

Automates the rotation of the given layer using its position.
If curved, works even if the trajectory is not horizontal, but is heavier to compute.

parameters

layer | AVLayer
radius | float, default 100.0
curved | boolean, default false

returns

true if successful, false if anything went wrong

Duik.morpher(layers)

Adds a "morpher", a slider to easily control interpolations of selected properties of the given layers.

parameters

layers | Array of AVLayer

returns

true if successful, false if anything went wrong

Duik.lensFlare(layers);

Rigs the layers to move like a lens flare. The first layer in the selection is the controller, with sliders for intensity and size; the other layers have a distance property to adjust their position along the lens flare.

parameters

layers | Array of AVLayer

returns

true if successful, false if anything went wrong

Duik.distanceLink(layer,property,parentLayer);

Links the property to the distance of parentLayer

parameters

layer | AVLayer containing the property
property | Property to rig
parentLayer | AVLayer which distance from layer is used to rig

returns

true if successful, false if anything went wrong

Duik.spring(property, layer, simulated);

Adds a spring effect on the property

parameters

property | Property
layer | AVLayer containing the property
simulated | if true, applies the simulated version of the spring, default: false

returns

true if successful, false if anything went wrong

Duik.copyAnim(layers, selectedKeysOnly, startTime, endTime)

Copies all the animations as *LayerAnim objects* (except expressions) on selected layers, and store them in the Array Duik.copiedAnim.
If selectedKeysOnly is true, copies only the selected keyframes, otherwise all the masks, effcts, and transformation properties will be copied, even if they are not animated (in this case, the value will be stored in the PropertyAnim.startValue). If you do not want to keep the properties without animation, you will have to loop through the arrays of PropertyAnim and check if PropertyAnim.keys.length > 0 to remove empty animations from the Arrays.
See *LayerAnim object*

parameters

layers | Array or Collection of AVLayers
selectedKeysOnly | boolean, true to copy only selected keys, default: false
startTime | float, default: start of the comp
endTime | float, default: end of the comp

returns

Array of LayerAnim

Duik.pasteAnim(layers, layerAnims, startTime, getLayerMethod)

Pastes all the animations in the Array of LayerAnim on layers, using layer names or layer indexes, beginning at startTime
See *LayerAnim object*

parameters

layers | Layers where to paste the animation
layerAnims | Array of LayerAnim, default: Duik.copiedAnim
startTime | float, default: comp.time
getLayerMethod | one of Duik.getLayers.NAME, Duik.getLayers.INDEX, Duik.getLayers.SELECTION_INDEX, default: Duik.settings.getLayerMethod

returns

integer, number of layers on which animations were pasted

Duik.rigPaint(layers)

Rigs the paint effects to be able to animate all strokes as if there was only one.

parameters

layers | Array of AVLayers or LayerCollection

returns

void

Duik.blink(layer, prop)

Adds a blink effect to the property.

parameters

layer | AVLayer
prop | Property

returns

true if successful, false if anything went wrong

Duik.lockProperty(layer, prop)

Locks the property with a simple expression.

parameters

layer | AVLayer
prop | Property

returns

void

Duik.scaleZLink (layers)

Links the distance of the layer from the camera to its scale so its apparent size won't change. If multiple cameras, include the camera used in the array.

parameters

layers | Array of Layer or LayerCollection

returns

void

Duik.timeRemap(layers)

Activates the time remapping of the layers, extending them to the length of the comp and adjusting the last keyframe.

parameters

layers | Array of Layer or LayerCollection
loopType | String, "in" or "out" or "none", default: "none"

returns

void

Duik.onionSkin(layers)

Activates or deactivates an onion skin on paint effects on the layer.

parameters

layer | AVLayer
activate | boolean, default: true
duration | integer, onion skin duration in frames, default: 5

returns

void

Duik.importRigInComp(comp, rigComp, rigName, progressBar, progressText, containingWindow)

Imports a rig in the comp, transferring and linking the controllers in the new comp, while keeping the rig precomposed.
The rig comp is duplicated, including precomps, renamed, and expressions are updated, so that one can import the same rig several times.
A Master Controller is created to move, scale and flip the imported rig.

All controllers created by Duik, and any layer which name begins with “C_” is considered a controller. The controllers should not be parented to any of the other layers, but they can be parented to other controllers and have zeros.
Any controller without zero will have one automatically added, this is needed to link them from the composition with the rig to the one where it's imported.

You can provide some ScriptUI Controls to display the progression of the importation, which can take some time. The method takes care of showing, updating and hiding the provided window.

parameters
comp | CompItem, the comp where to import the rig

rigComp | CompItem, the comp containing the rig
rigName | the name of this instance of the rig, must be unique in the project
progressBar | scriptUI ProgressBar, optionnal, to display the progression
progressText | Any scriptUI Control with a text property, needed if progressBar is provided
containingWindow | scriptUI Window containing the provided progressBar and progressText

returns
void

Duik.setup

Methods and attributes to correctly install libDuik & pseudo effects.

Duik.setup Attributes

Duik.setup.presetEffects

Name	Type	Description
presetEffects	string	The XML (as string object) to insert just before *</effects>* in After Effects presetEffects.xml to correctly install libDuik pseudo effects. This includes the version of of libDuik as an XML comment, which can be checked by *Duik.setup.checkPresetEffectsVersion* to ensure libDuik has been correcly installed.

Duik.setup Methods

Duik.setup.installPseudoEffects()
Duik.setup.checkPresetEffectsVersion()

Name	Description	Return
installPseudoEffects()	Automatically install pseudo effects in After Effects *presetEffects.xml*	void
checkPresetEffectsVersion()	Checks the version of installed libDuik pseudo effects, stored in *Duik.presetEffectsInstalledVersion*	void

Duik.setup.installPseudoEffects()

Tries to Automatically install pseudo effects in After Effects *presetEffects.xml.*
The installation can be checked with *Duik.checkPresetEffectsVersion()*, en then comparing *Duik.presetEffectsInstalledVersion* with *Duik.versionNumber.*
Example:

```
//install
Duik.installPseudoEffects();

//check
Duik.checkPresetEffectsVersion();

if (Duik.presetEffectsInstalledVersion !=  Duik.versionNumber) {
        //do something
} else {
        //continue loading your script
}
```

parameters:

none

returns

void

Duik.setup.checkPresetEffectsVersion()

Checks the version of installed libDuik pseudo effects, stored in *Duik.presetEffectsInstalledVersion*.
See ***Duik.setup.installPseudoEffects()*** for an example.

parameters:

none

returns

void

Duik.uiStrings

Contains all string names used by effects created by Duik.
You can set these strings to translate libDuik at runtime.
Default values are English names.

Duik.uiStrings Attributes

Duik.uiStrings.ik
Duik.uiStrings.wiggle
Duik.uiStrings.exposure
Duik.uiStrings.rotMorph
Duik.uiStrings.swing
Duik.uiStrings.wheel
Duik.uiStrings.lensFlare
Duik.uiStrings.distanceLink
Duik.uiStrings.spring
Duik.uiStrings.paintRig
Duik.uiStrings.flip

Name	Type	Description
ik	string	*"IK"*
wiggle	string	*"Wiggle"*
exposure	string	*"Exposure"*
rotMorph	string	*"Rotation Morph"*
swing	string	*"Swing"*
wheel	string	*"Wheel"*
lensFlare	string	*"Lens Flare"*
distanceLink	string	*"Distance Link"*
spring	string	*"Spring"*
paintRig	string	*"Paint Rig"*
flip	string	*“Flip”*

Duik.settings

Access to settings used by Duik.

Duik.settings Attributes

These attributes define some settings and preferences needed by Duik.
If you set them, they can be saved to be reloaded even if After Effects is shutdown, using *Duik.settings.save()*. If this method is not called, the settings will be set back to previous values if After Effects is shut down.
Saved settings must be loaded at runtime calling *Duik.settings.load()*.
Default values can be restored using *Duik.settings.restoreDefaults()*.

Duik.settings.controllerSize
Duik.settings.controllerType
Duik.settings.controllerSizeAuto
Duik.settings.controllerSizeHint
Duik.settings.boneType
Duik.settings.boneSize
Duik.settings.boneSizeAuto
Duik.settings.boneSizeHint
Duik.settings.boneColor
Duik.settings.morpherCreatesKeyframes
Duik.settings.getLayersMethod

Name	Type	Description	Default
controllerSize	integer	Size of controllers in pixels	100
controllerType	integer	Enumerated value, one of: Duik.layerTypes.NULL Duik.layerTypes.VECTOR	Duik.layerTypes.VECTOR
controllerSizeAuto	boolean	If true, controller sizes will be automatically adapted to comp size, according to *Duik.settings.controllerSizeHint*	true
controllerSizeHint	integer	Enumerated value, one of: Duik.sizes.SMALL Duik.sizes.MEDIUM Duik.sizes.BIG	Duik.sizes.MEDIUM
boneType	integer	Enumerated value, one of: Duik.layerTypes.NULL Duik.layerTypes.SOLID	Duik.layerTypes.SOLID
boneSize	integer	Size of bones in pixels	20
boneSizeAuto	boolean	If true, bone sizes will be automatically adapted to comp size, according to *Duik.settings.boneSizeHint*	true

boneSizeHint	integer	Enumerated value, one of: Duik.sizes.SMALL Duik.sizes.MEDIUM Duik.sizes.BIG	Duik.sizes.MEDIUM
boneColor	string	Hex value of the color of the bones, excluding the leading « # »	« FF0000 »
morpherCreatesKeyframes	boolean	If true, morpher will automatically create keyframes for each keyframe of the controlled properties	True
getLayersMethod	boolean	Enumerated value, one of: Duik.getLayers.NAME Duik.getLayers.INDEX Duik.getLayers.SELECTION_INDEX	Duik.getLayers.NAME

Duik.settings Methods

Duik.settings.save()
Duik.settings.load()
Duik.settings.restoreDefaults()

Name	Description	Return
save()	Saves Duik settings into After Effects preferences	void
load()	Loads Duik settings from After Effects preferences	void
restoreDefaults()	Restore default values to Duik settings	void

Duik.settings.save()

Saves Duik settings attributes into After Effects preferences (using app.settings.saveSetting())

Those settings can be loaded when the script runs using *Duik.settings.load()*. This allows to easily restore the settings set by the user even if After Effects is shut down.

parameters:

none

returns

void

Duik.settings.load()

Loads Duik settings attributes from After Effects preferences (using app.settings.getSetting())

This allows to easily restore the settings set by the user even if After Effects is shut down.

If this method is not called at runtime, default values will be loaded at first run.

parameters:

none

returns

void

Duik.settings.restoreDefaults()

Restore default values to Duik settings. These values will not be saved until Duik.settings.save() is called.

parameters:

none

returns

void

Duik.utils

Some useful methods.

Duik.utils Methods

Duik.utils.prepareProperty(property,isFX,index,depth,parentName)
Duik.utils.getPropertyDimensions(property)
Duik.utils.getLength(value1,value2)
Duik.utils.getAverageSpeed(layer,property)
Duik.utils.addPseudoEffect(layer,pseudoEffectName)
Duik.utils.getPuppetPins(effects)
Duik.utils.getDistance(layer1,layer2)
Duik.utils.rigProperty(layer,prop,pseudoEffect)
Duik.utils.deselectLayers()
Duik.utils.checkNames(comp)
Duik.utils.getItem(items, itemIndex)
Duik.utils.getKey(prop, keyIndex)
Duik.utils.getPropertyAnims(prop, selectedKeysOnly, startTime, endTime)
Duik.utils.getPropertyAnim(prop, selectedKeysOnly, startTime, endTime)
Duik.utils.setPropertyAnim(prop, propAnim, startTime)
Duik.utils.addKey(prop,key, startTime)
Duik.utils.getFirstKeyTime(prop)
Duik.utils.hasSelectedKeys(prop)
Duik.utils.convertCollectionToArray(collection)
Duik.utils.prepIK(layers)
Duik.utils.getControllers(layers)
Duik.utils.getAverageSpeeds(layers)
Duik.utils.replaceInExpressions(prop,oldString,newString)
Duik.utils.replaceInLayersExpressions(layers, oldString, newString)
Duik.utils.renameLayer(layer, newName, updateExpressions)
Duik.utils.renameItem(item, newName, updateExpressions)

Name	Description	Return
prepareProperty(property, isFX, index, depth, parentName)	Prepares property to be rigged	true if property can set expression, false otherwise
getPropertyDimensions(property)	Gets the dimensions of the property (1, 2 or 3), taking care of 2D layer positions (reported as 3D by AFX, but to be considdered as 2D)	integer, number of dimensions
getLength(value1, value2)	Gets the length between the values, whichever dimensions they are	float, length between the values
getAverageSpeed(layer, property)	Gets the average speed of the animated property, between its first and last keyframe only	float, average speed of the property
addPseudoEffect(layer, pseudoEffectName)	Adds a Duik predefined pseudo	Property, the effect

	effect to the layer	added
getDistance(layer1,layer2)	Measure distance between two layers	integer, distance between layers, in pixels
getPuppetPins(effects)	Gets all puppet pins from a layer effects	Array of Properties, all puppet pins found
rigProperty(layer, prop, pseudoEffect)	Performs some checks on the property and adds a pseudo effect on the layer	Property, the effect added
deselectLayers()	Deselects all layers	Void
checkNames(comp)	Checks for duplicate names among the layers of the comp, renaming them if found.	true if any layer was renamed
getItem(items, itemIndex)	Gets the item as if it were in a 0-based indexed Array, even if it is in a 1-based indexed Collection	Object, the item
getKey(prop, keyIndex)	Gets the keyframe at keyIndex on the property	KeyFrame object
getPropertyAnims(prop, selectedKeysOnly, startTime, endTime)	Gets the keyframe animations on the child properties of the prop, if it's a PropertyGroup (recursive), or the animation of the prop if it's a Property	Array of PropertyAnim objects
getPropertyAnim(prop, selectedKeysOnly, startTime, endTime)	Gets the keyframe animation of the Property	PropertyAnim object
setPropertyAnim(prop, propAnim, startTime)	Sets the animation on the property	boolean, true if succeeded
addKey(prop,key, startTime)	Adds a keyframe on the property	void
getFirstKeyTime(prop)	Gets the time of the first key on the property	float, time of the keyframe
hasSelectedKeys(prop)	Checks if the properties has keyframes which are selected	Boolean
convertCollectionToArray(collection)	Converts the given Collection to an array. If the parameter is already an Array, returns a copy of it.	Array
prepIK(layers)	Creates an *IKRig* object, automatically detecting each layer usage.	IKRig object
getControllers(layers)	Gets the controllers created by Duik found in the Array or Collection	Array of Controller objects

getAverageSpeeds(layers)	Gets the average variation speed of the selected properties in the layers	float, average speed
replaceInLayersExpressions(layers, oldString, newString)	Replaces all occurences of oldString by newString in all the expressions of all the layers.	void
renameLayer(layer, newName, updateExpressions)	Renames the layer, updating expressions in all the compositions of the project	void
renameItem(item, newName, updateExpressions)	Renames the item, updating expressions in all the compositions of the project, if the item is a CompItem	void

Duik.utils.prepareProperty(property,isFX,index,depth,parentName)

Prepare the given property to be rigged.
isFX, *index*, *depth*, *parentName* will be filled by the method with the values corresponding to this property.

parameters:

property | Property
isFX | boolean
index | integer
depth | integer
parentName | string

returns

true if property can set expression, false otherwise

Duik.utils.getPropertyDimensions(property)

Gets the dimensions of the property (1, 2 or 3), taking care of 2D layer positions (reported as 3D by AFX, but to be considdered as 2D)

parameters:

property | Property

returns

integer, number of dimensions

Duik.utils.getLength(value1, value2)

Gets the length between the values, whichever dimensions they are

parameters:

value1 | float or Array of float, first coordinates
value1 | float or Array of float, second coordinates

returns

float, length between the values

Duik.utils.getAverageSpeed(layer, property)

Gets the average speed of the animated property, between its first and last keyframe only.

parameters:

layer | AVLayer of the property
property | Property

returns

float, average speed of the property

Duik.utils.addPseudoEffect(layer, pseudoEffectFileName)

Adds a Duik predefined pseudo effect to the layer. The AFX preset file of the pseudo effect must be located in the same folder as libDuik.jsxinc and called « Duik_ » + pseudoEffectName + « .ffx ».
In the preset, the effect must be called pseudoEffectName.

parameters:

layer | AVLayer
pseudoEffectFileName | string, name of the file of the pseudo effect

returns

Property, the effect added

Duik.utils.getPuppetPins(effects)

Recursive method to find all puppet pins on a given layer, even if there is more than one puppet effect. You must provide the effects PropertyGroup of the layer.
Example : var pins = Duik.utils.getPuppetPins(app.project.activeItem.layer(1)(« Effects »);

parameters:

effects | PropertyGroup, the effects group of a layer

returns

Array of Property, the puppet pins

Duik.utils.getDistance(layer1,layer2)

Measures distance between two layers, in pixels.

parameters:

layer1 | AVLayer
layer2 | AVLayer

returns

integer, distance in pixels

Duik.utils.rigProperty(layer, prop, pseudoEffect)

Performs some checks on the property and adds a pseudo effect on the layer.
The AFX preset file of the pseudo effect must be located in the same folder as libDuik.jsxinc and called « Duik_ » + pseudoEffectName + « .ffx ».
In the preset, the effect must be called pseudoEffectName.

parameters:

layer | AVLayer
prop | Property
pseudoEffect | file name of the pseudo effect

returns

PropertyGroup, the effect added

Duik.utils.deselectLayers()

Deselects all layers

returns

void

Duik.utils.checkNames(comp)

Checks for duplicate names among the layers of the comp, renaming them if found. This method is called everytime libDuik creates an effect which involves expressions and more than one layer, to avoid any bug with expressions linking to wrong layers.

parameters:

comp | CompItem where are the layers which must be checked. Default: app.project.activeItem

returns

true if any layer was renamed, false otherwise.

Duik.utils.getItem(items, itemIndex)

After effects sometimes uses its own Collection class, which is very similar to Arrays, but the first element of a Collection is at index 1 instead of 0 as in an Array.

This can make it difficult to write functions which will work both on Array or Collections.

Example:

```
function doSomethingOnLayers(layers) {
	for (i = 0 ; i < layers.length ; i++) {
		var layer = layers[i];
		//do something
	}
}

//will work correctly, as selectedLayers is an Array beginning at index 0
doSomethingOnLayers(app.project.activeItem.selectedLayers);

 //will not work, as layers is a LayerCollection beginning at index 1
doSomethingOnLayers(app.project.activeItem.layers);
```

This method makes it possible to get an item both for an Array or a Collection, without knowing which type is given.

```
function doSomethingOnLayers(layers) {
	for (i = 0 ; i < layers.length ; i++) {
		var layer = Duik.utils.getItem(layers,i);
		//do something
	}
}

//both will work correctly
doSomethingOnLayers(app.project.activeItem.selectedLayers);
doSomethingOnLayers(app.project.activeItem.layers);
```

parameters:

items | Array or Collection
itemIndex | int, index where the item must be found

returns

Object, the item at itemIndex in items.

Duik.utils.getKey(prop, keyIndex)

Gets the keyframe at keyIndex on the property
see *KeyFrame object*

parameters:

prop | Property
keyIndex | int

returns

KeyFrame object

Duik.utils.getPropertyAnims(prop, selectedKeysOnly, startTime, endTime)

Gets the keyframe animations on the child properties of the prop, if it's a PropertyGroup (recursive), or the animation of the prop if it's a Property, beginning at startTime and ending at endTime.
This is a recursive method.
see *PropertyAnim object*

parameters:

prop | PropertyBase
selectedKeysOnly | boolean
startTime | float
endTime | float

returns

Array of PropertyAnim objects

Duik.utils.getPropertyAnim(prop, selectedKeysOnly, startTime, endTime)

Gets the keyframe animation of the Property
This is not a recursive method (it won't check child properties); see *Duik.utils.getPropertyAnims()* for the recursive method.
see *PropertyAnim object*

parameters:

prop | Property
selectedKeysOnly | boolean
startTime | float
endTime | float

returns

PropertyAnim object

Duik.utils.setPropertyAnim(prop, propAnim, startTime)

Sets the animation on the property, beginning at startTime
see *PropertyAnim object*

parameters:

prop | PropertyBase
propAnim | PropertyAnim object
startTime | float

returns

boolean, true if succeeded.

Duik.utils.addKey(prop,key, startTime)

Adds a keyframe on the property. You can offset the time by setting startTime
see *KeyFrame object*

parameters:

prop | PropertyBase
key | KeyFrame object
startTime | float, default: 0

returns

void

Duik.utils.getFirstKeyTime(prop)

Gets the time of the first key on the property.

parameters:

prop | Property

returns

float

Duik.utils.hasSelectedKeys(prop)

Checks if the properties has keyframes which are selected.

parameters:

prop | Property

returns

boolean

Duik.utils.convertCollectionToArray(collection)

Converts the given Collection to an array. If the parameter is already an Array, returns a copy of it.

parameters:

collection | Collection or Array

returns

Array

Duik.utils.prepIK(layers)

Creates an *IKRig* object, automatically detecting each layer usage.
The detection checks the hierarchy of the layers to find each layer usage.
If the detection fails, the IKRig object is created using the order of the layers in the Array or LayerCollection: the first are the layers, beginning by the last child, the last one is the controller.
Goal layers are detected by measuring the distance between the last child of the chain and the controller: goal layers and controllers should be at the same place.
See *IKRig object*.

parameters:

layers | Array of AVLayers or LayerCollection

returns

IKRig object

Duik.utils.getControllers(layers)

Gets the controllers created by Duik found in the Array or LayerCollection. If the Array or the LayerCollection are empty, or if not provided, gets the controllers found in the active comp.
See *Controller object*.

parameters:

layers | Array of AVLayers or LayerCollection

returns

Array of Controller objects.

Duik.utils.getAverageSpeed(layers)

Gets the average variation speed of the selected properties in the layers.

parameters:

layers | Array of AVLayers or LayerCollection

returns

float, the average speed.

Duik.utils.replaceInLayersExpressions(layers, oldString, newString)

Replaces all occurences of oldString by newString in all the expressions of all the layers.

parameters

layers | Array of AVLayers or LayerCollection
oldString | string
newString | string

returns

void

Duik.utils.renameLayer(layer, newName, updateExpressions)

Renames the layer, updating expressions in all the compositions of the project.

parameters

layer | Layer
newName | string
updateExpressions | boolean, default: true

returns

void

Duik.utils.renameItem(item, newName, updateExpressions)

Renames the item, updating expressions in all the compositions of the project if the item is a CompItem

parameters

item | Item
newName | string
updateExpressions | boolean, default: true

returns

void