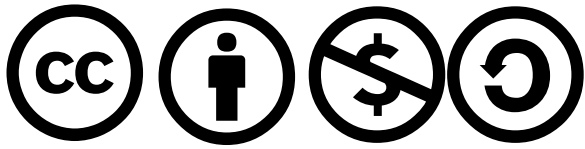


# Duik 15 Developer's guide

the complete manual of libDuik 15





**This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. To view a copy of this license, visit**

**<http://creativecommons.org/licenses/by-nc-sa/4.0/>**

**Author: Nicolas Dufresne [www.duduf.com](http://www.duduf.com)**

**Composition : Assia Chioukh**

## **Introduction**

**License**

**Including libDuik in your scripts**

**Installing libDuik**

**Using libDuik**

**Modifying libDuik**

## **Pseudo Effects List**

## **Objects**

**KeySpatialProperties Object**

**KeyFrame Object**

**PropertyAnim Object**

**MaskAnim Object**

**EffectAnim Object**

**LayerAnim Object**

**IKRig Object**

**PropertyDescription Object**

**Controller Object**

**TVPCamera**

**TVPCameraPoint**

**TVPProfileprof**

**TVPProfileprofPoint**

## **Duik**

**Enumerated Values**

**Attributes**

**Objects**

**Methods**

autoIK

goal

addController

addControllers

wiggle

adaptativeExposure

fixedExposure

addBones

addZeros

rotationMorph

swing

wheel

morpher

lensFlare

distanceLink

spring

copyAnim

pasteAnim

rigPaint

blink

lockProperty

scaleZLink

timeRemap

- onionSkin
- importRigInComp
- randomizeProperties
- randomizeStartTimes
- randomizeInPoints
- randomizeOutPoints
- moveAway
- groupPaint

## **Duik.setup**

### **Attributes**

### **Methods**

- installPseudoEffects
- checkPresetEffectsVersion

## **Duik.ui**

### **ScriptUI Objects**

### **Methods**

## **Duik.uiStrings**

### **Attributes**

## **Duik.settings**

### **Attributes**

### **Methods**

- save
- load
- restoreDefaults

## **Duik.bridge**

## **Duik.bridge.tvPaint**

### **Methods**

- parseCam
- loadCamFile

## **Duik.js**

### **Methods**

- escapeRegExp
- replaceAll
- random

## **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 you to easily include Duik functions into other scripts.

## License

Duik and libDuik are licensed under the GNU-General Public License version 3. This means they are free software, which offers four freedoms:

- the freedom to use the software for any purpose,
- the freedom to change the software to suit your needs,
- the freedom to share the software with your friends and neighbors, and
- the freedom to share the changes you make.



The complete source code along with a copy of the license of Duik and libDuik is available at:

<https://github.com/Duduf-dev/Duik/>

**This license does not allow you to use libDuik in a non-free or commercial software.** Any software using libDuik should be licensed under a free software license. See <http://www.fsf.org> for more information

## 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 several 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 beta 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.

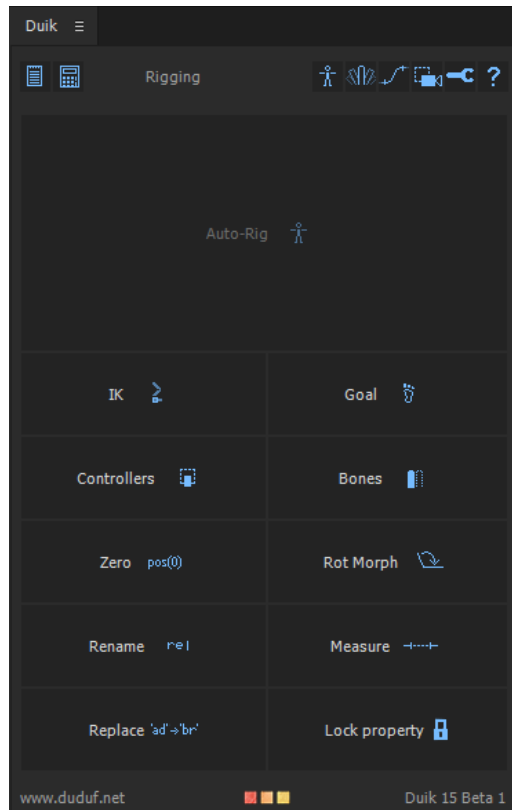
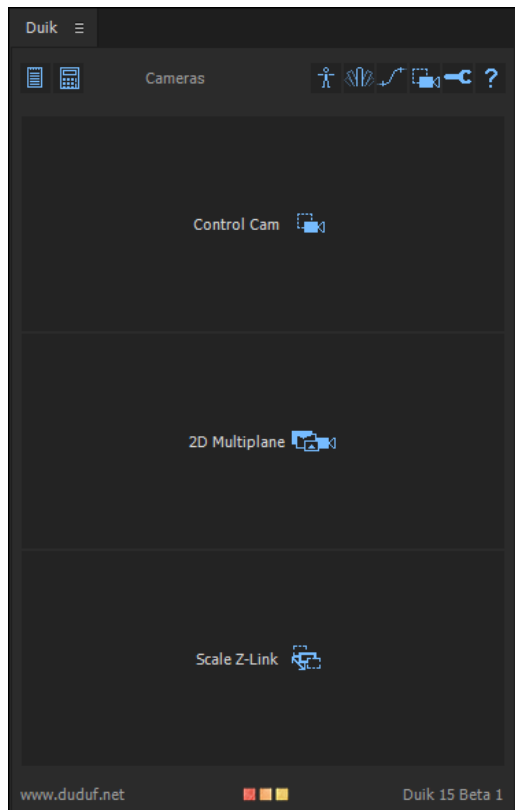
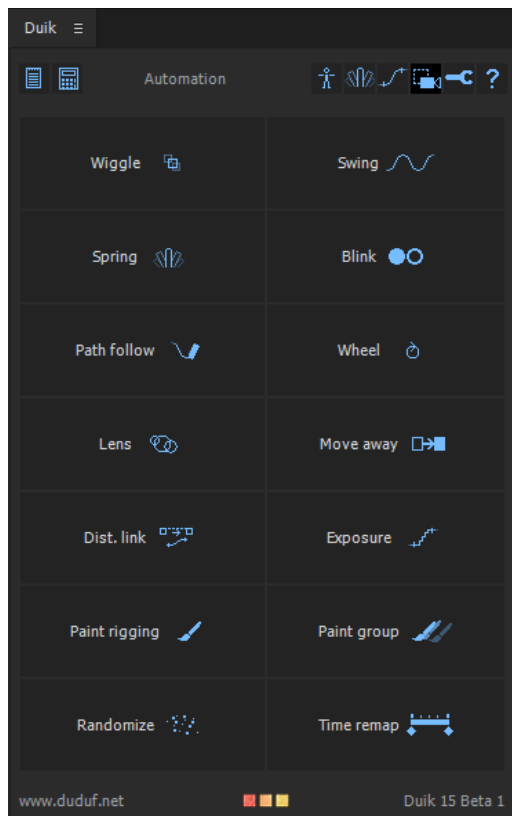
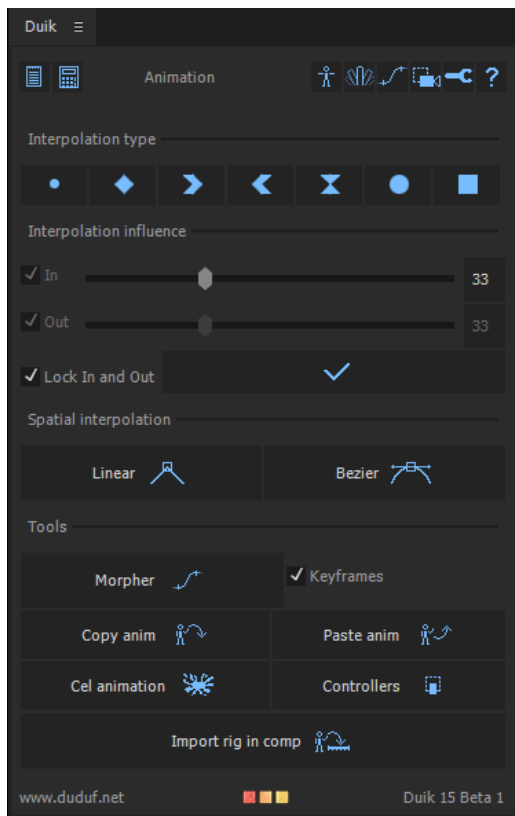
## Modifying libDuik

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*

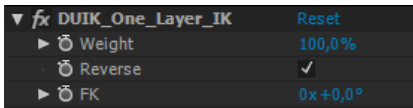
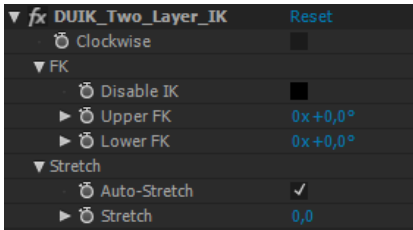


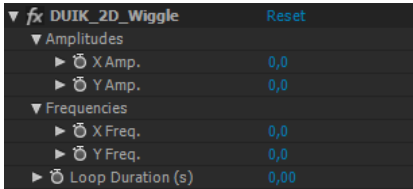
Here is a list of the effects available.

Those effects can be added on any layer with:


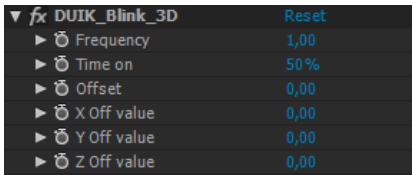
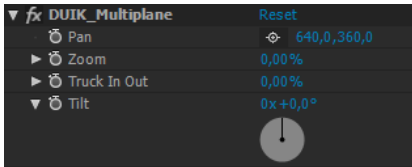

```
layer.effect.addProperty(matchName)
```

Example:

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

matchName	Description	Screenshot
<b><i>DUIK_One_Layer_IK</i></b>	Used by one layer IK	
<b><i>DUIK_Two_Layer_IK</i></b>	Used by two layer IK	
<b><i>DUIK_Three_Layer_IK</i></b>	Used by three layer IK	
<b><i>DUIK_3D_Wiggle</i></b>	Used for wiggle on 3D properties	
<b><i>DUIK_2D_Wiggle</i></b>	Used by wiggle on 2D properties	

<b><i>DUIK_1D_Wiggle</i></b>	Used by wiggle on 1D properties	<div> <div>fx DUIK_1D_Wiggle</div> <div>Reset</div> <div> ▶ <input type="checkbox"/> Amplitude 0,0 ▶ <input type="checkbox"/> Frequency 0,0 ▶ <input type="checkbox"/> Loop Duration (s) 0,00 </div> </div>
<b><i>DUIK_Exposure</i></b>	Used by exposure, in fixed mode	<div> <div>fx DUIK_Exposure</div> <div>Reset</div> <div> ▶ <input type="checkbox"/> Exposure (frames) 1,0 </div> </div>
<b><i>DUIK_RotMorph</i></b>	Used by Rotation Morph	<div> <div>fx DUIK_RotMorph</div> <div>Reset</div> <div> <div>Reference Layer 1. Dark Gray Solid 1</div> ▶ <input type="checkbox"/> Min. Angle 0x +0,0° ▶ <input type="checkbox"/> Max. Angle 0x +0,0° </div> </div>
<b><i>DUIK_Swing</i></b>	Used by Swing (oscillation)	<div> <div>fx DUIK_Swing</div> <div>Reset</div> <div> ▶ <input type="checkbox"/> Amplitude 0,0 ▶ <input type="checkbox"/> Frequency 0,0 ▶ <input type="checkbox"/> Offset 0,0 ▶ <input type="checkbox"/> Damping 0,0 </div> </div>
<b><i>DUIK_Wheel</i></b>	Used by Wheel	<div> <div>fx DUIK_Wheel</div> <div>Reset</div> <div> ▶ <input type="checkbox"/> Radius 100,0 <input type="checkbox"/> Reverse </div> </div>
<b><i>DUIK_LensFlare</i></b>	Used by Lens Flare on the layer of the center to control size and intensity	<div> <div>fx DUIK_LensFlare</div> <div>Reset</div> <div> ▶ <input type="checkbox"/> Intensity 100,0% ▶ <input type="checkbox"/> Scale 100,0% </div> </div>
<b><i>DUIK_LensFlareDistance</i></b>	Used by Lens Flare on flare layers to control their distance from the center	<div> <div>fx DUIK_LensFlareDistance</div> <div>Reset</div> <div> ▶ <input type="checkbox"/> Distance 0,0% </div> </div>
<b><i>DUIK_DistanceLink</i></b>	Used by Distance Link	<div> <div>fx DUIK_DistanceLink</div> <div>Reset</div> <div> <div>▼ Range</div> ▶ <input type="checkbox"/> Minimum Distance 0,0 ▶ <input type="checkbox"/> Maximum Distance 500,0 <input type="checkbox"/> Reverse </div> </div>
<b><i>DUIK_Spring</i></b>	Used by Spring on 2D and 3D properties	<div> <div>fx DUIK_Spring</div> <div>Reset</div> <div> ▶ <input type="checkbox"/> Elasticity 10,0 ▶ <input type="checkbox"/> Damping 5,0 </div> </div>
<b><i>DUIK_Spring_Bounce</i></b>	Used by spring on 1D properties, includes a checkbox called 'bounce'.	<div> <div>fx DUIK_Spring_Bounce</div> <div>Reset</div> <div> ▶ <input type="checkbox"/> Elasticity 10,0 ▶ <input type="checkbox"/> Damping 5,0 <input type="checkbox"/> Bounce </div> </div>
<b><i>DUIK_Paint_Rig</i></b>	Used by the paint rig tool to control the end, begin and diameter properties of the paint brushes	<div> <div>fx DUIK_Paint_Rig</div> <div>Reset</div> <div> ▶ <input type="checkbox"/> Start 0,00% ▶ <input type="checkbox"/> End 100,00% ▶ <input type="checkbox"/> Diameter 0 </div> </div>
<b><i>DUIK_Blink_1D</i></b>	Used by blink on 1D properties	<div> <div>fx DUIK_Blink_2D</div> <div>Reset</div> <div> ▶ <input type="checkbox"/> Frequency 1,00 ▶ <input type="checkbox"/> Time on 50% ▶ <input type="checkbox"/> Offset 0,00 ▶ <input type="checkbox"/> X Off value 0,00 ▶ <input type="checkbox"/> Y Off value 0,00 </div> </div>

<b><i>DUIK_Blink_2D</i></b>	Used by blink on 2D properties	
<b><i>DUIK_Blink_3D</i></b>	Used by blink on 3D properties	
<b><i>DUIK_Multiplane</i></b>	Used by 2D Multiplane cam	
<b><i>DUIK_Paint_Group</i></b>	Used by Paint groups	

## 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
<b><i>KeySpatialProperties</i></b>	Describes all spatial properties of a KeyFrame.
<b><i>KeyFrame</i></b>	Represents an animation keyframe of After Effects
<b><i>PropertyAnim</i></b>	Describes the keyframe animation of a given property
<b><i>MaskAnim</i></b>	Describes all the keyframe animations of the properties of a given Mask
<b><i>EffectAnim</i></b>	Describes all the keyframe animations of the properties of a given Effect
<b><i>LayerAnim</i></b>	Describes all the keyframe animations of the transformation, masks, and effects of a layer
<b><i>IKRig</i></b>	Describes an IK created by Duik (layers needed, type, goal, controller...)
<b><i>PropertyDescription</i></b>	Describes any property (useful to retrieve a property if the selection changes in the effects)
<b><i>Controller</i></b>	A controller created by Duik
<b><i>TVPCamera</i></b>	A camera imported from TVPaint
<b><i>TVPCameraPoint</i></b>	A spatial keyframe of a camera from TVPaint
<b><i>TVPProfileprof</i></b>	Temporal interpolation from TVPaint
<b><i>TVPProfileprofPoint</i></b>	A temporal keyframe from TVPaint

### KeySpatialProperties object attributes

Describes all spatial properties of a KeyFrame.

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

Name	Type	Description
<b><i>inTangent</i></b>	float or Array of float	In spatial tangent of the keyframe
<b><i>outTangent</i></b>	float or Array of float	Out spatial tangent of the keyframe
<b><i>continuous</i></b>	boolean	Spatial interpolation set to continuous
<b><i>autoBezier</i></b>	boolean	Spatial interpolation set to auto Bezier

<b>roving</b>	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
<b><i>time</i></b>	float	Time of the keyframe in the comp
<b><i>value</i></b>	Any AFX propertyValueType	Value of the keyframe
<b><i>inInterpolationType</i></b>	Enumerated value; one of: KeyframeInterpolationType.LINEAR KeyframeInterpolationType.BEZIER KeyframeInterpolationType.HOLD	In interpolation type of the keyframe
<b><i>outInterpolationType</i></b>	Enumerated value; one of: KeyframeInterpolationType.LINEAR KeyframeInterpolationType.BEZIER KeyframeInterpolationType.HOLD	Out interpolation type of the keyframe
<b><i>spatial</i></b>	boolean	True if the keyframe is on a spatial property, one of: PropertyValueType.ThreeD_SPATIAL PropertyValueType.TwoD_SPATIAL
<b><i>spatialProperties</i></b>	KeySpatialProperties	All spatial properties of the keyframe. See <a href="#"><i>KeySpatialProperties object attributes</i></a>
<b><i>inEase</i></b>	Array of AFX KeyframeEase objects	Incoming temporal ease of the keyframe
<b><i>outEase</i></b>	Array of AFX KeyframeEase objects	Outgoing temporal ease of the keyframe
<b><i>continuous</i></b>	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
<b><i>name</i></b>	string	Name of the animated Property
<b><i>keys</i></b>	Array of KeyFrames	Keyframes of the animation, see <a href="#"><i>KeyFrame object attributes</i></a>
<b><i>startValue</i></b>	Any AFX propertyValueType	First value of the animation. If there's no keyframe <code>PropertyAnim.keys.length == 0</code> , 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
<b><i>name</i></b>	string	Name of the animated Mask
<b><i>anims</i></b>	Array of PropertyAnim	Animations of the properties of the mask, see <a href="#"><i>PropertyAnim object attributes</i></a>

### 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
<b><i>name</i></b>	string	Name of the animated Effect
<b><i>matchName</i></b>	string	matchName of the animated Effect
<b><i>anims</i></b>	Array of PropertyAnim	Animations of the properties of the effect, see <a href="#"><i>PropertyAnim object attributes</i></a>

## 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
<b><i>name</i></b>	string	Name of the animated layer
<b><i>index</i></b>	string	Index of the animated layer
<b><i>transformAnims</i></b>	Array of PropertyAnim	Animations of the transformations, see <a href="#"><i>PropertyAnim object attributes</i></a>
<b><i>effectsAnims</i></b>	Array of EffectAnim	Animations of the effects, see <a href="#"><i>EffectAnim object attributes</i></a>
<b><i>masksAnims</i></b>	Array of MaskAnim	Animations of the masks, see <a href="#"><i>MaskAnim object attributes</i></a>

## IKRig object attributes

Describe an IK created by Duik.

*IKRig.type*

*IKRig.layer1*

*IKRig.layer2*

*IKRig.layer3*

*IKRig.goal*

*IKRig.controller*

Name	Type	Description
<b><i>type</i></b>	int	Type of the IK, either 1, 2, or 3. 0 if the IK is not valid.
<b><i>layer1</i></b>	AVLayer	First layer of the IK (the root, the top parent)
<b><i>layer2</i></b>	AVLayer or null	The second layer of the IK, if type is 2 or 3, or null if type is 1.
<b><i>layer3</i></b>	AVLayer or null	The third layer of the IK, if type is 3, or null if type is 1 or 2.
<b><i>goal</i></b>	AVLayer or null	A goal layer attached to the IK, or null
<b><i>controller</i></b>	AVLayer	The controller layer of the IK
<b><i>threeD</i></b>	boolean	true if this is a 3D IK (used for type 2 only)



<b><i>frontFacing</i></b>	boolean	true if the 3D layers face the front/back views, false if they face the right/left views.
<b><i>clockWise</i></b>	boolean	true if the IK bends clockwise. Used with type 2 and 3 only.
<b><i>created</i></b>	boolean	true if the IK has already been successfully created and exists in the comp.

## IKRig object methods

*IKRig.create()*

Name	Description	Return
<b><i>create()</i></b>	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 successful.

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*

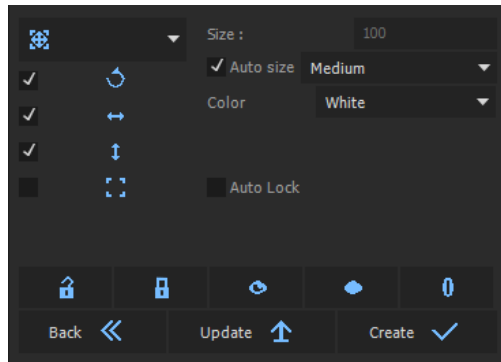
*PropertyDescription.parentName*

*PropertyDescription.dimensions*

*PropertyDescription.canSetExpression*

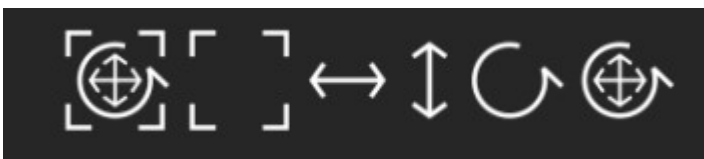
Name	Type	Description
<i>isEffect</i>	boolean	Property.parentProperty.isEffect
<i>index</i>	integer	Property.propertyIndex
<i>depth</i>	integer	Property.propertyDepth
<i>parentName</i>	string	Property.parentProperty.name
<i>dimensions</i>	integer	1, 2 or 3
<i>canSetExpression</i>	boolean	Property.canSetExpression

## Controller object attributes

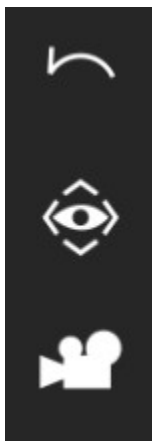


A controller created by Duik, which can have several shapes.

There are four transform shapes which can be combined:







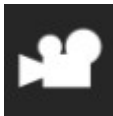


And three special shapes:



*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	Screenshot
<i>locked</i>	boolean	If true, transformation properties not controlled by the controller are	

		locked with a simple expression, to prevent inadvertently changing them	
<i>xPosition</i>	boolean	If true, the X Position of the controller may be animated	
<i>yPosition</i>	boolean	If true, the Y Position of the controller may be animated	
<i>rotation</i>	boolean	If true, the Rotation of the controller may be animated	
<i>scale</i>	boolean	If true, the Scale of the controller may be animated	
<i>arc</i>	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.	
<i>eye</i>	boolean	If true, the Position of the controller may be animated. The icon is an eye.	
<i>camera</i>	boolean	If true, the Position and rotation of the controller may be animated. The icon is a camera.	
<i>layer</i>	ShapeLayer	The controller layer	
<i>size</i>	float	The size of the controller (in % if type is <i>VECTOR</i> , pixels if type is <i>NULL</i> ) Set to 0 to use Duik.settings.controllerSize	
<i>type</i>	integer	Enumerated value, one of: Duik.layerTypes.NULL Duik.layerTypes.VECTOR	
<i>color</i>	Array of floats [R,V,B,A]	The color of the controller	

## Controller object methods

*Controller.lock()*

*Controller.unlock()*  
*Controller.update()*

Name	Description	Return
<b><i>lock()</i></b>	Locks the transformation properties not controlled by the controller, to prevent inadvertently changing them	void
<b><i>unlock()</i></b>	Unlocks the previously locked transformation properties. Note that before parenting a controller, it should be unlocked.	void
<b><i>update()</i></b>	Updates the shape of the controller, if its properties have changed	void

### TVPCamera object attributes

Describes a Camera imported from TVPaint

*TVPCamera.points*  
*TVPCamera.pointCount*  
*TVPCamera.profileprof*

Name	Type	Description
<i>points</i>	Array of <i><u>TVPCameraPoint</u></i>	The spatial keyframes of the camera animation
<i>pointCount</i>	integer	The number of spatial points of the camera
<i>profileprof</i>	<i><u>TVPProfileprof</u></i>	The temporal interpolation of the camera

### TVPCamera object methods

*TVPCamera.createNull(comp, links)*  
*TVPCamera.precompose(comp)*  
*TVPCamera.applyToLayer(camLayer, links)*

Name	Description	Return
<b><i>createNull(comp, links)</i></b>	Creates a Null object representing the TVPaint Camera in the comp	void
<b><i>precompose(comp)</i></b>	Precomposes all layers of the comp, and animates the resulting layer with the animation of the camera	void
<b><i>applyToLayer(camLayer, links)</i></b>	Applies the animation of the camera to the given layer	void

### TVPCameraPoint object attributes

A spatial keyframe of a camera from TVPaint

*TVPCameraPoint.x*  
*TVPCameraPoint.y*  
*TVPCameraPoint.zoom*  
*TVPCameraPoint.rotation*

Name	Type	Description
<i>x</i>	float	X position
<i>y</i>	float	Y position
<i>zoom</i>	float	Zoom value (from 0.0 to 1.0 or more)
<i>rotation</i>	float	Rotation (degrees)

### **TVPProfileprof object attributes**

A temporal interpolation from TVPaint

*TVPProfileprof.points*  
*TVPProfileprof.linear*  
*TVPProfileprof.pointCount*

Name	Type	Description
<i>points</i>	Array of <i>TVPProfileprofPoint</i>	Temporal keyframes
<i>linear</i>	boolean	Whether interpolation is linear or bezier
<i>pointCount</i>	integer	Number of temporal keyframes

### **TVPProfileprofPoint object attributes**

A temporal keyframe from TVPaint

*TVPProfileprofPoint.u*  
*TVPProfileprofPoint.v*

Name	Type	Description
<i>u</i>	float	Time coordinate of the key (from 0.0 to 1.0, 1.0 representing the end of the animation)
<i>v</i>	float	Value coordinate of the key (from 0.0 to 1.0, 1.0 representing the value of the last spatial point.

# Duik

## Duik Enumerated Values

Duik uses some predefined values to be simpler to use. Here are those values you can use with Duik settings, methods and attributes:

Name	Type	Value
Duik.sizes.SMALL	integer	0
Duik.sizes.MEDIUM	integer	1
Duik.sizes.BIG	integer	2
Duik.layerTypes.VECTOR	integer	2
Duik.layerTypes.NULL	integer	1
Duik.layerTypes.SOLID	integer	0
Duik.getLayers.INDEX	integer	0
Duik.getLayers.NAME	integer	1
Duik.getLayers.SELECTION_INDEX	integer	2
Duik.placement.TOP	integer	0
Duik.placement.BOTTOM	integer	1
Duik.placement.OVER_LAYER	integer	2
Duik.placement.UNDER_LAYER	integer	3
Duik.colors.WHITE	Array of floats	[1,1,1,1]
Duik.colors.RED	Array of floats	[1,0,0,1]
Duik.colors.GREEN	Array of floats	[0,1,0,1]
Duik.colors.BLUE	Array of floats	[0,0,1,1]
Duik.colors.CYAN	Array of floats	[0,1,1,1]
Duik.colors.MAGENTA	Array of floats	[1,0,1,1]
Duik.colors.YELLOW	Array of floats	[1,1,0,1]
Duik.colors.BLACK	Array of floats	[0,0,0,1]
Duik.colors.LIGHT_GRAY	Array of floats	[0.75,0.75,0.75,1]
Duik.colors.DARK_GRAY	Array of floats	[0.25,0.25,0.25,1]

## Duik Attributes

string *Duik.version*

float *Duik.versionNumber*

boolean *Duik.forceReload*

boolean *Duik.usePresets*

string *Duik.presetPath*

float *Duik.presetEffectsInstalledVersion*

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

## Duik Objects

*Duik.uiString*  
*Duik.settings*  
*Duik.utils*  
*Duik.setup*  
*Duik.js*  
*Duik.bridge*

Name	Description
<b><i>uiStrings</i></b>	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.
<b><i>settings</i></b>	Access to settings used by Duik.
<b><i>utils</i></b>	Some useful tools
<b><i>setup</i></b>	Methods and attributes to correctly install libDuik & pseudo effects.
<b><i>bridge</i></b>	Import / Export tools
<b><i>js</i></b>	General javascript tools

## Duik Methods


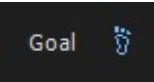
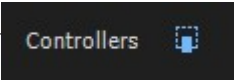
//TODO tri par ordre alphabétique

**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.**

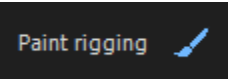
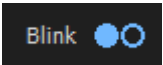
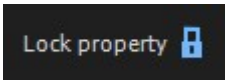
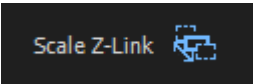
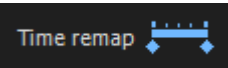
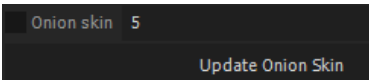
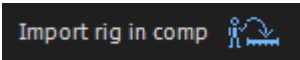
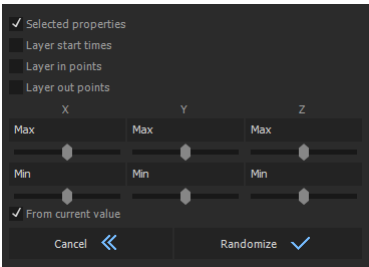
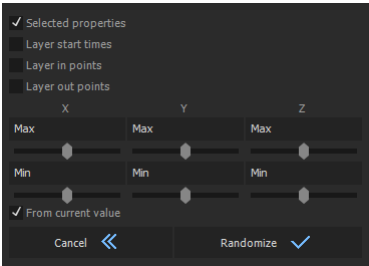
*Duik.autoIK(layers, clockWise, frontFacing)*  
*Duik.goal(layer, controller)*  
*Duik.addController(layer,color,rotation,xPosition,yPosition,scale,arc)*  
*Duik.addControllers(layers,color,rotation,xPosition,yPosition,scale,arc)*  
*Duik.oneLayerIK(controller,layer)*  
*Duik.twoLayerIK(controller,root,end,clockWise,frontFacing)*

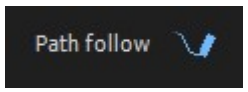
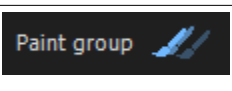
*Duik.threeLayerIK(controller,root,middle,end,clockWise)*  
*Duik.wiggle(layer,property,separateDimensions)*  
*Duik.threeDWiggle(layer,property,)*  
*Duik.twoDWiggle(layer,property)*  
*Duik.oneDWiggle(layer,property)*  
*Duik.adaptativeExposure(layer,property,precision,minExp,maxExp)*  
*Duik.fixedExposure(layer,property)*  
*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)*  
*Duik.randomizeProperties(props,fromCurrentVal,xMin,xMax,yMin,yMax,zMin,zMax)*  
*Duik.randomizeStartTimes(layers,fromCurrentVal,min,max)*  
*Duik.randomizeInPoints(layers,fromCurrentVal,min,max)*  
*Duik.randomizeOutPoints(layers,fromCurrentVal,min,max)*  
*Duik.pathFollow(layer)*  
*Duik.multiplane(numLayers)*  
*Duik.moveAway(layer)*  
*Duik.groupPaint(props)*

Name	Description	Return	Screenshot from Duik
<b><i>autoIK(layers, clockWise, frontFacing)</i></b>	Adds IK on the layers	true if successful, false if anything went wrong	
<b><i>goal(layer, controller)</i></b>	Adds a goal effect to the layer, which may be controlled by a controller	true if successful, false if anything went wrong	
<b><i>addController(layer, color, autoLock, rotation, xPosition, yPosition, scale, arc)</i></b>	Creates a null object (controller) at layer position and named by layer.name	Controller object	
<b><i>addControllers(layers, color, autoLock, rotation,</i></b>	For each layer, Creates a null object	Array of Controller objects	



<b><i>xPosition, yPosition, scale, arc</i></b>	(controller) at layer position and named by layer.name		
<b><i>wiggle(layer, property, separateDimensions)</i></b>	Adds a wiggle effect to given property	true if successful, false if anything went wrong	Wiggle 
<b><i>adaptativeExposure(layers, precision, minExp, maxExp, sync, layerSync)</i></b>	Adds exposure controls to the animation of the property.	true if successful, false if anything went wrong	Exposure 
<b><i>fixedExposure(layer, prop)</i></b>	Adds exposure controls to the animation of the property.	true if successful, false if anything went wrong	Exposure 
<b><i>addBones(layers)</i></b>	Adds bones to the layers	Array of AVLayer; bones	Bones 
<b><i>addZero(layer)</i></b>	Adds zero to the layer	AVLayer; zero	
<b><i>addZeros(layers)</i></b>	Adds zeros to the layers	Array of AVLayer; zeros	Zero pos(0)
<b><i>rotationMorph(layer, prop)</i></b>	Creates a rotation morph on the given property	true if successful, false if anything went wrong	Rot Morph 
<b><i>swing(layer,prop)</i></b>	Creates a swing on the given property	true if successful, false if anything went wrong	Swing 
<b><i>wheel(layer, radius, curved)</i></b>	Automates the rotation of the given layer using its position	true if successful, false if anything went wrong	Wheel 
<b><i>morpher(layers)</i></b>	Adds a slider to easily control interpolations of selected properties of the given layers.	true if successful, false if anything went wrong	Morpher 
<b><i>lensFlare(layers)</i></b>	Rigs the layers to move like a lens flare.	true if successful, false if anything went wrong	Lens 
<b><i>distanceLink(layer, property, parentLayer)</i></b>	Links the property to the distance of parentLayer	true if successful, false if anything went wrong	Dist. link 
<b><i>spring(property, layer, simulated)</i></b>	Adds a spring effect on the properties	true if successful, false if anything went wrong	Spring 
<b><i>copyAnim(layers, selectedKeysOnly, startTime, endTime)</i></b>	Copies the animation of the layers	Array of LayerAnim	Copy anim 
<b><i>pasteAnim(layers, layerAnims, startTime,</i></b>	Pastes the animations on the layers	int, the number of the layers on whichh an animtion was	Paste anim 

<b><i>getLayerMethod()</i></b>		pasted	
<b><i>rigPaint(layers)</i></b>	Rigs the paint effects to be able to animate all strokes as if there was only one.	Void	
<b><i>blink(layer, prop)</i></b>	Adds a blink effect to the property.	true if successful, false if anything went wrong	
<b><i>lockProperty(layer, prop)</i></b>	Locks the property with a simple expression.	void	
<b><i>scaleZLink(layers)</i></b>	Links the distance of the layer from the camera to its scale so its apparent size won't change.	void	
<b><i>timeRemap(layers)</i></b>	Activates the time remapping of the layers, extending them to the length of the comp and adjusting the last keyframe.	Void	
<b><i>onionSkin(layer, activate, duration)</i></b>	Activates or deactivates an onion skin on the paint effects of the layer.	void	
<b><i>importRigInComp(comp, rigComp, rigName)</i></b>	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	
<b><i>randomizeProperties(props, fromCurrentVal, xMin, xMax, yMin, yMax, zMin, zMax)</i></b>	Randomizes the values of the properties.	void	
<b><i>randomizeStartTimes(layers, fromCurrentVal, min, max)</i></b>	Randomizes start times of the given layers.	void	

<b><i>randomizeInPoints(layers, fromCurrentVal, min, max)</i></b>	Randomizes in points of the given layers.	void	
<b><i>randomizeOutPoints(layers, fromCurrentVal, min, max)</i></b>	Randomizes out points of the given layers.	Void	
<b><i>pathFollow(layer)</i></b>	Rigs the rotation of a layer so it follows its path	Void	
<b><i>multiplane(numLayers)</i></b>	Creates null objects rigged to easily animate a 2D multiplane camera.	void	
<b><i>moveAway(layer)</i></b>	Rigs the layer to be able to move it away from its parent with a simple slider	void	
<b><i>groupPaint(props)</i></b>	Rigs the paint effects to be able to animate all brushes as if there was only one.	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.adaptiveExposure(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, effects, 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.

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

returns

void

***Duik.randomizeProperties(props, fromCurrentVal, xMin, xMax, yMin, yMax, zMin, zMax)***

Randomizes the values of the properties.

Min and max values for each axis can be undefined: in this case, the axis won't be randomized.

parameters

props | Array of PropertyBase

fromCurrentVal | boolean, if true, min and max values are added to current property

value

returns

void

***Duik.randomizeStartTimes(layers, fromCurrentVal, min, max)***

Randomizes start times of the given layers.

Min and Max in seconds (comp time).

parameters

layers | Array of Layers or LayerCollection

fromCurrentVal | boolean, if true, min and max values are added to current start time

value

returns

void

***Duik.randomizeInPoints(layers, fromCurrentVal, min, max)***

Randomizes in points of the given layers.

Min and Max in seconds (comp time).

parameters

layers | Array of Layers or LayerCollection

fromCurrentVal | boolean, if true, min and max values are added to current in point

value

returns  
void

### ***Duik.randomizeOutPoints(layers, fromCurrentVal, min, max)***

Randomizes out points of the given layers.  
Min and Max in seconds (comp time).

parameters  
layers | Array of Layers or LayerCollection  
fromCurrentVal | boolean, if true, min and max values are added to current out point

value

returns  
void

### ***Duik.pathFollow(layer)***

Automates the rotation of the layer so it follows its path.

parameters  
layer | AVLayer

returns  
void

### ***Duik.multipane(numLayers)***

Creates null objects rigged to easily animate a 2D multipane camera.

parameters  
numLayers | integer, number of layers to create, default: 3

returns  
void

### ***Duik.moveAway(numLayer)***

Rigs the position of the layer to be able to move it away from its parent with a simple slider.

parameters  
layer | AVLayer

returns  
void

### ***Duik.groupPaint(props)***

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

parameters

props | Array of Properties (the brushes to rig)

returns

void

## Duik.setup

Methods and attributes to correctly install libDuik & pseudo effects.

### Duik.setup Attributes

*Duik.setup.presetEffects*

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

### Duik.setup Methods

*Duik.setup.installPseudoEffects()*

*Duik.setup.checkPresetEffectsVersion()*

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

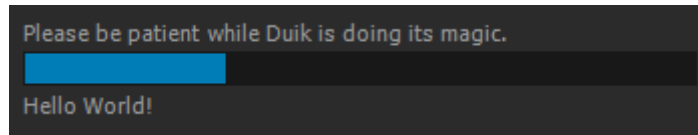
void

## Duik.ui

Contains attributes and methods to manipulate some user interface objects (progress bar, alerts...) displayed by libDuik

## Duik.ui ScriptUI Objects

*Duik.ui.progressPanel*  
*Duik.ui.progressGroup*  
*Duik.ui.progressBar*  
*Duik.ui.progressStatus*



Name	Type	Description
<i>progressPanel</i>	Window	Window containing the progress bar and status of libDuik
<i>progressGroup</i>	Group	The group in the Window <i>progressPanel</i> , used for the layout of child elements of the window.
<i>progressBar</i>	ProgressBar	The ProgressBar used by libDuik
<i>progressStatus</i>	StaticText	The text displayed behind the <i>progressBar</i>

## Duik.ui Methods

*Duik.ui.updateProgressPanel (val,status)*  
*Duik.ui.showProgressPanel (maxVal,status)*  
*Duik.ui.hideProgressPanel ()*

Name	Description	Return
<i>updateProgressPanel (val, status)</i>	Updates the progress panel.	Void
<i>showProgressPanel (maxVal, status)</i>	Initializes and displays the progress panel.	Void
<i>hideProgressPanel ()</i>	Hides the progress panel.	Void

### *Duik.ui.updateProgressPanel (val,status)*

Updates the progress panel, setting the value of the progress bar and the text of the status.

parameters:

val | integer, the value of the progress bar  
status | string, the text to display behind the progress bar

returns

void



### ***Duik.ui. showProgressPanel (maxVal,status)***

First, initializes the progress panel, setting the max value of the progress bar and the text to display behind it, then displays it.

parameters:

maxVal | integer, the max value of the progress bar  
status | string, the text to display behind the progress bar

returns

void

### ***Duik.ui. hideProgressPanel ()***

Hides the progress panel.

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*  
*Duik.uiStrings.moveAway*  
*Duik.uiStrings.multiplane*  
*Duik.uiStrings.camInfluence*

Name	Type	Default
<b><i>ik</i></b>	string	"IK"
<b><i>wiggle</i></b>	string	"Wiggle"
<b><i>exposure</i></b>	string	"Exposure"
<b><i>rotMorph</i></b>	string	"Rotation Morph"
<b><i>swing</i></b>	string	"Swing"
<b><i>wheel</i></b>	string	"Wheel"
<b><i>lensFlare</i></b>	string	"Lens Flare"
<b><i>distanceLink</i></b>	string	"Distance Link"
<b><i>spring</i></b>	string	"Spring"
<b><i>paintRig</i></b>	string	"Paint Rig"
<b><i>flip</i></b>	string	"Flip"
<b><i>moveAway</i></b>	string	"Distance from parent"
<b><i>multiplane</i></b>	string	"Multiplane"
<b><i>camInfluence</i></b>	string	"Camera Influence"

## 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*

*Duik.settings.bonePlacement*

*Duik.settings.ctrlPlacement*

*Duik.settings.controllerColor*

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

		size, according to <i>Duik.settings.boneSizeHint</i>	
<b><i>boneSizeHint</i></b>	integer	Enumerated value, one of: Duik.sizes.SMALL Duik.sizes.MEDIUM Duik.sizes.BIG	Duik.sizes.MEDIUM
<b><i>boneColor</i></b>	string	Hex value of the color of the bones, excluding the leading « # »	« FF0000 »
<b><i>morpherCreatesKeyframes</i></b>	boolean	If true, morpher will automatically create keyframes for each keyframe of the controlled properties	True
<b><i>getLayersMethod</i></b>	boolean	The method used to get layers (i.e. when pasting an animation) Enumerated value, one of: Duik.getLayers.NAME Duik.getLayers.INDEX Duik.getLayers.SELECTION_INDEX	Duik.getLayers.NAME
<b><i>bonePlacement</i></b>	integer	The placement of the bones in the comp. Enumerated value, one of: Duik.placement.TOP Duik.placement.BOTTOM Duik.placement.OVER_LAYER Duik.placement.UNDER_LAYER	Duik.placement.OVER_LAYER
<b><i>ctrlPlacement</i></b>	integer	The placement of the controllers in the comp. Enumerated value, one of: Duik.placement.TOP Duik.placement.BOTTOM Duik.placement.OVER_LAYER Duik.placement.UNDER_LAYER	Duik.placement.TOP
<b><i>controllerColor</i></b>	Array of integer	The color of the controllers, [R,G,B,A] or one of: Duik.colors.WHITE Duik.colors.RED Duik.colors.GREEN Duik.colors.BLUE Duik.colors.CYAN Duik.colors.MAGENTA Duik.colors.YELLOW Duik.colors.BLACK Duik.colors.LIGHT_GRAY Duik.colors.DARK_GRAY	Duik.colors.WHITE [1,1,1,1]

## Duik.settings Methods

*Duik.settings.save()*

*Duik.settings.load()*  
*Duik.settings.restoreDefaults()*

Name	Description	Return
<b><i>save()</i></b>	Saves Duik settings into After Effects preferences	void
<b><i>load()</i></b>	Loads Duik settings from After Effects preferences	void
<b><i>restoreDefaults()</i></b>	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.bridge

Tools for importing/exporting to/from After Effects.

## Duik.bridge.tvPaint

Tools to import and export assets from/to TvPaint

### Duik.bridge.tvPaint Methods

*Duik.bridge.tvPaint.parseCam(camString)*

*Duik.bridge.tvPaint.loadCamFile(camFile)*

Name	Description	Return
<b><i>parseCam(camString)</i></b>	Parses a string representing a camera exported from TVPaint.	<u><i>TVPCamera</i></u> object
<b><i>loadCamFile(camFile)</i></b>	Loads and parses a camera exported from TVPaint	<u><i>TVPCamera</i></u> object

#### ***Duik.bridge.tvPaint.parseCam(camString)***

Parses a string representing a camera exported from TVPaint, with its animation.

parameters:

camString | the string to parse

returns

TVPCamera, see *TVPCamera Object*

#### ***Duik.bridge.tvPaint.loadCamFile(camFile)***

Loads and parses a file representing a camera exported from TVPaint, with its animation.

parameters:

camFile | javascript File object to load

returns

TVPCamera, see *TVPCamera Object*

# Duik.js

General javascript related tools.

## Duik.js Methods

*Duik.js.escapeRegExp(string)*

*Duik.js.replaceAll(string, find, replace, caseSensitive)*

*Duik.js.random(min, max)*

Name	Description	Return
<b><i>escapeRegExp(string)</i></b>	Escapes all regular expressions special characters in the given string	string
<b><i>replaceAll(string, find, replace, caseSensitive)</i></b>	Replaces all occurrences of <i>find</i> by <i>replace</i> in the given string	string
<b><i>random(min, max)</i></b>	Random number between min and max	float

### ***Duik.js.escapeRegExp(string)***

Escapes all regular expressions special characters in the given string.

parameters:

string | string

returns

string, the modified string.

### ***Duik.js.replaceAll(string, find, replace, caseSensitive)***

Replaces all occurrences of *find* by *replace* in the given string.

parameters:

string | string to modify

find | string to search

replace | string, replacement

caseSensitive | boolean, whether to perform a caseSensitive search

returns

string, the modified string.

### ***Duik.js.random(min, max)***



Random number between min and max

parameters:

min | float or integer, the minimum value  
max | float or integer, the maximum value

returns

float, the random number.

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

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

<b><i>getAverageSpeeds(layers)</i></b>	Gets the average variation speed of the selected properties in the layers	float, average speed
<b><i>replaceInLayersExpressions(layers, oldString, newString)</i></b>	Replaces all occurrences of oldString by newString in all the expressions of all the layers.	void
<b><i>renameLayer(layer, newName, updateExpressions)</i></b>	Renames the layer, updating expressions in all the compositions of the project	void
<b><i>renameItem(item, newName, updateExpressions)</i></b>	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 position and scale (reported as 3D by AFX, but to be considered 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 occurrences 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