

Introduction

Including libDuik in your scripts

Installing libDuik

Using libDuik

Modifying libDuik

Pseudo Effects List

Duik

Attributes

Objects

Methods

IK

goal

addController

addControllers

wiggle

exposure

addBones

addZeros

rotationMorph

swing

wheel

morpher

lensFlare

distanceLink

spring

Duik.uiStrings

Attributes

Duik.settings

Attributes

Methods

save

load

restoreDefaults

Duik.utils

Methods

prepareProperty

getPropertyDimensions

getLength

getAverageSpeed

addPseudoEffect

getPuppetPins

getDistance

rigProperty

deselectLayers

checkNames

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 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> ».

- **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 to *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 set *Duik.forceReload* to *true* instead of *false* **inside libDuik itself**. You'll find this attribute just after the declaration of the *Duik* object in the beginning of 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:

layer.effect.addProperty(matchName) ;

Example: *app.project.activeItem.layer(1).effect.addProperty('DUIK_One_Layer_IK') ;*

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

Duik

Duik Attributes

string *Duik.version*

float *Duik.versionNumber*

boolean *Duik.forceReload*

Name	Type	Description
<i>version</i>	string, read-only	Version string of libDuik
<i>versionNumber</i>	float, read-only	Version number of libDuik
<i>forceReload</i>	boolean, read-only during run time	<p>When true, forces libDuik to be reloaded each time it is included in a script.</p> <p>When false, libDuik loads only on first run and then stays available until After Effects is shut down.</p> <p>This attribute should not be changed, unless you're editing libDuik.jsxinc itself and you need to do some testing and reload it without shutting down After Effects.</p> <p>To speed up launching of your scripts even at first launch, instead of including <i>libDuik.jsxinc</i>, you can copy it in the <i>Startup</i> subfolder of the <i>Scripts</i> folder of After Effects, and rename it to <i>libDuik.jsx</i>. This way, libDuik will be available to all scripts without <i>#include</i> macro.</p> <p>This attribute must be set by directly by editing libDuik.jsxinc to work.</p>

Duik Objects

Duik.uiString

Duik.settings

Duik.utils

Name	Description
<i>uiStrings</i>	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.
<i>settings</i>	Access to settings used by Duik.
<i>utils</i>	Some useful 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.IK(controller, layer1, layer2, layer3, goal, clockWise, threeD, frontFacing)

Duik.goal(layer, controller)

Duik.addController(layer)

Duik.addControllers(layers)

Duik.oneLayerIK(controller,layer)

Duik.twoLayerIK(threeD,controller,root,end,clockWise,frontFacing)

Duik.wiggle(layer,property,separateDimensions)

Duik.threeDWiggle(layer,property,)

Duik.twoDWiggle(layer,property)

Duik.oneDWiggle(layer,property)

Duik.exposure(layer,property,adaptative,limit,minExp,maxExp)

Duik.adaptativeExposure(layer,property,precision,minExp,maxExp)

Duik.fixedExposure(layer,property)

Duik.addBones(layers)

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)

Name	Description	Return
<i>IK(controller, layer1, layer2, layer3, goal, clockWise, threeD, frontFacing)</i>	Adds IK on the layers	true if successful, false if anything went wrong
<i>goal(layer, controller)</i>	Adds a goal effect to the layer, which may be controlled by a controller	true if successful, false if anything went wrong
<i>addController(layer)</i>	Creates a null object (controller) at layer position and named by layer.name	AVLayer; controller
<i>addControllers(layers)</i>	For each layer, Creates a null object (controller) at layer position and named by layer.name	Array of AVLayer; controllers
<i>wiggle(layer, property, separateDimensions)</i>	Adds a wiggle effect to given property	true if successful, false if anything went wrong
<i>exposure(layer, property, adaptative, precision, minExp, maxExp)</i>	Adds exposure controles to given property	true if successful, false if anything went wrong
<i>addBones(layers)</i>	Adds bones to the layers	Array of AVLayer; bones

<i>addZeros(layers)</i>	Adds zeros to the layers	Array of AVLayer; zeros
<i>rotationMorph(layer, prop)</i>	Creates a rotation morph on the given property	true if successful, false if anything went wrong
<i>swing(layer,prop)</i>	Creates a swing on the given property	true if successful, false if anything went wrong
<i>wheel(layer, radius, curved)</i>	Automates the rotation of the given layer using its position	true if successful, false if anything went wrong
<i>morpher(layers)</i>	Adds a slider to easily control interpolations of selected properties of the given layers.	true if successful, false if anything went wrong
<i>lensFlare(layers)</i>	Rigs the layers to move like a lens flare.	true if successful, false if anything went wrong
<i>distanceLink(layer, property, parentLayer)</i>	Links the property to the distance of parentLayer	true if successful, false if anything went wrong
<i>spring(property, layer, simulated)</i>	Adds a spring effect on the properties	true if successful, false if anything went wrong

Duik.IK(controller, layer1, layer2, layer3, goal, clockWise, threeD, frontFacing)

high-level method.

Adds IK on the layers

parameters:

controller | AVLayer
 layer1 | AVLayer
 layer2 | AVLayer or undefined
 layer3 | AVLayer or undefined
 goal | AVLayer or undefined
 clockWise | boolean, used only with two-layer and three-layer IK, default: false
 threeD | boolean, works only with two-layer IK, default: false
 frontFacing | boolean, default: false

returns

true if successful, false if anything went wrong

Duik.goal(layer, controller)

high-level method.

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)

high-level method.

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

parameters

layer | AVLayer

returns

AVLayer controller

Duik.addControllers(layers)

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

parameters

layers | Array of AVLayer

returns

Array of AVLayer controllers

Duik.wiggle(layer, property, separateDimensions)

high-level method.

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.exposure(layer, property, adaptative, limit, minExp, maxExp)

high-level method.

Adds exposure controls to given property.

parameters

layer | AVLayer of the property
property | Property
adaptative | boolean, default: true
limit | float, default: 100
minExp | integer, default : 1, minimum exposure
maxExp | integer, default : 4, maximum exposure

returns

true if successful, false if anything went wrong

Duik.addBones(layers)

high-level method.

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.addZeros(layers)

high-level method.

Adds a null object for each 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.rotationMorph(layer,prop)

high-level method.

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)

high-level method.

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)

high-level method.

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)

high-level method.

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.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

Name	Type	Description
<i>ik</i>	string	"IK"
<i>wiggle</i>	string	"Wiggle"
<i>exposure</i>	string	"Exposure"
<i>rotMorph</i>	string	"Rotation Morph"
<i>swing</i>	string	"Swing"
<i>wheel</i>	string	"Wheel"
<i>lensFlare</i>	string	"Lens Flare"
<i>distanceLink</i>	string	"Distance Link"
<i>spring</i>	string	"Spring"

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.controllerSizeAuto

Duik.settings.controllerSizeHint

Duik.settings.boneType

Duik.settings.boneSize

Duik.settings.boneSizeAuto

Duik.settings.boneSizeHint

Duik.settings.boneColor

Duik.settings.morpherCreatesKeyframes

Name	Type	Description	Default
<i>controllerSize</i>	integer	Size of controllers in pixels	100
<i>controllerSizeAuto</i>	boolean	If true, controller sizes will be automatically adapted to comp size, according to <i>Duik.settings.controllerSizeHint</i>	true
<i>controllerSizeHint</i>	integer	Enumerated value, one of: Duik.sizes.SMALL Duik.sizes.MEDIUM Duik.sizes.BIG	Duik.sizes.MEDIUM
<i>boneType</i>	integer	Enumerated value, one of: Duik.layerTypes.NULL Duik.layerTypes.SOLID	Duik.layerTypes.SOLID
<i>boneSize</i>	integer	Size of bones in pixels	20
<i>boneSizeAuto</i>	boolean	If true, bone sizes will be automatically adapted to comp size, according to <i>Duik.settings.boneSizeHint</i>	true
<i>boneSizeHint</i>	integer	Enumerated value, one of: Duik.sizes.SMALL Duik.sizes.MEDIUM Duik.sizes.BIG	Duik.sizes.MEDIUM
<i>boneColor</i>	string	Hex value of the color of the bones, excluding leading « # »	« FF0000 »

<i>morpherCreatesKeyframes</i>	boolean	If true, morpher will automatically create keyframes for each keyframe of the controlled properties	true
---------------------------------------	---------	---	------

Duik.settings Methods

Duik.settings.save()

Duik.settings.load()

Duik.settings.restoreDefaults()

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

Name	Description	Return
<i>prepareProperty(property, isFX, index, depth, parentName)</i>	Prepares property to be rigged	true if property can set expression, false otherwise
<i>getPropertyDimensions(property)</i>	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
<i>getLength(value1, value2)</i>	Gets the length between the values, whichever dimensions they are	float, length between the values
<i>getAverageSpeed(layer, property)</i>	Gets the average speed of the animated property, between its first and last keyframe only	float, average speed of the property
<i>addPseudoEffect(layer, pseudoEffectName)</i>	Adds a Duik predefined pseudo effect to the layer	Property, the effect added
<i>getDistance(layer1,layer2)</i>	Measure distance between two layers	integer, distance between layers, in pixels
<i>getPuppetPins(effects)</i>	Gets all puppet pins from a layer effects	Array of Properties, all puppet pins found
<i>rigProperty(layer, prop, pseudoEffect)</i>	Performs some checks on the property and adds a pseudo effect on the layer	Property, the effect added
<i>deselectLayers()</i>	Deselects all layers	Void
<i>checkNames(comp)</i>	Checks for duplicate names among the layers of the comp, renaming them if found.	true if any layer was renamed

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 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
value2 | 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.