Introduction

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 severeal scripts without having to include libDuik in all scripts.

Pseudo Effects

Instead of expression controls, Duik now makes an extensive use of Pseudo Effects. If those effects are not available, After Effects will warn for missing effects when using some functions of libDuik, but everything should work as expected anyway. If you want to disable those warnings, you will have to install the Pseudo Effects.

Copy all of the content of the file called *Duik_PresetEffects.xml* at the end of the file *presetEffects.xml* in the folder of After Effects, just before the last line *</effects>*.

In later versions, libDuik will be able to install Pseudo Effects itself (at the cost of one reboot of After Effects. You may use an installer to avoid that).

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 run of your scripts.

Modifying libDuik

If you're modifying libDuik and need to test it without having to reboot After Effects, 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.

Duik

Duik Attributes

string *Duik.version*
float *Duik.versionNumber*
boolean *Duik.forceReload*

Name	Type	Description
version	string, read-only	Version string of libDuik
versionNumber	float, read-only	Version number of libDuik
forceReload	boolean, read-only during run time	When true, forces libDuik to be reloaded each time it is included in a script. When false, libDuik loads only on first run and then stays available until After Effects is shut down. 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. To speed up launching of your scripts even at first launch, instead of including *libDuik.jsxinc*, you can copy it in the *Startup* subfolder of the *Scripts* folder of After Effects, and rename it to *libDuik.jsx*. This way, libDuik will be available to all scripts without *#include* macro. This attribute must be set by directly by editing libDuik.jsxinc to work.

Duik Objects

Duik.uiString
Duik.settings
Duik.utils

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

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

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

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
ik	string	*"IK"*
wiggle	string	*"Wiggle"*
exposure	string	*"Exposure"*
rotMorph	string	*"Rotation Morph"*
swing	string	*"Swing"*
wheel	string	*"Wheel"*
lensFlare	string	*"Lens Flare"*
distanceLink	string	*"Distance Link"*
spring	string	*"Spring"*

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

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

Duik.settings.save()

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

parameters:

none

returns

void

Duik.settings.load()

Loads Duik settings attributes from After Effects preferences (using app.settings.getSetting())
This allows to easily restore the settings set by the user even if After Effects is shut down.
If this method is not called at runtime, default values will be loaded at first run.

parameters:

none

returns

void

Duik.settings.restoreDefaults()

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

parameters:

none

returns

void

Duik.utils

Some useful methods.

Duik.utils Methods

Duik.utils.prepareProperty(property,isFX,index,depth,parentName)
Duik.utils.getPropertyDimensions(property)
Duik.utils.getLength(value1,value2)
Duik.utils.getAverageSpeed(layer,property)
Duik.utils.addPseudoEffect(layer,pseudoEffectName)
Duik.utils.getPuppetPins(effects)
Duik.utils.getDistance(layer1,layer2)
Duik.utils.rigProperty(layer,prop,pseudoEffect)
Duik.utils.deselectLayers()
Duik.utils.checkNames(comp)

Name	Description	Return
prepareProperty(property, isFX, index, depth, parentName)	Prepares property to be rigged	true if property can set expression, false otherwise
getPropertyDimensions(property)	Gets the dimensions of the property (1, 2 or 3), taking care of 2D layer positions (reported as 3D by AFX, but to be considdered as 2D)	integer, number of dimensions
getLength(value1, value2)	Gets the length between the values, whichever dimensions they are	float, length between the values
getAverageSpeed(layer, property)	Gets the average speed of the animated property, between its first and last keyframe only	float, average speed of the property
addPseudoEffect(layer, pseudoEffectName)	Adds a Duik predefined pseudo effect to the layer	Property, the effect added
getDistance(layer1,layer2)	Measure distance between two layers	integer, distance between layers, in pixels
getPuppetPins(effects)	Gets all puppet pins from a layer effects	Array of Properties, all puppet pins found
rigProperty(layer, prop, pseudoEffect)	Performs some checks on the property and adds a pseudo effect on the layer	Property, the effect added
deselectLayers()	Deselects all layers	Void
checkNames(comp)	Checks for duplicate names among the layers of the comp, renaming them if found.	true if any layer was renamed

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

Prepare the given property to be rigged.

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

parameters:

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

returns

true if property can set expression, false otherwise

Duik.utils.getPropertyDimensions(property)

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

parameters:

property | Property

returns

integer, number of dimensions

Duik.utils.getLength(value1, value2)

Gets the length between the values, whichever dimensions they are

parameters:

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

returns

float, length between the values

Duik.utils.getAverageSpeed(layer, property)

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

parameters:

layer | AVLayer of the property
property | Property

returns

float, average speed of the property

Duik.utils.addPseudoEffect(layer, pseudoEffectFileName)

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

parameters:

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

returns

Property, the effect added

Duik.utils.getPuppetPins(effects)

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

parameters:

effects | PropertyGroup, the effects group of a layer

returns

Array of Property, the puppet pins

Duik.utils.getDistance(layer1,layer2)

Measures distance between two layers, in pixels.

parameters:

layer1 | AVLayer
layer2 | AVLayer

returns

integer, distance in pixels

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

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

parameters:

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

returns

PropertyGroup, the effect added

Duik.utils.deselectLayers()

Deselects all layers

returns

void

Duik.utils.checkNames(comp)

Checks for duplicate names among the layers of the comp, renaming them if found.

parameters:

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

returns

true if any layer was renamed, false otherwise.