Animator

INHERITS FROM	Object
REQUIRES HEADER FILES	Animator.h
DEFINED IN	Extended Tools, version 0.0
AUTHOR	Scott Hess

CLASS DESCRIPTION

The Animator allows the programmer to create an object which will have animation properties, and then to cause it to animate, without explicitely using DPSTimedEntries. This behaviour requires the object to either be a target of an Animator object, or to have an Animator object as an outlet. This may be useful when the user would like two versions of a View subclass - one which animates, and one which does not. The Animator allows the programmer to only create one version, and connect it to an Animator object when animation is required.

The current Animator class is not fully supported by the InterfaceBuilder. Eventually, I would like to have the Animator act similar to a Control, converting lower-level events (in this case DPSTimedEntries) into higher level events. It should be fully configurable from within InterfaceBuilder via an Inspector panel, and the act of connecting an Animator to its target outlet should bring up a choice of actions to send. I have attempted to implement a custom palette for this, and the Animator class appears, and instances may be instantiated, but the Inspector panels do not work correctly. I believe that many of these problems are going to be fixed in the next release of the Interface Builder.

INSTANCE VARIABLES

Inherited from Object	Class	isa;
Declared in Animator	id	target;
	BOOL	running;
	float	timing
	float	threshold
	SEL	action
	DPSTimedEntry	te
	double	startTime
	double	elapsed

target The object the animator is to call

running	The current status of the Animator
timing	How long to pause between calls
threshold	At what priority to run
action	What message to send to target
te	Our timed entry
startTime	When the animator started running last
elapsed	The elapsed time between the start time and the last time that the animator called its target with the action

METHOD TYPES

Creating and Freeing an Animator	- free + new + setDefaultAction: +setDefaultRunning: + setDefaultThreshold: + setDefaultTiming:
Accessing the instance variables	- action - doubleValue - floatValue - intValue - resetValue - resetValue: - running - setAction: - setRunning: - setThreshold: - setTiming: - threshold - timing
Acting on the target	- sendAction - sendAction:to:
Interface methods	- start: - stop: - takeRunningFrom: - takeTimingFrom: - toggleRun:

Archiving	- awake - copy - read: - write:

IMPLEMENTED BY TARGET

Initialization	- setAnimator:

CLASS METHODS

new

+ new

Creates a new Animator object, with a nil target, and in the quiescent state.

setDefaultAction:

+ setDefaultAction:(SEL)*theAction*

Sets the default action method Animators should use on creation.

setDefaultRunning:

+ setDefaultRunning:(BOOL)*runningState*

Sets the default running state Animators should use on creation.

setDefaultThreshold:

+ setDefaultThreshold:(float)*theThreshold*

Sets the default threshold Animators should use on creation.

setDefaultTiming:

+ setDefaultTiming:(float)*theTiming*

Sets the default timing Animators should use on creation.

INSTANCE METHODS

action

–(SEL)**action**

Returns the action the Animator is currently calling the target with.

awake

– awake

Brings the Animator up animating, if it was animating when it went down, and if the target will respond .

copy

– copy

Makes a copy of the Animator object and returns it.

free

– free

Stops the current animation session (if it is animating), and frees up the Animator.

doubleValue

–(*double*)**doubleValue**

Returns the amount of time, in seconds, between the startup time of the Animator and the last time it called its target. Returns -1.0 if not currently running.

floatValue

–(*float*)**floatValue**

Returns the amount of time, in seconds, between the startup time of the Animator and the last time it called its target. Returns -1.0 if not currently running.

intValue

–(*int*) **intValue**

Returns the amount of time, in seconds, between the startup time of the Animator and the last time it called its target. Returns -1 if not currently running.

read:

– read:(NXTypedStream *)*stream*

Reads in the Animator from the typed stream *stream*.

resetValue:

– resetValue:(double)*value*

Resets the starting time of the Animator to *value*.

resetValue

– resetValue

Resets the starting time of the Animator to the current time.

running

– (BOOL)**running**

Return running status of the Animator.

sendAction

– sendAction

Sends the Animator's action to its target.

sendAction:to:

– sendAction:(SEL)*theAction* **to:***theTarget*

Sends the specified action to the specified target, with the receiving Animator as the only parameter.

setAction:

– setAction:(SEL)*theAction*

Sets the action for the Animator to call its target with.

setRunning:

– setRunning:(BOOL)*state*

Sets the running state of the Animator.

setTarget:

– setTarget:*anObject*

Sets the target to send to. If the target is the same as the current one, it returns right away. If not, it sets the target, and calls the target's setAnimator: method. This is useful for making sure that the target gets a chance to initialize the Animator's speed, etc. This should not be needed, as the setup should occur in the InterfaceBuilder. A side effect of this method is that if the setAnimator: method of the target works the same way (return right away if the animator is the same), the Animator and target can be connected as outlets in either direction, and you will not get an infinite loop.

setThreshold:

– setThreshold:(float)*threshold*

Sets the threshold for the Animtor to run at.

setTiming:

– setTiming:(float)*timing*

Sets the number of seconds between calls.

start:

– start:*sender*
Begins animation, and resets the startTime.

stop:

– stop:*sender*

Ends animation.

takeTimingFrom:

– takeTimingFrom:*sender*

Get the timing instance value from the sender.

takeRunningFrom:

– takerunningFrom:*sender*

Get the running instance value from the sender.

target

– target

Returns the Animator's target.

timing

– timing

Returns the Animator's timing.

threshold

– threshold

Returns the Animator's threshold.

toggleRun:

– toggleRun:*sender*

Toggles the running status of the Animator from running to not, or vice versa.

write:

– write:(NXTypedStream *)*stream*

Writes the receiving Animator to the typed stream *stream*.

TARGET METHODS

setAnimator:

– setAnimator:*anObject*

Send to target to allow setup of the Animator.