HOMEWORLD2

Mesh Animation to Game Event Binding (.madState files)

This document describes the format and usage of .madState files, which provide a system for binding mesh animations (stored in .mad files) to in-game events. .madState files are lua scripts which contain a number of functions that are run whenever certain in-game events happen. For example, you could use this system to bind an animation to the launch event of a ship. These functions are called while the game is running and for each time a particular event is called. Therefore, they should be kept quite simple. Generally, they are meant to trigger animations in ship-specific ways.

These functions are called from a SobWithMesh's MadState class. The MadState class is an encapsulation of an arbitrary number of boolean state variables. When the state of one of these variables changes, it calls a function in the .madState file, if it exists.

Mesh animations are deterministic in playback. Therefore, it is permissible to make the MadState dependent upon mesh animations and the game simulation dependent upon the MadState. This is one way to enable mesh animation to fit in with procedural animation. For example, we can require a ship to deploy its weapons (a mesh animation) before training them on targets (procedural animation).

Naming of .madState Functions

Functions in the .madState file are named as such:

`<ShipName>_<State>_On<Event>`

where:

ShipName is the name of the ship. Ex: VGR_HEAVYMISSILEFRIGATE.

State is the state variable. Ex: Launching. Following is a table events.

State name	Usage
Normal	Called when a ship is created.
Open	Called when a ship is in its open or "deployed" state.
Closed	Called when ships are no longer open or "deployed".
CodeRed	Called when a ship tries to fire its weapons.
CodeGreen	Called when a ship is finished firing its weapons (there's a delay before this is called)
ResourceStart	The ship is about to start resourcing, this is called when it starts to get in position.
ResourceDo	The ship is in latch position and harvesting.

ResourceEnd	The ship has launched from the resource and is about to head back.
RepairStart	The ship is about to get in position to repair something.
RepairDo	The ship is in position and is starting to repair.
RepairEnd	The ship has finished repairs.
DockPathOpen	The animation linked dock path (set in the hod file) has been booked. The docking / launching ship will wait for this state to be set.
DockPathClosed	The animation linked dock path is now free.
Launched	The ship is fully launched.
Docked	The ship is on the final approach for docking.
DefenseFieldActivate	The defense field is trying to activate, the game logic will not start until this state is set.
DefenseFieldDeActivate	The defense field is no longer active.
CloakFieldActivate	Cloaking is trying to activate.
CloakFieldDeactivate	Cloaking has stopped.
HyperspaceGateActivate	Hyperspace gate is in position and trying to link with its pair.
HyperspaceGateDeActivate	Hyperspace gate has delinked.
DoingFlightManeuver	The ship is performing some kind of flight maneuver.
CaptureActive	The ship has started to capture the target.
CaptureInActive	The ship is not capturing anything.
NIS00	Called by the NIS.
NIS01	Called by the NIS.
NIS02	Called by the NIS.

Event is one of the following:

Event	Usage
OnSet	Called when the specific state is set by the game code.
OnPause	Called when the animation associated with the state pauses.
OnEnd	Called when the animation associated with the state ends.

Example:

Here's an simplified example .madState script:

```
VGR_HEAVYMISSILEFRIGATE_CodeRed_OnSet = function(ship)
    setState(ship, "CodeRed", 0)
    startAnim(ship, "Open")
    startEffect(ship, "Open")
    setPauseTime(ship, "Open", 1000)
end

VGR_HEAVYMISSILEFRIGATE_CodeRed_OnPause = function(ship)
    setState(ship, "CodeRed", 1)
end
```

```
VGR_HEAVYMISSILEFRIGATE_CodeGreen_OnSet = function(ship)
    startAnim(ship, "Close")
    startEffect(ship, "Close")
    setTime(ship,"Close",0)
    setPauseTime(ship, "Close", 1000)
end
```

Obviously, this script is for the Vaygr Heavy Missile Frigate. This defines animations for going "Code Red" and "Code Green". When going "Code Red", the animation for opening the missile bay doors is triggered, and told to pause at its end (1000 is used, it's just a big number). By default the system which calls our script also sets the state variable so we need to unset it as we only want weapons to be active when the animation has finished. In the OnPause function we actually set the state. When going "Code Green", the animation for closing the missile bay doors is triggered and CodeGreen is set for us by the calling code.

Available functions:

From within these event functions, you can call any allowable Lua functions. However, we defined the following animation and state-specific functions:

`setState(ship, stateName, state)`		
Sets the specified state within the ship's MADState class. Will not trigger a call to the corresponding _OnSet Lua function. This is useful for making the setting of a state dependent upon an animation completing, or reaching a pause point.		
`ship`	`Sob*`	Passed from game
`stateName`	`string`	Name of state. See state name table above.
`state`	`bool (0/1)`	Is state on or off
`Returns`	`nothing`	

`getState(ship, stateName)`		
Queries the specified state within the ship's MADState class.		
`ship`	`Sob*`	Passed from game
`stateName`	`string`	Name of state. See state name table above.
`Returns`	`bool (0/1)`	Is state on or off
`startAnim(ship, animName)`		
Starts a mesh animation. Will not re-start an animation if already running. An assert will be generated if animation not found.		
`ship`	`Sob*`	Passed from game
`animName`	`string`	Name of mesh animation from .mad file.
`Returns`	`nothing`	

`startParamAnim(ship, animName, param)`
Starts a parameterized mesh animation. A parameterized animation is an animation where the time is specified by a parameter which ranges from 0..1. Will not re-start an animation if already running. An assert will be generated if animation not found.

ship	Sob*	Passed from game
animName	string	Name of mesh animation from .mad file.
param	string	Name of parameter. "HorizontalRotation" and "VerticalRotation" are built-in animations that determine the rotation based on the ship's rotational thrust, similar to how the engine glow is parameterized.
Returns	nothing	

stopAnim(ship, animName)		
Stops a mesh animation. Animation need not be running.		
ship	Sob*	Passed from game
animName	string	Name of mesh animation from .mad file.
Returns	nothing	

pauseAnim(ship, animName)		
Pauses a mesh animation. Animation need not be running.		
ship	Sob*	Passed from game
animName	string	Name of mesh animation from .mad file.
Returns	nothing	

unpauseAnim(ship, animName)		
Unpauses a mesh animation. Animation need not be running or paused.		
ship	Sob*	Passed from game
animName	string	Name of mesh animation from .mad file.
Returns	nothing	

setPauseTime(ship, animName, time)		
Specifies when the indicated animation will pause.		
ship	Sob*	Passed from game
animName	string	Name of mesh animation from .mad file.
time	float	Time (seconds) in animation when animation will pause. Pass a very large number to pause at the end.
Returns	nothing	

getPauseTime(ship, animName)		
Queries the time when the specified animation will pause.		
ship	Sob*	Passed from game

animName	string	Name of mesh animation from .mad file.
Returns	float	Time (seconds) in animation when animation will pause. A very large number means it will not pause.

setTime(ship, animName, time) Sets the current time in an animation (FF or Rew)		
ship	Sob*	Passed from game
animName	string	Name of mesh animation from .mad file.
time	float	Time (seconds) in animation to jump to. Will be clamped to animation bounds.
Returns	nothing	

getTime(ship, animName) Queries the current time within the animation.		
ship	Sob*	Passed from game
animName	string	Name of mesh animation from .mad file.
Returns	float	Time (seconds) in animation to jump to.

setLoopCount(ship, animName, nLoops) Sets the number of times the animation will loop. 0 by default.		
ship	Sob*	Passed from game
animName	string	Name of mesh animation from .mad file.
nLoops	int	Number of times the animation will loop
Returns	nothing	

getLoopCount(ship, animName) Queries the number of times the animation will loop. Current loop is included.		
ship	Sob*	Passed from game
animName	string	Name of mesh animation from .mad file.
Returns	int	Number of times animation will loop. Will be 0 for animations with no looping.

startEffect(ship, eventName) Starts playing an effect event as defined in the ship's .events file.		
ship	Sob*	Passed from game
effectName	string	Name of the effect to play.
Returns	Effect handle	Handle to started effect.

Inclusive and exclusive states

Some states can be inclusive and exclusive. This makes it easier to apply a certain animation state behavior to all ships. In Scripts/MeshAnimation.lua, there are two tables named inclusive and exclusive which list inclusive and exclusive pairs of states. When the first of an inclusive pair is set, the second is also set. When the first of an exclusive pair is set, the second is cleared.

These rules are applied similar to how states are applied by the .madState scripts: they do not call the lua functions and therefore cannot trigger animations.