for Unity 3d
by Altered Reality Entertainment
v1.6.0

Documentation

What is Sage?

Sage is a powerful and flexible Animation State Machine and Blend Graph system for Unity 3d. Complete with a fully featured graph based editor.

Overview

Key Terms

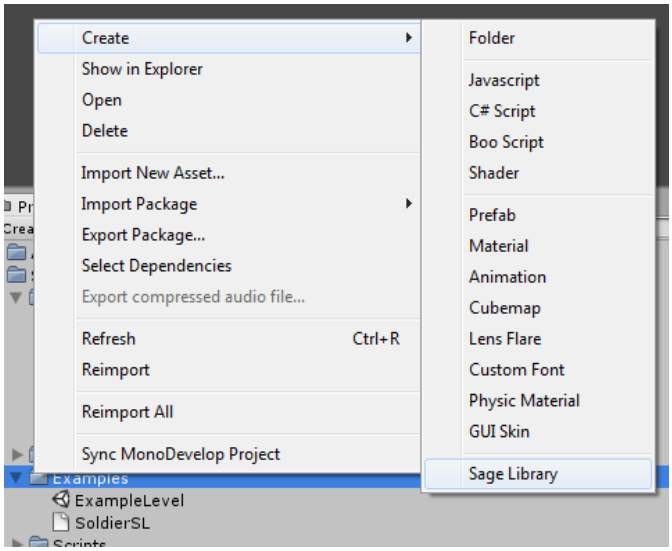
1. ***Sage Library***
   A Sage Library is the main assets that are created and used with Sage. They contain all of the Variables, State Machines, and Blend Graphs.
2. ***Sage Controller***
   A Sage Controller is a script that is attached to a GameObject that has an Animation component. This is the script that will actually affect the animation of a GameObject and controller a Sage Library.
3. ***Sage Testing GUI***
   The Sage Testing GUI is a script that is included with Sage that will fully expose all of the features of a Sage Library and Sage Controller, and make it easy to test your graphs at runtime.
4. ***Template Animation***
   The Template Animation is the Animation component that the Sage Editor will use as a source to pull available animation names from when selecting animations. It is possible to reference animations that are not in the currently selected Template Animation, however Sage will warn that this is the case.
5. ***Variable***
   A Variable is a value that can be set from a SageController and that will control what is happening in State Machines and Blend Graphs. A variable can also define a Min and Max value, as well as determine how fast it's value can be changed.
6. ***State Machine***
   A State Machine is a graph that contains discrete animation states that can be transitioned between. An example of some states could be “Attack”, “Move”, and “Idle”. A Parallel State Machine that always runs in a parent state machine, regardless of what the current state is.
7. ***Blend Graph***
   A Blend Graph is a graph that contains animations that are to be blended together. An example use for a blend graph would be to blend 8 movement directions together based on the direction a character is supposed to move.
8. ***Node***
   A Node is a functional element of a graph. Some examples of nodes are is a node to play an animation or a node to define a state in a state machine.
9. ***Port***
   A Port is a connection point that is used to Link nodes together. If a Port is on the left side of a Node, it is an Input Port. If it is on the right side of a Node, it is an Output Port.
10. ***Link***
    A Link is a connection between two nodes, using Ports as connection ports. Links are used to define state transitions, as well as to pass float values between nodes.

Workflow

Creation

This section will outline how to create new Sage Libraries, complete with Variables and State Machines. PLEASE NOTE: If detailed steps are left out of a step, please see the Reference section at the below this section for more info.

1. ***Create a Sage Library***



Right click on any Folder in the Project view, and select Create → Sage Library

2. ***Open your new Library in the Sage Editor***
   Select the new Library Asset in the Project View, and in the Inspector, click "Open in Sage Editor"
3. ***Select a Template Animation***
4. ***Add Variables***
5. ***Add State Machines***
6. ***Edit your State Machines and Blend Graphs***

IMPORTANT NOTE:

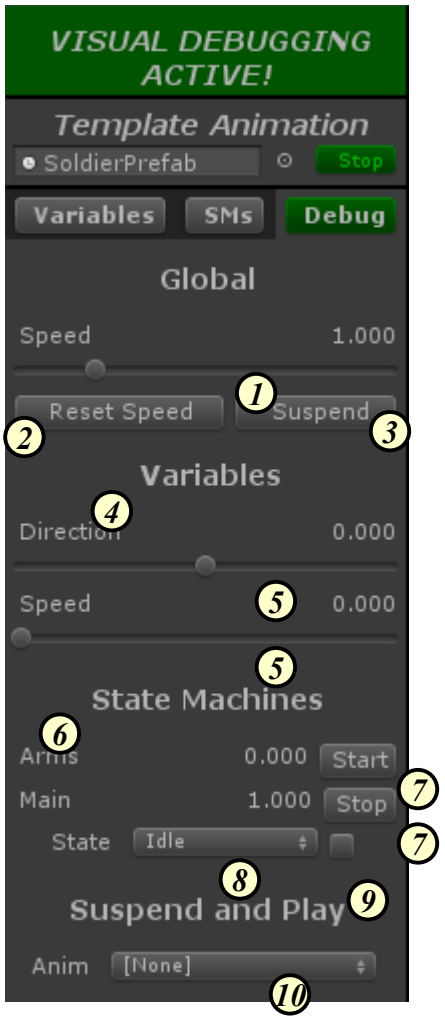
There does NOT have to be one Sage Library for each animated character, as Sage only stores animation references by string. So you can use a single library to drive the animation of multiple characters, as long as their animations are named the same thing!

Testing

Testing a Sage Library is VERY simple and straight forward...

1. ***Select a Template Animation Game Object by dragging and dropping on the game object control.***
2. ***Click the Start button to start previewing and debugging.***

After that, it will place your character in front of the scene view camera, and launch the debugging window, where you can adjust all of the variables, start/stop state machines, and transition/force the various states.



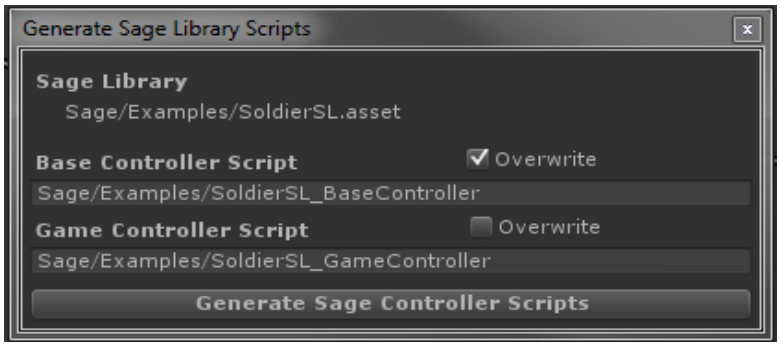
1. ***Global Speed Value Slider***
   This slider controls the global playback speed for the Sage Library that is being previewed.
2. ***Reset Global Speed***
   The resets the Global Speed to 1.0.
3. ***Suspend/Resume Library***
   This suspends the library in it's current state. When suspended, clicking this will resume to it's previous state.
4. ***Variable List***
   This is the list of variables that are in the current library.
5. ***Variable Value Slider***
   These sliders change the value of each variable.
6. ***State Machine List***
   This is where all of the State Machines in the current library are listed.
7. ***Stop/Start State Machine***
   This will start or stop State Machines in the current library.
8. ***States***
   This area lists all of the states in this State Machine. Select one from the drop down to transition to that state.
9. ***Force Transitions***
   If this is checked, when requesting to transition to a new state, it will instead force a transition and ignore normal transition paths.
10. ***Suspend and Play Animation***
    This will show a list of all animations in this library. Clicking on an animation will suspend the library, play the animation, and when finished, then resume the library.

Runtime Usage

This section will outline the recommended way to use your Sage Libraries at runtime while your game is running. First, you need to generate the game scripts that will be used to interact with your Sage Library...

1. ***Click the Generate Scripts button***

2. ***This will bring up the following dialog window...***



This dialog defines what scripts will be created, and whether or not it will overwrite existing scripts. As follows...

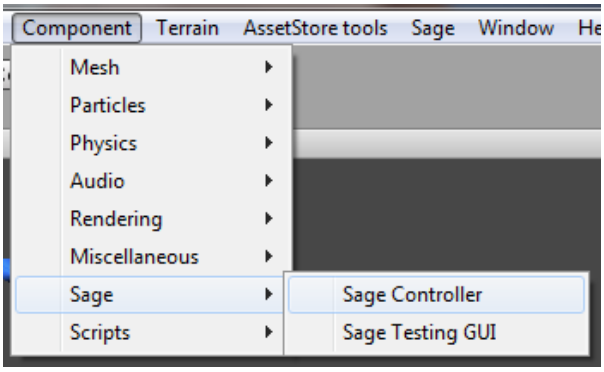
- Base Controller Script
  - This script contains all of the base functions for interacting with this Sage Library. Such as getting/setting variables, starting/stopping state machines, transitioning/forcing state changes, and even callbacks for when a state is entered/updated/exited.
- Game Controller Script
  - This script derives from the Base Controller Script. All custom game code for interacting with this Sage Library should go in here. For example, you can override On_MainSM_IdleState_Entered() to execute a custom action when the Idle State is entered in the Main State Machine.

3. ***Configure what script names to generate, whether they should overwrite, then click Generate Sage Controller Scripts***
4. ***Create a Game Object containing the desired Animation component.***
5. ***Attach the newly generated Game Script Component to the Game Object***
6. ***Access using gameObject.GetComponentInChildren<MyScriptName>()***

Alternate Runtime Usage

This section will outline an alternate way to use your Sage Libraries at runtime while your game is running. **<u>NOTE:</u> This was the primary method of interacting with a Sage Library before version 1.3, it still works, but it is recommended to use the above method, as it is easier to use.**

1. ***Create a Game Object containing the desired Animation component.***
2. ***Attach a Sage Controller to the Game Object***



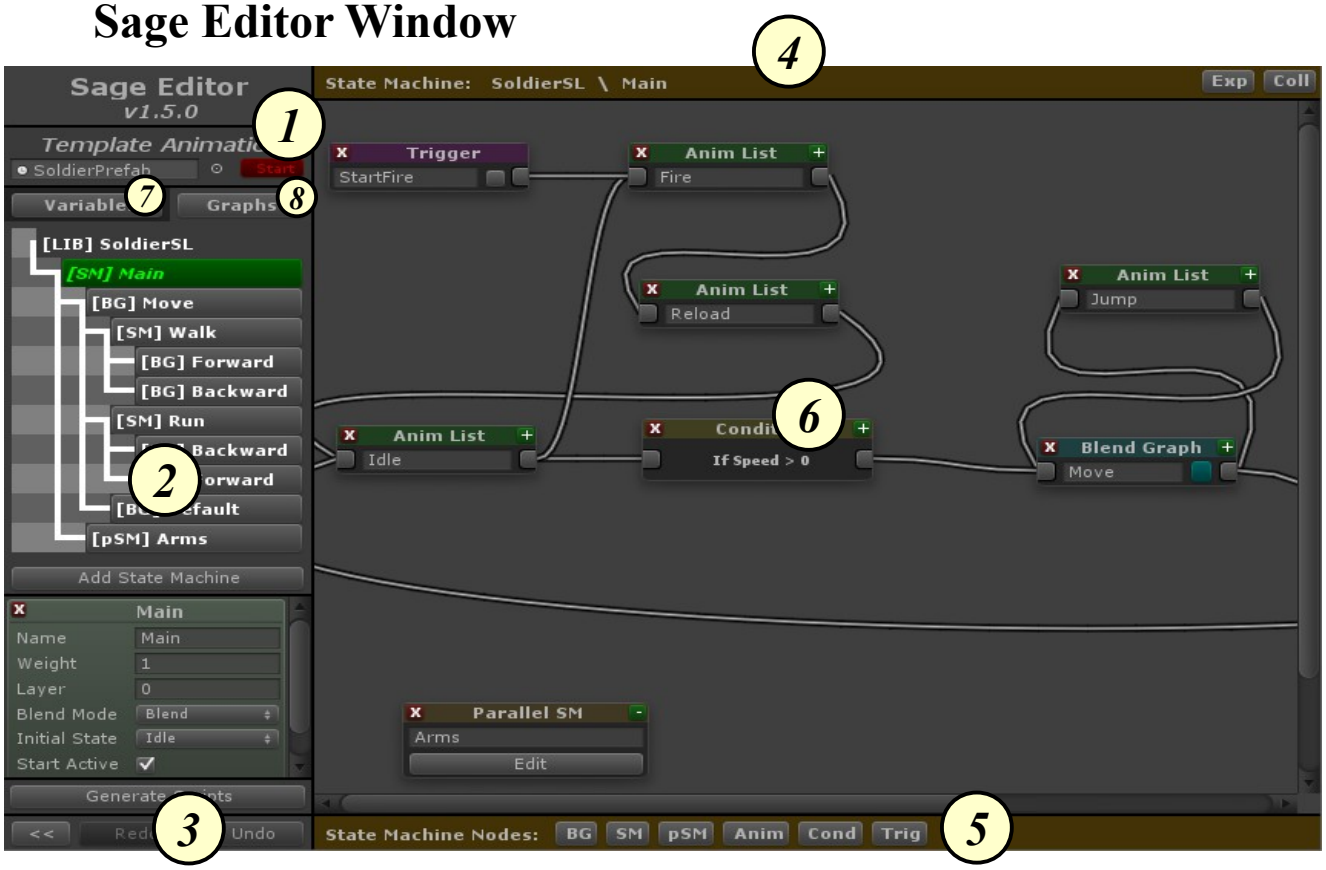
3. ***Set Variables and Trigger Transitions***

   To interact with a SageLibraryAsset, you must access it through the SageController class. For full documentation, please refer to SageCodeReference.pdf. Listed below are a few useful functions:

   1. **static SageController SageController.Get(GameObject gameObject)**
      This static function gets the SageController attached to a GameObject (or one of its children). This will return null if there is no SageController attached to it.
   2. **bool SageController.SetFloat(string variableName, float value)**
      This function sets a float variable on this SageController.
   3. **bool SageController.TransistionToState(string stateMachine, string newState)**
      This function causes a state machine to transition from it's current state to a new state, using only normal transition paths.
   4. **bool SageController.ForceState(string stateMachine, string newState)**
      This functions forces a state on a state machine, ignoring normal transistion paths.
   5. **bool SageController.StartStateMachine(string stateMachine)**
      This function starts a state machine on this SageController.
   6. **bool SageController.StopStateMachine(string stateMachine)**
      This function stops a state machine on this SageController.
   7. **string SageController.GetStateMachineCurrentState(string stateMachine)**
      This function gets the current state of an activate state machine.

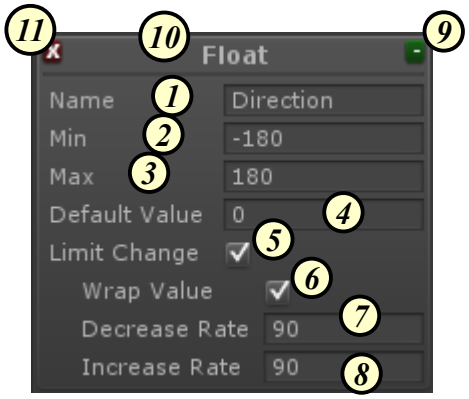
Reference

Sage Editor Window



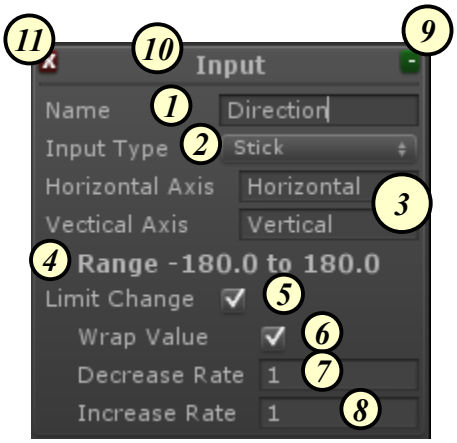
1. ***Template Animation***
   This is where you select the current Template Animation by dragging and dropping an asset game object that contains an Animation component. You can also start/stop previewing and debugging here.
2. ***Variables/Graphs***
   This is where you add/edit/delete both Variables and Graphs
3. ***Control Bar***
   This is where you can show/hide the side bar, and undo/redo any editor actions that are performed. You can also generate scripts from here.
4. ***Navigation Bar***
   This shows what graph is currently being edited, and it will allow you to easily go to the parent of a current graph if it has a parent. As well, all nodes can be Expanded or Collapsed here.
5. ***Toolbox Bar***
   This is where new nodes can be created and added to the current graph. The options in this area will change based on what type of graph is currently be edited.
6. ***Graph Area***
   This is where all graphs are viewed and edited.
7. ***Switch to Variables View***
8. ***Switch to Graphs View***

Float Variables



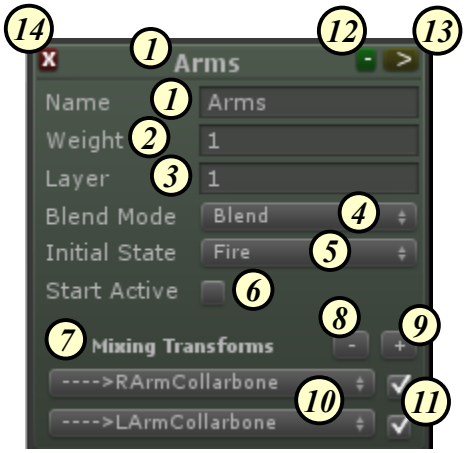
1. ***Name***
   This is the name of the variable, which is used to reference it in the graph and when changing the value of the variable on the SageController.
2. ***Min***
   This is the minimum value that this variable can be set to.
3. ***Max***
   This is the maximum value that this variable can be set to.
4. ***Default Value***
   This is the default value that this variable will be set to when the Sage Library is initialized.
5. ***Limit Change***
   This indicates if, when requesting to change this variable, if the value should change instantly, or if it should approach the requested value at a given rate.
6. ***Wrap Value***
   This indicates if, when Limit Change is active, if the value wraps around. For Instance, if your Min is 0.0 and Max is 1.0. And then your current value is 0.1 and target is 0.9. When approaching this value, it will approach is this manner...0.1, 0.075, 0.05, 0.025, 0.0, 0.975, 0.95, 0.925, 0.9.
7. ***Decrease Rate***
   If limit change is enabled, and the target value is less than the current value, this variable will approach the target value at this rate.
8. ***Increase Rate***
   If limit change is enabled, and the target value is greater than the current value, this variable will approach the target value at this rate.
9. ***Expand/Collapse***
   This will either collapse this float entry, and make it take up less room, or it will expand it to show more details.
10. ***Variable Type***
    This indicates what type this variable is. As of right now, only Float variables are supported.
11. ***Delete***
    This deletes this Variable from the Sage Library.

Input Variables



1. ***Name***
   This is the name of the input variable, which is used to reference it in the graph.
2. ***Input Type***
   This is the type of input variable. The current types are Button, Axis, and Stick.
3. ***Input Name(s)***
   This is the name or names of the type of input that is selected. It will show different entries based on the Input Type. For instance, "Button" will show "Button Name" here. Which is the name of the button to pass into the Input.Button check.
4. ***Value Range***
   This is the range of the value (min and max) as it will be used in this Sage Library.
5. ***Limit Change***
   This indicates if, when requesting to change this variable, if the value should change instantly, or if it should approach the requested value at a given rate.
6. ***Wrap Value***
   This indicates if, when Limit Change is active, if the value wraps around.
7. ***Decrease Rate***
   If limit change is enabled, and the target value is less than the current value, this variable will approach the target value at this rate.
8. ***Increase Rate***
   If limit change is enabled, and the target value is greater than the current value, this variable will approach the target value at this rate.
9. ***Expand/Collapse***
   This will either collapse this float entry, and make it take up less room, or it will expand it to show more details.
10. ***Variable Type***
    This indicates what type this variable is. As of right now, only Float variables are supported.
11. ***Delete***
    This deletes this Variable from the Sage Library.

State Machines



1. ***Name***
   This is the name of the variable, which is used to reference it in the graph and when changing the value of the variable on the SageController.
2. ***Weight***
   This is the base weight to apply to all animations that are played in this state machine. This is important to note, as it is possible to have multiple state machines active at the same time.
3. ***Layer***
   This is the layer to play all animations in the state machine on. This is important to note, as it is possible to have multiple state machines active at the same time.
4. ***Blend Mode***
   This is the blend mode to use when playing all animations in this state machine.
5. ***Initial State***
   This is the state to start the state machine in.
6. ***Start Active***
   This indicates if this State Machine should be started as active when the Sage Library containing this State Machine is started.
7. ***Mixing Transforms List***
   This is the list of mixing transforms to apply to all animation in this state machine. If any bones are in this list, the animations in this state machine will ONLY be applied to those bones, and no other bones. If this list is empty, all animations will be applied to all bones like normal.
8. ***Remove Mixing Transform***
   This removes a mixing transform entry.
9. ***Add Mixing Transform***
   This adds a mixing transform entry.
10. ***Mixing Transform Bone***
    This is the bone that this mixing transform entry refers to. Animations in this state machine will apply to this bone. The list of bones is populated from the Template Animation.
11. ***Mixing Transform Recursive***
    This indicates if this mixing transform entry is recursive. If it is recursive, the

animations in this state machine will ALSO apply to all of this bones children.

12. Expand/Collapse

This will either collapse this State machine entry, and make it take up less room, or it will expand it to show more details.

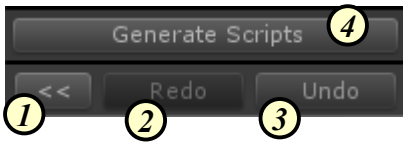
13. Edit Graph

This will make this State Machine the currently active graph in the Graph Editor.

14. Delete

This deletes this State Machine from the Sage Library.

Control Bar



1. ***Hide/Show Side Bar***

   This hides the side bar, making more room to sure the graph area. If hidden, it will show it again.

2. ***Redo***

   This will redo the last Sage Editor action that was undone

3. ***Undo***

   This will undo the last Sage Editor action that was done.

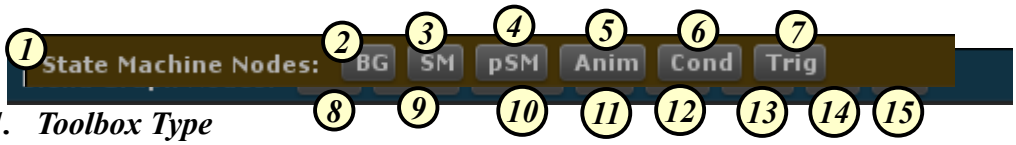
4. ***Generate Scripts***

   This will launch the Generate Scripts dialog. See the “Runtime Usage” section above for more details.

Navigation Bar

1. ***Graph Type***
   This indicates the type of the currently active graph. The background color also indicates type as follows: Orange is a State Machine Graph, Blue is a Blend Graph, and Red indicates no graph is active.
2. ***Graph Path***
   This displays the path of the current graph that is active. Starting with the Sage Library asset name, then the parent of the currently active graph (if it has a parent), and then the name of the graph itself.
3. ***Edit Parent Button***
   Clicking this button edits the parent of the currently active graph.
4. ***Expand/Collapse All Nodes***
   This will expand or collapse all nodes in the current grpah.

Toolbox Bar



1. ***Toolbox Type***
   This shows which version of the toolbox is currently active. All of the buttons will create a new node and place it in the bottom left of the graph area.

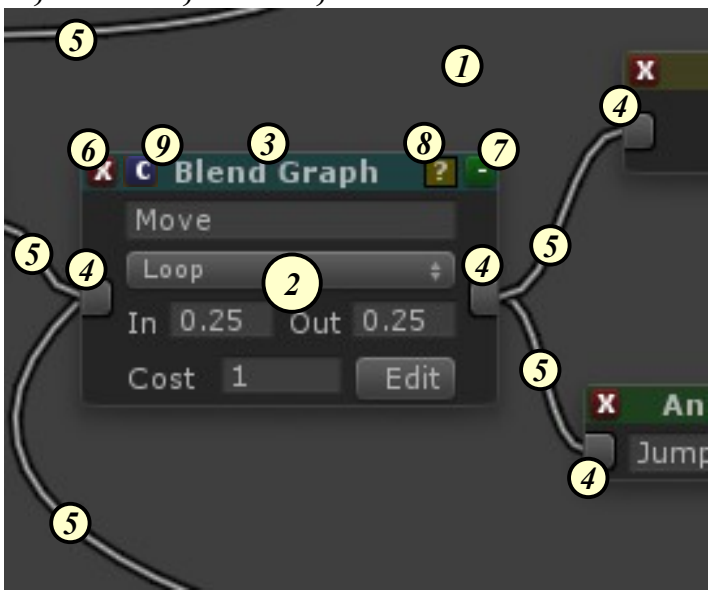
State Machine Graph Toolbox Buttons

2. ***Create Blend Graph Node***
3. ***Create State Machine Node***
4. ***Create Parallel State Machine Node***
5. ***Create Anim List Node***
6. ***Create Condition Node***
7. ***Create Trigger Node***

***Blend Graph* Toolbox Buttons**

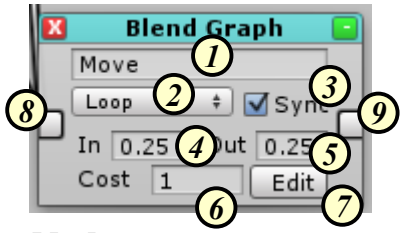
8. ***Create Variable Node***
9. ***Create Animation Node***
10. ***Create Blend Node***
11. ***Create Direction Node***
12. ***Create Min Node***
13. ***Create Max Node***
14. ***Create Blend Graph Node***
15. ***Create State Machine Node***

Graph Area, Nodes, Ports, and Links



1. ***Empty Graph Area***
   Right clicking and dragging on any empty graph area will pan and move the graph area view.
2. ***Node***
   This is a node, which is the basis of all Sage Library graphs.
3. ***Node Title***
   This is the title of the node. Clicking here and dragging will move the node around.
4. ***Port***
   This is a Port. A port on the left of a node is an Input Port, a Port on the right of a node is a Output Port. To create a link and connect an Output Port to an Input Port, left click on the port and drag to another port and release. To remove all links to a port, place the mouse over the port and right click. Also, if you hold the mouse over any port, it will display the name of the port. NOTE: If a port is of the type Float, and it has multiple inputs, the final input value will be the result of all the input values multiplied together.
5. ***Link***
   This is a Link, which is a logical connection between two nodes. Creating Links between nodes is the main way in which Sage works.
6. ***Delete Node Button***
   This will delete a node.
7. ***Expand/Collapse Node Button***
   This will either collapse a node, causing it to take up less room in the graph. Or, if already collapsed, it will expand it to expose full options.
8. ***Warning Tooltip***
   This element will ONLY show up if there is something potentially wrong with how the node is configured. Place the mouse over this element to display the Warning.
9. ***Copy Node***
   This copies this node to be pasted into any graph of the same type.

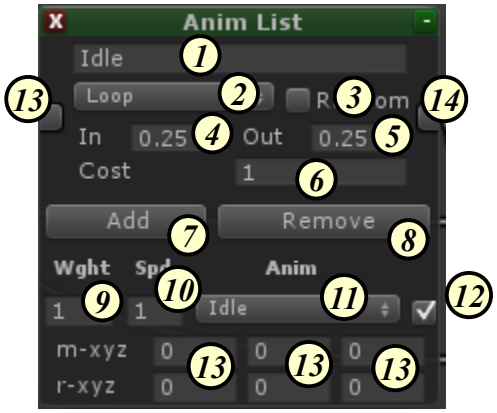
State Machine Graph Nodes



Blend Graph State Node

The Blend Graph State Node is used to define a State in the State Machine that will contain an internal Blend Graph that will blend between animations while this state is active. The internal Blend Graph can be edited by clicking on the "Edit" button.

1. ***State Name***
   This is the name of this State, which is used to identify this state in the state machine.
2. ***Wrap Mode***
   This is the wrap mode to use when playing the animations contained in the blend graph.
3. ***Sync Anims***
   This indicates if all animations in this blend graph should have their playback speed synced in order to allow them to blend better together. This works exactly like Animation.SyncLayer, except only applies to animations in this layer.
4. ***In Time***
   When transitioning into this state, this is the amount of time (in seconds) that it should take to fully blend into this states animations.
5. ***Out Time***
   When transitioning away from this state, this is the amount of time (in seconds) that it should take to fully blend out of this states animations.
6. ***Path Cost***
   This is the cost used for this state when the State Machine attempts to create a path between states. This can happen either because a state has finished playing, or the SageController has requested to transition to another state. Set this value very high to make it less likely to automatically transition through this state.
7. ***Edit Blend Graph***
   This will open the Blend Graph for this state in the Graph Area.
8. ***In Port***
   This is the in port for this state. Any links into this port indicate any potential transitions into this state.
9. ***Out Port***
   This is the out port for this state. Any links out of this port indicate any potential transitions out of this state.



Anim List State Node

The Anim List Graph State Node is used to define a State in the State Machine that contains a list of animations that will be randomly selected when this state is activated.

1. ***State Name***
   This is the name of this State.
2. ***Wrap Mode***
   This is the wrap mode to use when playing an animation.
3. **Random on Loop**
   This indicates if when set to loop, that a new animation in the list will randomly be chosen when one loop is finished.
4. ***In Time***
   When transitioning into this state, this is the amount of time (in seconds) that it should take to fully blend into this states animations.
5. ***Out Time***
   When transitioning away from this state, this is the amount of time (in seconds) that it should take to fully blend out of this states animations.
6. ***Path Cost***
   This is the cost used for this state when the State Machine attempts to create a path between states.
7. ***Add Animation***
   This will add a new animation to the list of potential animations.
8. ***Remove Animation***
   This will remove the last animation in the list of animations.
9. ***Chance***
   This is the chance an animation will be played when this state is entered.
10. ***Speed***
    This is the speed at which to play an animation.
11. ***Animation***
    This is an animation to play.

12. Apply Movement

This indicates while this animation is active, if movement should be applied to the character.

13. Movement Amount

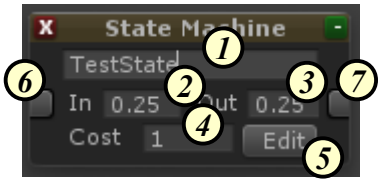
This is the amount of movement to apply to the character per second, multiplied by the playback speed of the animation. This movement also blends in and out with the animation.“m-xyz” is the movement to apply. “r-xyz” is the rotation to apply.

14. In Port

This is the in port for this state. Any links into this port indicate any potential transitions into this state.

15. Out Port

This is the out port for this state. Any links out of this port indicate any potential transitions out of this state.



State Machine State Node

The State Machine State Node is used to define a State in the State Machine that contains nested state machine that will run while this state is active.

1. ***State Name***

   This is the name of this State.

2. ***In Time***

   When transitioning into this state, this is the amount of time (in seconds) that it should take to fully blend into this states animations.

3. ***Out Time***

   When transitioning away from this state, this is the amount of time (in seconds) that it should take to fully blend out of this states animations.

4. ***Path Cost***

   This is the cost used for this state when the State Machine attempts to create a path between states.

5. ***Edit State Machine Graph***

   This will open the State Machine for this state in the Graph Area.

6. ***In Port***

   This is the in port for this state. Any links into this port indicate any potential transitions into this state.

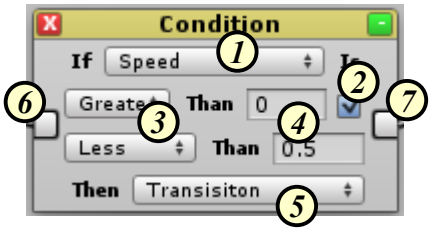
7. ***Out Port***

   This is the out port for this state. Any links out of this port indicate any potential transitions out of this state.

Parallel State Machine Node

The Parallel State Machine Node is used to define a Child State machine in the State Machine that will can run in parallel to the parent state machine, and will get blended in and out with the parent state machine as well.

1. ***State Machine Name***
   This is the name of this State Machine.
2. ***Edit State Machine Graph***
   This will open the State Machine for this state in the Graph Area.



Condition Node

This node is used to automatically trigger transitions between two states. As well, it can also be used to prevent automatic transitions between states. If the condition defined is not true, a transition is not possible. If it is setup to transition when it's condition becomes true, it will automatically trigger a transition between the current state connected to the In Port to the state with the lowest Cost connected to its Out Port.

Also, if this Node has no Input links, it will be considered a “Forced Condition” node. In this mode, if it is setup to transition when true, it will force a transition to the state node linked through it's output with the lowest “In Cost”

1. ***Variable***
   This is the variable to use in the comparison.
2. ***Compound Comparison Enabled***
   This indicates if this is a compound comparison, such as “Variable > 0.0 and Variable <= 0.5”.
3. ***Comparison Type***
   This is the comparison type to use, such as “Less Than” or “Equal”.
4. ***Comparison Value***
   This is the constant value to use in the comparison. Such as “Speed Less Than 0.25”.

5. ***Reaction***
   This is what action should be done when the condition becomes true. This action will ONLY be performed if a state currently connected to the In Port is the active state.
6. ***In Port***
   The links into this port indicate which transitions are controlled by this condition.
7. ***Out Port***
   This links out of this port indicate which transitions are controlled by this condition.

Trigger Node

This node is used to trigger transitions between two states, initiated from a message sent to the SageController game object. For example, to trigger this node, you would need to call the following code:
"gameObject.SendMessage("Trigger", "StartFire");

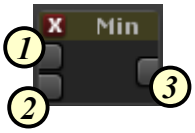
1. ***Trigger Name***
   This is the name of the trigger that will cause this node to initiate a transition.
2. ***Force Transition***
   Indicates if, when trigger, the transition initiated should be forced, or only use normal transition paths.
3. ***Out Port***
   This links out of this port indicate which transitions are controlled by this trigger.

Blend Graph Nodes

Variable Node

This node is used to simply output what the current value of the selected variable is, between 0.0 and 1.0. For example, if you are using the Speed variable, and the valid range is 1.0 to 3.0. If the value is currently 1.5, this node will output 0.25.

1. ***Variable***
   This is the variable to output from this Variable Node.
2. ***Value Port***
   This port will output the current normalized value of the selected variable, from 0.0 to 1.0.



Min Node

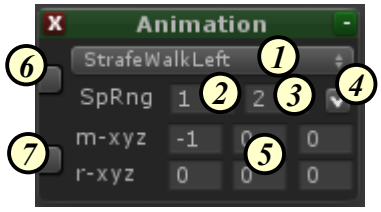
This node takes two input floats, and will output the minimum of the two values.

1. ***Value 1 Port***
   The first input value to compare.
2. ***Value 2 Port***
   The second input value to compare.
3. ***Min Value Port***
   The minimum value of the two input ports.

Max Node

This node takes two input floats, and will output the maximum of the two values.

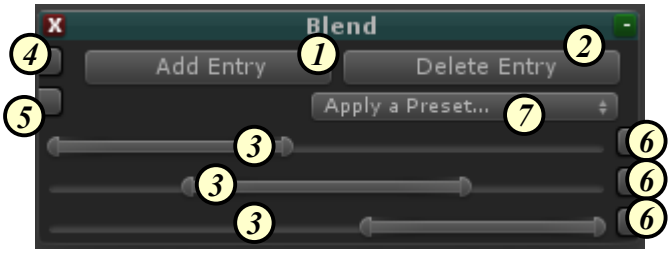
1. ***Value 1 Port***
   The first input value to compare.
2. ***Value 2 Port***
   The second input value to compare.
3. ***Min Value Port***
   The maximum value of the two input ports.



Animation Node

This node is used to play an animation in a blend graph. While the blend graph is active, this animation will continually play, using the passed in weight as the weight of the animation while being played back.

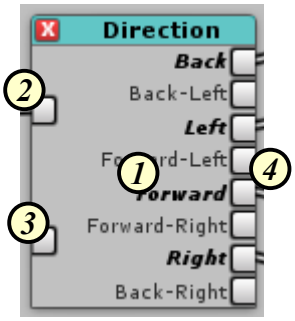
1. ***Animation***
   This is the animation to play with this node.
2. ***Speed***
   This is the speed of playback for the animation in this node.
3. ***Max Speed***
   This is the max playback speed of the animation to lerp to based off of the input speed port. (Only visible when there is an input connected to Speed)
4. ***Apply Movement***
   This indicates while this animation is active, if movement should be applied to the character.
5. ***Movement Amount***
   This is the amount of movement to apply to the character per second, multiplied by the playback speed of the animation. This movement also blends in and out with the animation. “m-xyz” is the movement to apply. “r-xyz” is the rotation to apply.
6. ***Weight Port***
   This port inputs what the current weight of the animation playback should be.
7. ***Speed Port***
   This port controls the current playback speed. As the input goes from 0.0 to 1.0, the playback speed goes from Speed to Max Speed.



Blend Node

This node is used to define how a blend graph will blend between an input value and multiple output values. The node contains most of the "work" that a blend graph does.

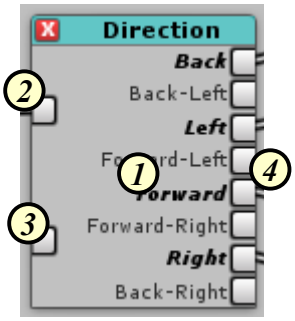
1. ***Add Blend Entry***
   This adds a blend entry to this blend node.
2. ***Remove Blend Entry***
   This removes the last blend entry from this blend node.
3. ***Blend Range***
   This area defines how each blend entry blends with the other blend entries. The active area on each entry bar indicates it's current blend range. Any part of this area that doesn't overlap with another blend area, will output 1.0 whenever the Blend Source is in that section. When a blend area overlaps with another blend area, it will cross fade between entries. When the Blend Source is over an empty area in a blend entry, it would output 0.0. IMPORTANT: No more than two blend entrys should overlap at any given time!
4. ***Blend Source Port***
   This is the input float value which is used to blend between all blend entries.
5. ***Output Multiplier Port***
   This input float value is used to multiply the outputs of each blend entry. This is useful if you want to pass in 0.0 or any other value between 0.0 and 1.0 to cause the entire blend node to blend based off of another blend nodes output.
6. ***Blend Result Port***
   This is the output float value of the final blend result of a blend entry. From 0.0 to 1.0.
7. ***Apply Preset***
   This will allow you to quickly and easily apply a preset layout for blending to all entries of this blend node.



Direction Node

This node is used to define how a blend graph will blend directional animations. The node contains most of the "work" that a blend graph does for blending direction movement animations.

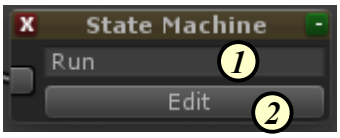
1. ***Blend Directions***
   This is the list of directions that can be output from this node. Bold Italic text indicates it is an active direction that will be blended for this direction node, normal text indicates inactive. If a direction output port is connected, it will use that to blend the direction.
2. ***Blend Source Port***
   This is the input float value which is used to blend between all directions. For the direction blend node, this indicates a direction/rotation where 0.0 is backwards, rotating counter clockwise, up to where 1.0 is backwards again.
3. ***Output Multiplier Port***
   This input float value is used to multiply the outputs of each blend entry. This is useful if you want to pass in 0.0 or any other value between 0.0 and 1.0 to cause the entire blend node to blend based off of another blend nodes output.
4. ***Blend Direction Output Ports***
   These ports are the ports that are used to output the blend results of this direction node. If you want your blend graph to play specific direction animations, all you need to do is connect to the proper output port to use that direction in the blend operation. Disconnected ports will automatically be ignored.



Blend Graph Node

This node is a nested blend graph that can be blended in and out just like an animation can.

1. ***State Machine Name***
   This is the name of this State Machine.
2. ***Edit Blend Graph***
   This will open the Blend Graph for this node in the Graph Area.



State Machine Node

This node is a nested state machine that can be blended in and out just like an animation can.

1. ***State Machine Name***
   This is the name of this State Machine.
2. ***Edit Blend Graph***
   This will open the Blend Graph for this node in the Graph Area.

Source Code License

A full source code license is available at an additional cost. Source code includes a full site license for Sage as well. Any questions about the source code can be answered by contacting us at the email address below.

If you have already purchased Sage and are interested in the Sage Source Code License, please send me an email at AlteredRealityEnt@yahoo.com with your invoice number (if bought from Unity Asset Store) or email (if bought from our Website), for a coupon for $75 off the Sage Source Code.

The Source Code License can be purchased here: www.alteredr.com/sage

Support

Twitter

twitter.com/AlteredR

Email

AlteredRealityEnt@yahoo.com

Website

www.alteredr.com

Forums

forums.alteredr.com

Credits

Lead Programmer and Architect

Andrew Thayer

Special Thanks

Andrew Taylor
Jim Weinhart
Michael Howard
Leslie Thayer
Caiden Thayer

Software Disclaimer