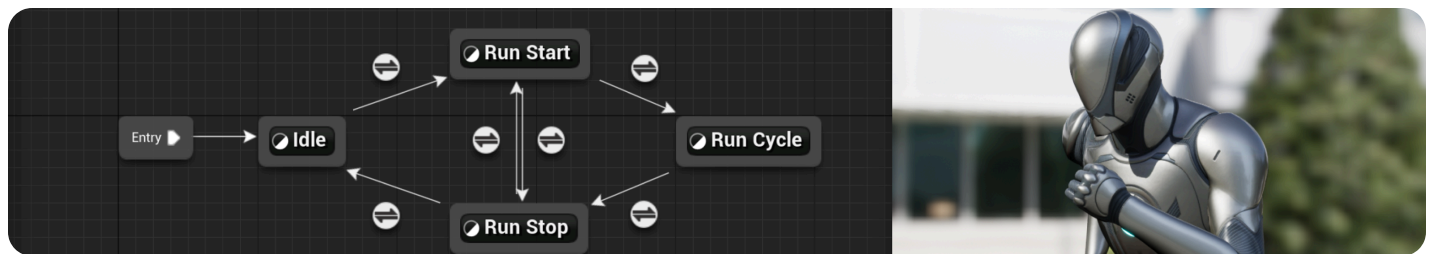


# State Machines

Create logic-based branching animation by using State Machines.



**State Machines** are modular systems you can build in **Animation Blueprints** in order to define certain animations that can play, and when they are allowed to play. Primarily, this type of system is used to correlate animations to movement states on your characters, such as idling, walking, running, and jumping. With State Machines, you will be able to create **states**, define animations to play in those states, and create various types of **transitions** to control when to switch to other states. This makes it easier to create complex animation blending without having to use an overly complicated Anim Graph.

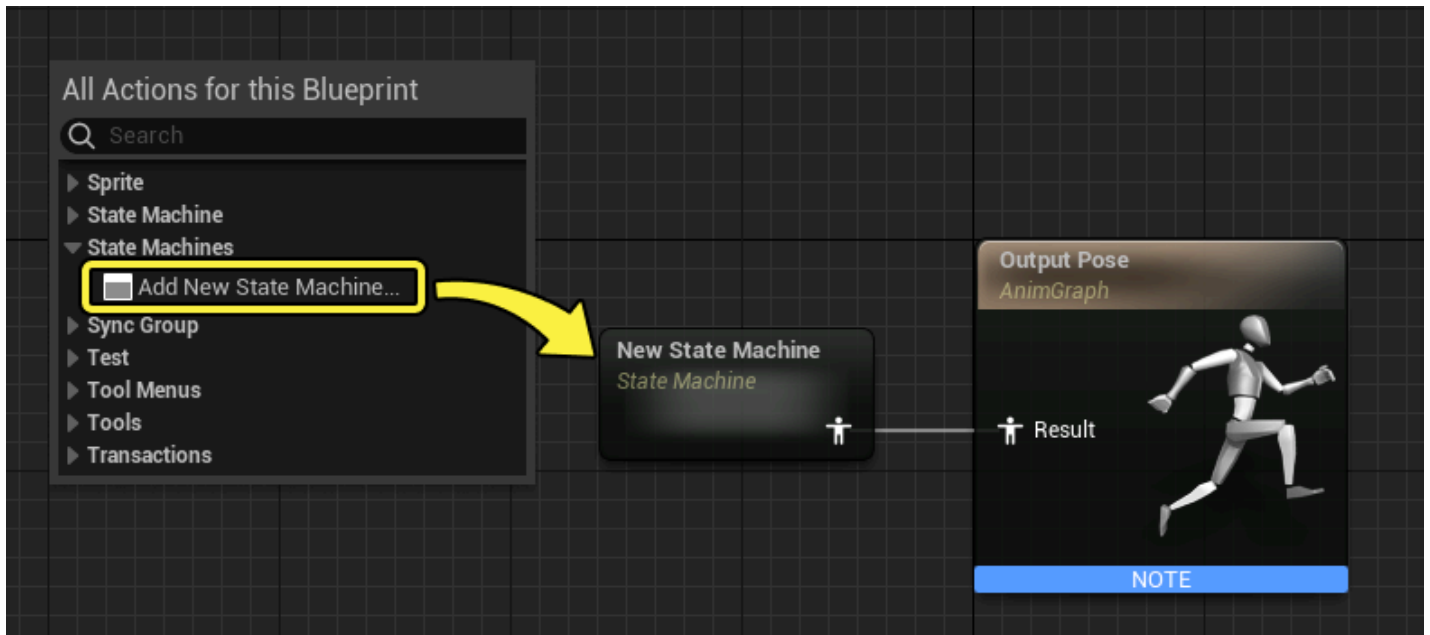
This document provides an overview of how to create and use State Machines, states, and transitions in Animation Blueprints.

## Prerequisites

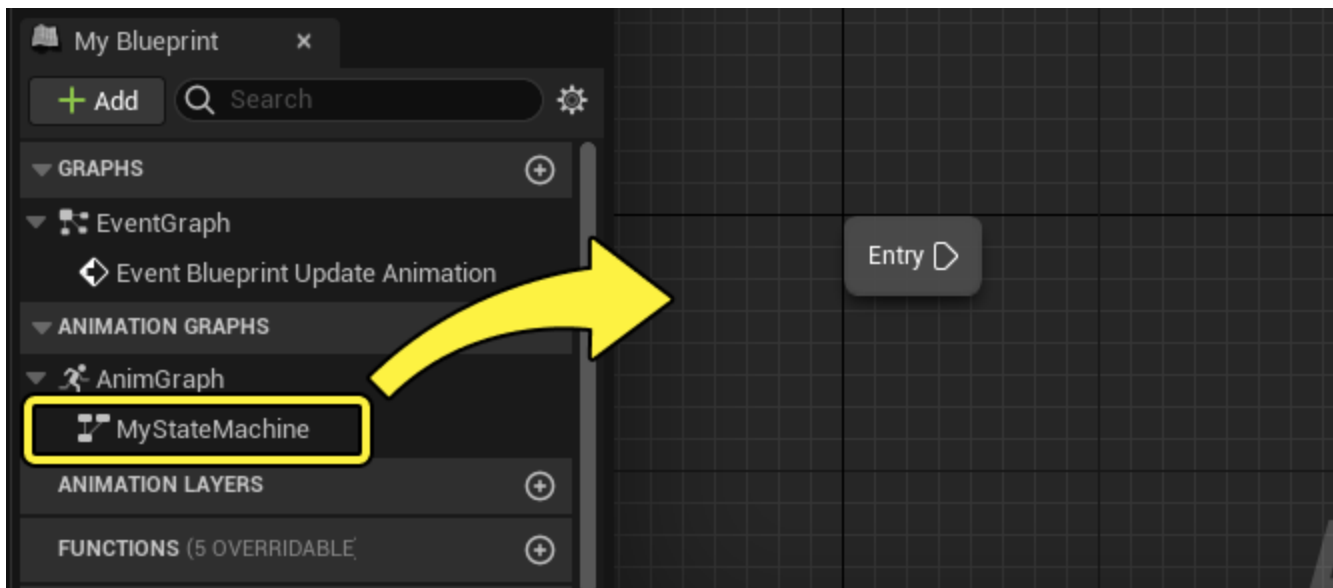
- State Machines are created within [Animation Blueprints](#), therefore you should have an understanding of how to use Animation Blueprints and their [interface](#).
- Your project contains a character with a movement component so that you can build states that react to input. The [Third Person Template](#) can be used if you do not have one.

# Creation and Setup

State Machines are created within the [Anim Graph](#). To create one, right-click in the **Anim Graph** and select **State Machines > Add New State Machine**. Connect it to the **Output Pose**.



State Machines are subgraphs within the Anim Graph, therefore you can see the State Machine graph within the **My Blueprint** panel. Double-click it to open the State Machine.

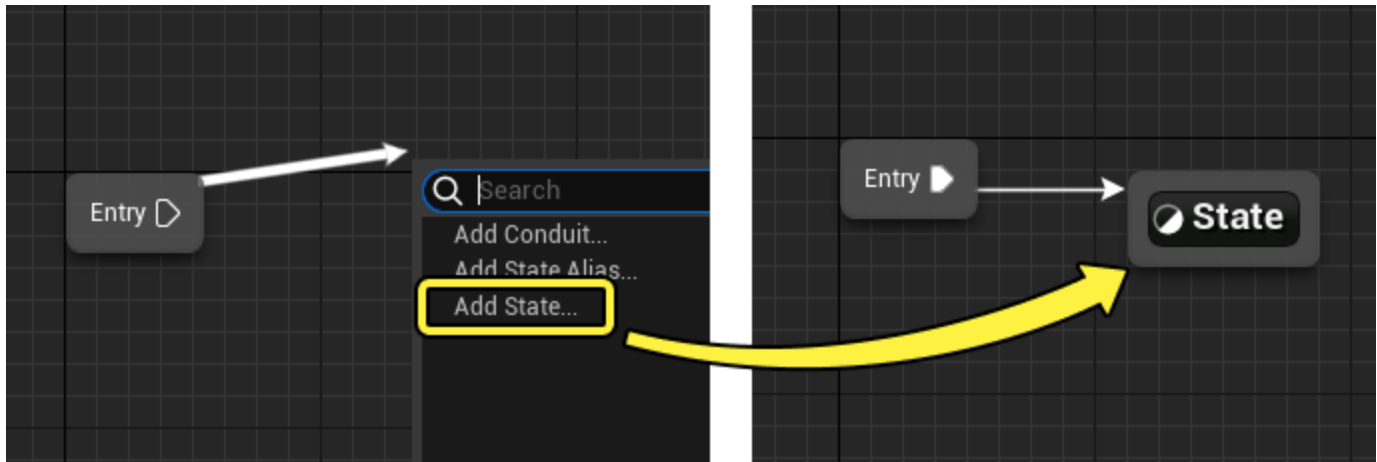


 You can also double-click the State Machine node in the Anim Graph to open it.

# Entry Point

All State Machines begin with an **entry** point, which is typically used to define the **default state**. In most common locomotion setups, this would be the character idle state.

To create the default state, click and drag the **entry** output pin and release the mouse, which will expose the context menu. Select **Add State**. This will create the new state and connect it to the entry output, making this state active by default.



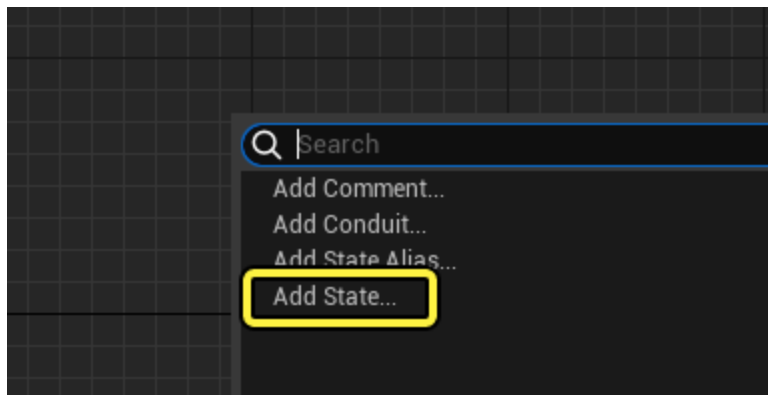
## States

States are organized sub-sections within a State Machine that can transition to and from each other regularly. States themselves contain their own Anim Graph layer, and can contain any kind of animation logic. For example, an idle state may just contain a character's idle animation, whereas a weapon state may contain additional logic for shooting and aiming. Whatever logic is used, the purpose of a state is to produce a final animation or pose unique to that state.

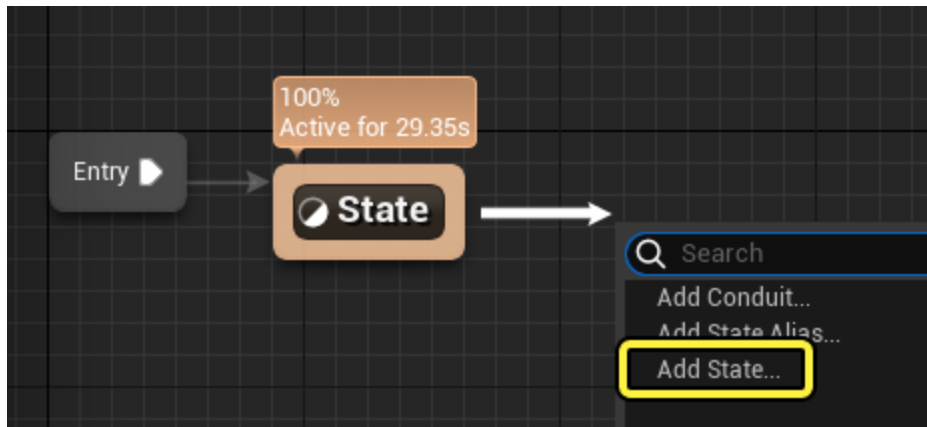
## Creating States

States can be created in the following ways:

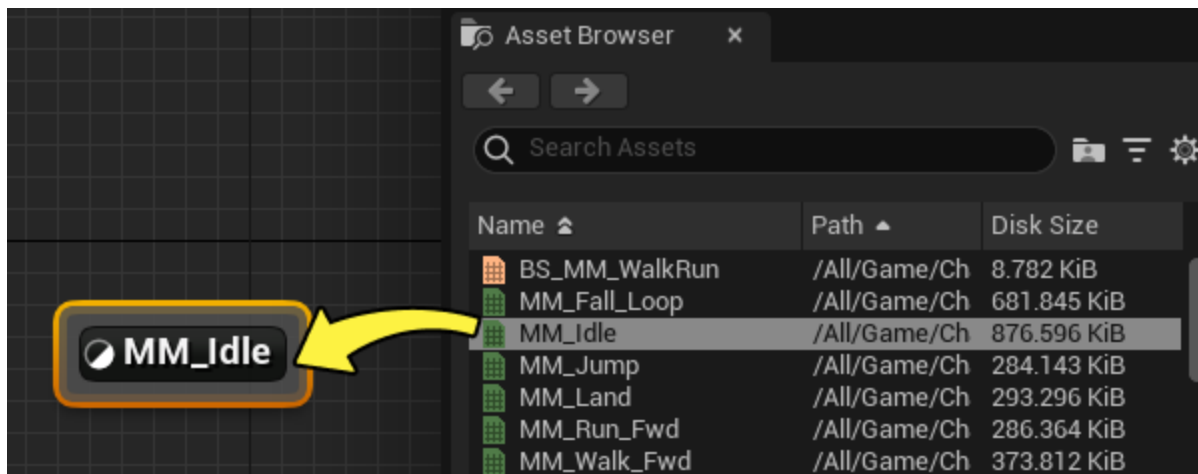
- Right-click in the State Machine graph and select **Add State**.



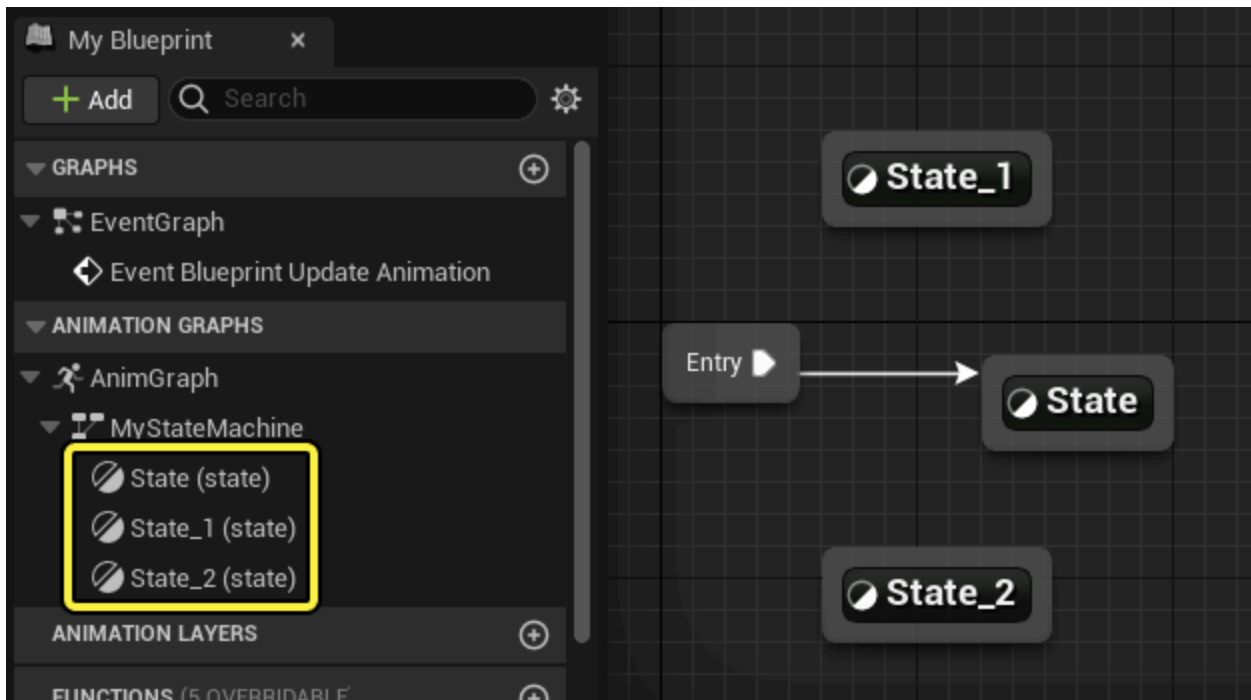
- Click and drag off of the border of a state (or entry output), then release the mouse and select **Add State**. This will also connect it to the previous state with a [transition](#).



- Drag an **Animation Asset** into the State Machine graph from the **Content Browser** or **Asset Browser**. This also adds the animation to the state and connects it to its **Output Pose**.

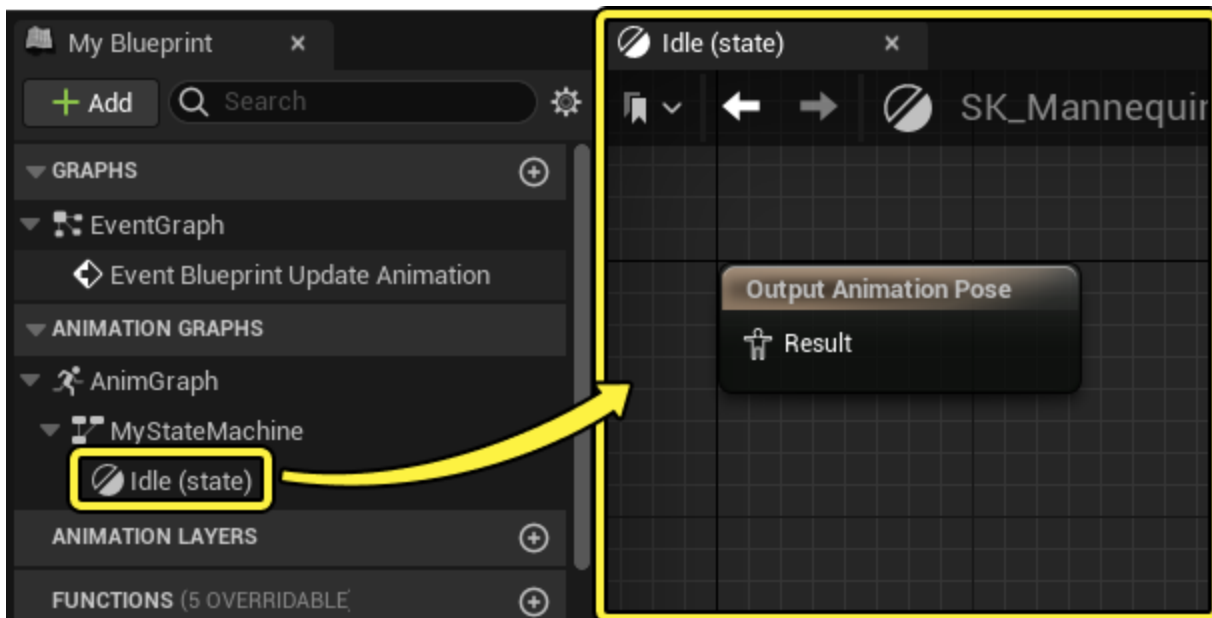


State Machines can have as many states as needed, and they also display as subgraphs under the State Machine.

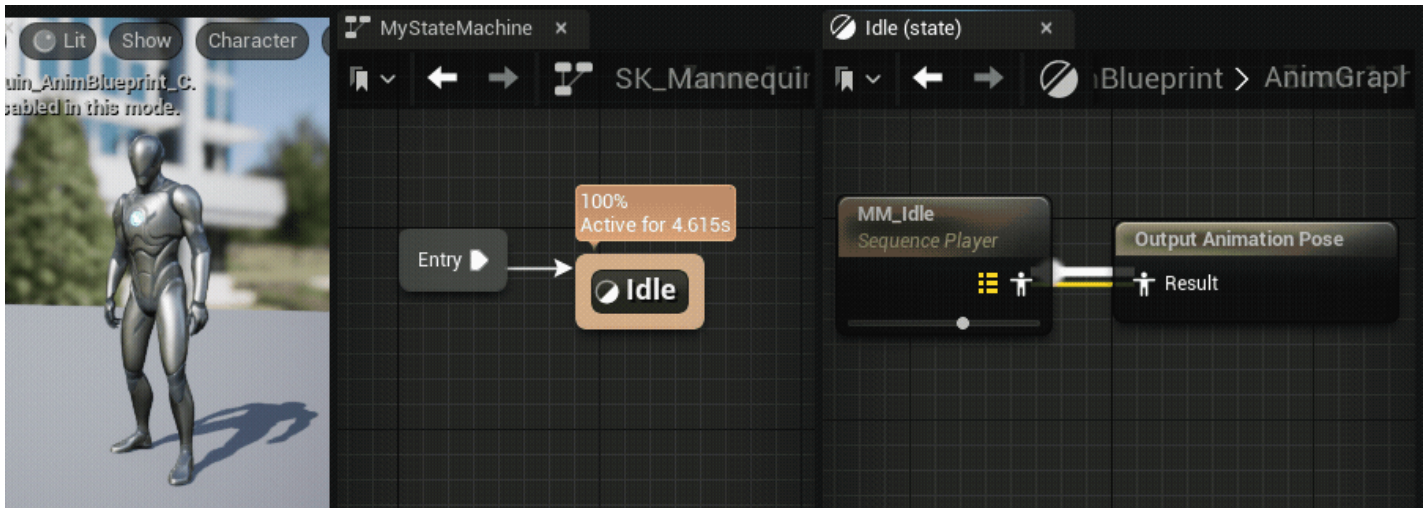


## Editing States

To view the internal operation of a state, you can either double-click it in the **My Blueprint** panel, or double-click the node itself in the **State Machine** graph. This will open the state.

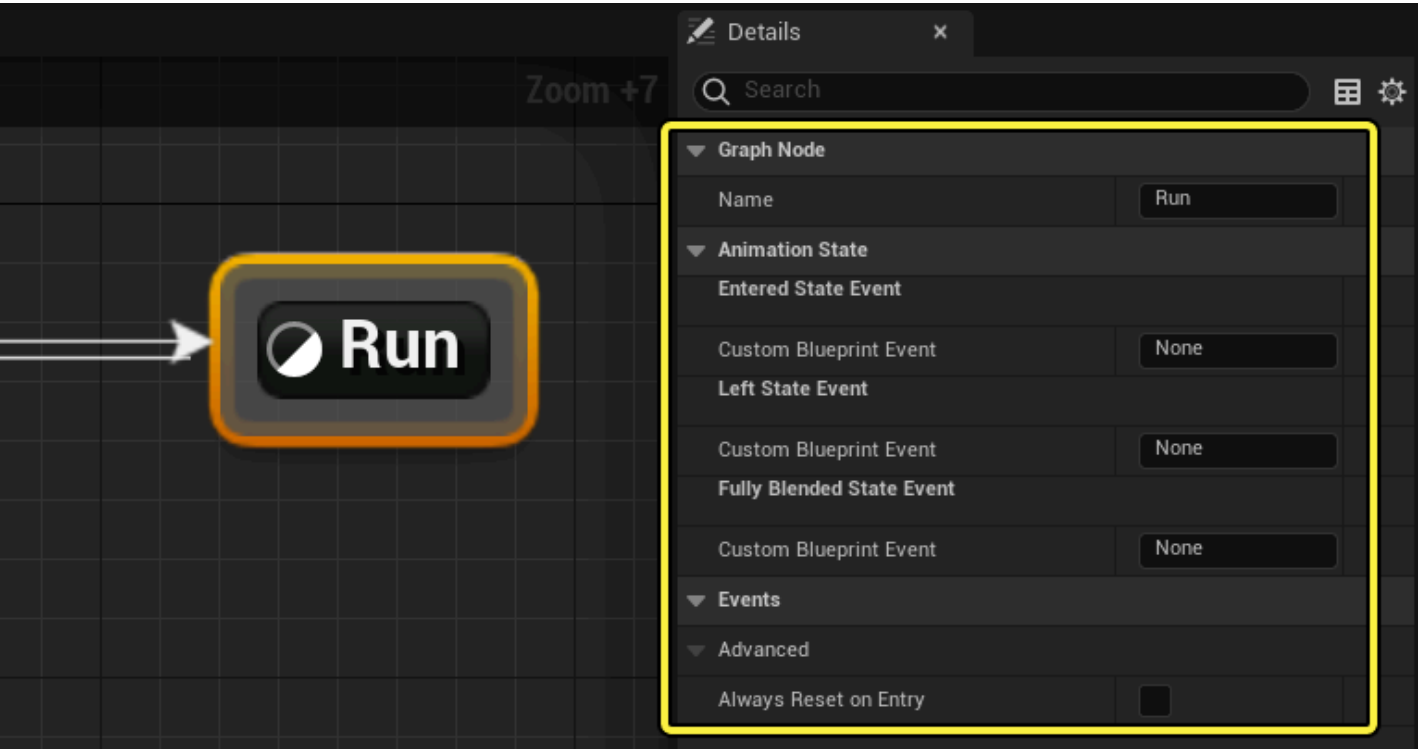


Like Anim Graphs, states contain a final **Output Pose** node to connect your animation logic to. When the state is active, this logic will execute. When a different state is active, this logic will no longer execute. In this idle state example, an idle animation is connected to the Output Pose. When this state is active, this resulting animation will play.

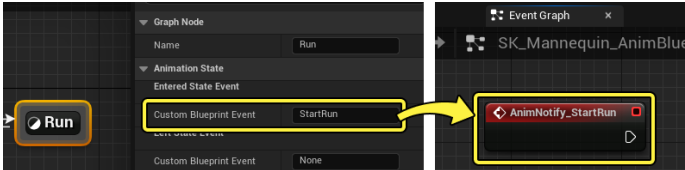


## State Properties

When a state is selected, you can view and edit the following properties in the **Details** panel.

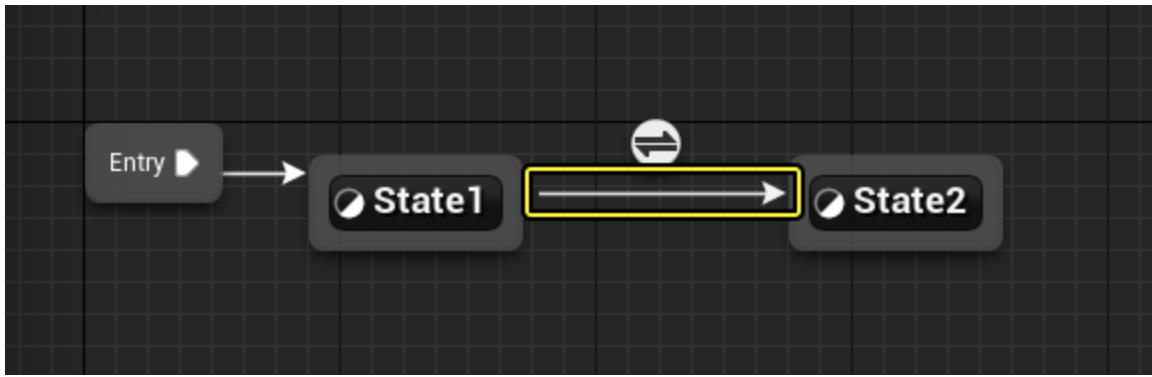


Name	Description
<b>Name</b>	The name of the selected state.
<b>Entered State Event (Custom Blueprint Event)</b>	Creates a <a href="#">Skeleton Notify</a> with the name used in the <b>Custom Blueprint Event</b> field. This notify will execute when the state becomes active and starts

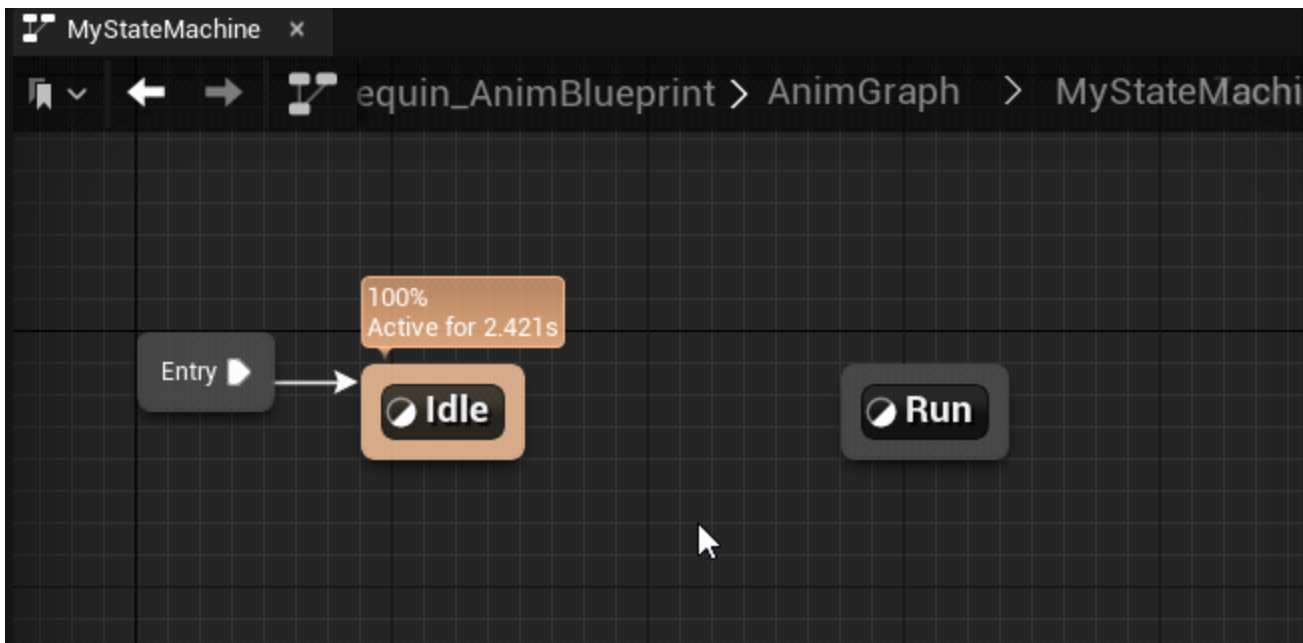
Name	Description
	<p>to transition. As with normal Skeleton Notifies, you can access the event by creating it in the Animation Blueprint's <b>Event Graph</b>.</p> 
<b>Left State Event (Custom Blueprint Event)</b>	<p>Creates a Skeleton Notify with the name used in the <b>Custom Blueprint Event</b> field. This notify will execute when starting to blend to another state.</p>
<b>Fully Blended State Event (Custom Blueprint Event)</b>	<p>Creates a Skeleton Notify with the name used in the <b>Custom Blueprint Event</b> field. This notify will execute when this state is fully blended to.</p>
<b>Always Reset on Entry</b>	<p><b>Enabling</b> this will cause all animations within this state to re-initialize to their default values. In most cases, this means the following:</p> <ul style="list-style-type: none"> <li>• Sequence players will restart at the animation start time.</li> <li>• Properties will initialize at their default values.</li> </ul> <p>If <b>disabled</b>, then all animations and their properties will maintain their previous playback state and other properties upon leaving and then returning to this state. In other words, the animations will "pick up where they left off".</p>

## Transitions

To control which states can blend to another, you can create **transitions**, which are links between states that define the structure of your State Machine.

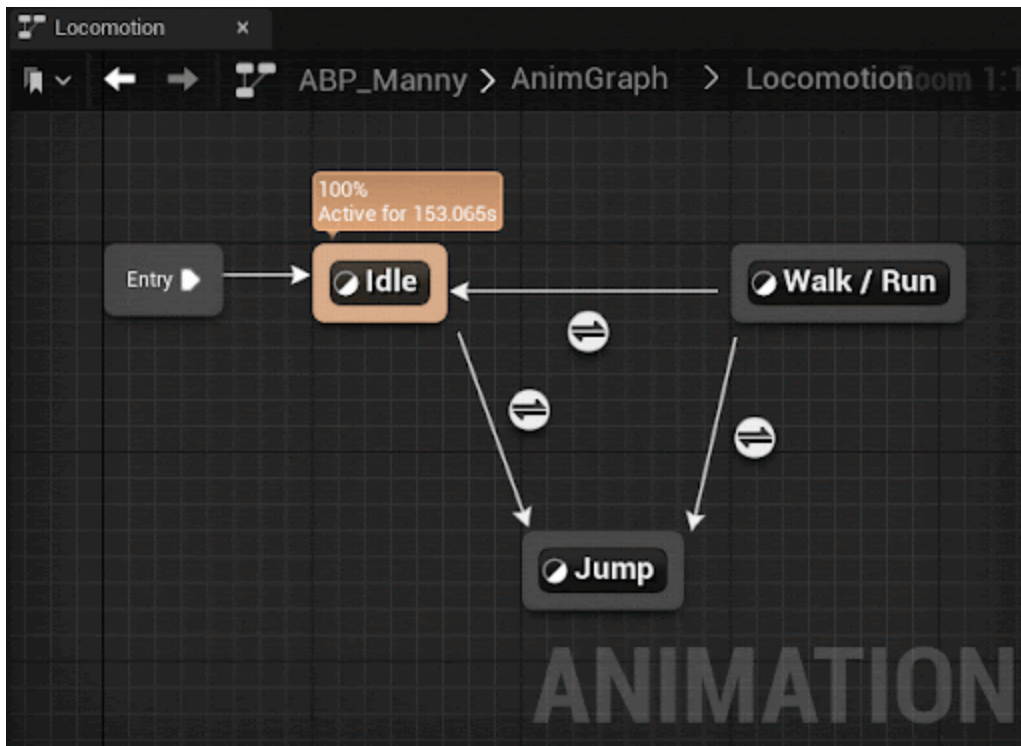


To create a transition, drag from a state border to another state. In this example the **Idle state** is connected bi-directionally to the **Run state**, which is a common setup for basic locomotion State Machines. Transitions are single-direction, so if two states are intended to transition back and forth, you need to create a transition for each direction.



You can also rebind existing transition logic by selecting a transition node and dragging it to a different state. You can rebind multiple transition nodes at once by dragging the transition arrow to a new binding.





To learn more about **transitions** and **transition rules**, refer to the [Transitions](#) page.

## Transition Rules

Blend between states in your State Machine using Transitions.

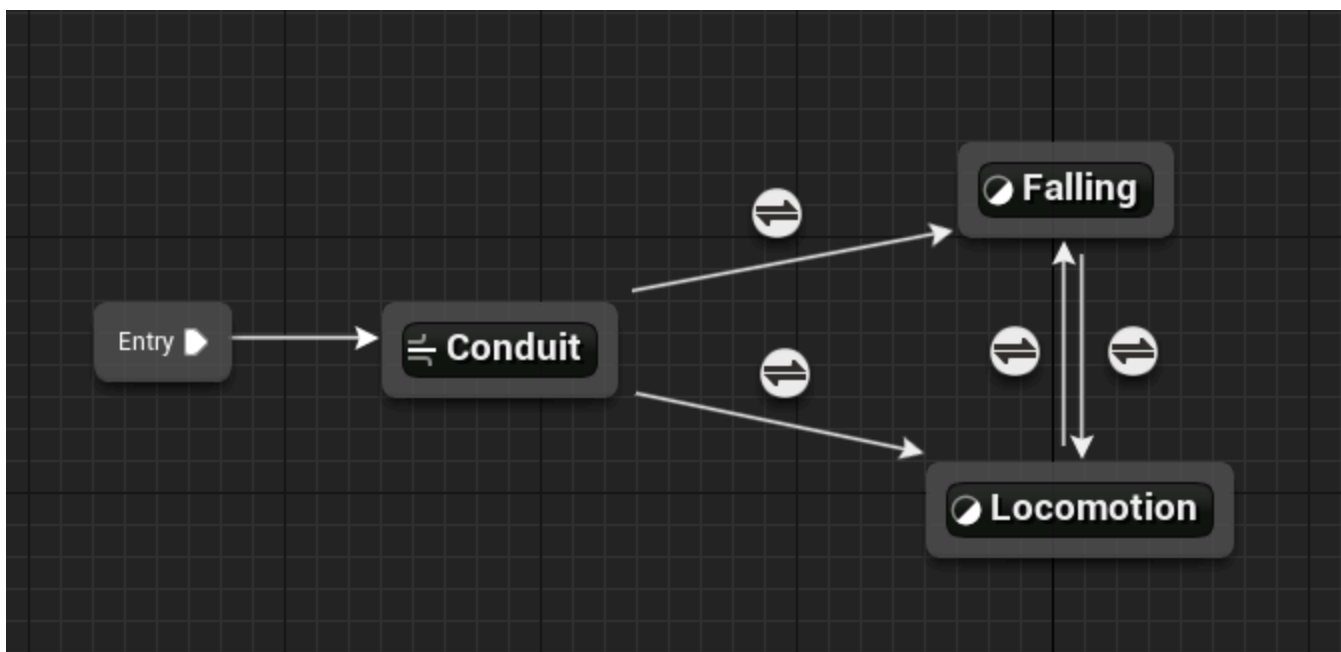
# Conduits

While ordinary transitions can be used for 1-to-1 transitions between states, **conduits** can be used to create 1-to-many, many-to-1, or many-to-many transitions. Because of this, conduits serve as a more advanced and shareable transition resource.

To create a conduit, right-click in the State Machine graph and select **Add Conduit**.



There are several ways you can use conduits. One example might be to use them to diverge your State Machine's entry point. You can then use transitions from the conduit to select which state should start as the default. This example can be useful when re-initializing a State Machine if you were overwriting it with another animation, such as an [Animation Montage](#).



The above example requires **Allow Conduit Entry States** to be enabled in the [State Machine Details panel](#).

Conduits contain their own [transition rules](#), which can be located by double-clicking on the conduit node, or by opening the conduit graph from the My Blueprint panel. By default, conduit transitions rules will return false. In most cases you may just want to enable **Can**

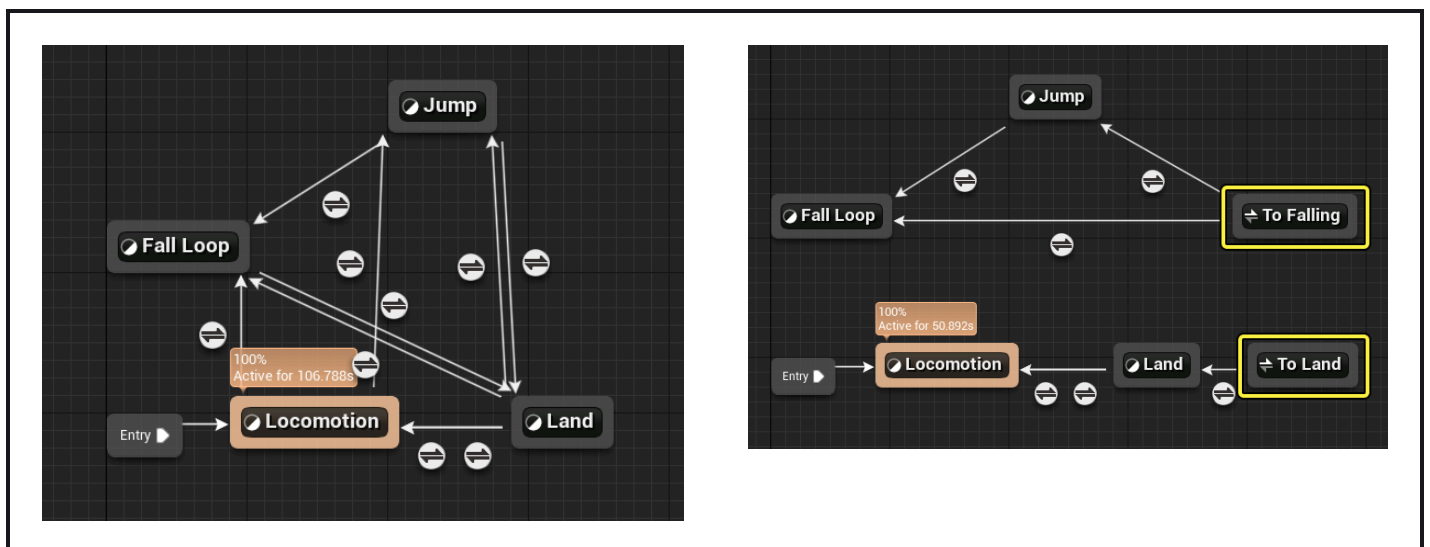
**Enter Transition**, and create transition rule logic on the individual transitions in and out of the conduit.



Refer to the [Transition Rules](#) page for more information on transitions and transition rules.

## State Alias

As you build more complicated State Machines with many states and ways to transition between them, you may want to use **state aliases** to improve your graph. State aliases are shortcut-type nodes you can add to your State Machine to reduce line clutter, consolidate transitions, and improve the readability of your graph.



State Machine with no alias usage

State Machine using aliases

To create a state alias, right-click in the **State Machine** graph and select **Add State Alias**.

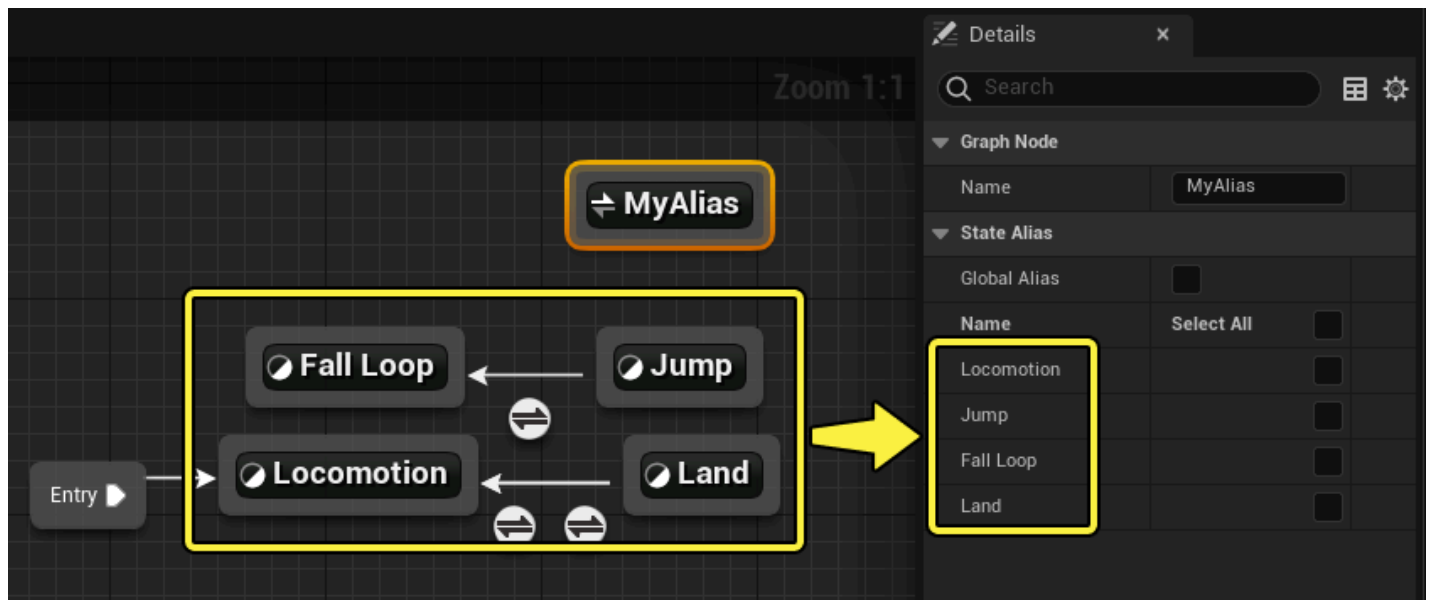


State aliases work by defining which states can transition into it, then connecting the alias to other states using the normal transition method. Click the state alias node, and in the **Details** panel you can observe the following:

- Each state within your State Machine is listed as a property. If you enable that state, it causes that state to adopt the transitions and rules that you make from the alias to other states. In other words, this is where you define which states are "coming into" the alias.
- Enabling **Global Alias** makes all states come into the alias. Although you can enable all listed states which causes the same behavior, enabling Global Alias will also include any new states created later.

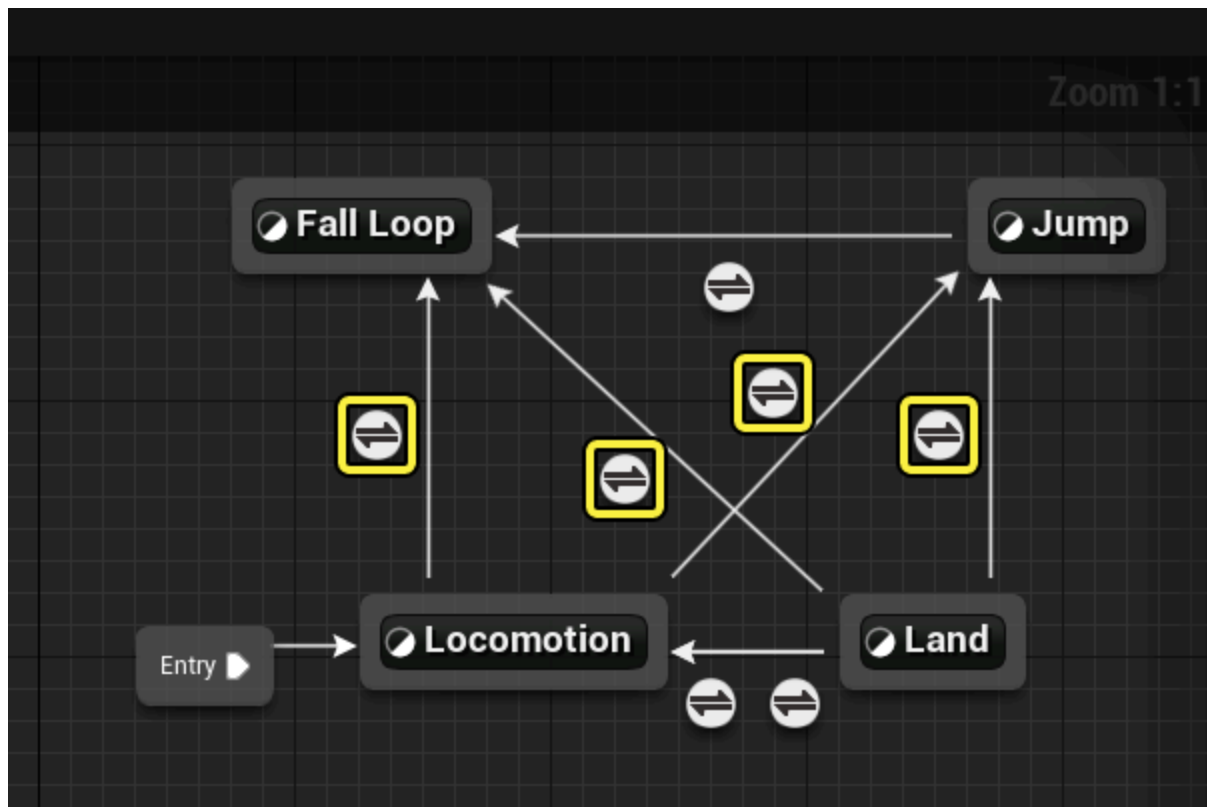


Global Alias is best used in a limited way with single-frame input and finite-duration states, such as interaction, attacks, or other similar animations. Using Global Alias for states with indefinite lengths may require additional complicated logic between all other state transitions to ensure your other states are not always transitioning to it.



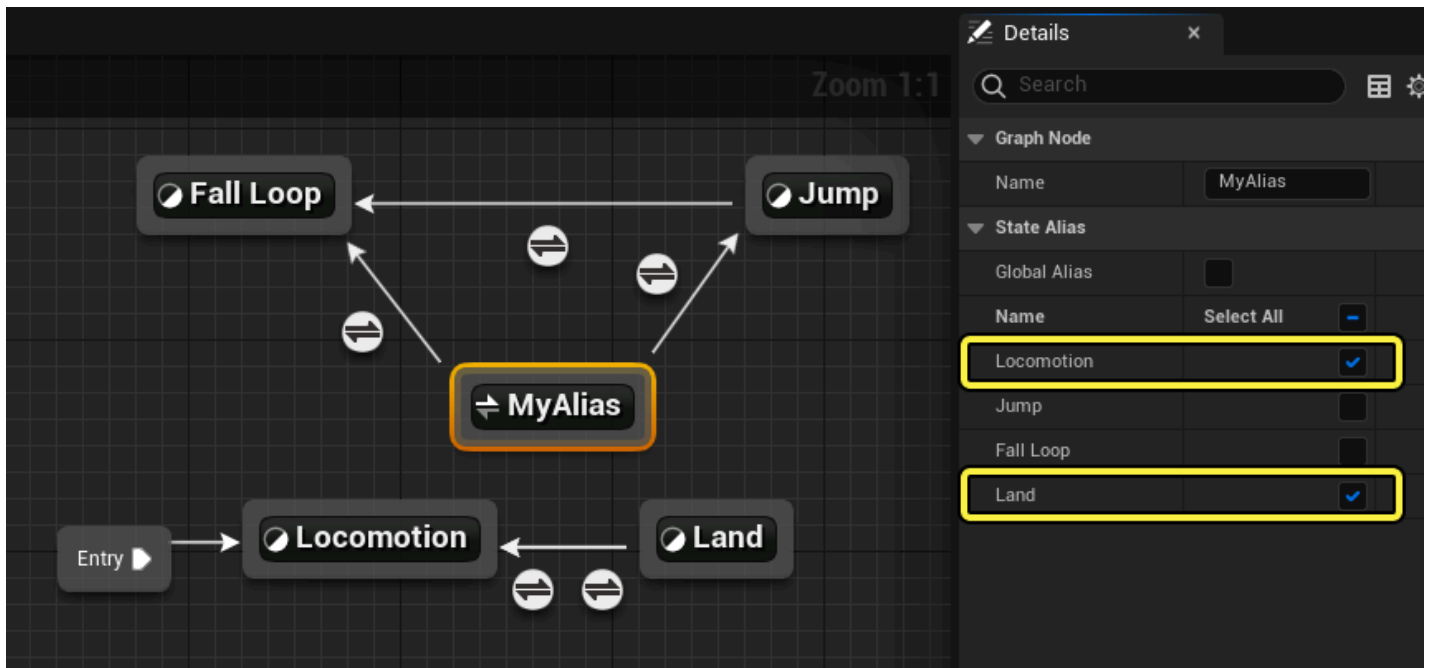
## Alias Example

In this example, a somewhat simple State Machine requires the **land** and **locomotion** states to transition to both the **jump** and **fall loop** states. Four transitions in total are being used, each with their own transition rules.



State aliases can be used to clean up this graph. To achieve the same effect, you can do the following:

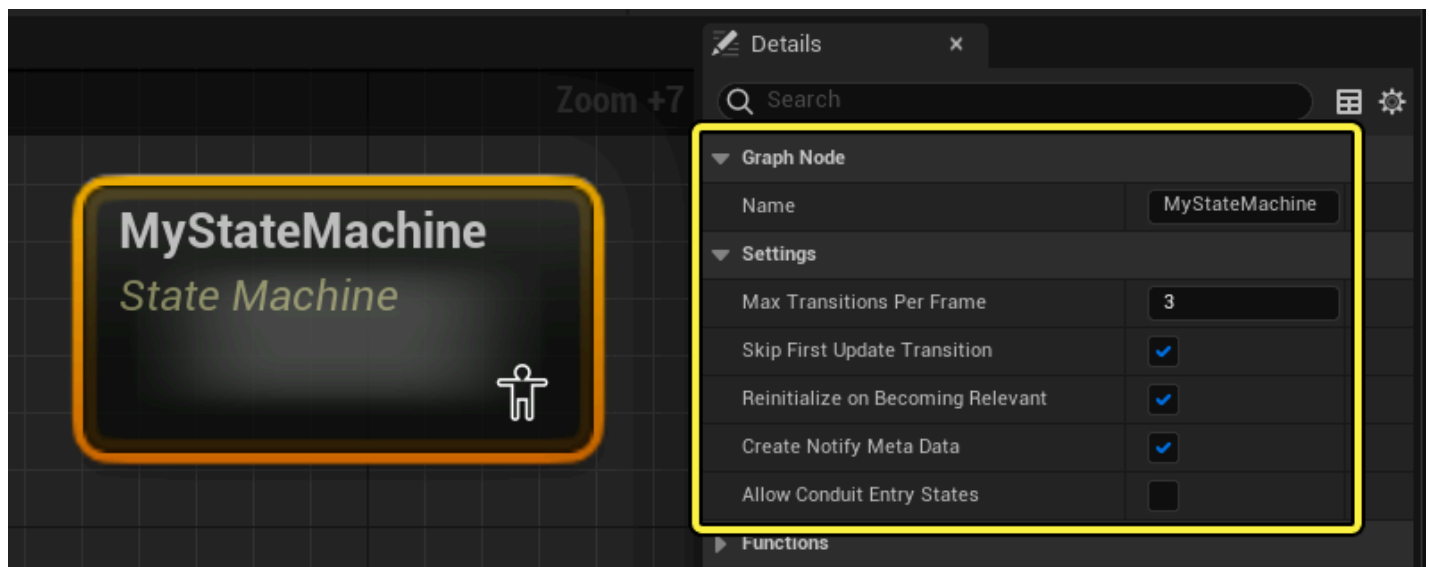
- Create a state alias and transition it to both the **fall loop** and **jump** states.
- Select the state alias and enable **locomotion** and **land**.



Because state aliases consolidate transitions from all enabled states, using state aliases means that these states share the same transition rules and properties. If you want certain transitions to have different rules, blend durations, or other properties, then you should create unique transitions for those states instead.

## State Machine Properties

State Machines contain the following properties in the **Details** panel.



Name	Description
<b>Name</b>	The name of the selected State Machine.
<b>Max Transitions Per Frame</b>	<p>This number defines how many transitions or <b>decisions</b> can occur in a single frame or tick. If your state machine has many states and transitions where more than one transition can be true at a given time, you may want to set this number to <b>1</b>. This makes it so that only one decision can be made at a time, preventing competing decisions and transitions.</p>
<b>Skip First Update Transition</b>	<p>When a State Machine becomes <a href="#">relevant</a>, it initializes into the default state connected to the <b>Entry</b> point. At that point, normal State Machine processes begin and any valid transitions are taken.</p> <ul style="list-style-type: none"> <li>• If this property is enabled, then any non-default states that are valid transition targets upon initialization will be immediately transitioned to.</li> <li>• If disabled, then any valid transition targets will blend normally.</li> </ul> <p>This is to allow for any non-default states to behave as if they were the default state, if that is desired behavior. For example, if you have a simple idle and run State Machine, where <b>idle</b> is the default state and can transition to <b>run</b>. If <b>Skip First Update Transition</b> is enabled, and if the run transition rule is <b>true</b> at the time of the State Machine's</p>

Name	Description
	initialization, then run will be initialized and blended to 100% immediately.
<b>Reinitialize on Becoming Relevant</b>	Enabling this reinitializes the first entered state when the State Machine becomes relevant. This setting operates similarly to the per-state property <b>Always Reset on Entry</b> , but only resets the first initialized state entered.
<b>Create Notify Meta Data</b>	When using <a href="#">Animation Notify Functions</a> in your transition rules, enabling this allows for all relevant data to be sent to these notify functions. If this is disabled, then none of the notify functions will work.
<b>Allow Conduit Entry States</b>	Enabling this allows <a href="#">conduits</a> to be used as entry states, allowing for variable default states depending on the conduit's transition rules.