Action

Action List

Action Lists let you manage groups of actions. You can set them to run in parallel, series or just a single random action.

Parameters:

- **runType**: Set to serial to run each action after the previous finishes, parallel to run all actions at the same time, or single random to pick a random action to run.

Add Target To List Action

Adds the targeted GameObject to a list.

Parameters:

- **list**: The list the GameObject will be added to
- **target**: The GameObject to add to the list.
- **allowDuplicates**: Should a GameObject have the ability to be added to the list more than once?

Apply Force Action

Applies an amount of force to the target, potentially every fixed update.

Parameters:

- **target**: The object(s) to apply force to.
- **forceAmount**: The amount of force to apply along each axis.
- **forceMode**: The force mode to use for 3d operations.
- **forceMode2D**: The force mode to use for 2d operations.
- **continuous**: If true, this will apply force every fixed update until stopped.

Apply Torque Action

Applies an amount of torque to the target, potentially every fixed update.

Parameters:

- **target**: The object(s) to apply torque to.
- **torqueAmount**: The amount of torque to apply along each axis.
- **forceMode**: The force mode to use for 3d operations.
- **forceMode2D**: The force mode to use for 2d operations.
- **continuous**: If true, this will apply torque every fixed update until stopped.

Axis Input Action

Forwards input axis events to a camera or player controller.

Parameters:

- **target**: The object where the CATEventManager to be used for events can be found.

- **inputAxisName**: The name of the Input Axis to use.
- **inputType**: The type of Event to fire on Input.

Calculate Nav Mesh Path Action

Calculates a path along the NavMesh and stores it in a Vector3 List Value.

Parameters:

- **start**: Start position for the path.
- **end**: End position for the path.
- **areaMask**: Defines which NavMesh areas are allowed. Defaults to all.
- **storeIn**: Vector3 List Value to store the resulting path in.
- **clear**: Clear the list first.

Change Quest Stage Action

Changes the current stage of this quest. The previously active stage will be set to inactive. Must be used in a quest.

Parameters:

- **stageName**: the name of the stage to set active.

Change State Action

Changes the current state of the targeted State Machine.

Parameters:

- **target**: State Machine to change the state of.
- **stateName**: Name of the state to change to.

Clear List Value Action

Clears the list pointed to by targetList.

Parameters:

- **targetList**: The list to be cleared (can be any type)

Clear Navigation Destination Action

Clears the destination of the targeted NavMeshAgent(s).

Parameters:

- **target**: The object(s) to set the destination of.

Clear Particle System Action

Clears all particles from the targeted particle system.

Parameters:

- **target**: the particle system to clear.

Complete Quest Action

Immediately mark this Quest as Complete. The Complete Actions from the active Steps, Stage, and Quest will be executed as well.

Conditional Action

Requires a Condition to be childed to it. If the Condition is met, the Actions underneath the Condition will Fire.

Controller Animation Action

Connects an animation parameter to a controller event.

Parameters:

- **controllerEventType**: Event to listen for on the controller.
- **animationTarget**: The object which will be animating.
- **parameterName**: The animation parameter name.
- **revertOnStop**: If true, the parameter will be set back to its initial value when this action is stopped.

Controller Event Action

Sends an event to a camera or player controller.

Parameters:

- **target**: The object where the CATEventManager to be used for events can be found.
- **inputValue**: The input value to send.
- **inputType**: The type of Event to fire.

Create Character Action

Creates a new character for the logged in account.

Parameters:

- **characterName**: Name of the new character.
- **saveTo**: An optional value to save the newly created character to.

Delayed Action

This action will run sub-actions after waiting a specified duration and will delay any sibling actions below it in a serial Action List

Parameters:

- **duration**: the duration in seconds to delay

Delete Character Action

Deletes a new character from the logged in account.

Parameters:

- **character**: The character to delete.

Destroy Object Action

Completely removes a GameObject from the scene.

Parameters:

- **target**: The object(s) to be destroyed.

Dont Destroy On Load Action

Sets a GameObject to not be destroyed on a scene change.

Parameters:

- **target**: The object(s) to not destroy on load.
- **singleton**: Make this object a singleton (only allow one object with this name)

Emit Particles Action

Immediately causes the targeted particle system to emit count particles.

Parameters:

- **target**: the particle system to emit from.
- **count**: the number of particles to emit.

Enable Gravity Action

Sets the gravity status of the target(s).

Parameters:

- **target**: The target(s) to change the gravity status of.
- **useGravity**: The gravity status to set.
- **revertOnStop**: Should the previous gravity status be reset when the Action stops?

Enable Particle Colliders Action

Enables or disables particle colliders in the targeted particle system.

Parameters:

- **target**: the particle system to enable or disable colliders on.
- **enable**: whether to enable or disable colliders.

Enable Particle Triggers Action

Enables or disables particle triggers in the targeted particle system.

Parameters:

- **target**: the particle system to enable or disable triggers on.
- **enable**: whether to enable or disable triggers.

Enable State Machine Action

Starts a targeted State Machine.

Parameters:

- **target**: The State Machines to start.
- **enable**: Whether to enable or disable the State Machine.

Enter Area Action

Immediately enter a new area.

Parameters:

- **areaID**: The ID of the new area.
- **realmID**: The ID of the realm the area is in.

Face Target Action

Rotates the target object towards the face target object.

Parameters:

- **target**: The object to rotate.
- **faceTarget**: The object that the target will rotate towards.
- **durationOrSpeed**: The amount of time the rotation will take, or the speed of the rotation.
- **pause**: The delay which affects the duration.
- **movementType**: Options for how the rotation will be executed.
- **curve**: How the object will rotate over a duration.
- **revert**: Should the rotation of the target revert on Stop?

Fade Screen Action

Fades the screen either in or out to a specified color or Texture.

Parameters:

- **fadeType**: Whether to fade in or out.
- **fadeColor**: The color to fade to.
- **fadeTexture**: The texture to fade to.
- **fadeDuration**: How long the fade should take in seconds.
- **curve**: The rate of the fade over the duration.
- **sortingLayer**: What sorting layer to have the fade happen on.

Fire Event Action

Fires an event defined in the Event Manager.

Parameters:

- **target**: The EventManager you want to fire the event from. If none, it will fire the event globally.
- **eventName**: The name of the event you wish to fire.

Get Characters Action

Retrieves the list of characters for the logged in account.

Parameters:

- **target**: The character list to save the results to.

Get List Length Action

Gets the length of the list pointed to by targetList and saves it in storeIn.

Parameters:

- **targetList**: The list to get the length of (can be any list type)
- **storeIn**: The value to save the list length into.

Get Particles Action

Store the particles in a list.

Parameters:

- **target**: the particle system.
- **storeIn**: the list to store the particles in.

Get Targets From List Action

Returns a List of Targets from a GameObject List.

Parameters:

- **list**: The List of GameObjects to pull from.
- **all**: Should all GameObjects in the list be pulled?
- **start**: The first index to pull a GameObject from.
- **end**: The last index to pull a GameObject from.
- **remove**: Should the pulled GameObjects be removed from the original list?
- **random**: Select targets at random from the range.
- **count**: Number of random targets to select if random is set.

Grant Quest Action

Check the Grant Conditions of the specified Quest, and if successful, it will grant the Quest to the player.

Parameters:

- **questName**: the name of the quest to grant.

Instanced Action

Spawns a prefab with an Action attached to it and runs that Action.

Parameters:

- **prefab**: The prefab with the Action attached.

Load Multiple State Machines Action

Loads all State Machines stored in a specified Storage Level.

Parameters:

- **storageLevel**: The level of storage to load the State Machines from.
- **template**: The State Machine template to set to the loaded State Machines.
- **parent**: Where to parent the loaded State Machines.

Load Quest Action

Add a quest (or quests) to the list of loaded quests based on a prefab.

Parameters:

- **questPrefab**: the prefahb of the quest(s) to load.

Load Scene Action

Loads a scene from scene name or build index. Using the scene name 'current' will load the current scene. Using the scene name 'next' or 'previous' will load the next and previous scene by build index.

Parameters:

- **sceneNameOrIndex**: The name or build index of the scene.
- **mode**: How to load the scene.
- **allowActivation**: Activate the scene as soon as it's ready?
- **loadProgress**: Value to store the progress of loading in.

Load State Machine Action

Loads a single State Machine stored in a specified Storage Level using a specified Storage Key.

Parameters:

- **storageLevel**: The level of storage to load the State Machine from.
- **storageKey**: The unique key the State Machine is stored under.
- **target**: The State Machine template or target to set to the loaded State Machines.
- **parent**: Object to parent the new State Machine to.
- **overwrite**: If true, the target will be overwritten with the loaded data. Otherwise a copy will be made.

Log Action

Creates a console log entry for debugging.

Parameters:

- **type**: The type of Debug Log to use.
- **logs**: The information to print to the console.

Log In Action

Logs in to an account with the supplied credentials.

Parameters:

- **username**: The user name to use.
- **password**: The password to use.

Looping Action

Will loop any number of Actions a number of times depending on the parameters.

Parameters:

- **delay**: The delay between loops in seconds.
- **loops**: The number of loops to perform. A value of 0 will loop an infinite number of times.
- **checkAtBeginning**: Should the loop check the optional condition at the start or end of the loop.

Modify Bool Value Action

Modifies a Bool to toggle it's value.

Parameters:

- **sourceValue**: The source Bool to modify. The source will not actually be modified unless the source is also the target.
- **negate**: Should the source be set to it's opposite value?
- **targetValue**: Where to set the result of the modification.
- **revertOnStop**: When this Action stops, should the previous value be reset to the target?

Modify Color Value Action

Modifies a Color based on a specified amount and sets the modified Color to the target.

Parameters:

- **sourceValue**: The source Color to modify. The source will not actually be modified unless the source is also the target.
- **operation**: What modification to perform on the source.
- **modificationAmount**: How much the source should be modified by.
- **targetValue**: Where to set the result of the modification.
- **revertOnStop**: When this Action stops, should the previous value be reset to the target?

Modify Float Value Action

Modifies a Float based on a specified amount and sets the modified Float to the target.

Parameters:

- **sourceValue**: The source Float to modify. The source will not actually be modified unless the source is also the target.
- **operation**: What modification to perform on the source.
- **modificationAmount**: How much the source should be modified by.
- **targetValue**: Where to set the result of the modification.
- **revertOnStop**: When this Action stops, should the previous value be reset to the target?

Modify Integer Value Action

Modifies an Integer based on a specified amount and sets the modified Integer to the target.

Parameters:

- **sourceValue**: The source Integer to modify. The source will not actually be modified unless the source is also the target.
- **operation**: What modification to perform on the source.
- **modificationAmount**: How much the source should be modified by.
- **targetValue**: Where to set the result of the modification.
- **revertOnStop**: When this Action stops, should the previous value be reset to the target?

Modify String Value Action

Modifies a String based on a specified operation and sets the modified String to the target.

Parameters:

- **sourceValue**: The source String to modify. The source will not actually be modified unless the source is also the target.
- **operation**: What modification to perform on the source.
- **modification**: What to modify the source with.
- **toReplace**: If using the Replace operation, the targeted characters in the source to replace with the modification.
- **targetValue**: Where to set the result of the modification.
- **revertOnStop**: When this Action stops, should the previous value be reset to the target?

Modify Vector2 Value Action

Modifies a Vector2 based on a specified amount and sets the modified Vector2 to the target.

Parameters:

- **sourceValue**: The source Vector2 to modify. The source will not actually be modified unless the source is also the target.
- **operation**: What modification to perform on the source.
- **modificationAmount**: How much the source should be modified by.
- **targetValue**: Where to set the result of the modification.
- **revertOnStop**: When this Action stops, should the previous value be reset to the target?

Modify Vector3 Value Action

Modifies a Vector3 based on a specified amount and sets the modified Vector3 to the target.

Parameters:

- **sourceValue**: The source Vector3 to modify. The source will not actually be modified unless the source is also the target.
- **operation**: What modification to perform on the source.
- **modificationAmount**: How much the source should be modified by.
- **targetValue**: Where to set the result of the modification.
- **revertOnStop**: When this Action stops, should the previous value be reset to the target?

Mouse Drag Action

Moves the target(s) to the current cursor position.

Parameters:

- **target**: The object(s) to move.
- **offset**: The offset from the cursor. The X and Y offsets are offset relative to the cursor and the Z offset the is distance from the Main Camera.
- **X**: If checked, movement is enabled on the X axis.
- **Y**: If checked, movement is enabled on the Y axis.
- **Z**: If checked, movement is enabled on the Z axis.

Move Action

Moves the targeted object(s).

Parameters:

- **target**: The object(s) to change the position of.
- **destination**: The position to move the target(s) to.
- **durationOrSpeed**: If using the Duration Movement Type, how long the move should take in seconds. For all else how fast the targets should move to the destination. Set to 0 for instant movement.
- **curve**: If using the Duration Movement Type, the rate of movement over the duration.
- **movementType**: How the target(s) should move.
- **x**: Should moving be enabled on the X-Axis?
- **y**: Should moving be enabled on the Y-Axis?
- **z**: Should moving be enabled on the Z-Axis?
- **clampToGround**: Should the target(s) be clamped to the ground?
- **clampDistance**: If clamping the target(s) to the ground, how offset between the target(s) and the ground.

Move Texture Action

Offsets the texture towards the desired offset value.

Parameters:

- **target**: The object to change the texture offset of.
- **revert on stop**: revert to original offset once the transition is over
- **offset**: desired offset
- **duration**: How long the offset change should take. 0 is instant.
- **movement type**: duration, speed or constant movement.
- **curve**: Used to modify how the offset is evaluated.

Navigate Action

Sets the destination of the targeted NavMeshAgent(s).

Parameters:

- **target**: The object(s) to set the destination of.
- **destination**: The new destination position.
- **animationTarget**: The target for animations.
- **walkParameterName**: The name of the parameter to set when walking.

On Start And Stop Action

Action to run other actions when it starts or finishes. Any actions underneath will be run at start unless they are marked as Stop Actions, in which case they will be run on stop.

Open Web Page Action

Opens a URL in a browser.

Parameters:

- **url**: The URL to go to.

Pause Particle System Action

Pauses or unpauses the targeted particle system.

Parameters:

- **target**: the particle system to pause or unpause.
- **pause**: whether to pause or unpause.

Play Particle System Action

Plays the targeted particle system.

Parameters:

- **target**: the particle system to play.

Play Sound Action

Plays a specified AudioClip.

Parameters:

- **target**: Target to play the clip from. Target requires an AudioSource component.
- **clip**: The sound to play. If not specified, whatever sound is attached to the target will play.
- **loop**: When ticked the clip loops until the action is stopped.
- **keepPlayingAfterStop**: If ticked and the action is stopped, the clip continues playing until done.
- **volume**: Clip volume.
- **pitch**: Clip pitch.
- **reverb**: Clip reverb.
- **steroPan**: Clip stero pan.

Quit Game Action

Instantly quits the application.

Remove Quest Action

Remove an Active Quest without completing it. The active Steps, Stage, and Quest will be set to Inactive.

Parameters:

- **questName**: the name of the quest to remove.

Reset Quest Action

Reset all progress on an Active or completed Quest.

Parameters:

- **questName**: the name of the quest to reset.

Rotate Action

Scales the targeted object(s).

Parameters:

- **target**: The object(s) to change the rotation of.
- **size**: The orientation to rotate the target(s) to.
- **durationOrSpeed**: If using the Duration Movement Type, how long the move should take in seconds. For all else how fast the targets should move to the destination. Set to 0 for an instant rotation.
- **movementType**: How the target(s) should scale.
- **curve**: If using the Duration Movement Type, the rate of scaling over the duration.

Save State Machine Action

Saves the targeted State Machine(s).

Parameters:

- **target**: The State Machine(s) to save.

Scale Action

Scales the targeted object(s).

Parameters:

- **target**: The object(s) to change the scale of.
- **size**: The size to scale the target(s) to.
- **duration**: If using the Duration Movement Type, how long the scaling should take in seconds. Set to 0 for it to happen instantly.
- **curve**: If using the Duration Movement Type, the rate of scaling over the duration.
- **movementType**: How the target(s) should scale.

- **x**: Should scaling be enabled on the X-Axis?
- **y**: Should scaling be enabled on the Y-Axis?
- **z**: Should scaling be enabled on the Z-Axis?

Scale Texture Action

Scales the texture towards the desired offset value.

Parameters:

- **target**: The object to change the texture offset of.
- **revert on stop**: revert to original offset once the transition is over
- **offset**: desired scale
- **duration**: How long the offset change should take. 0 is instant.
- **movement type**: duration, speed or constant movement.
- **curve**: Used to modify how the offset is evaluated.

Select Character Action

Selects a character to play as.

Parameters:

- **character**: The character to select.

Set Angular Velocity Action

Sets the angular velocity to a Rigidbody(2D) attached to the target(s).

Parameters:

- **target**: The object(s) to set the angular velocity of.
- **newVelocity**: The new angular velocity to set.
- **revertOnStop**: When this action stops, should the previous angular velocity be reset to the target(s)?

Set Bool Animation Param Action

Sets a boolean value to a specified parameter on the targeted Animatior(s).

Parameters:

- **target**: The object(s) to access the boolean parameter of. Requires an Animator component.
- **parameterName**: The name of the boolean parameter on the target you want to change.
- **value**: The new boolean value to set to the parameter.
- **revertOnStop**: When this Action stops, should the previous value of the parameter be reset?

Set Camera Angle Action

Sets the Camera Angle on a Camera Controller.

Parameters:

- **target**: Camera Controller object you want to set the Camera Angle of.
- **angle**: The new Camera Angle.

- **revertOnStop**: Should the previous Angle be reset when the Action stops?

Set Camera Bob Action

Sets the amount of Bob on a Camera Controller.

Parameters:

- **target**: Camera Controller object you want to set the amount of Bob on.
- **bobAmount**: The new amount of Bob.
- **revertOnStop**: Should the previous Bob amount be reset when the Action stops?

Set Camera Distance Action

Sets the Camera Distance on a Camera Controller.

Parameters:

- **target**: Camera Controller object you want to set the Camera Distance of.
- **distanceValue**: The new distance value (0-1)
- **revertOnStop**: Should the previous distance be reset when the Action stops?

Set Camera Target Action

Sets the Targets on a Camera Controller.

Parameters:

- **target**: Camera Controller object you want to set the Targets of.
- **cameraTargets**: The new Camera Targets.
- **revertOnStop**: Should the previous targets be restored when the Action stops?

Set Collider Enabled Action

Sets the enabled status of the targeted Collider(s).

Parameters:

- **target**: The object(s) to set the new enabled status to.
- **isEnabled**: The new enabled status.
- **revertOnStop**: When this Action stops, should the previous enabled status be reset to the target(s)?

Set Color Action

Sets the color to the specified Type in the name.

Parameters:

- **target**: The object to change the color of.
- **duration**: How long the color change should take. 0 is instant.
- **gradient**: The colors to set the targeted object's color to.
- **curve**: Used to modify how the gradient is evaluated.

Set Color Value Action

Sets a color value to another color value.

Set Component Enabled Action

Sets a Component's enabled state.

Parameters:

- **target**: The Component to set the enabled state of.
- **enable**: Should the component be enabled or disabled?
- **componentType**: Which Component type to look for.
- **revertOnStop**: Should the previous value be restored when this Action stops?

Set Enabled Action

Sets a GameObject's Active state.

Parameters:

- **target**: The GameObject to set the Active state of.
- **isEnabled**: Should the GameObject be set Active or Inactive?
- **revertOnStop**: Should the previous value be reset to the Target when this Action stops?

Set Float Animation Param Action

Sets a float value to a specified parameter on the targeted Animatior(s).

Parameters:

- **target**: The object(s) to access the float parameter of. Requires an Animator component.
- **parameterName**: The name of the float parameter on the target you want to change.
- **value**: The new float value to set to the parameter.
- **revertOnStop**: When this Action stops, should the previous value of the parameter be reset?

Set Game Gravity Action

Sets the gravity for the entire game.

Parameters:

- **newGravity**: The new gravity to set. Default gravity is -9.81.
- **revertOnStop**: Should the previous gravity be reset when this Action stops?

Set Game Object List Value Action

Sets a GameObject List to a GameObject List Value.

Set Game Object Value From Target Action

Sets a GameObject Value to a target.

Parameters:

- **value**: The GameObject to set.
- **target**: The object to set the value on.
- **revertOnStop**: When the action stops, do you want to set the previous GameObject value back onto the target?

Set Game Volume Action

Sets the volume for the entire game.

Parameters:

- **volume**: The new volume to set.
- **revertOnStop**: When this Action stops, should the previous volume be reset?

Set Integer Animation Param Action

Sets a float value to a specified parameter on the targeted Animatior(s).

Parameters:

- **target**: The object(s) to access the float parameter of. Requires an Animator component.
- **parameterName**: The name of the float parameter on the target you want to change.
- **value**: The new float value to set to the parameter.
- **revertOnStop**: When this Action stops, should the previous value of the parameter be reset?

Set Integer Value Action

Sets an integer value to another value.

Set Kinematic Action

Sets the kinematic status of the target(s).

Parameters:

- **target**: The target(s) to change the kinematic status of.
- **isKinematic**: The kinematic status to set.
- **revertOnStop**: Should the previous kinematic status be reset when the Action stops?

Set Light Cookie Action

Sets a Light Cookie onto the target(s).

Parameters:

- **target**: The Light object you want to set the Light Cookie of.
- **cookie**: The Light Cookie to set.
- **revertOnStop**: Should the previous Light Cookie be reset when the Action stops?

Set Light Flare Action

Sets a Flare object onto a Light object.

Parameters:

- **target**: The Light object you want to set the Flare of.
- **flare**: The Flare to set.
- **revertOnStop**: Should the previous Flare be reset when the Action stops?

Set Light Intensity Action

Sets the intensity of a Light object.

Parameters:

- **target**: The Light object you want to set the intensity of.
- **intensity**: The new intensity to set.
- **revertOnStop**: Should the previous intensity be reset when the Action stops?

Set Light Range Action

Sets the range of a Light object.

Parameters:

- **target**: The Light object you want to set the range of.
- **range**: The new range to set.
- **revertOnStop**: Should the previous range be reset when the Action stops?

Set Light Spot Angle Action

Sets the spot angle of a Light object.

Parameters:

- **target**: The Light object you want to set the spot angle of.
- **spotAngle**: The new spot angle to set.
- **revertOnStop**: Should the previous spot angle be reset when the Action stops?

Set Light Type Action

Sets the Light type of a Light object.

Parameters:

- **target**: The Light object you want to set the Light type of.
- **lightType**: The new Light type to set.
- **revertOnStop**: Should the previous Light type be reset when the Action stops?

Set Material Action

Sets a Material to the target(s).

Parameters:

- **target**: The object(s) to set the Material to.
- **material**: The Material to set.
- **revertOnStop**: When this Action stops, should the previous Material be reset to the target(s).

Set Navigation Speed Action

Sets the speed of the targeted NavMeshAgent(s).

Parameters:

- **target**: The object(s) to set the speed of.
- **speed**: The new speed.
- **revertOnStop**: Whether to revert this action when it stops.

Set Navigaton Destination Action

Sets the destination of the targeted NavMeshAgent(s).

Parameters:

- **target**: The object(s) to set the destination of.
- **destination**: The new destination position.

Set Parent Action

Sets the target(s) to be parented to the specified parent target.

Parameters:

- **target**: The object(s) to change the parent of.
- **parent**: The new parent.

Set Particle System Time Action

Sets the elapsed time on a particle system.

Parameters:

- **target**: the particle system to set the time on.
- **time**: the time to set.

Set Particles Action

Set the particles from a list.

Parameters:

- **target**: the particle system set.
- **particles**: the list to retrieve the particles from.

Set Random Float Action

Generates a random float and sets it to the targeted value.

Parameters:

- **target**: The value to set the random float to.
- **minValue**: The smallest float that the random number should be. This number is inclusive, meaning only numbers equal to or greater than the value will be generated.
- **maxValue**: The largest float that the random number should be. This number is inclusive, meaning only numbers less than or equal to the value will be generated.
- **revertOnStop**: Saves the target value and resets it to the target if the Action stops.

Set Random Game Object Action

Picks a random GameObject from a list and sets it to the targeted value.

Parameters:

- **target**: The value to set the random GameObject to.
- **gameObjects**: The List of GameObjects to randomly pick from.
- **revertOnStop**: Saves the target value and resets it to the target if the Action stops.

Set Random Int Action

Generates a random int and sets it to the targeted value.

Parameters:

- **target**: The value to set the random int to.
- **minValue**: The smallest int that the random number should be. This number is inclusive, meaning only numbers equal to or greater than the value will be generated.
- **maxValue**: The largest int that the random number should be. This number is exclusive, meaning only numbers less than the value will be generated.
- **revertOnStop**: Saves the target value and resets it to the target if the Action stops.

Set Random Seed Action

Used to set the seed for Unity's Random class.

Parameters:

- **seed**: The new seed to generate Random numbers from.
- **randomSeed**: Will override seed and generate a new seed randomly.
- **revertOnStop**: Will set the Random State to the previous Random State before the new seed was applied when the Action stops.

Set Random String Action

Picks a random string from a list and sets it to the targeted value.

Parameters:

- **target**: The value to set the random string to.
- **gameObjects**: The List of GameObjects to randomly pick from.
- **revertOnStop**: Saves the target value and resets it to the target if the Action stops.

Set Random Vector3 Action

Generates a random Vector3 and sets it to the targeted value.

Parameters:

- **target**: The value to set the random Vector3 to.
- **minValue**: The smallest floats for each axis that the random number should be. Each axis is inclusive, meaning only numbers equal to or greater than the value will be generated.
- **maxValue**: The largest floats for each axis that the random number should be. Each axis is inclusive, meaning only numbers less than or equal to the value will be generated.
- **revertOnStop**: Saves the target value and resets it to the target if the Action stops.

Set Sound Pitch Action

Sets the pitch of an AudioSource component.

Parameters:

- **target**: The object where the AudioSource component can be found.
- **pitch**: The new value for the pitch.
- **revertOnStop**: If true, the original pitch value will be restored when the action is stopped.

Set Sound Volume Action

Sets the specified volume to the target(s).

Parameters:

- **target**: The object(s) to set the volume of.
- **volume**: The new volume to set.
- **revertOnStop**: When this Action stops, should the previous volume be reverted to the target(s)?

Set Targets Action

Sets the new target(s) to a list of Actions and runs the Actions.

Parameters:

- **newTargets**: The new targets to set.

Set Time Scale Action

Sets Unity's native time scale to a value from 0.0f to 1.0f.

Parameters: greater lerps it to the target time scale over duration.

Set Trigger Animation Param Action

Sets a trigger on a specified parameter on the targeted Animatior(s).

Parameters:

- **target**: The object(s) to access the trigger parameter of. Requires an Animator component.
- **parameterName**: The name of the boolean parameter on the target you want to change.

Set UI Context Action

Set the context for a UI element (and its children).

Parameters:

- **target**: The UI element(s) to set the context on.
- **newOwner**: The new owner for the context.
- **newTargets**: The new targets for the context.

Set UI Image Action

Sets the sprite of a UI Image.

Parameters:

- **target**: The object where the Image component can be found.
- **spriteValue**: The new Sprite for the Image.
- **revertOnStop**: If true, the original Sprite will be restored when the action is stopped.

Set UI Slider Position Action

Sets the position of a UI Slider or Scrollbar.

Parameters:

- **target**: The object where the Slider or Scrollbar component can be found.
- **floatValue**: The new value for the Slider/Scrollbar.
- **revertOnStop**: If true, the original value will be restored when the action is stopped.

Set UI Text Action

Sets a series of string values to a targeted Text.

Parameters:

- **target**: The object(s) to set the text to.
- **body**: The body of the text.
- **prefix**: Any text to go before the body.
- **postfix**: Any text to go after the body.
- **revertOnStop**: When this action is stopped, should the original value of the target be reset to the target?
- **keepUpdated**: Should the Text on the target be continuously updated?

Set UI Toggle Action

Sets the isOn value of UI Toggle component.

Parameters:

- **target**: The object where the Toggle component can be found.
- **boolValue**: The new isOn value for the Toggle component.

- **revertOnStop**: If true, the original isOn value will be restored when the action is stopped.

Set Vector3 Value Action

Sets a Vector3 to a Vector3 Value.

Set Vector3 Value From Target Action

Sets a Vector3 Value to a target.

Parameters:

- **value**: The Vector3 to set.
- **target**: The object to set the value on.
- **revertOnStop**: When the action stops, do you want to set the previous Vector3 value back onto the target?

Set Velocity Action

Sets the velocity to a Rigidbody(2D) attached to the target(s).

Parameters:

- **target**: The object(s) to set the velocity of.
- **newVelocity**: The new velocity to set.
- **revertOnStop**: When this action stops, should the previous velocity be reset to the target(s)?

Simulate Particle System Action

Advance the simulation of a particle system by some time.

Parameters:

- **target**: the particle system to simulate.
- **time**: the number of seconds to simulate.

Spawn Action

Spawns an object.

Parameters:

- **target**: The position(s) to spawn the objects.
- **parent**: The object the spawned objects should be parented to.
- **prefab**: The prefab to spawn.
- **rotation**: A rotation to be applied to the spawned objects.
- **revertOnStop**: Should this Action revert to the state it was in before it ran when it stops?
- **startState**: If spawning an object with a State Machine attached, the name of the initial state.
- **setTo**: The value to set the spawned GameObject to.

Spawn Row Action

Spawns a set of objects in a row.

Parameters:

- **target**: The position(s) to spawn the objects.
- **parent**: The object the spawned objects should be parented to.
- **count**: How many objects should be spawned in the row.
- **spacing**: The offset between spawned objects.
- **axis**: Which axis to spawn the row of objects on.
- **prefab**: The prefab to spawn.
- **rotation**: A rotation to be applied to the spawned objects.
- **revertOnStop**: Should this Action revert to the state it was in before it ran when it stops?
- **startState**: If spawning an object with a State Machine attached, the name of the initial state.
- **storeIn**: An optional value to save the spawned GameObjects to.

Start Timer Action

Starts a Timer Value.

Parameters:

- **timer**: The timer to start.
- **reset**: Should the timer be reset to the startingTime?
- **startingTime**: The time the timer starts at.

Stop Particle System Action

Stops the targeted particle system.

Parameters:

- **target**: the particle system to stop.

Stop Sound Action

Stops the first AudioSource on the target(s)

Parameters:

- **target**: The object(s) to stop sound on.

Stop Timer Action

Stops a Timer Value.

Parameters:

- **timer**: The timer to stop.

Translate Action

Moves the targeted object(s) on its local axis.

Parameters:

- **target**: The object(s) to change the position of.

- **amount**: The amount to move the target(s) by.
- **durationOrSpeed**: If using the Duration Movement Type, how long the move should take in seconds. For all else how fast the targets should move to the destination. Set to 0 for instant movement.
- **curve**: If using the Duration Movement Type, the rate of movement over the duration.
- **movementType**: How the target(s) should move.
- **clampToGround**: Should the target(s) be clamped to the ground?
- **clampDistance**: If clamping the target(s) to the ground, how offset between the target(s) and the ground.

Triggered Action

Requires at least one childed Action and exactly one Trigger. When the Trigger Fires, all childed Actions will run.

Parameters:

- **triggerForNewTargets**: If any targets are added, should this run?

Unload Quest Action

Unload a loaded Quest.

Parameters:

- **questName**: the name of the quest to unload.
- **inactiveOnly**: If set, the quest will only be removed if it is inactive.

Unload Scene Action

Unloads a scene from scene name or build index. Using the scene name 'current' will unload the current scene.

Parameters:

- **sceneNameOrIndex**: The name or build index of the scene.
- **allowActivation**: Activate the scene as soon as it's ready?
- **loadProgress**: Value to store the progress of unloading.

Trigger

Animation Event Trigger

Trigger that fires when an Animator fires an Event with a specific payload. Use the Method names 'OnAnimationEventString', 'OnAnimationEventInt', and 'OnAnimationEventFloat' respectively with the Animation Event.

Parameters:

- **target**: The object where the Animator components can be found.
- **eventName**: The string value payload of the animation event to look for.

Animation State Trigger

Trigger that fires when one or more an Animators are in a certain state.

Parameters:

- **target**: The object where the Animator components can be found.
- **requirement**: How many of the animators must be in the specified state to trigger.
- **stateName**: The name of the animation state to look for.

Bool Animation Param Trigger

Trigger that fires when a bool parameter on one or more Animator components changes.

Parameters:

- **target**: The object(s) where the Animator component can be found.
- **parameterName**: The name of the bool animation parameter to look for.
- **comparison**: The value to compare the parameter value against. The result of the comparison is the state that is passed when this Trigger Fires.

Bool Value Trigger

Will fire whenever the bool value on this trigger is toggled.

Parameters:

- **value**: The Bool Value that is used to determine if the trigger fires.

Change State Trigger

Triggers when the targeted State Machine's state changes to stateName.

Parameters:

- **target**: The State Machine to check the state of.
- **stateName**: The name of the State to check for.

Collision Trigger

Triggers when the target collides with the collide target. If collide target is set to none then it will fire on any collision.

Parameters:

- **target**: The object to check for collisions on.
- **collideTarget**: The object to collide with. If none, any object will trigger.
- **isTrigger**: Whether to listen for a trigger collision (if your collider is set as a trigger, use this).
- **layerMask**: The layer to ignore when checking the RayCast.

Enable Particle Triggers Action

Enables or disables particle triggers in the targeted particle system.

Parameters:

- **target**: the particle system to enable or disable triggers on.
- **enable**: whether to enable or disable triggers.

Enter Area Trigger

Fires when the player is in a specific area.

Parameters:

- **areaID**: The ID of the area to check.

Enter Realm Trigger

Fires when the player is in a specific realm.

Parameters:

- **realmID**: The realm ID of the realm to check.

Event Trigger

Subscribes the target(s) to a specified Event and Fires when the specified event Fires.

Parameters:

- **target**: The target(s) to subscribe the Event to. Set to None for a Global Event.
- **eventName**: The name of the Event to subscribe to.

Float Animation Param Trigger

Trigger that fires when a float parameter on one or more Animator components changes.

Parameters:

- **target**: The object(s) where the Animator component can be found.
- **parameterName**: The name of the float animation parameter to look for.
- **comparison**: The value to compare the parameter value against. The result of the comparison is the state that is passed when this Trigger Fires.

Float Value Trigger

Triggers if the target value fulfils the comparison type of the comparison value.

Parameters:

- **target**: The target float to compare.
- **comparisonType**: The type of comparison to make.
- **comparison**: The float value the target should be compared against.
- **range**: If using the Range Comparison Type, the range in which the target value must be compared to the comparison value.

Game Object Value Trigger

Triggers if the target value fulfils the comparison type of the comparison value.

Parameters:

- **target**: The target GameObject to compare.
- **comparison**: The type of comparison to make.
- **compareTo**: The value the target should be compared against.

Input Axis Trigger

Trigger that fires when movement on the 'axisName' is detected.

Parameters:

- **axisName**: The input axis name to listen to. This needs to be set on Edit-> Project Settings-> Input.
- **comparisonType**: The type of comparison to use on the axis. Set to None to trigger on any change.
- **range**: For use in the range comparison type to specify the range.

Input Button Trigger

Trigger that fires when a specific named input button is pressed or released.

Parameters:

- **buttonName**: The input button name to listen to. This needs to be set on Edit-> Project Settings-> Input.
- **direction**: The direction of the button press (Up or Down) to trigger on.

Integer Animaton Param Trigger

Trigger that fires when an integer parameter on one or more Animator components changes.

Parameters:

- **target**: The object(s) where the Animator component can be found.
- **parameterName**: The name of the integer animation parameter to look for.
- **comparison**: The value to compare the parameter value against. The result of the comparison is the state that is passed when this Trigger Fires.

Integer Value Trigger

Triggers if the target value fulfils the comparison type of the comparison value.

Parameters:

- **target**: The target integer to compare.
- **comparisonType**: The type of comparison to make.
- **comparison**: The float value the target should be compared against.
- **range**: If using the Range Comparison Type, the range in which the target value must be compared to the comparison value.

Is Enabled Trigger

Triggers if the Target GameObject's active state changes.

Parameters:

- **target**: The Target GameObject to check the active state of.
- **evaluationRequirement**: For multiple targets, how many need to be enabled to fire.

Is Visible Trigger

Triggers if an object becomes visible or invisible.

Parameters:

- **target**: The objects you want to check the visibility of.
- **evaluationRequirement**: For multiple targets, how many need to be visible to fire.

Keyboard Trigger

Trigger that fires when a specific key (or any key) is pressed or released.

Parameters:

- **anyInput**: If true, this will trigger on any input instead of a specific key.
- **key**: The key to trigger on.
- **direction**: The direction of the keypress to trigger on.

Leave Area Trigger

Fires when the player leaves a specific area.

Parameters:

- **areaID**: The ID of the area to check.

Leave Realm Trigger

Fires when the player leaves a specific realm.

Parameters:

- **realmID**: The realm ID of the realm to check.

Log In Trigger

Fires when the player is logged in.

Mouse Button Trigger

Triggers if the chosen Mouse Button is pressed.

Parameters:

- **button**: Which Mouse Button you want to Fire the Trigger.
- **direction**: The direction to press the chosen Button.

Particle Count Trigger

Checks the number of currently spawned particles.

Parameters:

- **target**: the particle system to check.
- **comparisonType**: The comparison to use.
- **count**: The count to compare to.
- **range**: The range for range comparisons.
- **evaluationRequirement**: How many targets must match.

Particle System Alive Trigger

Checks if the particle system has active particles or is emitting.

Parameters:

- **target**: the particle system to check.
- **evaluationRequirement**: How many targets must match.

Particle System Emitting Trigger

Checks if the particle system is emitting.

Parameters:

- **target**: the particle system to check.
- **evaluationRequirement**: How many targets must match.

Particle System Playing Trigger

Checks if the particle system is playing.

Parameters:

- **target**: the particle system to check.
- **evaluationRequirement**: How many targets must match.

Particle System Time Trigger

Checks the elapsed time of the particle system.

Parameters:

- **target**: the particle system to check.
- **comparisonType**: The comparison to use.
- **time**: The time to compare to.
- **range**: The range for range comparisons.
- **evaluationRequirement**: How many targets must match.

Particle Trigger

Triggers when a particle enters a trigger.

Parameters:

- **target**: the particle system to check.
- **triggerType**: whether to check enter/exit or inside/outside.
- **storeIn**: Value to store the affected particles in.

Player Jump Trigger

Fires if the player is jumping.

Press Trigger

Triggers if the Left Mouse Button is pressed on a target.

Parameters:

- **target**: The target(s) to be clicked on.
- **direction**: The direction to press the Left Mouse Button.

Proximity Trigger

Triggers if any objects get within the specified radius of the target(s).

Parameters:

- **target**: The objects to check the area of.
- **radius**: How far from the target should be searched.
- **collideTarget**: Will only Fire if a specific target is within the radius.
- **triggerForNewTargets**: Should this trigger for new targets?
- **layerMask**: The layer to ignore when checking the radius.

Quest Active Trigger

Triggers when the specified Quest becomes Active.

Parameters:

- **questName**: The name of the Quest to check.

Quest Stage Active Trigger

Triggers when the specified Stage of a Quest becomes Active.

Parameters:

- **questName**: The name of the Quest.
- **stageName**: The name of the Stage to check.

Quest Step Active Trigger

Triggers when the specified Step of a Quest becomes Active.

Parameters:

- **questName**: The name of the Quest.
- **stageName**: The name of the Stage.
- **stepName**: The name of the Step to check.

Quests Loaded Trigger

Triggers when the Quest service finishes loading quest progress.

Raycast Trigger

Fires whenever a Collide Target's collider is hit by the ray between Start and End.

Parameters:

- **start**: The start position of the ray
- **end**: The end position of the ray
- **layerMask**: A layer mask to apply to the raycast
- **collideTarget**: The target to test for. Set to None for any target.
- **triggerForNewTargets**: Should this trigger re-trigger with true if a new collide target is hit by the ray.
- **is2D**: Determines whether to do the raycast in 2D or 3D.

Select Character Trigger

Fires when the player selects a character.

Set Trigger Animation Param Action

Sets a trigger on a specified parameter on the targeted Animatior(s).

Parameters:

- **target**: The object(s) to access the trigger parameter of. Requires an Animator component.
- **parameterName**: The name of the boolean parameter on the target you want to change.

String Value Trigger

Fires when the condition to compare the string is met.

Parameters:

- **target**: Target string.
- **comparison**: Type of comparison.
- **compareTo**: String to compare target to.

Swipe Trigger

Fires if the screen is swiped in a specified direction.

Parameters:

- **direction**: The direction to swipe.
- **maximumDuration**: The maximum amount of time a swipe should take.
- **minimumDistance**: How far the swipe should travel to be valid.

Time Scale Changed Trigger

Subscribes the target(s) to a Time.timeScale changes made through

Parameters: CATime and Fires when CATime.timeScale has been updated

Trigger List

Combines multiple triggers into one, only firing when the evaluation requirement is met.

Parameters:

- **evaluationRequirement**: Specifies how many child triggers need to be active before firing.

Triggered Action

Requires at least one childed Action and exactly one Trigger. When the Trigger Fires, all childed Actions will run.

Parameters:

- **triggerForNewTargets**: If any targets are added, should this run?

UI Button Trigger

Triggers when a UI Button is clicked.

Parameters:

- **target**: The object where the UI Button can be found.

UI Image Changed Trigger

Triggers when the sprite value on a UI Image component changes.

Parameters:

- **target**: The object where the UI Image can be found.

UI Slider Position Trigger

Triggers when the target UI Slider's value changes.

Parameters:

- **target**: The object where the UI Slider can be found.

UI Text Changed Trigger

Triggers when the text value on a UI text component changes.

Parameters:

- **target**: The object where the UI Text can be found.

UI Toggle Group Trigger

Triggers when a specific Toggle from a Toggle Group is switched on.

Parameters:

- **target**: The object where the Toggle Group component can be found.

UI Toggle Trigger

Triggers when the target UI Toggle changes.

Parameters:

- **target**: The object where the UI Toggle can be found.

Value Changed Trigger

Subscribes to a value and Fires when it is modified.

Parameters:

- **target**: The value to subscribe to.

Vector3 Value Trigger

Triggers if the target value fulfils the comparison type of the comparison value.

Parameters:

- **target**: The target vector to compare.
- **comparisonType**: The type of comparison to make.
- **comparison**: The vector value the target should be compared against.
- **scalarComparison**: Comparison for magnitude, dot product, and distance.
- **scalar**: Used for scalar comparison.
- **range**: If using the Range Comparison Type, the range in which the target value must be compared to the comparison value.

Velocity Changed Trigger

Triggers if the Target's Rigidbody or Rigidbody2D velocity changes on one or more axes and depending on the

Parameters: VelocityEvaluation mode either when the Target's velocity changes by way of Accelerating or Decelerating.

- **target**: The Target's Rigidbody or Rigidbody2D to check the velocity of.
- **changeType**: For a Target's velocity, whether to fire the trigger based on Acceleration or Deceleration changes.
- **x**: Whether or not to track velocity changes on the X axis.
- **y**: Whether or not to track velocity changes on the Y axis.
- **z**: Whether or not to track velocity changes on the Z axis.

Condition

Animation State Condition

Finds the first Animator attached to the target(s) and compares the animation state(s) to the specified comparison value.

Parameters:

- **target**: The object(s) where the Animator components can be found.
- **requirement**: How many of the animators must be in the specified state for the condition to succeed.
- **stateName**: The name of the animation state to look for.

Bool Animation Param Condition

Finds the first Animator attached to the target(s) and compares the specified bool parameter to the specified comparison value.

Parameters:

- **target**: The object(s) where the Animator components can be found.
- **parameterName**: The name of the bool animation parameter to look for.
- **passRequirement**: The number of bools that must be true for this condition to succeed.

Bool Value Condition

Checks a bool value.

Parameters:

- **targetValue**: The bool value to be checked.

Chance Condition

Checks the specified percentage against a randomly generated percentage (out of 100%).

Parameters:

- **percent**: The percentage of the time you want this condition to be met.

Condition List

Evaluates one or more child conditions using the evaluation requirement to determine if the condition passes.

Parameters:

- **evaluationRequirement**: Determines how many child conditions must succeed in order to be true.

Conditional Action

Requires a Condition to be childed to it. If the Condition is met, the Actions underneath the Condition will Fire.

Current State Condition

Checks the target(s) State Machine to see if it's current State matches the specified state name.

Parameters:

- **target**: The object(s) to check.
- **stateName**: The name of the State to check for.

Float Animation Param Condition

Finds the first Animator attached to the target(s) and compares the specified float parameter to the specified comparison value.

Parameters:

- **target**: The object where the Animator components can be found.
- **parameterName**: The name of the float parameter to look for.
- **comparison**: The value to compare against the float parameter.
- **comparisonType**: How to compare the two values.
- **requirement**: How to determine if the condition succeeds.
- **range**: If using the Range Comparison Type, the range in which the float parameter must be compared to the comparison value.

Float Value Condition

Evaluates two floats and runs if the comparison type is met.

Parameters:

- **targetValue**: The target float to compare.
- **comparisonType**: The type of comparison to make.
- **comparisonValue**: The float value the target should be compared against.
- **range**: If using the Range Comparison Type, the range in which the target value must be compared to the comparison value.

Game Object Value Condition

True if the target value fulfils the comparison type of the comparison value.

Parameters:

- **target**: The target GameObject to compare.
- **comparison**: The type of comparison to make.
- **compareTo**: The value the target should be compared against.

Has State Condition

Checks to see if the target(s) contain a State with a specified name.

Parameters:

- **target**: The object(s) to check for the State.
- **stateName**: The name of the State to check for.
- **evaluationRequirement**: Parameters for if the evaluation passes or fails.

Has Targets Condition

Checks to see if there are any targets.

Parameters:

- **target**: The object(s) to check the targets of.

In Area Condition

Check if the player is in a specific area.

Parameters:

- **areaID**: The ID of the area to check.

In Realm Condition

Checks if the player is in a specific realm.

Parameters:

- **realmID**: The realm ID of the realm to check.

Integer Animation Param Condition

Finds the first Animator attached to the target(s) and compares the specified integer parameter to the specified comparison value.

Parameters:

- **target**: The object where the Animator components can be found.
- **parameterName**: The name of the integer parameter to look for.
- **comparison**: The value to compare against the integer parameter.
- **comparisonType**: How to compare the two values.
- **requirement**: How to determine if the condition succeeds.
- **range**: If using the Range Comparison Type, the range in which the integer parameter must be compared to the comparison value.

Integer Value Condition

Evaluates two integers and runs if the comparison type is met.

Parameters:

- **targetValue**: The targeted integer to compare.
- **comparisonType**: How you want to compare the integers.
- **comparisonValue**: The integer value the target should be compared against.
- **range**: If using the Range Comparison Type, the range in which the target value must be compared to the comparison value.

Is Enabled Condition

Checks to see if the target(s) are enabled or not.

Parameters:

- **target**: The object(s) to check the enabled status of.

- **amountEnabled**: Parameters for if the evaluation passes or fails.

Is Quest Active Condition

Checks if a specified Quest is currently Active.

Parameters:

- **questName**: The name of the quest to check.

Is Quest Completed Condition

Checks if a specified Quest has been Completed

Parameters:

- **questName**: The name of the Quest to check.

Is Quest Stage Active Condition

Checks if a specified Stage on a Quest is currently Active.

Parameters:

- **questName**: The name of the Quest.
- **stageName**: The name of the Stage.

Is Quest Stage Completed Condition

Checks if a specified stage on an Active Quest has been Completed.

Parameters:

- **questName**: The name of the Quest.
- **stageName**: The name of the Stage.

Is Quest Step Active Condition

Checks if a specified Step on an Active Quest is Active.

Parameters:

- **questName**: The name of the Quest.
- **stageName**: The name of the Stage.
- **stepName**: The name of the Step.

Is Quest Step Completed Condition

Checks if a specified Step on an Active Quest has been Completed.

Parameters:

- **questName**: The name of the Quest.

- **stageName**: The name of the Stage.
- **stepName**: The name of the Step.

Is Visible Condition

Checks to see whether the target(s) are visible to any Cameras.

Parameters:

- **target**: The object(s) to check the visibility of.
- **evaluationRequirement**: Parameters for if the evaluation passes or fails.

Keyboard Condition

Checks to see if the specified key is being pressed in the specified direction.

Parameters:

- **anyInput**: Overrides 'key' and accepts any keyboard or mouse input instead.
- **key**: The Key to check.
- **direction**: The direction the button is being pressed in to check.

Logged In Condition

Checks if the player has logged in.

Mouse Button Condition

Checks to see if the specified Mouse Button is being pressed in a specified direction.

Parameters:

- **mouseButton**: The Mouse Button to check.
- **direction**: The direction the button is being pressed in to check.

Nav Mesh Path Distance Condition

Calculates a path along the NavMesh and compares the total distance of the path to the passed in value.

Parameters:

- **start**: Start position for the path.
- **end**: End position for the path.
- **areaMask**: Defines which NavMesh areas are allowed. Defaults to all.
- **comparisonType**: What comparison to use.
- **useDistanceBetweenPoints**: If enabled, the direct distance between the start and end points will be added to compareTo when checking the path distance.
- **compareTo**: The number to compare the path distance to.
- **range**: If comparisonType is range, this field is used as the range.
- **evaluationRequirement**: if multiple rays are cast, how many must pass the test.

Nav Mesh Ray Cast Condition

Casts a ray along the NavMesh and succeeds if there is an unobstructed path between the start and end.

Parameters:

- **start**: Start position for the ray.
- **end**: End position for the ray.
- **areaMask**: Defines which NavMesh areas are allowed. Defaults to all.
- **evaluationRequirement**: if multiple rays are cast, how many must pass the test.

Particle Count Condition

Checks the number of currently spawned particles.

Parameters:

- **target**: the particle system to check.
- **comparisonType**: The comparison to use.
- **count**: The count to compare to.
- **range**: The range for range comparisons.
- **evaluationRequirement**: How many targets must match.

Particle System Alive Condition

Checks if the particle system has active particles or is emitting.

Parameters:

- **target**: the particle system to check.
- **evaluationRequirement**: How many targets must match.

Particle System Emitting Condition

Checks if the particle system is emitting.

Parameters:

- **target**: the particle system to check.
- **evaluationRequirement**: How many targets must match.

Particle System Paused Condition

Checks if the particle system is paused.

Parameters:

- **target**: the particle system to check.
- **evaluationRequirement**: How many targets must match.

Particle System Playing Condition

Checks if the particle system is playing.

Parameters:

- **target**: the particle system to check.
- **evaluationRequirement**: How many targets must match.

Platform Condition

Checks the platform against the current running platform.

Parameters:

- **platform**: Build platform to check for.

Position Condition

Evaluates the target's position and compares it against the specified position.

Parameters:

- **target**: The object(s) to check the position of.
- **position**: The position to compare the target position(s) against.
- **comparison**: How to compare the two positions.
- **x**: Include X Position values in the evaluation?
- **y**: Include Y Position values in the evaluation?
- **z**: Include Z Position values in the evaluation?
- **range**: If using the Range Comparison, the range that the target position should be in compared to the position to be considered valid.

Proximity Condition

Checks all targets for any other objects in a given radius.

Parameters:

- **target**: The objects to check the area of.
- **radius**: How far from the target should be searched.
- **collideTarget**: Will only be true if a specific target is within the radius.
- **layerMask**: The layer to ignore when checking the radius.

Ray Cast Condition

Checks to see if there are any objects between the target(s) and the destination(s).

Parameters:

- **source**: The object(s) to shoot the ray from.
- **collideTarget**: The object to collide with. If none, any object will trigger.
- **destination**: The position(s) where the ray should end.
- **layerMask**: The layer to ignore when checking the RayCast.

Scene Loaded Condition

Checks if the named scene is currently loaded.

Parameters:

- **sceneName**: the name of the scene to check.

Selected Character Condition

Checks if the player has selected a character.

String Value Condition

Evaluates two strings and runs if the comparison type is met.

Parameters:

- **targetValue**: The targeted string to compare.
- **comparisonType**: How you want to compare the strings.
- **comparisonValue**: The string that will be compared against.
- **caseSensitive**: Should the string comparison be case sensitive?
- **trimWhiteSpace**: Should whitespace be trimmed at the start and end?

String Value Length Condition

Checks the length of a string against the compasison type condition.

Parameters:

- **target**: The object to check the length of.
- **comparison**: Comparison Type.
- **compareTo**: Number to compare the string length to.
- **range**: If using the Range Comparison Type, the range in which the target value must be compared to the comparison value.

Time Scale Condition

Evaluates a float value and Time.timeScale

Parameters:

- **comparisonType**: The type of comparison to make.
- **comparison**: The float value the target should be compared against.
- **range**: If using the Range Comparison Type, the range in which the target value must be compared to the comparison value.

UI Slider Condition

UI Condition for slider values.

Parameters:

- **target**: The object where the Slider component can be found.
- **comparison**: How to compare the Slider's position to the compare value.
- **compareValue**: Float value to compare against.
- **rangeValue**: Float value only for use with InRange type comparison.

UI Toggle Condition

UI Condition for a toggle.

Parameters:

- **target**: The object where the Toggle component can be found.
- **value**: What the toggle value should be for the condition to succeed.

UI Toggle Group Condition

UI Condition for a toggle group.

Parameters:

- **target**: The object where the Toggle Group component can be found.
- **toggleName**: The name of the Toggle that must be on for the condition to succeed.

Vector3 Value Condition

Evaluates two vectors and runs if the comparison type is met.

Parameters:

- **targetValue**: The targeted vector to compare.
- **comparisonType**: How you want to compare the vectors.
- **comparisonValue**: The vector that will be compared against.
- **caseSensitive**: Should the vector comparison be case sensitive?
- **trimWhiteSpace**: Should whitespace be trimmed at the start and end?