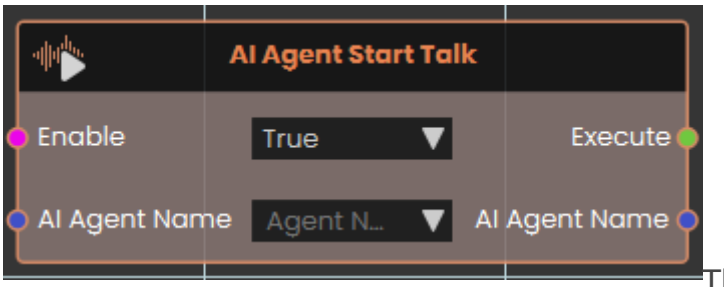
Events

- AI Agent
- SceneNode \ Dynamic Menu
- SceneNode \ Query
- Scene Node \ Overlap
- SceneNode \ Triggered
- Controller
- Keyboard
- User
- Variable
- On Message Received
- Voice Command
- Execution
- Snapping
- sceneNode \ Attributes

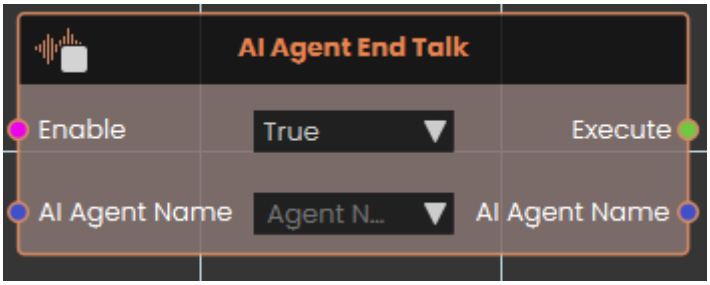
AI Agent

AI Agent Start Talk



The **AI Agent Start Talk** node activates the exact moment an assigned AI Agent begins delivering its response.

AI Agent End Talk



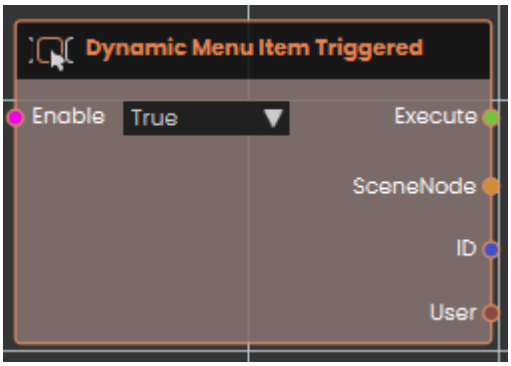
The **AI Agent End Talk** node activates as soon as the assigned AI Agent finishes delivering its response.

For more information, please watch the AI Start/End talk tutorial:

https://www.youtube.com/embed/YEMeCS8QcFY

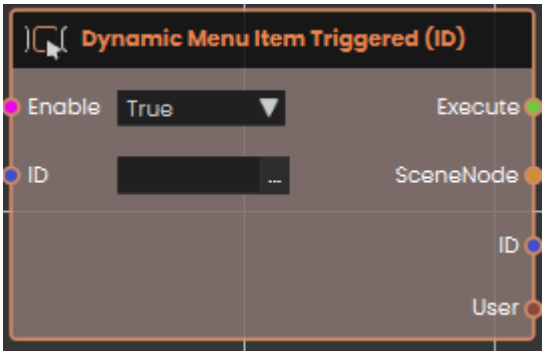
SceneNode \ Dynamic Menu

Dynamic Menu Item Triggered



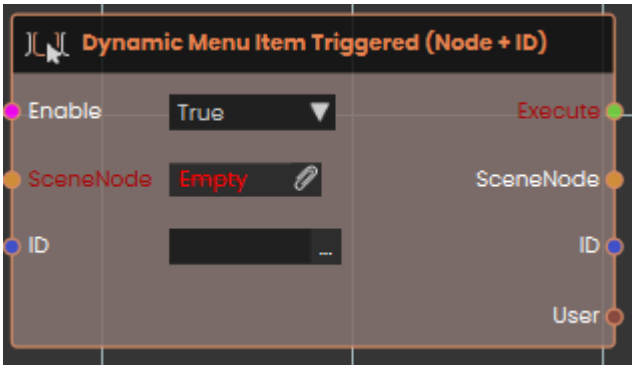
The **Dynamic Menu Item Triggered** event node acts as a universal listener, activating the moment a user interacts with *any* dynamic menu button within the VR scene. Upon execution, it outputs the specific **SceneNode** the menu is attached to, the **ID** of the clicked button, and the **User** who triggered it.

Dynamic Menu Item Triggered(ID)



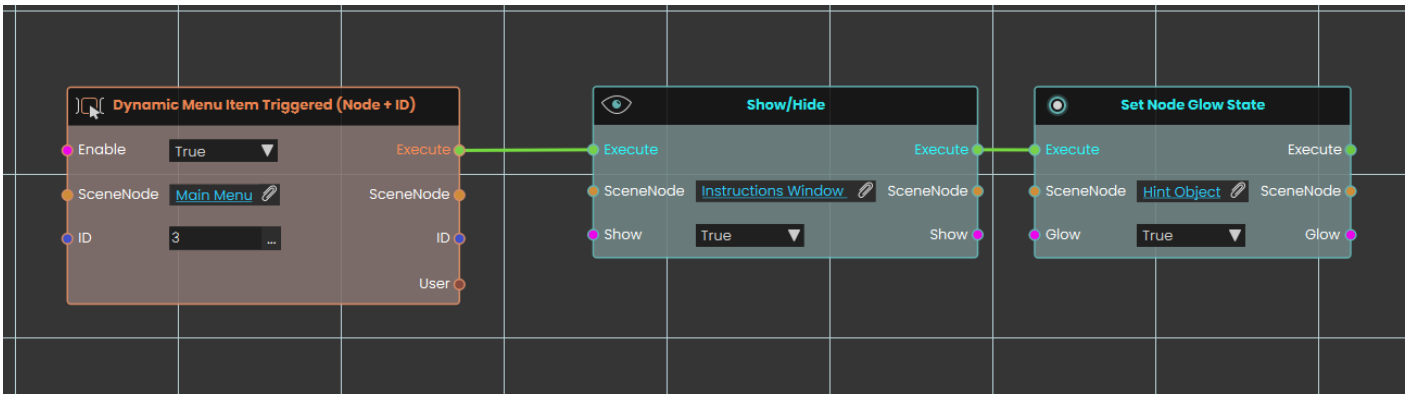
The **Dynamic Menu Item Triggered (ID)** event node activates only when a button with a specific **ID** is pressed. It ignores all other button interactions, allowing you to trigger a specific function

Dynamic Menu Item Triggered(Node + ID)



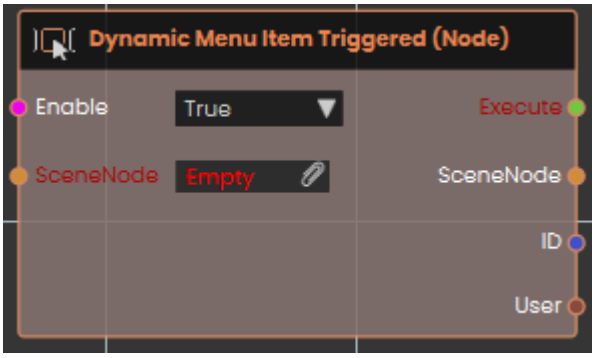
The **Dynamic Menu Item Triggered (Node + ID)** event node provides the most specific control by activating only when a button with a precise **ID** is pressed on a menu linked to a specific **SceneNode**. This prevents execution overlap by ensuring the logic only triggers for one unique button on one specific object.

Example:



In this example, the **Dynamic Menu Item Triggered (Node + ID)** node is configured to act as a "Help" button. By setting the **SceneNode** to "Main Menu" and the **ID** to "3", the system listens exclusively for that exact button press on that specific menu object. Once the user clicks this button, the execution flow passes to a **Show/Hide** node configured to display an "Instructions Window" by setting its state to True. Immediately following this, the flow triggers a **Set Node Glow State** node, which applies a visual highlight to a designated "Hint Object" to guide the user.

Dynamic Menu Item Triggered(Node)



The **Dynamic Menu Item Triggered (Node)** event node activates only when a button is pressed on a menu attached to a designated **SceneNode**. This node isolates interactions to a single object, ensuring that the logic only runs for the menus linked to that specific item in the scene.

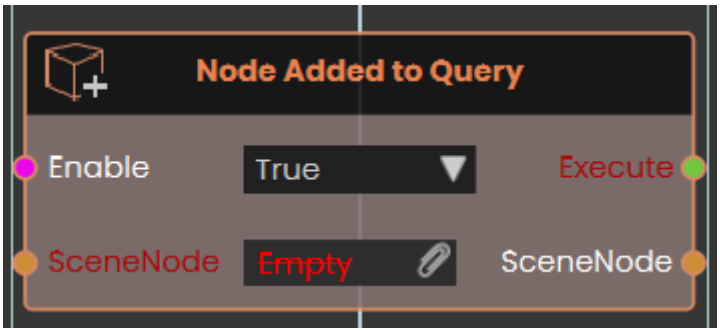
Dynamic Menu Triggered nodes are used in the following add-ons:

https://www.youtube.com/embed/8NtfNkBr_Z0

https://www.youtube.com/embed/7Ct8UpF9ce4

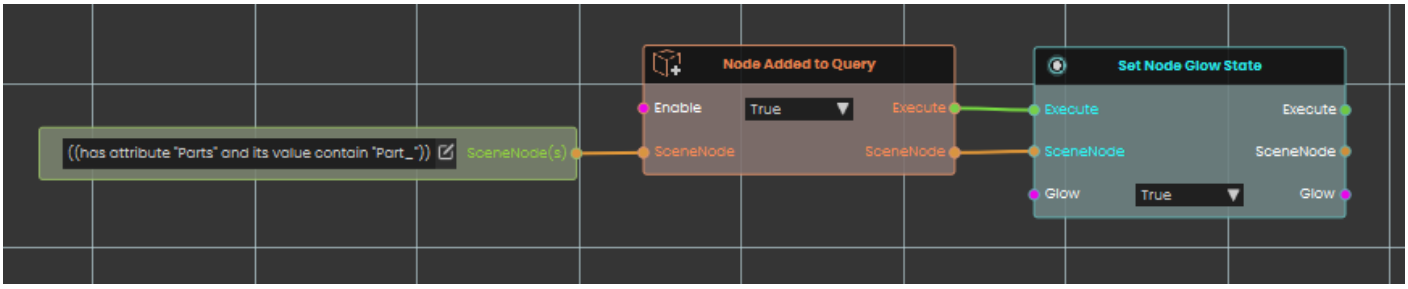
SceneNode \ Query

Node Added to Query



The Node Added to Query event node constantly monitors the scene and triggers its Execute output whenever a 3D object's attributes are modified to match the specific conditions defined by a connected Scene Node Query. By plugging a query into the node's SceneNode input, it listens for any object that newly meets these criteria—essentially being "added" to the query's list of valid results—and subsequently passes both the execution signal and the specific SceneNode that triggered the event through their respective outputs, allowing for dynamic, attribute-driven logic without needing to manually target individual objects.

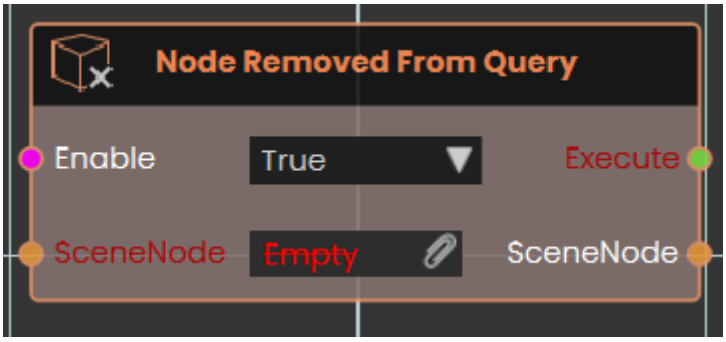
Example:



In this example, a Scene Node Query is configured to continuously look for any object that has an attribute named "Parts" with a value containing "Part_". This query is connected directly to Node Added to Query event node. During the experience, whenever an object

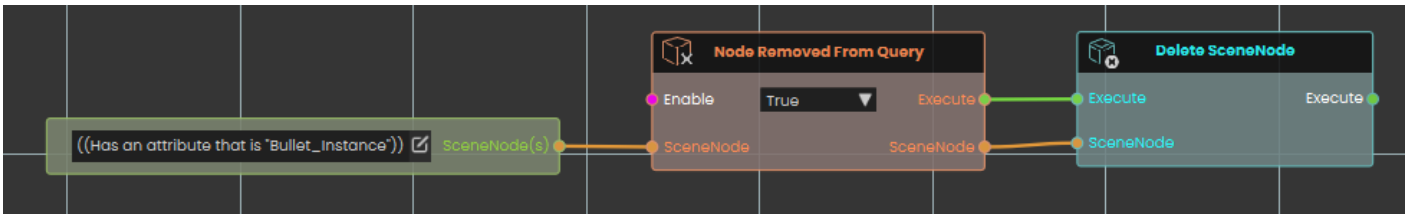
has its attributes updated to successfully meet this exact condition, the event node automatically triggers and passes that specific object through its SceneNode output into a Set Node Glow State node.

Node Removed From Query



The Node Removed From Query event node continuously monitors the scene and triggers its Execute output whenever a 3D object's attributes change so that they no longer match the conditions of a connected Scene Node Query. By connecting a query to the node's SceneNode input, it watches for any object that falls out of the query's criteria—essentially being "removed" from the list of valid results.

Example:

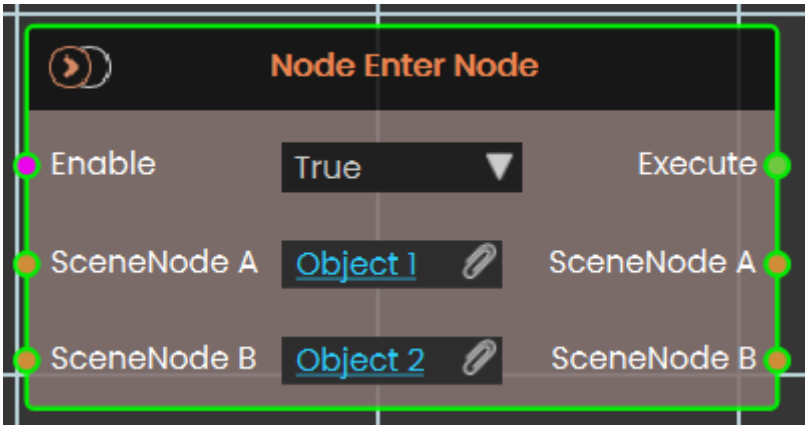


In this example, a Scene Node Query is configured to track any object that has an attribute named "Bullet_Instance". This query is connected directly to the SceneNode input of the Node Removed From Query event node. During the experience, if an object's attributes are altered so it no longer possesses the "Bullet_Instance" attribute (for example, if the attribute is removed after a collision), the event node automatically triggers and passes that specific object through its SceneNode output into a Delete

SceneNode node.

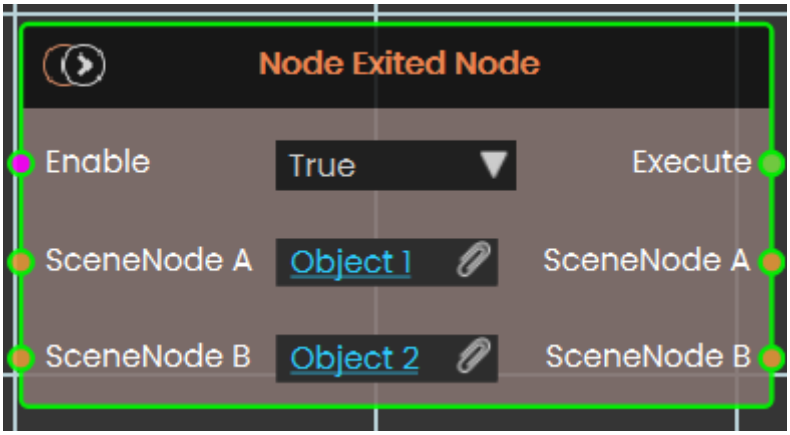
Scene Node \ Overlap

Node Enter Node



The **Node Enter Node** event is triggered when Scene Node A enters Scene Node B. Both nodes can be any nodes in the assembly tree — for example, Node A could be a screwdriver and Node B could be a screw.

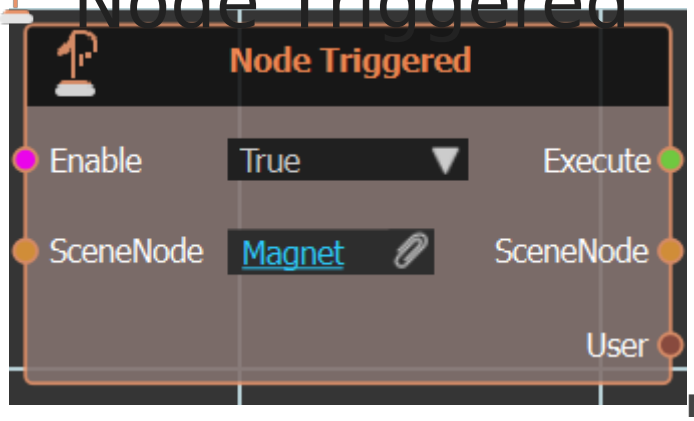
Node Exited Node



The **Node Excited Node** event is triggered when Scene Node A exits Scene Node B. Both nodes can be any nodes in the assembly tree — for example, Node A could be a screwdriver and Node B could be a screw.

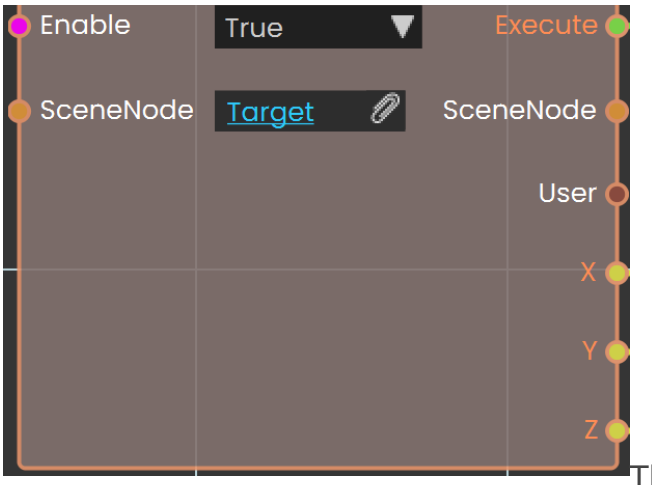
SceneNode \ Triggered

Node Triggered



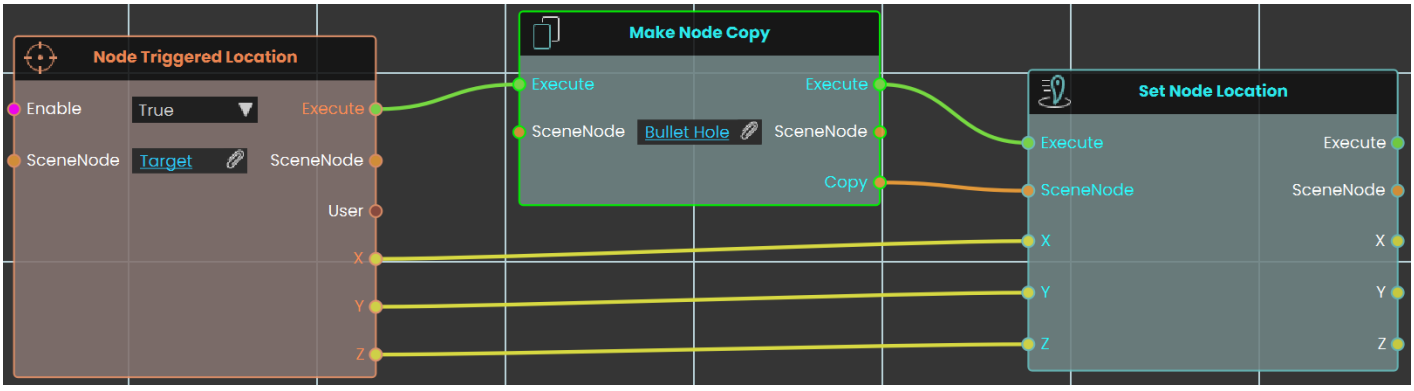
Node Triggered event is executed when the selected Scene Node is triggered within VR environment.

Node Triggered Location



The **Node Triggered Location** is used to capture the specific location or position of a triggered event within the VR environment. It enables tracking where a particular interaction occurred, which can be useful for dynamic object placement, analytics, or event-triggered responses based on spatial coordinates.

Example

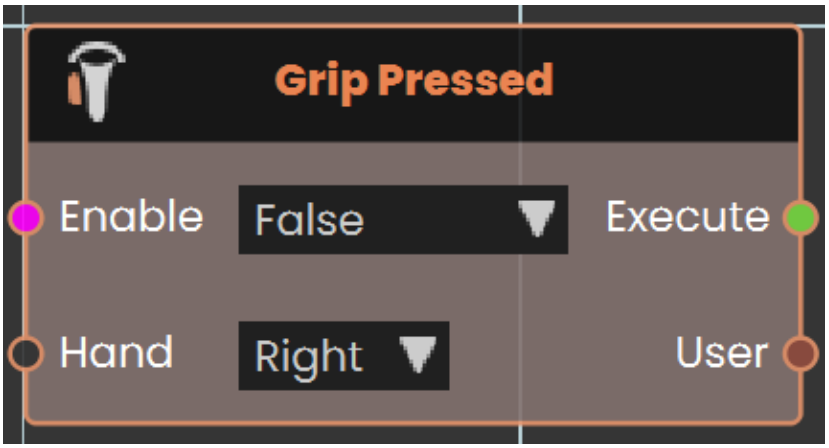


In this example, the **Node Triggered Location** is used to capture the location where the user triggers the target. When the target is hit, the node records the exact spot, and a bullet hole is placed using the **Set Node Location**. The **Make Node Copy** is used to enable the user to shoot more than one bullet, creating a new bullet hole at each triggered location. This setup allows for repeated interactions, with each shot creating a

new bullet hole in the correct spot.

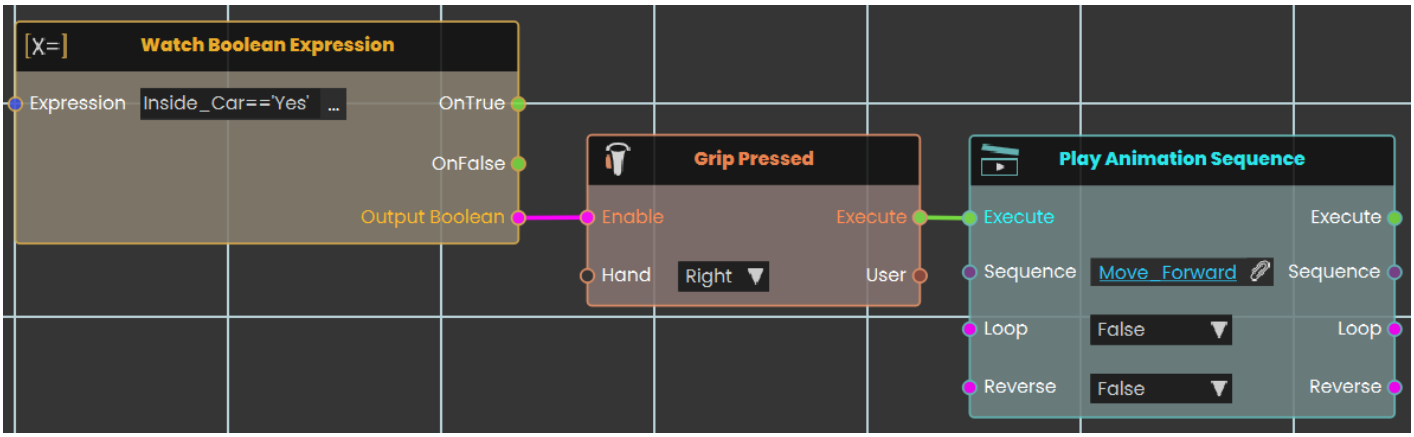
Controller

Grip Pressed



The **Grip Pressed node** enables the user to detect when the VR controller's grip is pressed. By connecting a response to this node, the specified action is executed each time the grip is pressed, allowing for interactive controls within the VR Experience.

Example

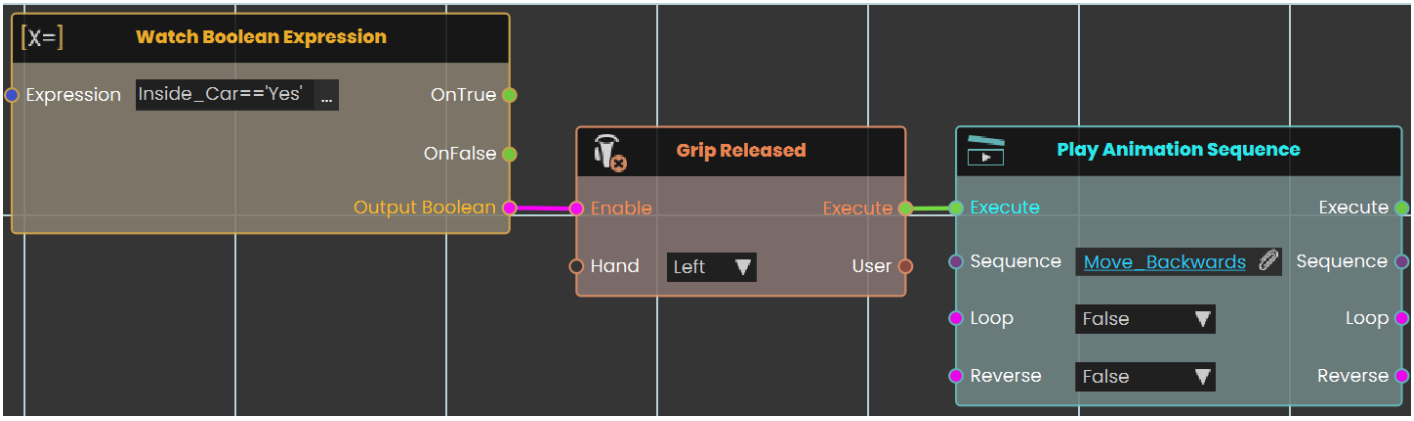


In this example, a **Grip Pressed node** is used to execute a response. Once the Grip press is triggered, the Grip Pressed node checks the boolean value. If the condition is met, the connected response executes, and the animation plays during the VR Experience.

Grip Released

The **Grip Released node** enables the user to detect when the VR controller's grip is released. By connecting a response to this node, the specified action is executed each time the grip is released, allowing for interactive controls within the VR Experience.

Example



In this example, a **Grip Released node** is used to execute a response. Once the Grip released is triggered, the Grip Released node checks the boolean value. If the condition is met, the connected response executes, and the animation plays during the VR Experience.

Hand

Hand gestures in VR are becoming more natural and intuitive than ever. No need for the user to fully close hand to grab objects — now, a drawer can be opened effortlessly using just two fingers, just like in real life.

With enhanced gesture control, users can interact with virtual world in a way that feels smooth, and precise, check this tutorial.

https://www.youtube.com/embed/YkPAQTO5GKE

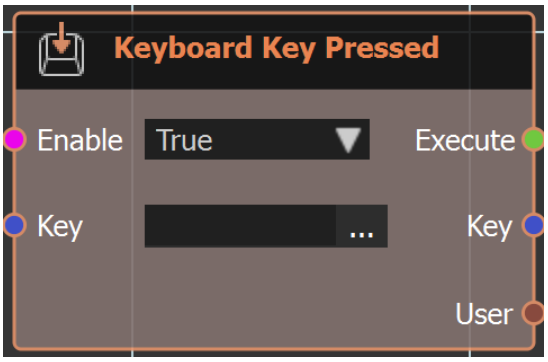
Keyboard

Adding an interactive 3D menu in VR is simple, but using the same menu on a desktop can be challenging, as you have to adjust your view every time you want to make a change.

To solve this, "**Key Press**" and "**Key Release**" events were introduced. By separating these events, designers now have more control when creating VR experiences.

Key Pressed

Right click in the Dynamic Training Builder and type in the



"Keyboard" or look under "**Events**" to get this node. As the name implies when a keyboard key is pressed this event is triggered.

Example

Key Released

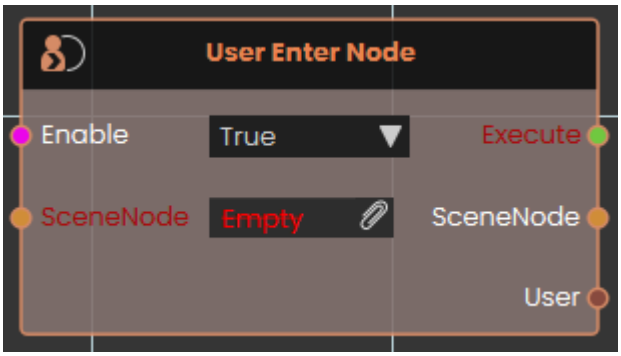
Same as above node but when key is released.

Check this tutorial to learn more about these events.

https://www.youtube.com/embed/ACQPYu1sdf0

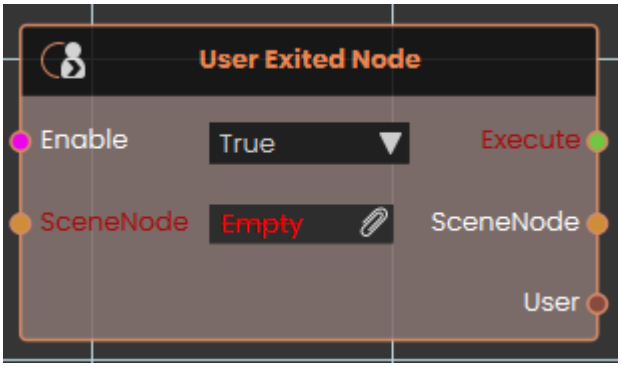
User

User Enter Node



The **User Enter Node event node** continuously monitors the physical boundaries of a specific 3D object and activates the moment a user's avatar collides with or enters its volume. While enabled, it passes forward a reference to the specific User involved before continuing the logic flow. This functionality applies to all participants within multiplayer VR collaboration scenes.

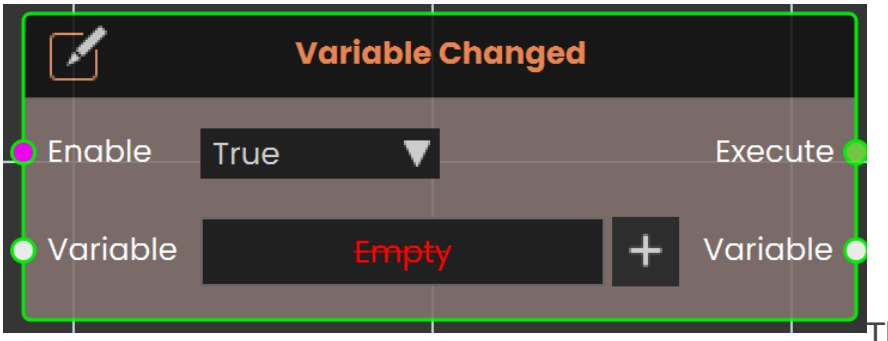
User Exit Node



The **User Exited Node event node** tracks the physical boundaries of a specific 3D object and activates the exact moment a user's avatar stops colliding with or fully exits its volume. While enabled, the node watches the targeted SceneNode and triggers its output as soon as the user completely separates from the object, passing forward a reference to the specific User who left before continuing the execution logic. And like **User Enter Node**, it also applies to all participants within multiplayer VR collaboration scenes.

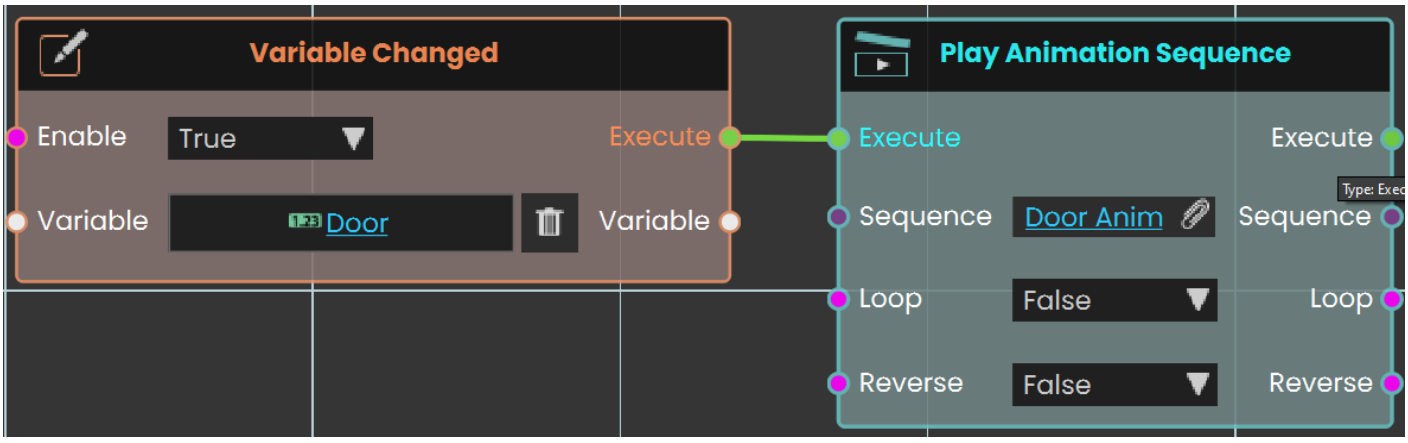
Variable

Variable Changed



The **Variable Changed** event activates when the value of a specified variable is modified. This event helps users detect real-time changes and trigger actions accordingly, enabling dynamic and responsive interactions within the VR Experience.

Example



In this example, the **Variable Changed** event is set to monitor changes in the variable **Door**. When the value of this variable is modified, the event triggers the **Play Animation Sequence** node, which activates the **Door Anim** animation sequence.

On Message Received

SimLab Composer 14 introduces WebSocket support, enabling seamless integration with external systems, allowing it to work with external hardware, co-simulation engines, and websites, unlocking numerous possibilities.

Use the node **On Message Received** to trigger an event when a message is received.

The **On Message Received** node triggers an event when an unhandled message is received from the external connection. This node allows you to access the incoming message and take appropriate action based on its content. It is useful for responding to external data or commands and processing messages within the VR Experience.

This node is related to Open External Connection node

Input Ports:

- Enable

Output Ports:

- Execute
- Message

The WebSocket nodes (Open External Connection, Send Message, and Receive Message) are exclusively available in the Ultimate Edition.

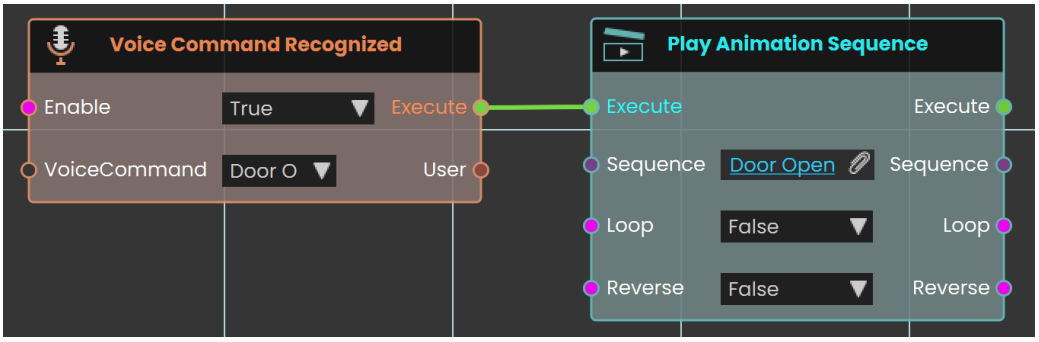
Voice Command

Voice Command Recognized

The **Voice Command Recognized event** is used to activate a response when the user says a specific command. This event listens for predefined voice commands and triggers the associated actions or responses within the VR environment upon recognizing the command.

Example

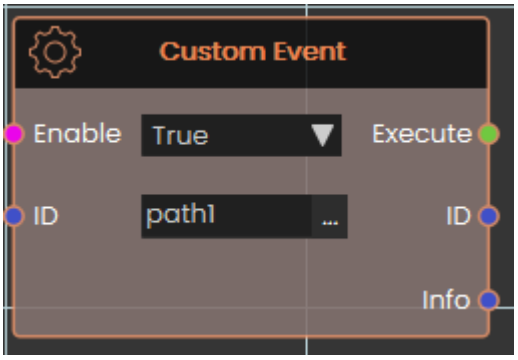
In this example, a new voice command named "Door Open" is created by accessing the Voice Command section from the Interaction menu and adding the command in the Voice Commands window.



The **Voice Command Recognized event** is used to link the newly created "Door Open" command. When the user says "Open the door", the door open sequence is triggered and plays.

Execution

Custom Event



This event is activated manually by the user. It's useful when you want multiple different paths or actions to lead to the same response.

To set it up, use the **Trigger Custom Event** response and assign it a unique ID (e.g., `path1`). You can place this trigger in as many locations as needed.

Whenever a **Trigger Custom Event** is called, it will execute the **Custom Event** that has a matching ID.

If you want to pass additional info, you can use the info field in **Trigger Custom Event** response, this data will be passed to the even in the info port

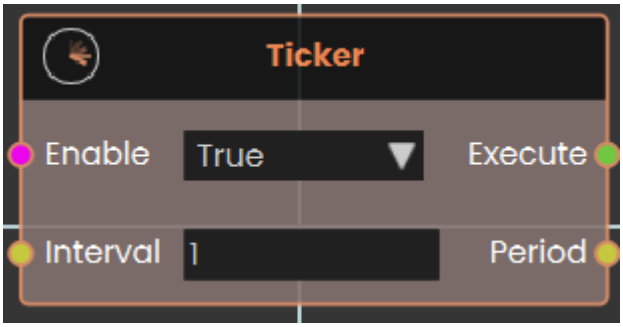
---

Scene Start

This event is triggered automatically when the scene begins. Use it to initialize elements at startup, such as playing videos, running background animations, or any other setup tasks.

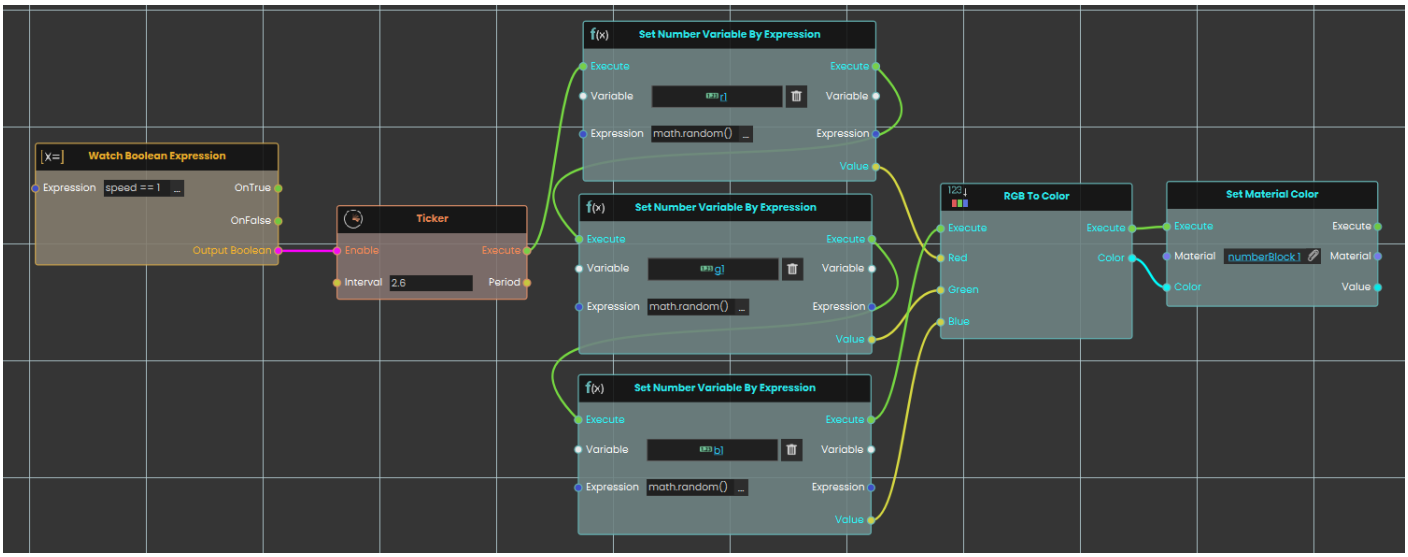
Note: Since this event fires the moment the scene loads, it's recommended to add a small delay before making additional scene changes — this ensures all scene elements have fully initialized before anything else runs.

Ticker



The **Ticker** event node functions as a continuous, time-based loop that repeatedly activates subsequent nodes at a consistent rate. As long as its Enable input is set to *True*, the node constantly triggers its Execute output every time the duration specified in the numerical Interval input (measured in seconds) elapses. Once activated, it continually outputs this time value through the Period pin, allowing the logic flow to perform recurring actions or background checks for as long as the node remains enabled.

Example:



In this example, we use the Ticker node to create a loop that randomly changes a material's color every 2.5 seconds:

1. A **Watch Boolean Expression** node continuously monitors the "speed1" variable and activates the **Ticker** node once the value equals 1.
2. While enabled, the **Ticker** node creates a continuous loop that triggers its execution output every 2.5 seconds.
3. The Ticker simultaneously executes three **Set Number Variable By Expression** nodes to generate random values using the `math.random()` function for the variables r1, g1, and b1.
4. These three random numbers are passed into an **RGB To Color** node, where they are combined to formulate a new color.
5. Finally, this new color is sent to a **Set Material Color** node, which immediately applies the randomly generated color to the target material.

Error

Fires whenever an error happens while your experience is running, and hands you a message describing what went wrong.

What it does

This node is a safety net. Unlike most nodes, you do not trigger it from an earlier node — it listens on its own in the background while your experience runs. If an error occurs, its **Execute** output fires and its **Error Message** output gives you the text of that error.

Instead of letting a problem pass unnoticed, you can respond to it — show the user a clear message, make a note of what happened, or move things to a safe state.

Turning it on and off

The **Enable** input controls whether the event is listening. It is **on by default**, so the node starts catching errors as soon as your experience begins. If you only want to catch errors during a certain part of your experience, connect a Yes/No value to **Enable** and switch it on and off as needed; while it is off, the event stays quiet and its **Execute** output will not fire.

Inputs

Port	Type	What to connect
Enable	True / false	*Optional.* Whether the event is listening for errors. **On** by default — leave it on to catch errors the whole time, or connect a Yes/No value to switch it on and off.

Outputs

Port	Type	What you get
Execute	Trigger	Fires the moment an error occurs while the event is enabled. Wire it to whatever should happen in response.
Error Message	Text	A short description of the error that occurred.

Example

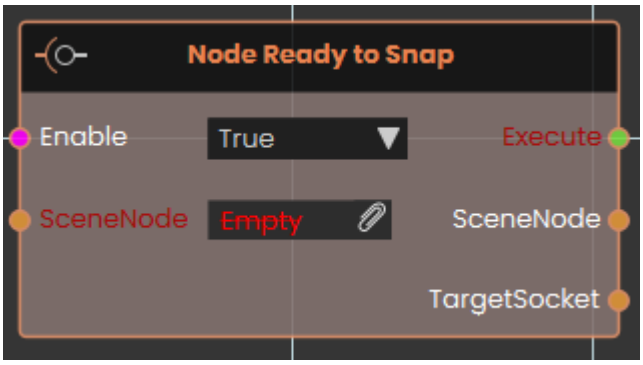
Enable input	**On** (the default)
Execute output	Fires when an error happens
Error Message output	`Could not load the requested file.` — the exact wording depends on what went wrong

Tips

- **Respond to the problem.** Wire **Execute** to something that handles it — show the user the **Error Message**, make a note of it, or move the scene to a safe state — rather than letting the error pass silently.
- **Show the message.** The **Error Message** output is plain text, so you can display it on a panel or pass it into another node.
- **Limit when it listens.** Leave **Enable** on to watch the whole time, or turn it off during parts of your experience where you would rather handle problems yourself.

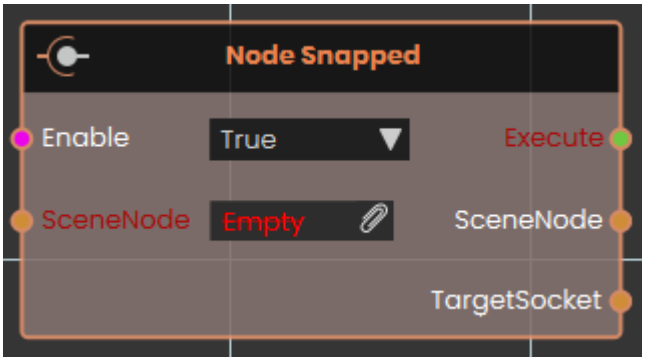
Snapping

Node Ready To Snap



The **Node Ready to Snap** event node activates the exact moment a designated SceneNode enters the valid snapping range of an eligible target. This node is highly useful for triggering immediate visual or audio feedback—such as highlighting a destination socket green or playing a hovering sound—letting the user know the object is ready to be released and snapped.

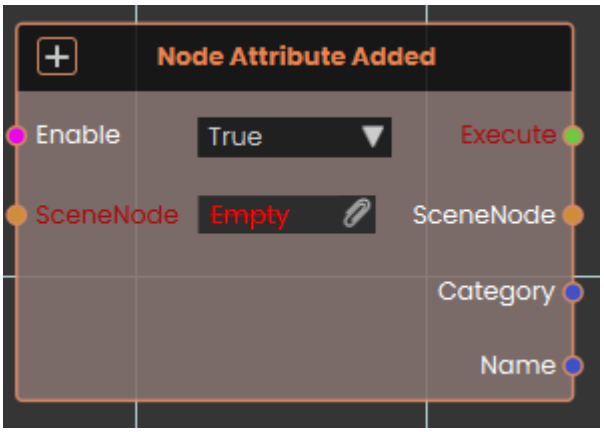
Node Snapped



The **Node Snapped** event node activates immediately after a SceneNode has successfully completed its snapping action onto a target socket. This event is typically used to seamlessly progress a sequence, such as locking the snapped part in place, playing a mechanical click sound, or advancing the user to the next step of an assembly training scenario.

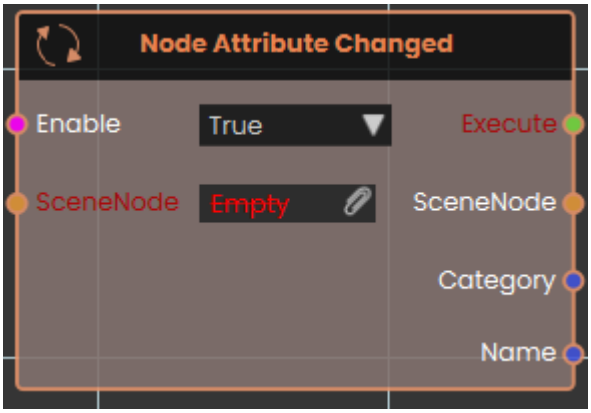
sceneNode \ Attributes

Node Attribute Added



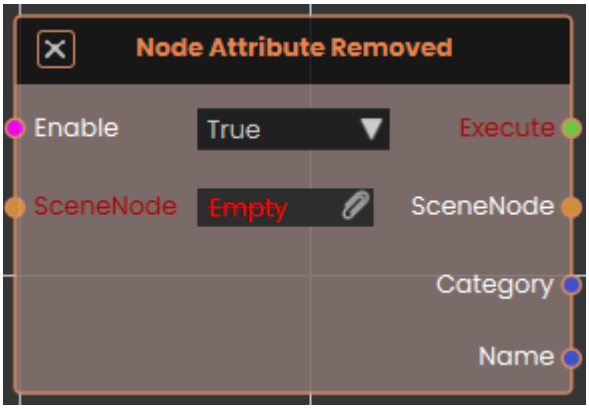
The **Node Attribute Added** event node continuously monitors the targeted SceneNode and activates whenever a completely new attribute is attached to it. This event is typically used in conjunction with **Set Node Attribute nodes** to trigger subsequent actions. Once activated, it outputs the affected SceneNode along with the specific Category and Name of the newly created attribute.

Node Attribute Changed



The **Node Attribute Changed** event node observes the targeted SceneNode and activates whenever the value of any of its existing attributes is modified. Once activated, the node outputs the affected SceneNode alongside the exact Category and Name of the altered attribute.

Node Attribute Removed



The **Node Attribute Removed** event node tracks the targeted SceneNode and activates the moment an existing attribute is deleted from the object. This event is typically used alongside the **Remove Node Attribute** node to initiate reactions to data clearance. Once activated, it outputs the affected SceneNode forward along with the specific Category and

Name of the deleted attribute.