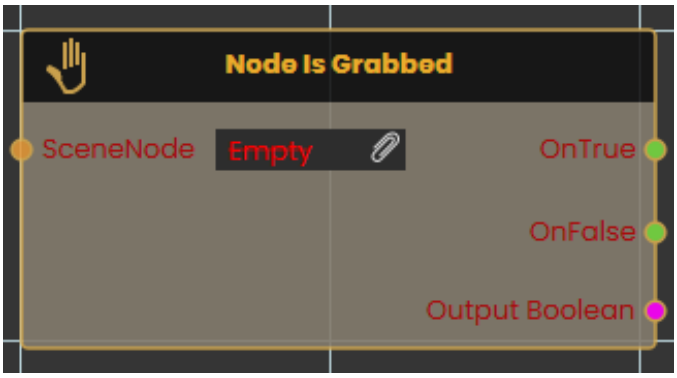
States

- Node Is Grabbed
- Watch Boolean Expression
- Boolean
- Assembly
- Overlap

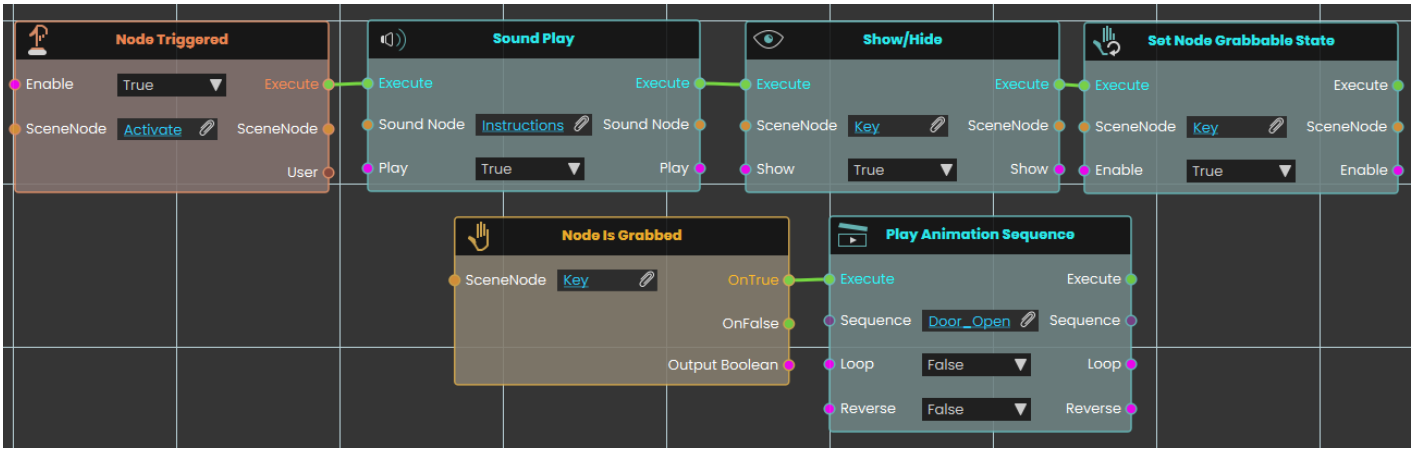
Node Is Grabbed

Node Is Grabbed



The **Node Is Grabbed** node enables the user to check if the assigned node is grabbed with **OnTrue/OnFalse** ports to execute responses or **Output Boolean** port that is compatible with event nodes.

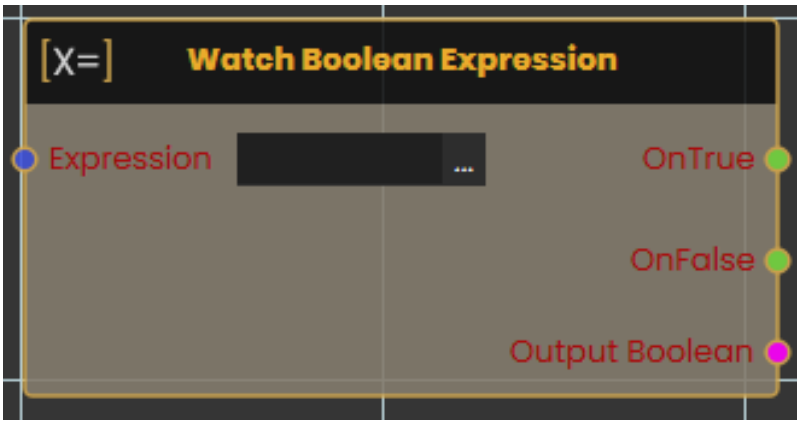
Examples



In this example, a **Node Is Grabbed** node is used to check if the node assigned is grabbed. Once the object named **Activate** is triggered, the sound named Instructions will play, and the object named Key will show up and become grabbable. Once the object named **Key** is grabbed, the sequence named **Door_Open** will play.

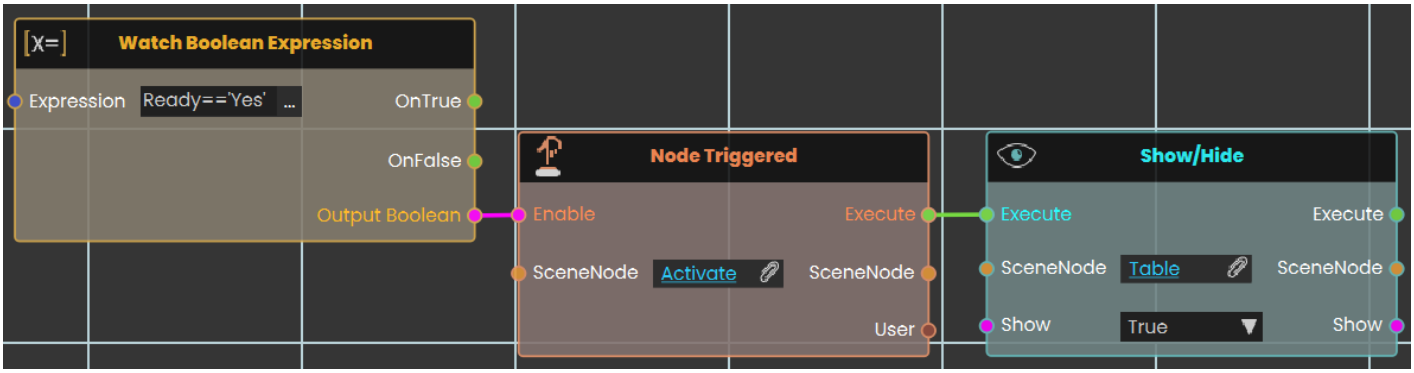
Watch Boolean Expression

[X=] Watch Boolean Expression

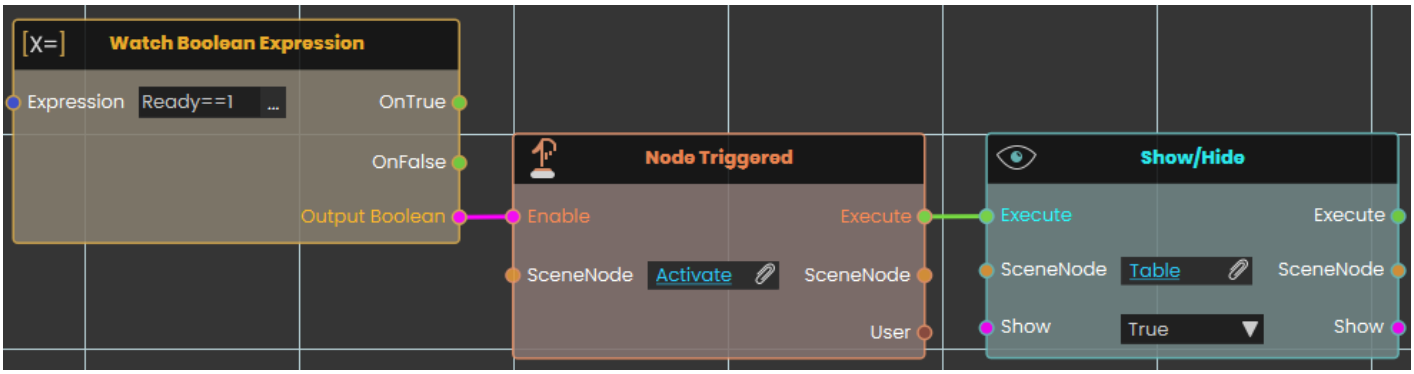


The **Watch Boolean Expression** node enables the user to check the value of specific variable(s) with **OnTrue/OnFalse** ports to execute responses or **Output Boolean** port that is compatible with event nodes.

Examples



In this example, a **Watch Boolean Expression** node is used to check the value of **a string variable** to determine if it is true or false, depending on the result, the object named Activate can be triggered. Once the event is triggered, the object named Table will be shown during the VR Experience.



In this example, a **Watch Boolean Expression** node is used to check the value of **a number variable** to determine if it is true or false, depending on the result, the object named Activate can be triggered. Once the event is triggered, the object named Table will be shown during the VR Experience.

Boolean

These nodes work with **true/false** values — the yes/no answers your scene produces, such as whether a door is open, a switch is on, or a trainee is standing in the right place. An operator takes one or two of these true/false values and works out a new one, so you can combine several conditions into a single answer and react to it.

What's on this page

- **And Operation** — the answer is true only when *both* values are true.
- **Or Operation** — the answer is true when *at least one* value is true.
- **Not Operation** — flips a single value to its opposite.

The **And** and **Or** nodes hand you the answer in two ways at once: a true/false **Result** you can read whenever you like, and two triggers — **OnTrue** and **OnFalse** — that fire at the moment the answer changes. Use whichever suits your scene.

True / false logic

And Operation

Checks two true/false values and tells you whether *both* of them are true.

What it does

This node looks at two true/false values you feed in and combines them: the result is true only when **both** are true. If either one is false — or both are — the result is false. Think of it as the word “and” in everyday speech: the door is unlocked *and* the trainee is wearing a helmet.

The node keeps watching both inputs and updates as they change. The **Result** output always holds the current answer, ready to read at any time. The two trigger outputs fire only at the moment the answer flips: **OnTrue** fires the instant the combined answer becomes true, and **OnFalse** fires the instant it becomes false. If an input changes but the overall answer stays the same, neither trigger fires.

Inputs

Port	Type	What to connect
Boolean A	True / false	The first true/false value you want to check — for example, whether a switch is on.
Boolean B	True / false	The second true/false value to check alongside the first — for example, whether a safety guard is closed. Both must be true for the result to be true.

Outputs

Port	Type	What you get
OnTrue	Trigger	Fires once at the moment both values become true. Wire this to whatever should happen when the combined answer turns true.
OnFalse	Trigger	Fires once at the moment the answer turns false — that is, when at least one of the two values stops being true. Wire this to whatever should happen when the combined answer turns false.
Result	True / false	The current combined answer: true when both values are true, otherwise false. You can read this at any time.

Example

Boolean A input	Machine power is on — `true`
Boolean B input	Safety guard starts open (`false`), then the trainee closes it (`true`)

OnTrue output	Fires the instant the guard closes — now both are true, so the answer flips to true and this trigger fires once (you could use it to start the machine)
Result output	`true`

Tips

- Use the **Result** output when you just need the current answer somewhere else; use **OnTrue** / **OnFalse** when you want something to happen the moment the answer changes.
- The triggers fire only on a change. If both values are already true when your scene starts, **OnTrue** won't fire until the answer leaves and returns to true.
- Need to check that *either* value is true instead of both? Use the **Or Operation** node. Need to flip a single true/false value? Use the **Not Operation** node.

Or Operation

Checks two true/false values and tells you whether at least one of them is true.

What it does

This node looks at its two true/false inputs and works out a single combined answer. The combined answer is **true** whenever at least one of the inputs is true, and only becomes **false** when both inputs are false. You can read this answer at any time from the Result output.

The node keeps watching both inputs and re-checks the answer whenever either one changes. The moment the combined answer flips to true, the **OnTrue** trigger fires once; the moment it flips back to false, the **OnFalse** trigger fires once. If an input changes but the overall answer stays the same, neither trigger fires.

Inputs

Port	Type	What to connect
Boolean A	True / false	The first true/false value to check — for example whether a door is open.
Boolean B	True / false	The second true/false value to check — for example whether a window is open.

Outputs

Port	Type	What you get
OnTrue	Trigger	Fires once at the moment the combined answer becomes true — that is, when at least one input has just turned true. Connect this to whatever should happen when that switch-on moment occurs.
OnFalse	Trigger	Fires once at the moment the combined answer becomes false — that is, when both inputs have just become false. Connect this to whatever should happen when that switch-off moment occurs.
Result	True / false	The current combined answer: true if at least one input is true, false only when both are false. You can read this at any time.

Example

Boolean A input	Door is open — `false`
Boolean B input	Window is open — changes from `false` to `true`
Result output	`true` (at least one is now open)
OnTrue output	Fires once, because the combined answer just flipped from false to true — you could use it to start a “something is open” warning.

Tips

- Use **Result** when you just want the current answer to feed into another node, and use **OnTrue** / **OnFalse** when you want something to happen the moment the answer changes.
- The triggers fire only on a change. If the answer is already true and an input change keeps it true, **OnTrue** will not fire again.
- If you need the answer to be true only when *both* inputs are true, use the **And Operation** node instead.



Not Operation

Flips a true/false value to its opposite.

What it does

This node takes a single true/false value and gives you back the reverse. If the value coming in is true, the result is false. If it comes in as false, the result is true.

It’s handy when you want to react to the opposite of something — for example, doing something only when a door is *not* open, or when a switch is *not* turned on. The value you connect is not changed; you simply get a new, flipped result out.

Inputs

Port	Type	What to connect
Input	True / false	The true/false value you want to flip — for example, whether a light is on or whether an object is visible.

Outputs

Port	Type	What you get
Output	True / false	The opposite of what you connected: `false` when the input is true, and `true` when the input is false.

Example

Input	`true` (the door is open)
Output	`false` — so “the door is not open” is false

Tips

- Use this when you want to act on the opposite of a condition without setting up a separate check — for example, to know when something is *not* happening.

Related

- **Branch** — use the **Result** from any of these nodes to send your scene down one path or another.
- Chain these three together to build a bigger check — for example, flip a value with **Not Operation** before feeding it into **And Operation** or **Or Operation**.

Assembly

The nodes on this page report on SimLab's **VR assembly** system. They are **states**: each one keeps an eye on a single part and continuously reports a simple true/false answer about where that part is in the assembly — for example "is it this part's turn to be put on?" or "is this part fully assembled?" A state is not an action you run once and finish; it is a live condition the scene keeps checking the whole time it is playing.

In a VR assembly, parts go together in a set order, and each part has a state of its own — think of a screw: all the way *out* (taken apart), *seated in place but not tightened* (partway together), or *tightened* (fully together). The nodes below let your scene react to where a part is in that process — whether it is this part's turn to be added or removed, and whether a part is completely assembled or completely apart.

Every one of these nodes gives you the same three things to work with:

- **OnTrue** — a trigger that fires the moment the answer becomes true.
- **OnFalse** — a trigger that fires the moment the answer becomes false.
- **Output Boolean** — the current true/false value, ready to read at any time or to send into a **Branch** node.

The two triggers fire only when the answer *changes* — not over and over while it stays the same. These nodes only watch and report; they never move, assemble, or change anything in your scene.

What's on this page

- **Node Can Assemble** — is it this part's turn to be put on next?
- **Node Can Disassemble** — is it this part's turn to be taken off next?
- **Node Fully Assembled** — is this part completely assembled (in place and secured)?
- **Node Fully Disassembled** — is this part completely taken apart (all the way out)?

Looking for other state nodes — checking whether two objects overlap, or whether the viewer is overlapping an object? See the **States** page.

Node Can Assemble

Reports whether a part in a VR assembly is currently allowed to be put on next — that is, whether it is this part’s turn in the assembly order.

What it does

In a VR assembly, every part has to go on in a set order, and some parts can only be added once the parts before them are already in place. This node watches one part and keeps answering a simple yes/no question: is it this part’s turn yet? The answer is `true` as long as everything that must come before this part has already been assembled, so it is now ready to be put on. While some earlier part is still missing, the answer is `false`.

It only watches and reports — it never assembles the part or changes anything in the scene. You connect what should happen on the “ready” side and the “not yet” side, and read the current answer whenever you like.

Inputs

Port	Type	What to connect
SceneNode	Scene node	The part you want to check — for example a `Bolt_01` object from your assembly. The node reports whether this part is currently allowed to be assembled next.

Outputs

Port	Type	What you get
OnTrue	Trigger	Fires once at the moment the part becomes ready to assemble — that is, when the last part that had to come before it is finally in place. Wire this to whatever should happen when it is this part’s turn, such as highlighting it or showing a hint.

Port	Type	What you get
OnFalse	Trigger	Fires once at the moment the part is no longer ready — for example if an earlier part is removed again, so it stops being this part's turn. Wire this to whatever should happen when the part is not yet ready.
Output Boolean	True / false	The current answer, ready to read at any time: `true` while the part is allowed to be assembled next, `false` while it is not yet its turn. Send this into a Branch node, or use it anywhere you need the current yes/no value.

Example

SceneNode input	`Bolt_01`
Output Boolean output	`true` — the `Housing` and `Gear` that come before `Bolt_01` are already assembled, so it is now the bolt's turn
OnTrue output	Fires the instant `Gear` is seated — the bolt is now ready, so you could highlight it to guide the trainee to fit it next

Tips

- Use the **Output Boolean** when you just need the current answer somewhere else (for example in a Branch); use **OnTrue** / **OnFalse** when you want something to happen the moment a part becomes ready or stops being ready.
- The triggers fire only on a change. If the part is already ready when your scene starts, **OnTrue** won't fire until the answer leaves and returns to true.
- This node only reports readiness — it never assembles the part. Combine it with your assembly steps to guide the trainee through the parts in the right order.

Node Can Disassemble

Reports whether a part is currently allowed to be taken off next in a VR assembly — that is, whether every part that has to come off before it has already been removed, so it's now this part's turn.

What it does

In a VR assembly, parts come apart in a set order — the pieces on top have to be removed before the ones underneath. This node watches one part and answers a simple live question: “Is it this part’s turn to be taken off?” The answer is true when all the parts that must be removed ahead of it are already off, and false while something on top of it is still attached.

This node only watches and reports — it never takes the part off or changes anything in the scene. The answer updates on its own as the assembly is taken apart, so you can use it to know exactly when a part becomes ready to remove.

Inputs

Port	Type	What to connect
SceneNode	Scene node	The part you want to watch — for example a bolt or a panel. The node reports whether this part is currently allowed to be taken off next.

Outputs

Port	Type	What you get
OnTrue	Trigger	Fires the moment the part becomes ready to be taken off — the instant it becomes this part’s turn. Wire it to whatever should happen then, such as highlighting the part or showing a “remove this” hint. It fires once at that moment, not continuously.
OnFalse	Trigger	Fires the moment the part is no longer the one allowed to come off next. Wire it to whatever should happen then, such as turning off a highlight. It fires once at that moment.
Output Boolean	True / false	The current answer as a true/false value — true while it’s this part’s turn to be removed, false otherwise. You can read it at any time or send it into a Branch node.

Example

SceneNode input	`Bolt_01`
OnTrue output	fires when the cover above `Bolt_01` has been removed and it's now the bolt's turn to come off
OnFalse output	fires if `Bolt_01` is no longer the next part allowed to be removed
Output Boolean output	`true` once the bolt is ready to be taken off

Tips

- Use **OnTrue** to guide the user — for example, light up the next part the moment it becomes removable so they always know what to take off next.
- This node reports readiness only; it does not take the part off. Pair it with your normal grab-and-remove interaction to act on the part itself.
- Send the **Output Boolean** into a Branch node when you want a check elsewhere in your logic to react to whether a part is ready to be removed.

 Node Fully Assembled

Watches one object and reports whether it is completely assembled — fully in place *and* secured.

What it does

This node keeps an eye on the object you connect and answers a simple yes/no question while your scene runs: is this object fully assembled? In SimLab's assembly system a part moves through three stages — think of a screw. With the screw all the way out, the part is disassembled. Once it is dropped into place but not yet tightened, it is only partially assembled. When it is finally tightened down, it is fully assembled. This node reports **true** only at that last stage, when the object is completely in place and secured. As long as the object is still loose, only seated, or not yet positioned, the answer stays **false**.

It only watches and reports — it never moves, tightens, or changes the object in any way. You use its answer to drive the rest of your scene: play a confirmation sound when a part is fully assembled, show a warning while it still isn't, or feed the current answer into a Branch node.

Inputs

Port	Type	What to connect
SceneNode	Scene node	The object you want to watch — the part whose assembly you care about, such as `Bolt_01`. The node reports on this object's own assembly state.

Outputs

Port	Type	What you get
OnTrue	Trigger	Fires once the moment the object becomes fully assembled. Wire it to whatever should happen when the part is finished — a success chime, a checkmark, the next step in your guide.
OnFalse	Trigger	Fires once the moment the object stops being fully assembled — for example if it is loosened or taken back apart. Wire it to whatever should happen when the part is no longer complete.
Output Boolean	True / false	The current answer — `true` while the object is fully assembled, `false` otherwise. You can read it at any time or send it straight into a Branch node.

Example

SceneNode input	`Bolt_01`
OnTrue output	Fires when `Bolt_01` is tightened all the way down — play a confirmation sound and mark the step complete.
OnFalse output	Fires if `Bolt_01` is later loosened or removed.
Output Boolean output	`true` once the bolt is fully assembled, `false` while it is still loose or only seated.

Tips

- A part that is only dropped into place but not tightened still counts as **false** here — this node reports true only when the part is both in place and secured.

- **OnTrue** and **OnFalse** each fire just once, right at the moment the answer changes. If you instead need to react to the answer continuously, read **Output Boolean** — it always holds the current value.

Node Fully Disassembled

Reports whether the object you connect is completely taken apart — all the way out of place, not just loosened.

What it does

Connect one object and this node keeps a live yes/no answer to the question “is this part fully disassembled?” In an assembly scene a part can be in three states — think of a screw: all the way out (**fully disassembled**), dropped into place but not tightened (**partially assembled**), or tightened down (**fully assembled**). The answer here is **true** only when the object is all the way out, and **false** while it is partially assembled or fully assembled.

The node only watches and reports — it never moves the object or changes anything in the scene. The answer can flip at any time as the user works: each time it flips, one of the two triggers fires, and a true/false value is always available to read.

Inputs

Port	Type	What to connect
SceneNode	Scene node	The object you want to watch — the part whose assembly state you care about, for example a `Bolt_01` . The node reports on this object’s own state.

Outputs

Port	Type	What you get
OnTrue	Trigger	Fires once at the moment the object becomes fully disassembled — the instant it comes all the way out of place. Wire this to whatever should happen when the part is completely removed.

Port	Type	What you get
OnFalse	Trigger	Fires once at the moment the answer turns false — that is, the instant the object is no longer fully out (it has been seated back into place, even loosely). Wire this to whatever should happen when the part is back in.
Output Boolean	True / false	The current answer, ready to read at any time: **true** while the object is fully disassembled, **false** otherwise. Send it into a Branch node, or use it anywhere a true/false value is needed.

Example

SceneNode input	`Bolt_01`
OnTrue output	Fires the instant `Bolt_01` is backed all the way out — you could use it to mark the disassembly step complete.
OnFalse output	Fires the instant `Bolt_01` is seated back into place — even just dropped in, before tightening.
Output Boolean output	`true` while the bolt is fully out, `false` once it is seated or tightened.

Tips

- Use **Output Boolean** when you just need the current answer somewhere else; use **OnTrue** / **OnFalse** when you want something to happen the moment the part comes out or goes back in.
- The triggers fire only on a change. If the part is already fully out when your scene starts, **OnTrue** won’t fire until it is put back and then removed again.
- “Partially assembled” (in place but not tightened) still reports **false** here — this node turns true only when the part is all the way out. To react to the tightened, fully-in state instead, use the matching fully-assembled state node.

Overlap

The nodes on this page are **states**. Each one keeps an eye on part of your scene and continuously reports a simple true/false answer about it — for example “are these two objects touching?” or “is the viewer standing inside this area?” A state is not an action you run once and finish; it is a live condition the scene keeps checking the whole time it is playing.

Every one of these nodes gives you the same three things to work with:

- **OnTrue** — a trigger that fires the moment the answer becomes true.
- **OnFalse** — a trigger that fires the moment the answer becomes false.
- **Output Boolean** — the current true/false value, ready to read at any time or to send into a **Branch** node.

The two triggers fire only when the answer *changes* — not over and over while it stays the same. These nodes only watch and report; they never move, assemble, or change anything in your scene.

This page covers the **overlap** states. For states that follow a part through a **VR assembly** — whether it can be put on or taken off, or is fully assembled — see the **Assembly** page.

Overlap

These nodes report whether things are sharing the same space in your scene. **Overlap** means two volumes intersect — they are touching or passing through one another. One node compares two objects you choose; the other compares an object against the **viewer** (the person moving through the scene in VR).

Nodes Overlap

Watches two objects in your scene and reports whether they are overlapping in space — that is, whether their 3D shapes share the same area.

What it does

You give this node two objects from your scene. It keeps watching them while your scene runs and answers one simple yes/no question: are these two objects overlapping right now? The answer is true whenever the two objects' 3D volumes share the same space, and false whenever they are apart.

This node only watches and reports — it never moves, changes, or touches the objects themselves. It gives you a true/false answer plus two triggers so you can make something happen the moment the two objects start touching, or the moment they come apart.

Inputs

Port	Type	What to connect
SceneNode A	Scene node	The first object to check — for example a `Gear`. It must be a different object from SceneNode B. The viewer's start position cannot be used here; to test the viewer against an object, use the User Overlap Node instead.
SceneNode B	Scene node	The second object to check — for example a `Housing`. It must be a different object from SceneNode A. The viewer's start position cannot be used here either.

Outputs

Port	Type	What you get
OnTrue	Trigger	Fires once the instant the two objects start overlapping. Wire it to whatever should happen the moment they come together.
OnFalse	Trigger	Fires once the instant the two objects stop overlapping and move apart. Wire it to whatever should happen the moment they separate.

Port	Type	What you get
Output Boolean	True / false	The current answer: true while the two objects are overlapping, false while they are apart. You can read it at any time or feed it into a Branch node.

Example

SceneNode A input	`Gear`
SceneNode B input	`Housing`
OnTrue output	Fires the moment the `Gear` slides into the `Housing` and they begin to overlap
OnFalse output	Fires the moment the `Gear` is pulled back out and the two no longer overlap
Output Boolean output	`true` while the `Gear` is inside the `Housing`

Tips

- OnTrue and OnFalse each fire just once, at the moment the answer changes — they do not keep firing while the two objects stay overlapping. If you need to act on the ongoing state rather than the moment it changes, read the Output Boolean instead.
- The two objects must be different. Pick two separate scene objects for SceneNode A and SceneNode B.
- To check whether the viewer is overlapping an object, use the User Overlap Node — the viewer’s start position is not accepted here.

User Overlap Node

This node reports whether the person viewing the scene is currently sharing the same space as a chosen object — in other words, whether the viewer is standing in or passing through it.

What it does

You give it one object from your scene. While the scene runs, the node keeps watching the viewer’s position and answers a single live question: is the viewer overlapping that object right now? The answer is `true` while the viewer’s space and the object’s space

share the same area, and `false` the rest of the time.

This node only watches and reports — it never moves, hides, or changes the object or the viewer in any way. You decide what happens by wiring its outputs to other nodes. The two triggers fire at the moment the answer changes: one the instant the viewer steps into the object, the other the instant they step back out. The true / false output always holds the current answer, so you can read it whenever you like.

Inputs

Port	Type	What to connect
SceneNode	Scene node	The object in your scene you want to test the viewer against — for example a doorway, a safety zone, or a part such as `Housing`. The node reports `true` while the viewer is sharing the same space as this object. (The viewer start position cannot be used here.)

Outputs

Port	Type	What you get
OnTrue	Trigger	Fires once the moment the viewer starts overlapping the object. Wire it to whatever should happen when the viewer steps in — play a sound, show a message, open a door.
OnFalse	Trigger	Fires once the moment the viewer stops overlapping the object. Wire it to whatever should happen when the viewer leaves — the matching “step out” reaction.
Output Boolean	True / false	The current answer as a `true` / `false` value: `true` while the viewer is inside the object, `false` otherwise. Read it directly or send it into a Branch node to choose between two paths.

Example

SceneNode input	`RestrictedZone`
OnTrue output	Fires when the viewer walks into `RestrictedZone` — wire it to a warning sound.
OnFalse output	Fires when the viewer leaves `RestrictedZone` — wire it to stop the warning.
Output Boolean output	`true` while the viewer is inside the zone, `false` when they are outside

Tips

- Use **OnTrue** and **OnFalse** as a matched pair when you want one reaction as the viewer enters and another as they leave — for example showing and then hiding a hint panel.
- When you only need to check the answer at a specific point (rather than react the instant it changes), feed **Output Boolean** into a Branch node instead.
- Pick an object whose space is big enough to comfortably contain the viewer, such as a room or a marked floor area, so the overlap is easy to trigger.