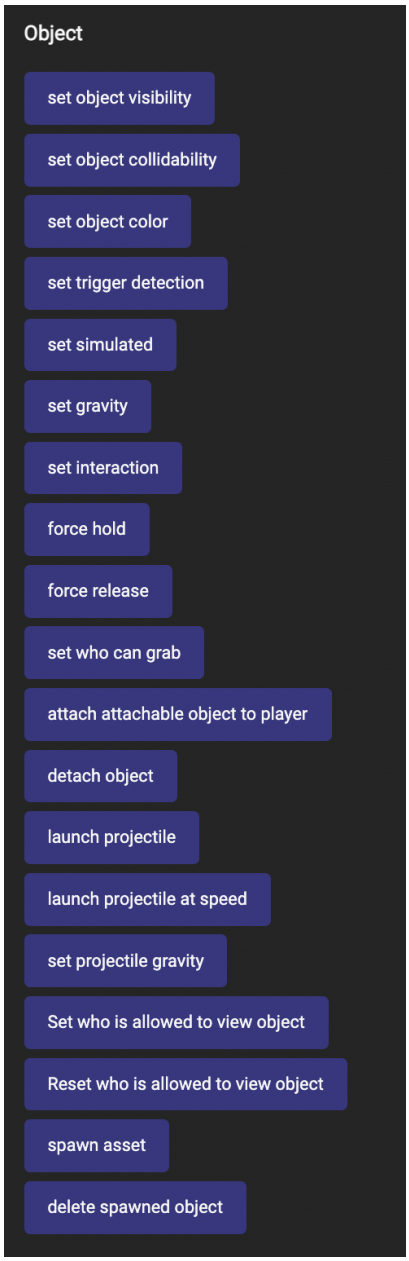
Actions

Object

Actions > Object



Codeblocks that control various object properties.

set object visibility

Actions > Object > set object visibility

Show or hide an object.

Appearance in Composition Pane

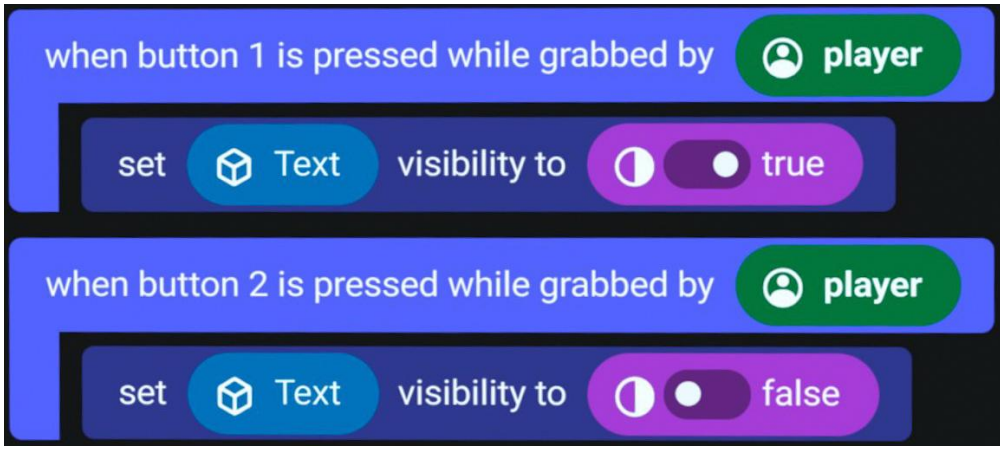
Description

Set object visibility will make an object visible or not, based on the boolean true or false.

Parameters

object, boolean

Example 1: Make a text object visible



When the player presses button 1 on their controller while holding an object with this script, attached, the text object becomes visible. When button 2 is pressed, the text is made invisible.

set object collidability

Actions > Object > set object collidability

Makes an object solid or not.

Appearance in Composition Pane

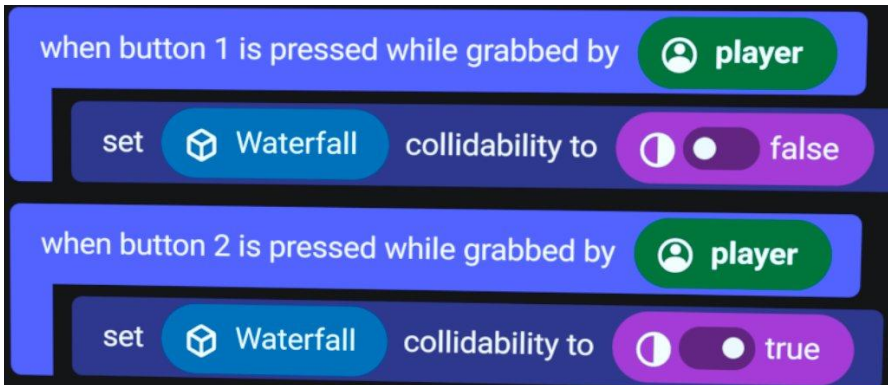
Description

Set object collidability will make an object non-collidable when false, and collidable when true.

Parameters

object, boolean

Example 1: Secret Waterfall



Imagine if this script was attached to a secret rock, and when button 1 is pressed the waterfall would become non collidable and you could access a secret cave! When button 2 is pressed the waterfall would become collidable again.

set object color

Actions > Object > set object color

Changes the color of an object.

Appearance in Composition Pane

Description

Set Object Color changes the color of an object, and accepts RGB (Red, Green, Blue) color values ranging from 0 to 1.

Parameters

object, color

Example 1: Color an object red

When trigger is entered by object we can color the **object** red.

See Also

- Values > Value Input > color input

set trigger detection

Actions > Object > set trigger detection

Enable or disable a trigger's ability to detect.

Appearance in Composition Pane

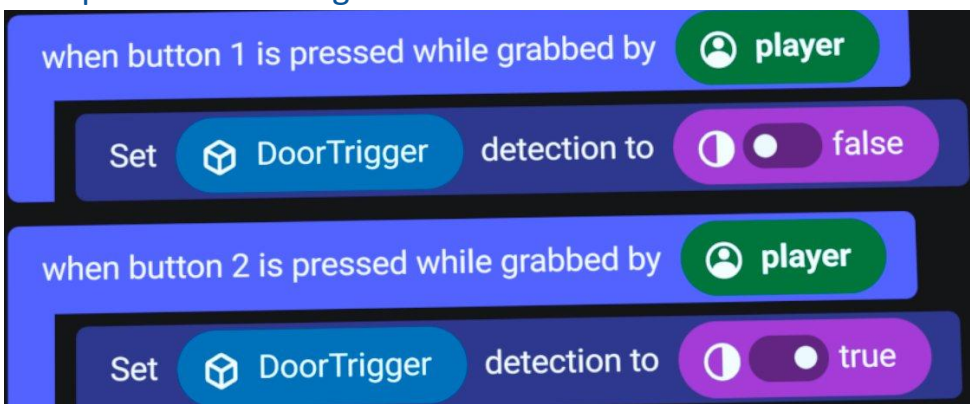
Description

Set trigger detection turns a trigger on or off, changing whether it can continue detecting objects or players, depending on which is selected in the object's property panel.

Parameters

object, boolean

Example 1: Lock Sliding Door



Imagine if this script was attached to a key, and when button 1 is pressed a sliding door trigger would become disabled. When button 2 is pressed the trigger would be reenabled allowing the door to open again.

set simulated

Actions > Object > set simulated

Turns on and off an object's "simulated" property.

Appearance in Composition Pane

Description

In an object's property panel you will find a toggle for simulated. If **set simulated** is *false*, the object cannot be moved by physics or grabbed by a player. If a player is holding an object when **set simulated** is set to false, the object is released from the player's hand.

Parameters

object, boolean

Set an object's simulation to the boolean true or false.

Example 1: Disable simulation with an event

The **when trigger is exited by object** we **set simulated** on the *obj* to disable its movement. You can imagine this event affecting a robot that mistakenly leaves its designated area.

See Also

- Actions > Object > set gravity

set gravity

Actions > Object > set gravity

Enables or disables gravity on an object.

Appearance in Composition Pane

Description

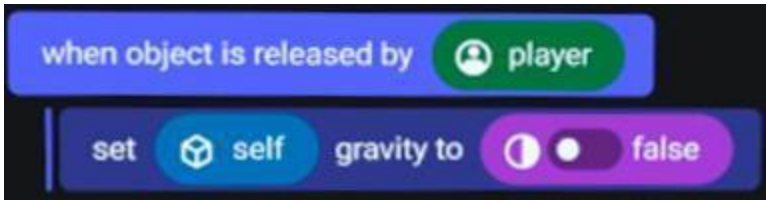
An object's gravity property can be found in its properties panel under physics. If **gravity** is set to *false*, then the object will act like it's floating in space. If *true*, the object will regain gravity.

Parameters

object, boolean

Sets the object's gravity to the boolean true or false.

Example 1: Disable gravity with an event



When object is released by player we **set gravity** to *false*. Imagine if this script was attached to an object released in a zero-gravity environment.

See Also

- Actions > Object > set simulated

set interaction

Actions > Object > set interaction

Set the interaction mode on an object; you can choose between Grabbable, Physics, or Both.

Appearance in Composition Pane

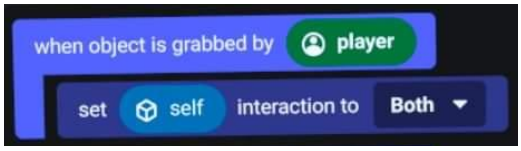
Description

An object must already have interaction enabled in its property panel for this code block to work. Then, this code block enables you to switch between the interaction modes (grabbable, physics, and both) while the world is running.

Parameters

object - the object to change the interaction on
[Grabbable/Physics/Both] - the new interaction mode for the object

Example: Pick an Apple



What it does: An apple is sitting in a tree. Once the player grabs it, it then has gravity when they let go (as if they had picked it from the tree).

How it works: The object starts off with interaction set to “Grabbable” in its property panel. When it is grabbed it then changes its model to “Both” meaning that it is now both grabbable and physics-enabled; so when the object is released, gravity will then act on the object (whereas it didn’t before).

force hold

Actions > Object > force hold

Force a player to automatically grab and hold an object.

Appearance in Library

Appearance in Composition Pane

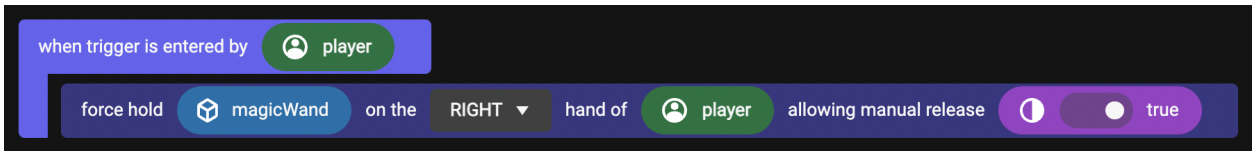
Description

Force hold affects an object causing a player to automatically grab it. You can specify whether or not the player can let go, otherwise it must be released with the *"force release"* code block.

Parameters

object - the object to be held.
hand - which hand the player will hold the object with.
player - the player that will now hold the object.
release - a boolean determining if the player can let go of the object.

Example: Auto-held magic wand



When the player walks into the trigger, the magic wand suddenly goes into the person's hand. They can let go of the wand if they like, since the last parameter is set to true.

See Also

Actions > Object > force release

Events > Grab Events > when object is grabbed by player

force release

Actions > Object > force release

Releases an object from a player's hand.

Appearance in Composition Pane

Description

Force release affects a grabbed object and causes the player to release it. **Force release** does not prevent future grabs.

Parameters

object

Example 1: You Don't Own Me

In this example, if the player is not our *indexplayer,* the *player* will be forced to release *self* immediately after grabbing it.

See Also

- Actions > Object > force hold
- Events > Grab Events > when object is released by player

set who can grab

Actions > Object > Set Who Can Grab

Set who can grab an object.

Appearance in Composition Pane

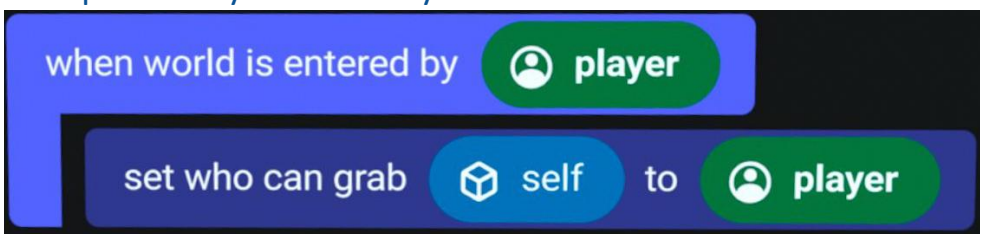
Description

Set who can grab an object allows you to specify players who can use an object. When updated it overrides any previous assignee(s). This means you may want to consider making a playerID list variable, adding that player to the list, and then setting who can grab to be that list of players, otherwise only the one person specified can grab.

Parameters

object, player or player list

Example 1: Only Newest Player Can Grab



In this example, only the most recent player to enter the world can grab the object.

See Also

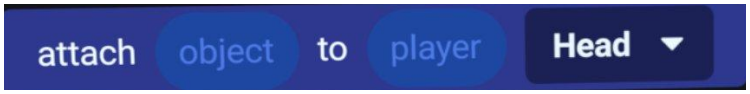
Operators > Lists > Add To List

attach attachable object to player

Actions > Object > attach attachable object to player

Attach an object to a player.

Appearance in Composition Pane



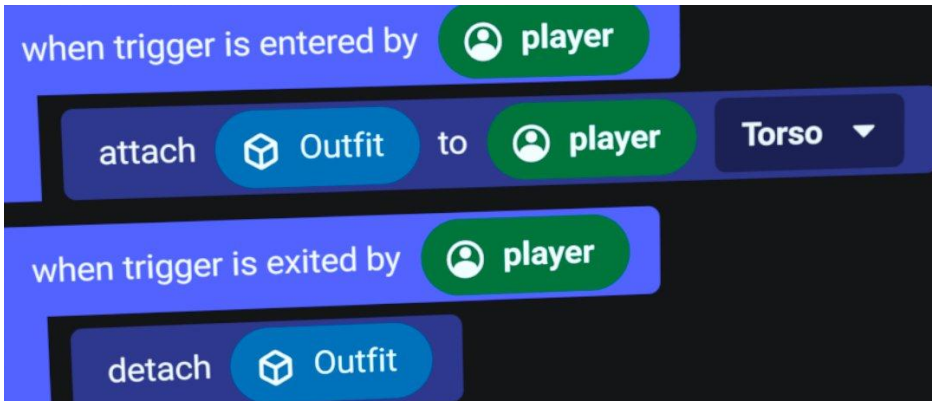
Description

Allows you to attach an attachable object to a player. You can adjust attach settings in the property panel of the attachable object.

Parameters

object, player, destination

Example 1: Try Me On



When a player walks into a trigger, the outfit is attached. And then detached when they exit the trigger.

detach object

Actions > Object > detach object

Detach an object that had been previously attached to a player.

Appearance in Composition Pane

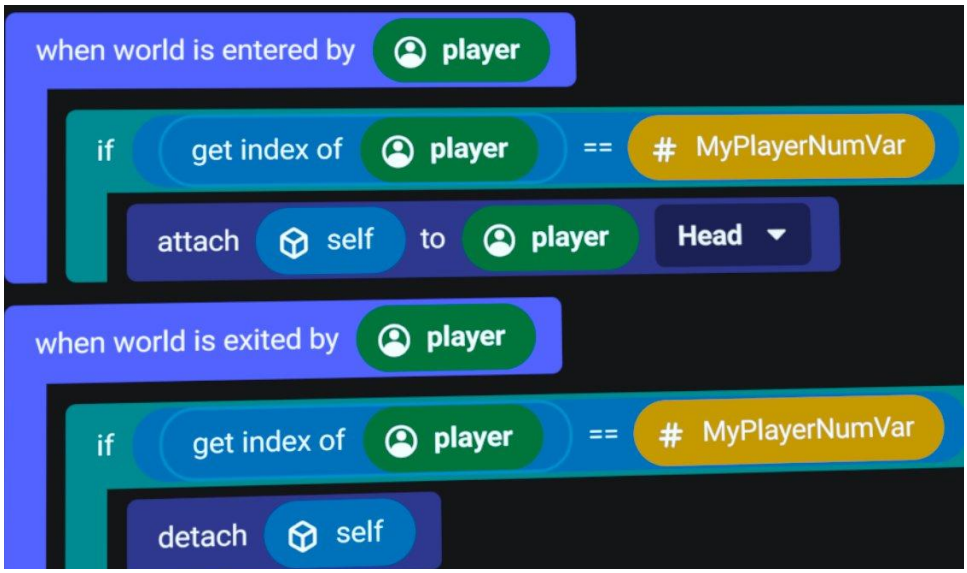
Description

Detaches an object that had been previously attached to a player. Works whether attached by a script or by the player themself.

Parameters

object

Example 1: Simple HUD



Attaching this script to an attachable object and setting the MyPlayerNumVar to be a number starting at 0, then duplicating a copy for each possible player that enters your world. You will then see that when the player enters the world, if their player index equals MyPlayerNumVar, self is attached, and when they leave the world it is detached. If you place a text object in the attachable object, it could even function like a simple HUD!

launch projectile

Actions > Object > launch projectile

Launches a projectile.

Appearance in Composition Pane

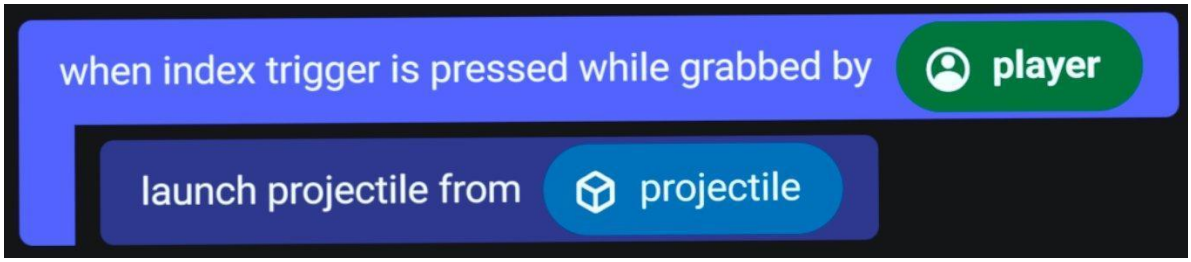
Description

Launches a projectile from a **projectile launcher gizmo** in the direction the projectile launcher is pointing. The **speed**, **gravity**, and other behaviors are set in the projectile launcher's properties.

Parameters

object

Example 1: launch a projectile



When a **player** grabs an **object** with this script attached and presses their **index trigger**, a projectile is launched from the reference *projectile launcher gizmo*.

launch projectile at speed

Actions > Object > launch projectile at speed

Launches a projectile at a set speed.

Appearance in Composition Pane

Description

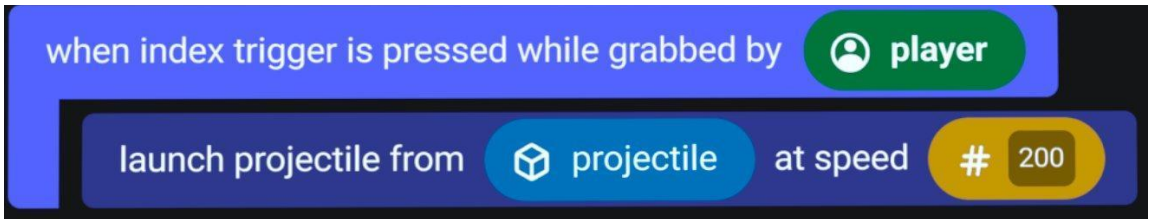
Launches a projectile at a number *speed* from a **projectile launcher gizmo**. The gravity and other behaviors can be set in the projectile launcher's property panel.

Parameters

object, number

Launches a projectile from the **object** at a **number** in meters per second.

Example 1: launch a projectile at 200 meters per second



When a player presses their **index trigger**, a projectile is launched from the *projectile launcher gizmo* at 200 meters per second.

set projectile gravity

Actions > Object > set projectile gravity

Sets the gravity of the projectile.

Appearance in Composition Pane

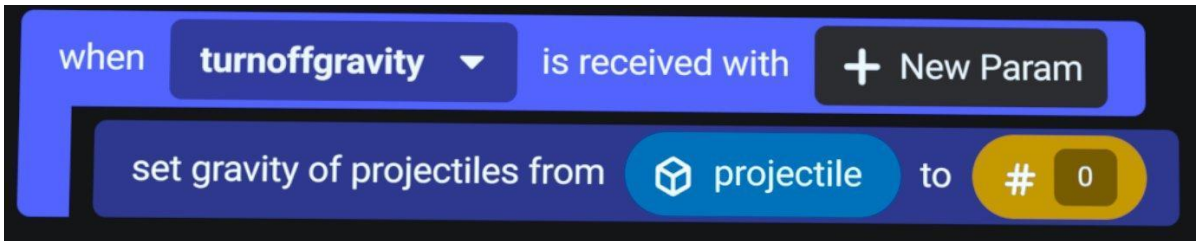
Description

Changes the gravity setting on a projectile launcher gizmo affecting its projectiles.

Parameters

object, number

Example 1: Turn off the projectile's gravity



When the ***turnoffgravity*** event is received, the gravity of the projectiles are set to 0.

set who is allowed to view object

Actions > Object > set who is allowed to view object

Set it so that only a specific player or set of players can see an object.

Appearance in Composition Pane

Description

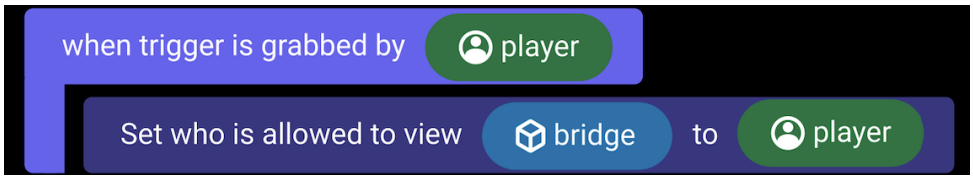
The collection of “allowed to view” codeblocks enable control over which players have permission to see an object. If you don’t have permission then you cannot see the object. If you do have permission, then whether you can see it is controlled by the object’s visibility (e.g. with the “set visibility” codeblock.

This codeblock sets the permissions for an object. You can pass in either a list of players or a single player variable and then only those players have permission to see the object.

Parameters

object - the object on which to enable per-player visibility settings.
player or player list - the list of players that can see the object. It will be invisible for everyone else. You can also pass in a player variable to set it so only that player can see the object.

Example: Grab the magic wand to see the bridge



What it does: There is a secret bridge across the valley that you can only see if you grab the wand.

How it works: When a grab event occurs on the magic wand, the person that grabbed the wand is set to be the only person that can see the bridge. In this example, the person can still see the bridge even if they let go (at least until someone else grabs it and then becomes the new person to be the only one to see the bridge).

See Also

- **Actions > Object > reset who is allowed to view object**
- **Actions > Object > get who is allowed to view object**

reset who is allowed to view object

Actions > Object > reset who is allowed to view object

Removing per-player visibility settings on an object so that all players see it the same.

Appearance in Composition Pane

Description

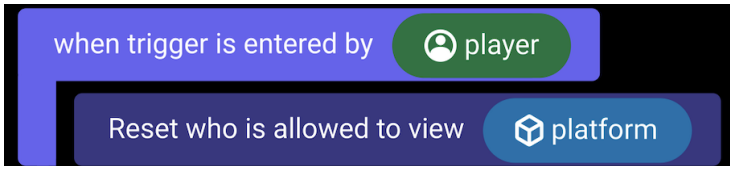
The collection of “allowed to view” codeblocks enable control over which players have permission to see an object. If you don’t have permission then you cannot see the object. If you do have permission, then whether you can see it is controlled by the object’s visibility (e.g. with the “set visibility” codeblock.

This codeblock removes all per-player permissions that have been set and returns the object to its default state, where everyone has permission and anyone new that joins will automatically have permission too (which is the default behavior for all objects in Horizon).

Parameters

object - the object on which to reset per-player visibility settings so that all players see the same for the item again.

Example: Once one person stands on the platform everyone can see it



What it does: There is a platform that players can earn the ability to see; once one person stands on the platform, everyone can then see it.
How it works: See the example for “set who is allowed to view object” to see how to make it so only one person can see an object. This example then adds the two lines of code above, where there is a trigger encompassing the platform, and once someone stands on it, the view permissions are reset (making it so that all players can see it - as long as its visibility hasn’t been set to false).

See Also

- **Actions > Object > set who is allowed to view object**
- **Actions > Object > get who is allowed to view object**

spawn asset

Actions > Object > spawn asset

Spawn a new instance of an asset into a running world. The spawn may fail if there isn't enough capacity available.

Appearance in Library

Appearance in Composition Pane

Description

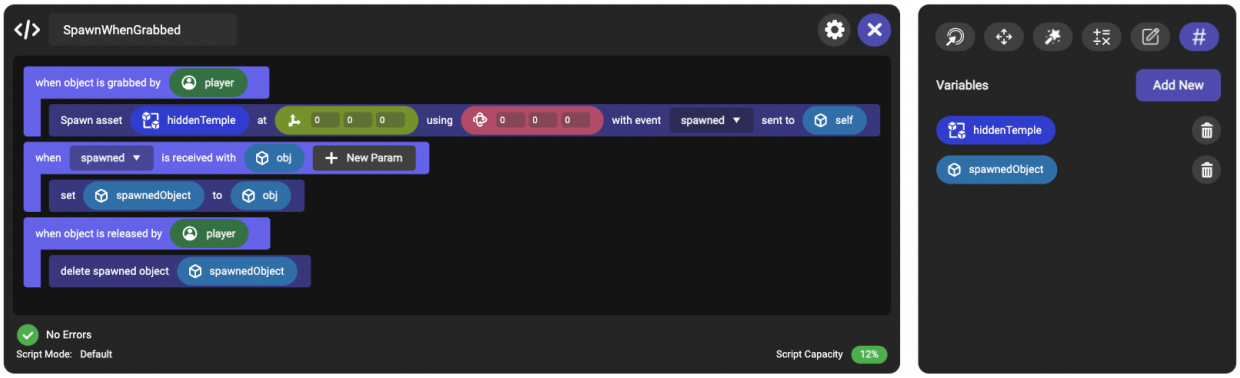
Dynamically spawn an asset into a world. This only works if there is enough capacity left in the world to spawn the asset. Once the asset spawns, Horizon will update that world's current capacity. As you spawn in assets the capacity will keep increasing. When you reach a point where there is no capacity left to spawn an object, this code block will fail. There is currently no way to detect that failure has occurred.

When an asset is spawned, the event you specify will be sent to the object you specify. This event will include a reference parameter to the newly spawned object.

Parameters

asset - The asset to spawn.
position - The location the object should be at when it spawns. If the spawned object is static then it cannot be moved again from this location.
rotation - The orientation the object should have when it spawns. If the spawned object is static then it cannot be rotated again from this orientation.
event - The callback event. When the object finishes spawning, this event will be sent to the object variable. The event is sent with an object parameter containing the spawned object.
object - The receiver object. When the object finishes spawning, the event will be sent to this object along with an object parameter containing the newly spawned object.

Example: Spawn an object when grabbed



What it does: This example spawns a hidden temple whenever a player grabs an object. The temple despawns if the player lets go of the object.

How it works: When a person grabs the object the script is one, we spawn the *hiddenTemple* asset and specify that the event *spawned* to be sent to *self* when the spawn completes. When we receive the *spawned* event we save the spawned object in a variable, so that we can despawn it when the object is released.

See Also

- **Actions > Object > delete spawned object**

delete spawned object

Actions > Object > delete spawned object

Delete an object that was previously spawned, removing the objects and freeing up their capacity.

Appearance in Library

Appearance in Composition Pane

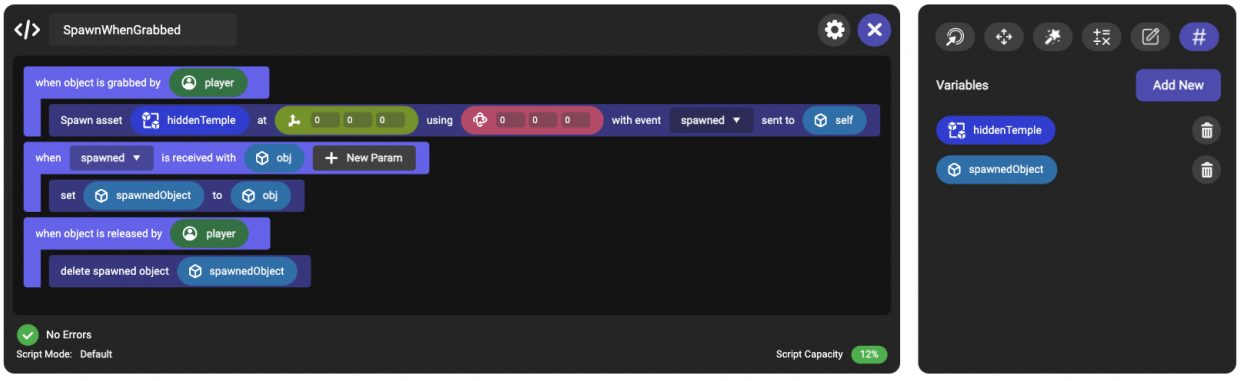
Description

If an object was spawned using the *"spawn asset"* code block then you can use this code block to delete it from the current running world, and free up the capacity that it took when it spawned.

Parameters

object - The object to delete. This object must have been spawned using the *"spawn asset"* code block.

Example: Spawn an object when grabbed



What it does: This example spawns a hidden temple whenever a player grabs an object. The temple despawns if the player lets go of the object.

How it works: When a person grabs the object the script is one, we spawn the *hiddenTemple* asset and specify that the event *spawned* to be sent to *self* when the spawn completes. When we receive the *spawned* event we save the spawned object in a variable, so that we can despawn it when the object is released.

See Also

Actions > Object > delete spawned object

Text

Actions > Text

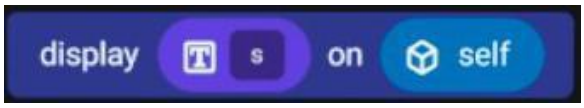
Text gizmo controls.

display text

Actions > Text > display text

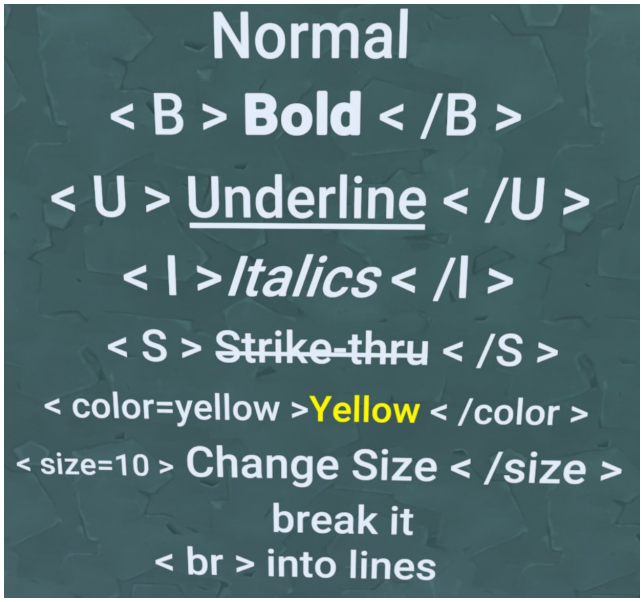
Sets the displayed text on a text gizmo.

Appearance in Composition Pane



Description

Display text controls what string value will be displayed on a text gizmo and does not affect the text gizmo's visibility. If a non-string variable is used, the value must be inside a **variable as string** codeblock. A text gizmo can display up to 1000 characters. Seen below are a few text formatting options from the "Intro to Gizmos" tutorial series: remove spaces when using:



Parameters

string, object

Example 1: Display a player's name

When player enters the world we **display text** to show the player's name on the text gizmo *self*.

See Also

- Operators > Player > name of player
- Values > Type Casting > variable as string
- Values > Value Input > string input
- https://www.oculus.com/horizon-worlds/learn/tutorial/text-formatting/

Animation

Actions > Animation

Controls an object’s recorded animation.

play animation

Actions > Animation > play animation

Plays an object's animation.

Appearance in Composition Pane

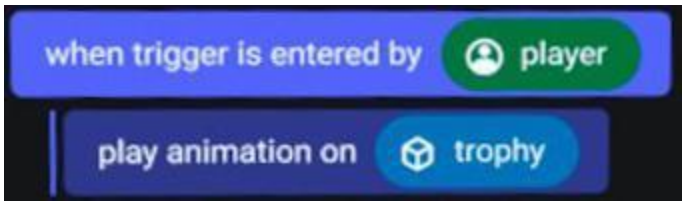
Description

Play animation will initiate playback of an object's animation that was previously recorded in its Properties Panel, via Behavior > Motion > Animated.

Parameters

object

Example 1: Play an animation



Imagine a player enters a trigger, causing a *trophy* to rise up.

See Also

- Actions > Animation > stop animation

pause animation

Actions > Animation > pause animation

Pauses an object's animation.

Appearance in Composition Pane

Description

Pause animation holds playback of an object's animation, this does not reset the animation.

Parameters

object

Example 1: Pause an animation

Imagine if this script was attached to a model train's remote control. **When index trigger is released** by a player, the **pause animation** codeblock pauses a *train*'s animation.

See Also

- Actions > Animation > play animation
- Actions > Animation > stop animation

stop animation

Actions > Animation > stop animation

Stops an object's animation.

Appearance in Composition Pane

Description

Stop animation will end playback of an object's animation, and it will reset the animation.

Parameters

object

Example 1: Stop and rewind an animation

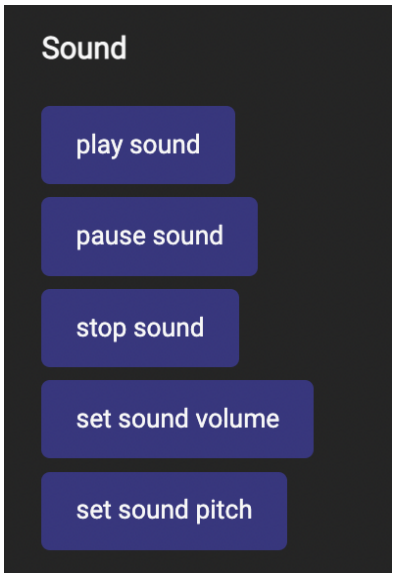
Imagine this script is attached to a model train's remote control. Because **stop animation** also rewinds an animation, **when button 1 is pressed** the *train* would return to its departure point.

See Also

- Actions > Animation > play animation

Sound

Actions > Sound



Controls for a sound, or audio gizmo.

play sound

Actions > Sound > play sound

Play a sound.

Appearance in Composition Pane

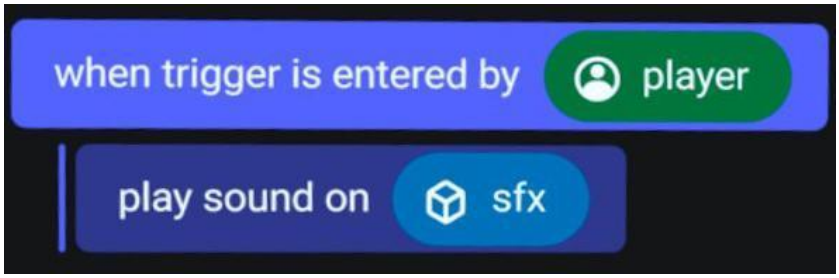
Description

Play sound will cause either a sound block or a sound recorder gizmo to play.

Parameters

object

Example 1: Play sound on trigger enter



Imagine a player walks into a puddle and a splash sound is played.

See Also

- Actions > Sound > stop sound

pause sound

Actions > Sound > pause sound

Pauses a sound.

Appearance in Composition Pane

Description

Pause sound will pause the playback of either a sound gizmo or a sound recorder gizmo. A subsequent call to **play sound** will resume playback from the point where it was paused.

Parameters

object

Example 1: Pause playback

Imagine a pause button that a player can press to pause music.

See Also

- Actions > Sound > play sound

stop sound

Actions > Sound > stop sound

Stop a sound.

Appearance in Composition Pane

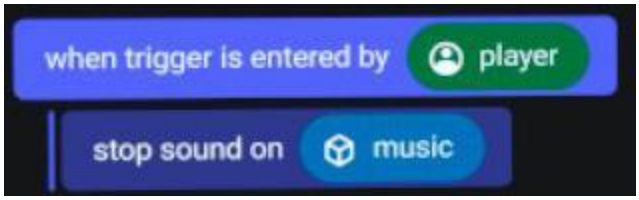
Description

Stop sound will stop and reset the playback of either a sound gizmo or a sound recorder gizmo.

Parameters

object

Example 1: Stop playback



Imagine a player being able to stop music playback in your world. Because **stop sound** rewinds the recording, a subsequent call to **play sound** will play the recording from its beginning.

See Also

- Actions > Sound > play sound

set sound volume

Actions > sound > set sound volume

Set the volume of a sound gizmo.

Appearance in Library

Appearance in Composition Pane

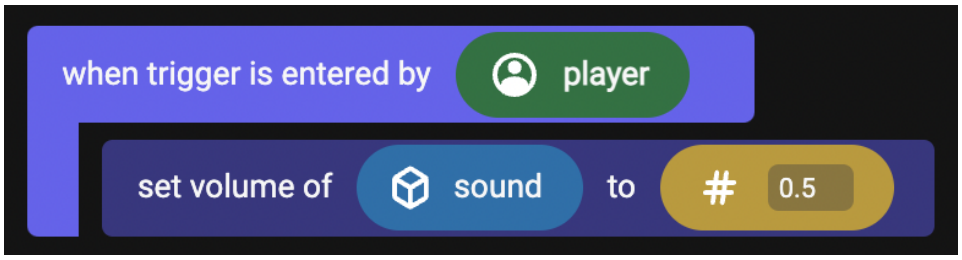
Description

This code block will instantly change the volume of a sound block to the value set in the number parameter of the code block. *Note: the range for setting the volume is from 0 to 1, representing the percent of full volume.*

Parameters

object (self) - the reference for the sound block.
number - the value needed to set the volume.

Example: tigger enter changes volume



What it does: When a player enters the trigger the volume reduces to 0.5 (i.e 50%) from 1.0 (i.e 100%).

How it works: When a player enters the trigger we use the code block on the sound gizmo to change the volume to 0.5 (i.e 50%), this will be an instant change to the volume of the sound block. Note that you are changing the volume of the sound to this number hence if you already have it set to 0.5 (i.e 50%) this will not add to it or change it.

set sound pitch

Actions > sound > set sound pitch

Set the pitch of a sound gizmo.

Appearance in Library

Appearance in Composition Pane

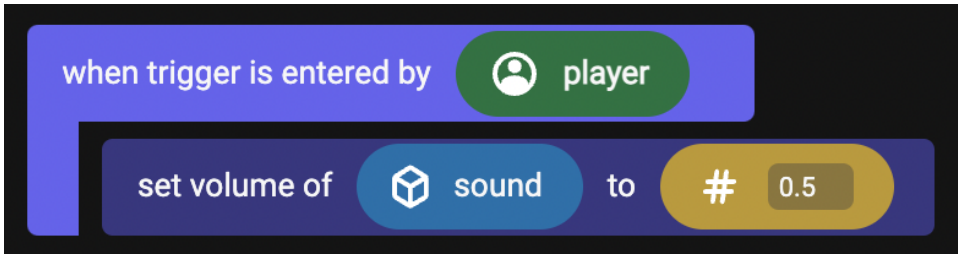
Description

This code block will instantly change the pitch of a sound block to the value set in the number parameter of the code block. *Note: the range for setting the pitch is from -24 to 24, You can not change the pitch of music gizmos.* The number represents musical semitones; e.g. a value of 12 raises the pitch 1 octave, -12 is down 1 octave, 24 is up 2 octaves, etc.

Parameters

object (self) - the reference for the sound block.
number - the value needed to set the pitch.

Example: tigger enter changes pitch



What it does: When a player enters the trigger the pitch of the sound block goes up from 0 to 10.

How it works: When a player enters the trigger we use the code block on the sound gizmo to change the pitch to 10, this will be an instant change to the pitch of the sound block. Note that you are changing the pitch of the sound to this number hence if you already have it set to 10 this will not add to it or change it.

VFX

Actions > VFX

Controls the behavior of a visual and trail effect gizmos.

play visual fx

Actions > VFX > play visual fx

Plays a visual or trail effect gizmo.

Appearance in Composition Pane

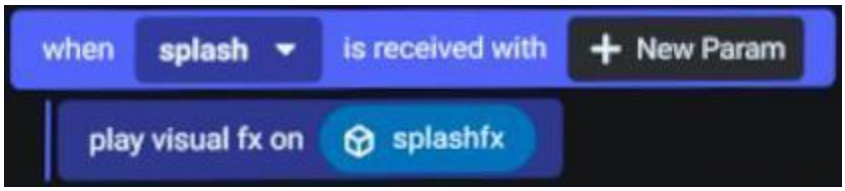
Description

Play visual fx on a TrailFx or ParticleFx gizmo.

Parameters

object

Example 1: Play a visual fx when splash is received



When the event *splash* is received, we **play visual fx** on the ParticleFx gizmo *splashfx*.

See Also

- Actions > VFX > stop visual fx

stop visual fx

Actions > VFX > stop visual fx

Stops a visual or trail effect gizmo.

Appearance in Composition Pane

Description

Stop visual fx on a TrailFx or ParticleFx gizmo.

Parameters

object

Example 1: Stop trail fx

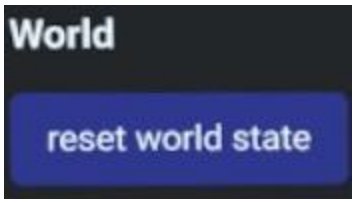
Imagine if this script was attached to a plane's remote control. When button 1 is released it would stop the trail fx.

See Also

- Actions > VFX > play visual fx

World

Actions > World



Codeblocks to control the world.

reset world state

Actions > World > reset world state

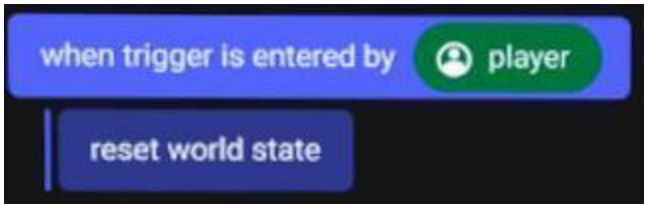
Resets the world back to its initial state.

Appearance in Composition Pane

Description

When called, **reset world state** resets the game world as if it had just been instantiated. Player positions and objects attached to players are not reset.

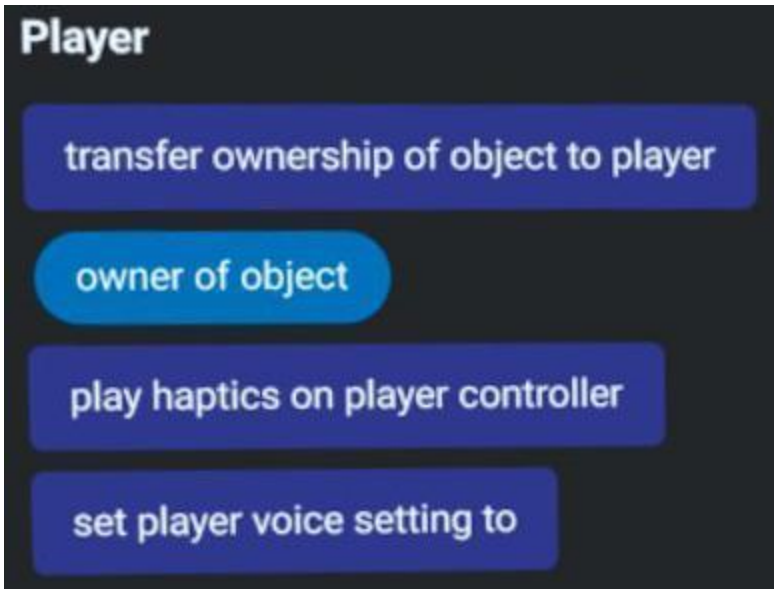
Example 1: Reset world with a trigger



In this example, **when trigger is entered by player** fires the **reset world state** codeblock. Imagine the trigger covering a “restart game” button.

Player

Actions > Player



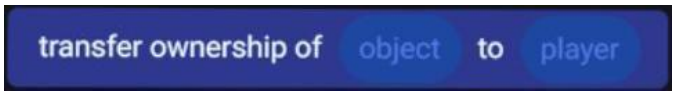
Codeblocks that control ownership, haptics, and player voice settings.

transfer ownership of object to player

Actions > Player > transfer ownership of object to player

Transfers the ownership of an object to a player

Appearance in Composition Pane



Description

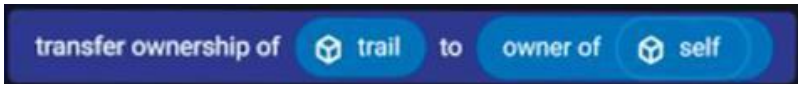
Transfers ownership of an object to a player, this happens automatically when an object is grabbed or attached to a player. But can be done manually with this codeblock for other objects and gizmos. Owning an object means that the player or server (via server player) determines the truth of an object's current position. This is great when paired with local scripting.

When an object's script mode is set to local, an ownership transfer of that object will cause the script to reset, the world start event fires and all variables are set back to their initial values. You may need to wait a fraction of a second before sending events to the newly started local script.

Parameters

object, player

Example 1: Transfer ownership to the owner of self



Imagine you have a baseball with a trail fx, when the player grabs the baseball you could transfer the ownership of the trail to the player who now owns the baseball. This creates a more responsive look and feel.

See Also

- Values > Value Input > server player
- Actions > Player > owner of object

owner of object

Actions > Player > owner of object

The owner of an object.

Appearance in Composition Pane

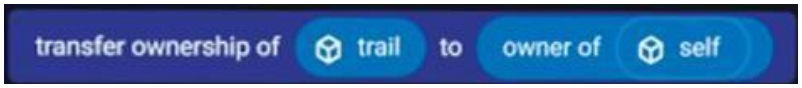
Description

Owner of object returns the player who owns the object.

Parameters

object

Example 1: Transfer ownership to the owner of self



Imagine you have a baseball with a trail fx, when the player grabs the baseball you could transfer the ownership of the trail to that player as they now own the baseball.

See Also

- Values > Value Input > server player
- Actions > Player > transfer ownership of object to player

play haptics on player controller

Actions > Player > play haptics on player controller

Play controller haptics.

Appearance in Composition Pane

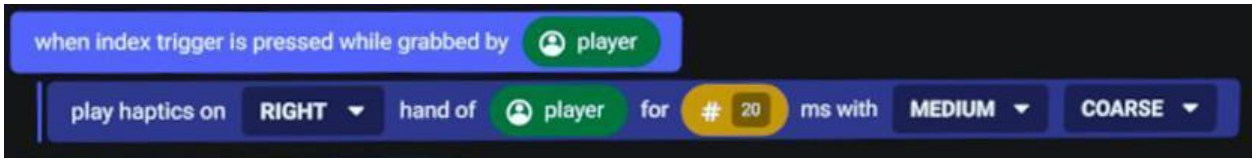
Description

Play haptics on controller will initiate a haptics vibration on a hand controller for a duration in ms. It can either play on the player's right hand or left hand. The intensity can be set to either *Very_light, Light, Medium,* or *Strong.* There are three types of vibrations: *Soft, Course,* and *Sharp*. *Soft* uses a sine wave, *Course* uses a square wave, and *Sharp* uses a saw wave.

Parameters

controller, player, number, intensity, vibration

Example 1: Play haptics on index trigger press



Applying haptics is a great way to give responsive feedback to the player for actions, like in this case, pressing their index trigger.

set player voice setting to

Actions > Player > set player voice setting to

Control how far a player's voice can travel.

Appearance in Composition Pane

Description

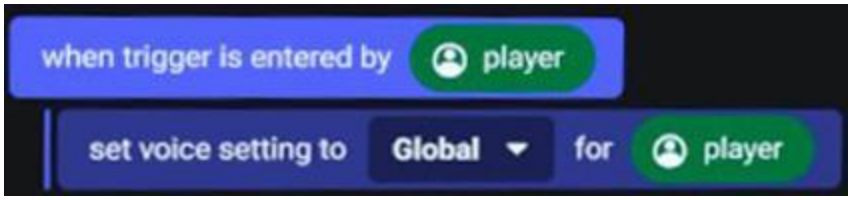
Different worlds will have different needs for voice range, with the default being set in the Environment Gizmo. The setting can be adjusted with this codeblock to either Mute, Whisper, *Environment, Default, Global, Nearby,* or *Extended.*

Environment sets it to the world's default voice setting. Mute, silences a player, Whisper is for people right next to each other. *Nearby* reduces how far people can hear the player. *Default* sets it to a standard distance. *Extended* allows people to hear the player further away than the *Default* setting. *Global* allows everyone in the world to hear the player and has slight direction.

Parameters

range, player

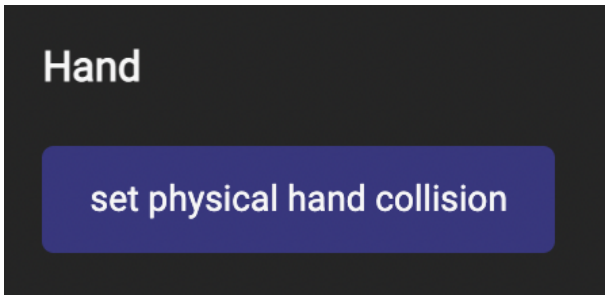
Example 1: Alter a player's voice setting.



Imagine this script attached to a trigger on a theater's stage. Performers would then get Global audio. You can pair this with the trigger exit event to return a player's voice back to the environment setting.

Hand

Actions > Hand



Code blocks for controlling actions related to player hands.

set physical hand collision

Actions > Hand > set physical hand collision

Control how hands collide with objects in the world.

Appearance in Library

Appearance in Composition Pane

Description

All code blocks affecting player physical hand collison must have the “Can hands collide with physical objects” and/or “Can hands collide with static objects” enabled for dynamic or static objects respectively. This setting can be found by navigating to the player settings section under the “World” tab in the build menu.

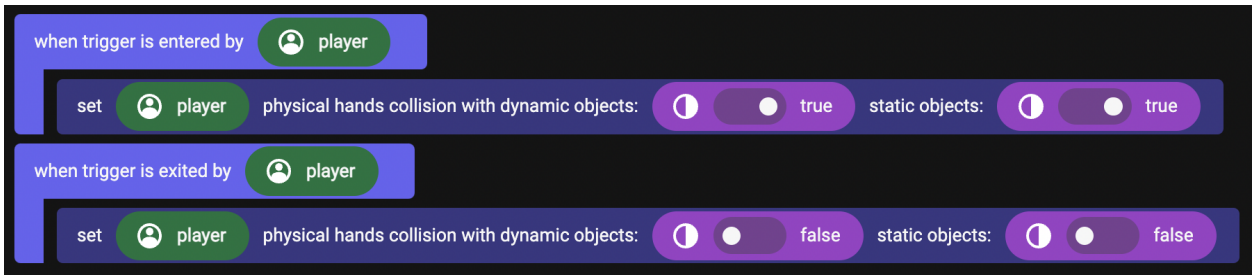
Enabling/Disabling physical hand collisons toggles the physical interaction a player can have with their hands by colliding their hands with static or dynamic objects. Toggling between set player hands collision with dynamic objects will enable/disable players ability to interact with dynamic objects by colliding their hands with the object. Similarly toggling between set player hands collision with static objects will enable/disable players ability to interact with static objects by colliding with their hands with the object.

Note: Players' hands will only interact with objects if they are moving their hands in physical space. If a person is moving forward by holding forward on the thumbstick and bumps into a wall the hands will not collide with the wall, instead if they are standing still and then try to interact with the wall by physically moving their hands in physical space then they will collide with the wall.

Parameters

player - the player to check
boolean dynamic objects - Toggle hand collisions with dynamic objects
boolean static objects - Toggle hand collisions with static objects

Example: trigger enter enable physical hands/trigger exit disable physical hands

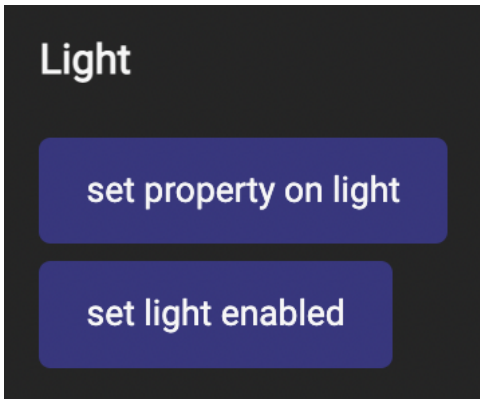


What it does: When a player enters the trigger, their hands can collide with dynamic and static objects and when they exit the trigger their hands will not collide with either dynamic or static objects.

How it works: When a player enters the trigger, the players' hands are set to collide with dynamic and static objects, by physically moving their hands the player is able to interact with objects by colliding their hands with objects. Once they leave the trigger collisions on their hands is diabled and they can no longer interact with objects by physically moving their hands.

Light

Actions > Light



Code blocks for configuring dynamic light gizmos.

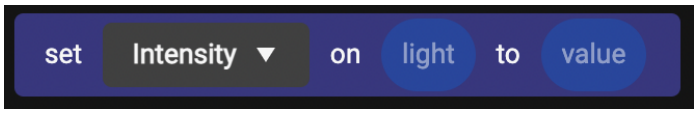
set property on light

Actions> Light> set property on light

Configure properties on dynamic light gizmos.

Appearance in Library

Appearance in Composition Pane



Description

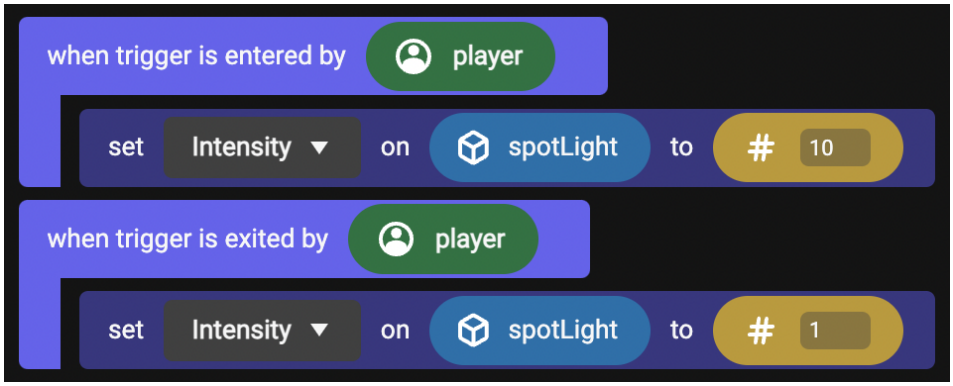
Dynamic light gizmos allow the lighting in a world to be dynamic (instead of static and never changing) by moving lights, turning them on/off, and modifying their properties. This code block allows modifying the following properties:

- **Intensity** - the brightness of a dynamic light on a scale of 0-10.
- **FalloffDistance** - the light intensity will decrease as you get farther away; beyond this distance the light will have no effect.
- **Spread** - the angle (0 to 180 degrees) of the cone projected from the center of a dynamic light with a "spot" light type. This will not affect dynamic lights with a light type of "point".

Parameters

property - one of the properties listed above.
light - the dynamic light gizmo to modify.
value - the value the property should now have.

Example: Under the spotlight



What it does: When a player enters the trigger a spotlight gets brighter, highlighting the player. When the player exits the trigger the light goes back to a low intensity.

How it works: The script should be run on a trigger. When the trigger is entered and exited we modify the light property accordingly. Note that this example doesn't work well with multiple players. You'd likely want to keep track if the trigger is empty or not.

set light enabled

Actions> Light> set light enabled

Turn dynamic light gizmo on and off.

Appearance in Library

Appearance in Composition Pane

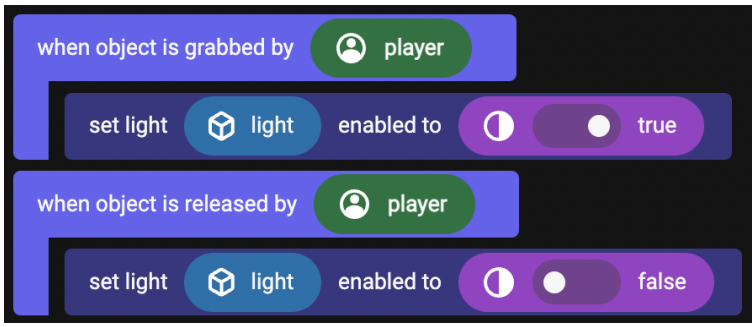
Description

Dynamic light gizmos allow the lighting in a world to be dynamic (instead of static and never changing) by moving lights, turning them on/off, and modifying their properties. This code block allows you to turn dynamic lights on and off.

Parameters

light - the dynamic light gizmo to modify.
enabled - a boolean specifying if the light should now be on or off.

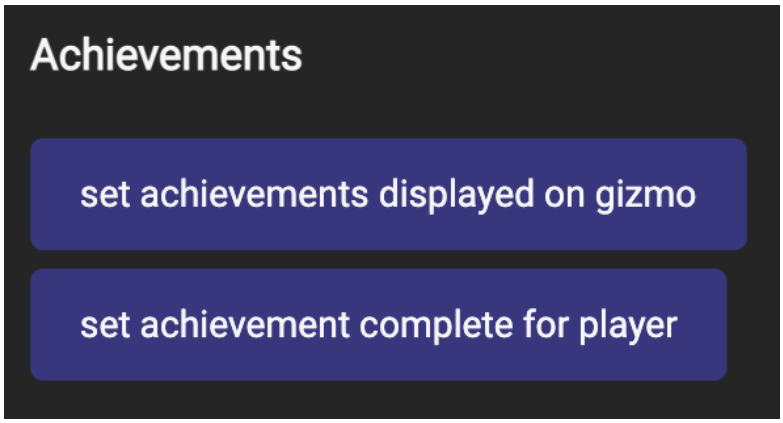
Example: Flashlight



When a player grabs an object the light turns on. When the player releases the object the light turns off. This script should be placed on a grabbable object group that has a dynamic light inside the group.

Achievements

Actions > Achievements



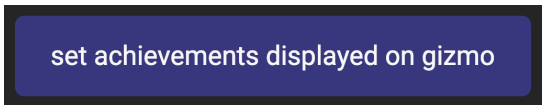
Code blocks for configuring dynamic light gizmos.

set achievements displayed on gizmo

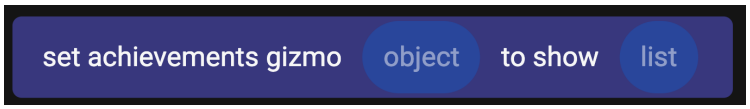
Actions > Achievements > set achievements displayed on gizmo

Configure which achievements are visible on an achievements gizmo.

Appearance in Library



Appearance in Composition Pane



Description

This code block allows you to limit which achievements are visible in a specific Achievements Panel gizmo, as well as the order of the achievements that are displayed. Each gizmo can have a different list of achievements configured for display. This configuration is global to all players, i.e. there is no way to set achievement ordering or visibility on a player by player basis.

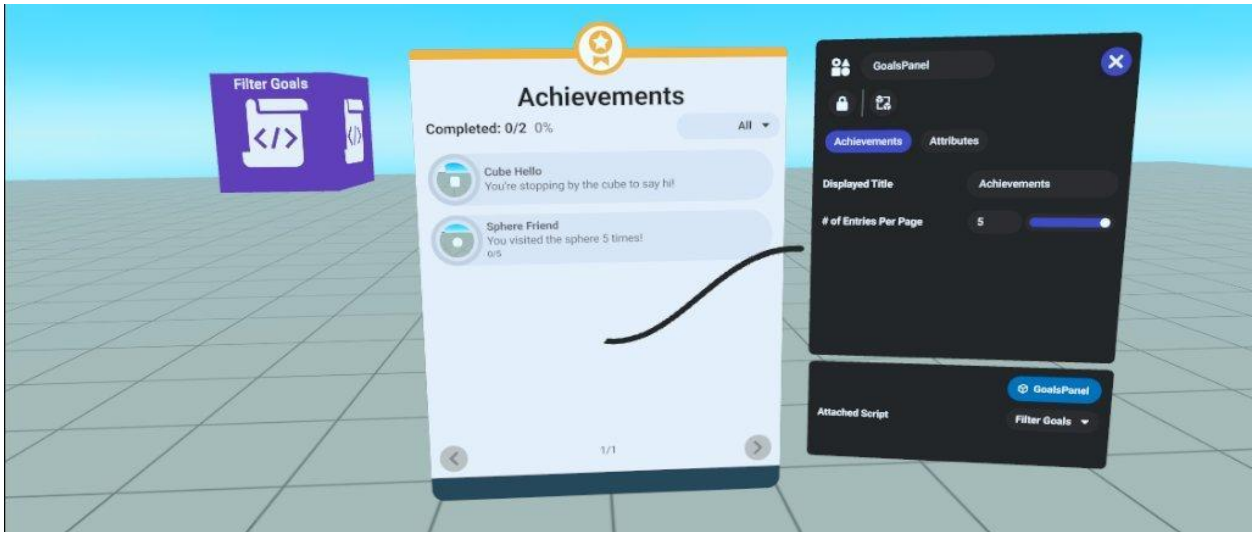
You can use this code block to group achievements into multiple different panels for organization purposes (especially useful if you have a lot of achievements). Also, you can use this feature to hide certain achievements from view so that the player doesn't know they exist until they complete them. Note that completed achievements do not appear on the configured panel if they are not in the provided list. There is no way to make an achievement unlisted before completion, but then listed after completion.

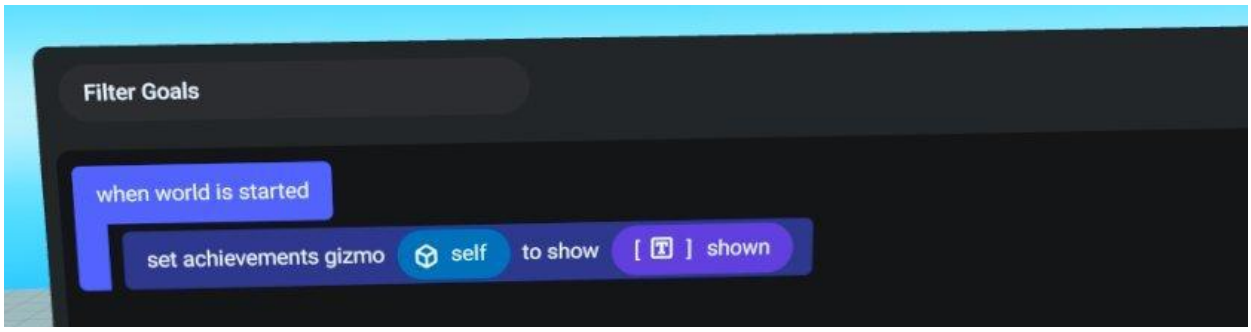
Parameters

object: the achievement panel gizmo on which you want to filter and/or reorder achievements.
list: a list of strings containing the Script IDs of the achievements you want to be displayed in the provided Achievement Panel object reference. The order of the achievement Script IDs in the list defines the order they will be displayed in the panel gizmo. Any achievements with Script IDs *not* provided in the list will *not* be displayed in the panel.

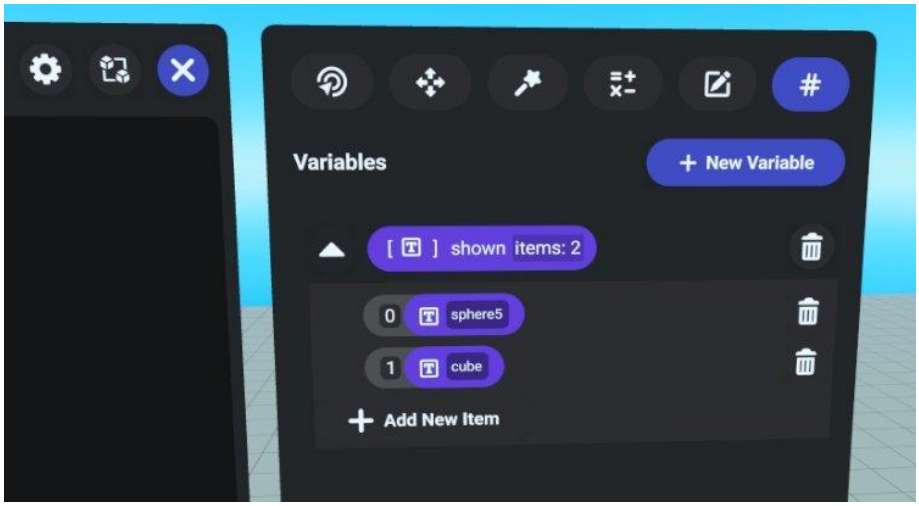
Example: Achievements Filter

The following example shows how you can use a script to filter and reorder the items that appear on an achievements panel gizmo.





You simply need to set a list of Script ID strings for the achievements (not the Names of the achievements) you want displayed, in the order you want them displayed, on the script gizmo.



This is an ideal use case for the Variable panel inline configuration of default value entries for text lists. In this case, the Script IDs of the achievements we want displayed are ‘sphere5’ and ‘cube’.

set achievement complete for player

Actions > Achievements > set achievement complete for player

Mark an achievement as completed for a given player.

Appearance in Library

Appearance in Composition Pane

Description

This code block is used to mark an achievement complete or incomplete, regardless of type. Note that while Tracked Persistent achievements are auto completed by the runtime when they reach their PPV threshold, they can also be marked complete explicitly.

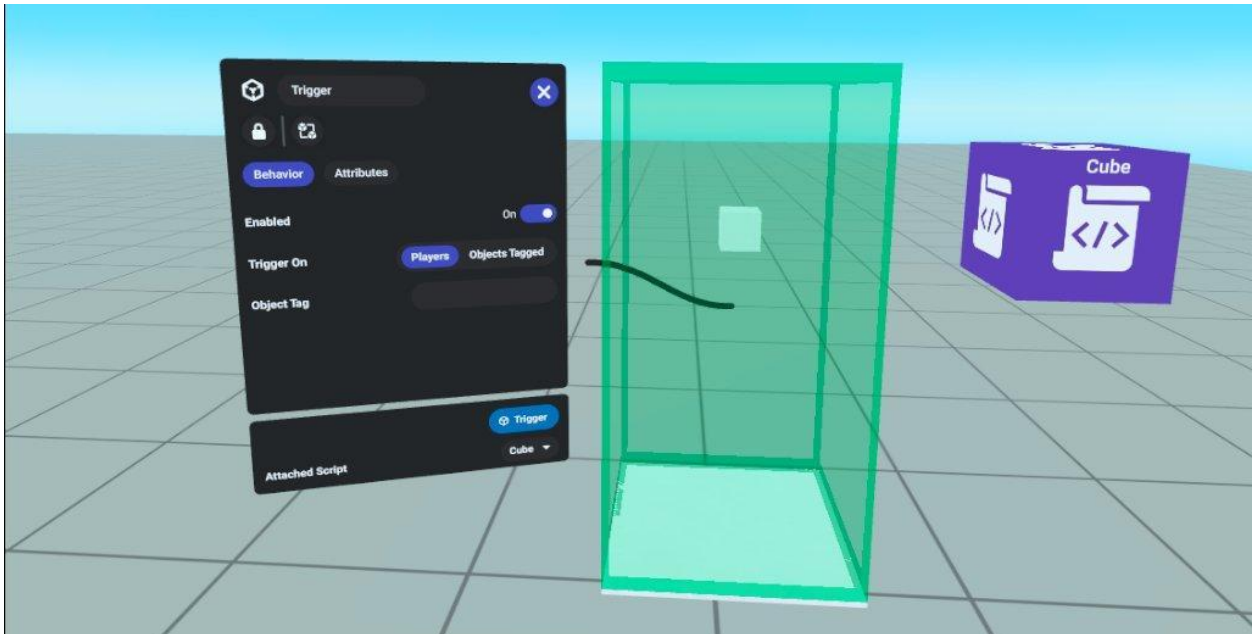
Parameters

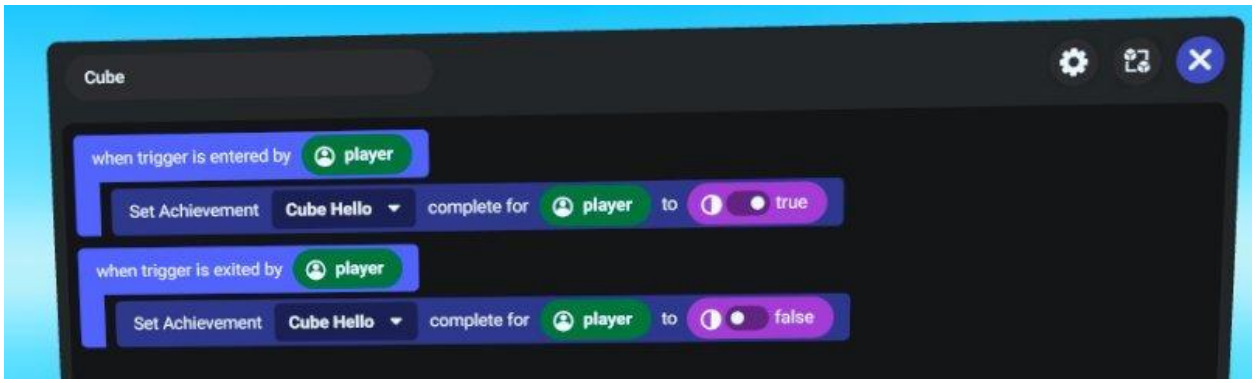
Achievement: the name of a configured achievement. It is *not* the Script ID.
player: the player for which you want to update an achievement value.
value: the new completion value for the achievement. If the achievement already has this value, nothing happens. If the achievement is changing to complete (i.e. true), a popup will appear in front of the player indicating completion. If changing completion state, any Achievement Panel gizmos configured to display the achievement will update its display accordingly.

Note that if this achievement is a Tracked Persistent achievement, and the underlying PPV value still exceeds the configured threshold, marking such an achievement incomplete will have no effect. It will remain completed.

Example: Hello Cube

In the following example, there is a Player Trigger with a script attached that sets the ‘Cube Hello’ achievement complete when the player enters the trigger, and incomplete when the player leaves the trigger.

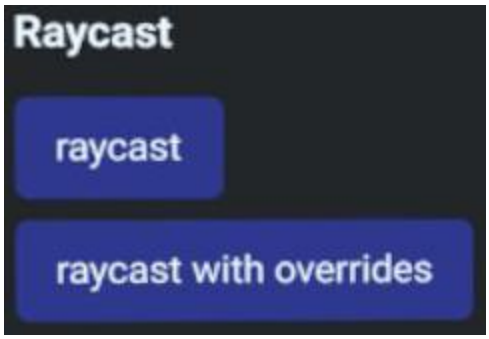




The achievement to mark complete or incomplete is referenced by Name, not ScriptID, via the drop down list.

Raycast

Actions > Raycast



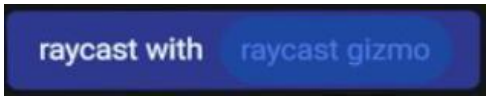
Codeblocks to control a raycast gizmo.

raycast

Actions > Raycast > raycast

Allows a raycast gizmo to get data in the same event as the **raycast** codeblock.

Appearance in Composition Pane



Description

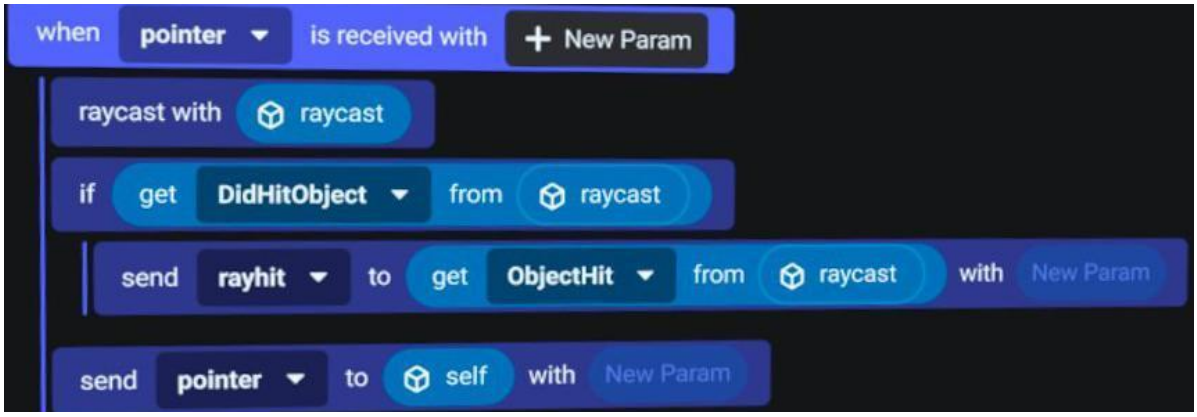
Raycast will cause the referenced raycast gizmo to fire a ray in the direction of the gizmo.

The data gathered by the raycast can then be extracted with **get raycast data**. If raycasting at objects, the raycast gizmo must be configured to register hits on objects with a specific object tag; see the Behavior tab of the raycast gizmo's properties panel and the object's properties panel under Attributes > Tag.

Parameters

object

Example 1: Laser pointer



The *pointer* event continues looping to constantly fire the *raycast*. When *DidHitObject* is true, we send the *rayhit* event to the object that was hit.

See Also

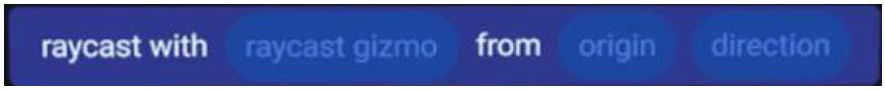
- Actions > Raycast > raycast with overrides
- Operators > Raycast > get raycast data

raycast with overrides

Actions > Raycast > raycast with overrides

Enables a raycast gizmo with overrides for position and direction.

Appearance in Composition Pane



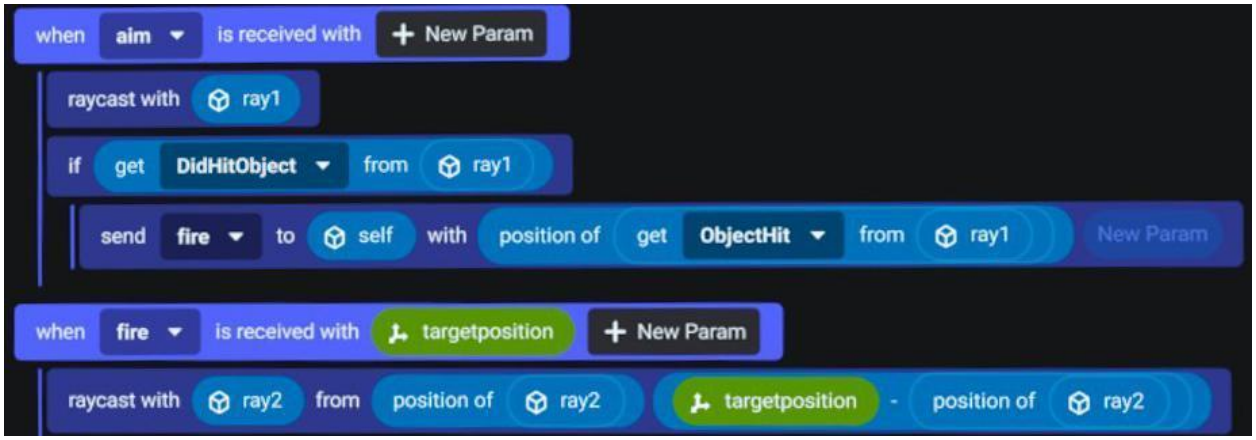
Description

Raycast with overrides will cause the specified raycast gizmo to fire a data-gathering beam from an origin position, with a given direction. A raycast gizmo must be referenced in the first parameter, but will fire from the given position and direction. The data gathered by the raycast can then be extracted with the **get raycast data**.

Parameters

object, vector, vector

Example 1: Aiming scope



Imagine this script attached to an aiming scope, when the event *aim* is received, **raycast** causes *ray1* to fire a beam in its direction. If the **get raycast data** registers a hit, then the event *fire* is sent back to *self*, passing in a parameter containing the hit position. When *fire* is received, a second raycast gizmo, *ray2*, fires from its position directly at the positional coordinates that were passed in.

See Also

- Operators > Raycast > get raycast data
- Operations > Object Transform > position of object

Popup

Actions > Popup

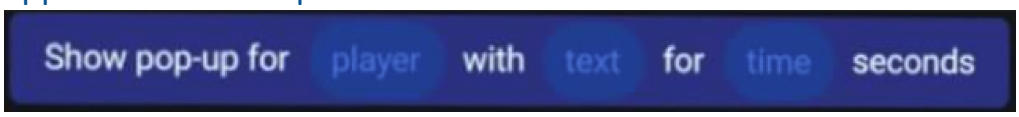
Codeblocks to create popups in front of players.

show simple popup for player

Actions > Popups > show simple popup for player

Shows a text message in a floating panel to just one player.

Appearance in Composition Pane



Description

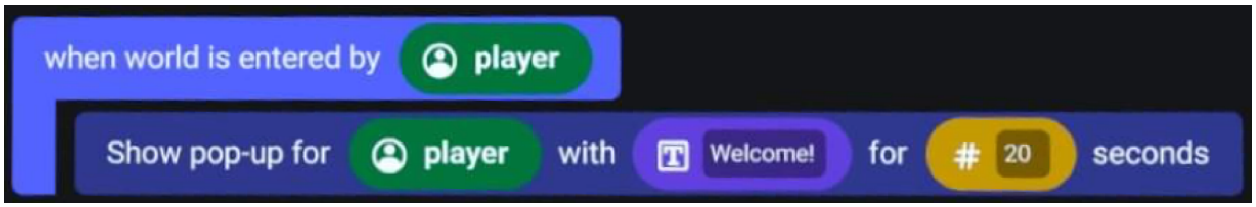
Show a message to a single player that only they can see. You can use this to tell a player that they earned a powerup, that they just unlocked a reward, that they are secretly the villain this round of the game, and so much more. You can configure the text shown and how long it will be shown for.

Parameters

player - the player to show the message to
text - the string that will be shown to the player
time - the number of seconds the popup will be visible for

Example: A welcome message

What it does: When a player enters the world they see a popup that says “Welcome!” It goes away after 20 seconds.



How it works: We use the “when world is entered by player” codeblock to detect that a new player has arrived and then use the “show pop-up” code block to show them the greeting.

See Also

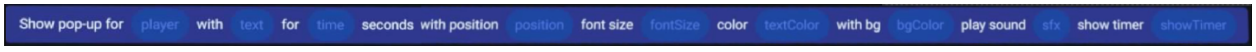
- **Actions > Popups > show popup for player**
- **Actions > Popups > show popup for all players**

show popup for player

Actions > Popups > show popup for player

Shows a text message in a floating panel to just one player. Configure colors, timing, and sound.

Appearance in Composition Pane



Description

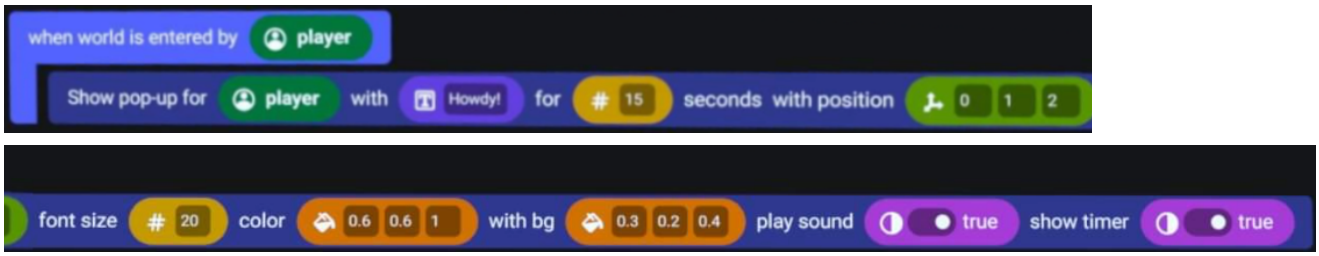
Show a message to a single player that only they can see. You can use this to tell a player that they earned a powerup, that they just unlocked a reward, that they are secretly the villain this round of the game, and so much more. You can configure the text shown, how long it will be shown for, the position of the panel, the font size, the text color, the background color, if a sound is heard when it appears, and whether it shows a countdown timer or not.

Parameters

player - the player to show the message to
text - the string that will be shown to the player
time - the number of seconds the popup will be visible for
position - a vector offset of the panel from its default location; for example (0, 1, 0) will render the panel one meter higher
fontSize - the size the font should be
textColor - the color of the text in the panel
bgColor - the background color of the panel
sfx - a boolean controlling whether a sound is played when the panel appears
showTimer - a boolean controlling if a countdown timer should be shown that depicts how many seconds until the popup disappears.

Example: A welcome message

What it does: When a player enters the world they see a popup that says “Howdy!” It goes away after 15 seconds and shows a countdown for how many seconds until it vanishes.



How it works: We use the “when world is entered by player” codeblock to detect that a new player has arrived and then use the “show pop-up” code block to show them the greeting with custom colors and a countdown timer.

See Also

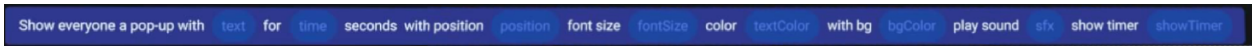
- **Actions > Popups > show simple popup for player**
- **Actions > Popups > show popup for all players**

show popup for all players

Actions > Popups > show popup for all players

Shows a text message in a floating panel to all players. Configure colors, timing, and sound.

Appearance in Composition Pane



Description

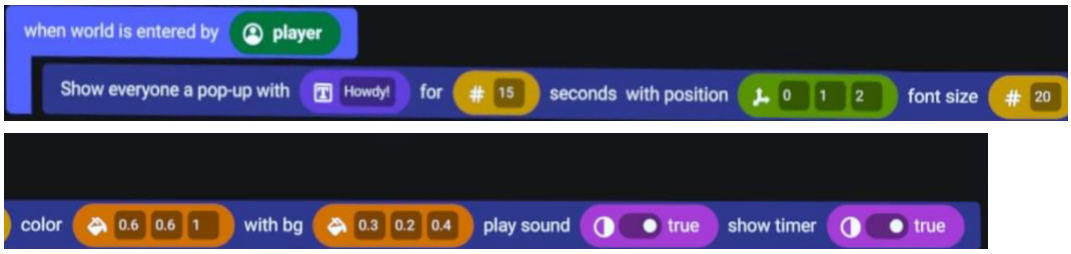
Show a message to all players in the world. Even though they all get the same message, they cannot see one another's message panels. You can use this to tell a player that they earned a powerup, that they just unlocked a reward, that they are secretly the villain this round of the game, and so much more. You can configure the text shown, how long it will be shown for, the position of the panel, the font size, the text color, the background color, if a sound is heard when it appears, and whether it shows a countdown timer or not.

Parameters

text - the string that will be shown to the player
time - the number of seconds the popup will be visible for
position - a vector offset of the panel from its default location; for example (0, 1, 0) will render the panel one meter higher
fontSize - the size the font should be
textColor - the color of the text in the panel
bgColor - the background color of the panel
sfx - a boolean controlling whether a sound is player when the panel appears
showTimer - a boolean controlling if a countdown timer should be shown that depicts how many seconds until the popup disappears.

Example: A welcome message

What it does: When a player enters the world everyone sees a popup with custom colors that says “howdy” so that they know there is a new player somewhere to say hi to. It goes away after 15 seconds and shows a countdown for how many seconds until it vanishes.



How it works: We use the “when world is entered by player” codeblock to detect that a new player has arrived and then use the “show pop-up for all players” codeblock to show everyone a greeting with custom colors and a countdown timer.

See Also

- **Actions > Popups > show simple popup for player**
- **Actions > Popups > showpopup for player**