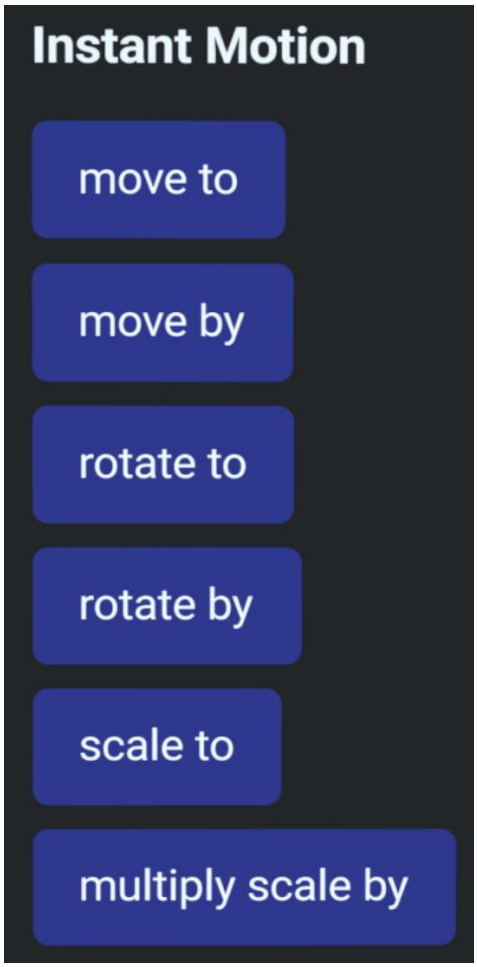
Motion

Instant Motion

Motion > Instant Motion



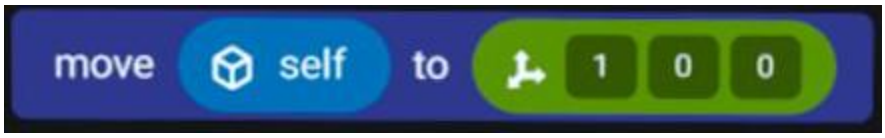
Cause instant changes to an object, that object needs to be set to "Interactive" or "Animated."

move to

Motion > Instant Motion > move to

Instantly moves the object to the coordinates provided.

Appearance in Composition Pane



Description

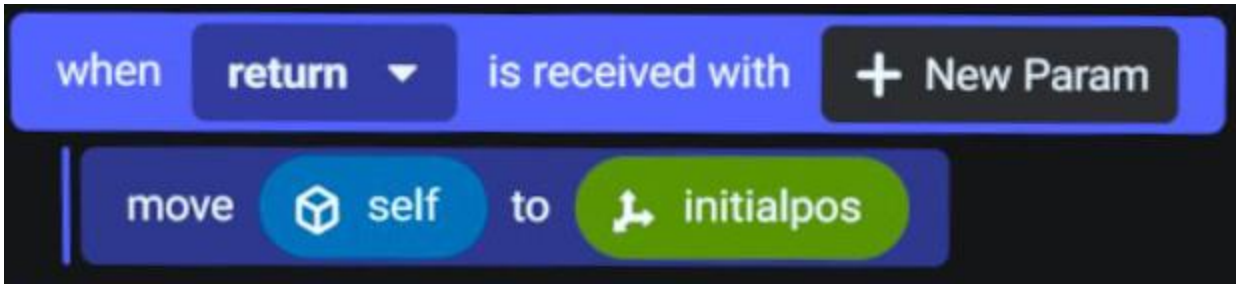
Instantly moves the object to the provided vector coordinates. The coordinates are relative to the world's grid. The movement is similar to teleportation, as it will not interact with any collidable objects between the starting and ending positions.

Parameters

object, vector

The object moves to the vector position.

Example 1: Return to initial position



In this example the object self moves to the *initialpos* vector. This vector can be set in the object's property panel, or alternatively on world start using **set to** and **position of object**.

See Also

- Operations > Object Transform > position of object

move by

Motion > Instant Motion > move by

Instantly moves the object relative to its current position.

Appearance in Composition Pane

Description

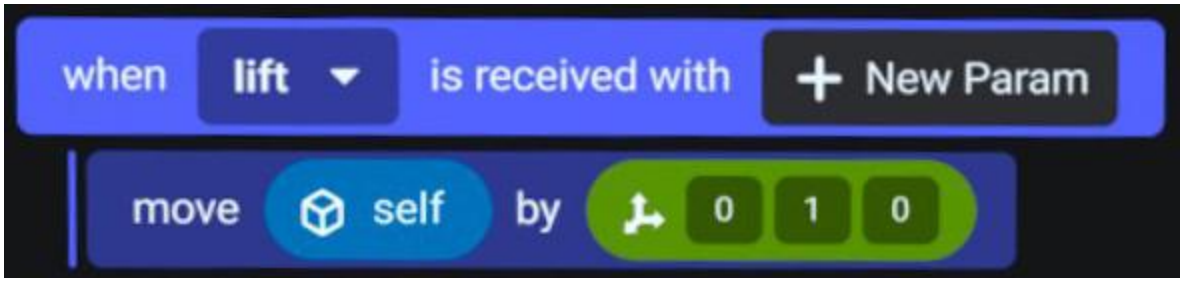
Instantly moves the object by the vector provided. Think of this like adding to the current position. The movement is similar to teleportation, as it will not interact with any collidable objects between the starting and ending positions.

Parameters

object, vector

Moves the object by the vector.

Example 1: Move object upwards



When *lift* is received, self is moved upwards along the y axis by 1 meter.

See Also

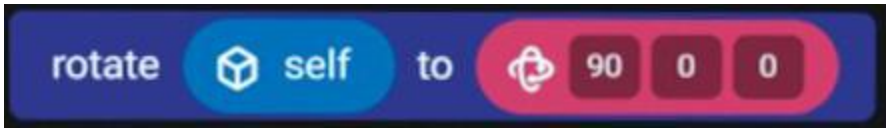
- Values > Value Input > vector input

rotate to

Motion > Instant Motion > rotate to

Instantly rotates the object to the rotation provided.

Appearance in Composition Pane



Description

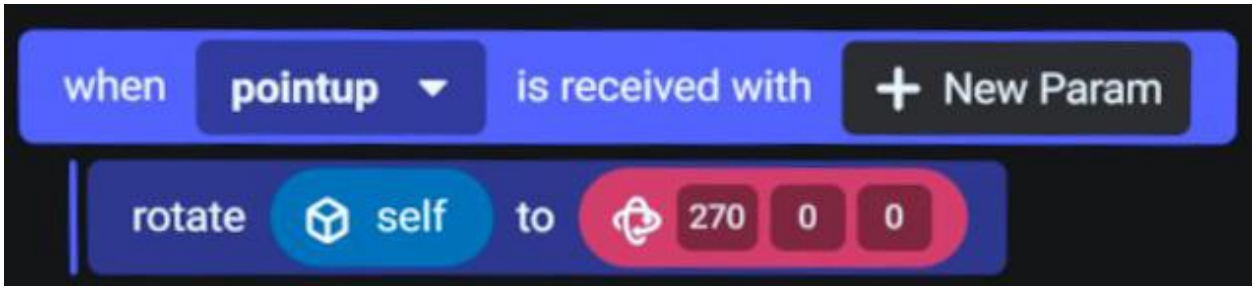
Instantly rotates the object to the rotation. Rotations along one axis can affect the other axis, this makes understanding rotation amounts very difficult. Consider rotating the object into position, and getting the rotation value from the attributes tab of it's property panel.

Parameters

object, rotation

Rotates the object to the x,y,z rotation.

Example 1: Rotate object to an absolute angle



The *pointup* event uses **rotate to** in order to rotate *self* to the provided rotation.

See Also

- Values > Value Input > rotation input
- Motion > Instant Motion > rotate by

rotate by

Motion > Instant Motion > rotate by

Instantly rotates the object by adding the rotation to the current rotation.

Appearance in Composition Pane

Description

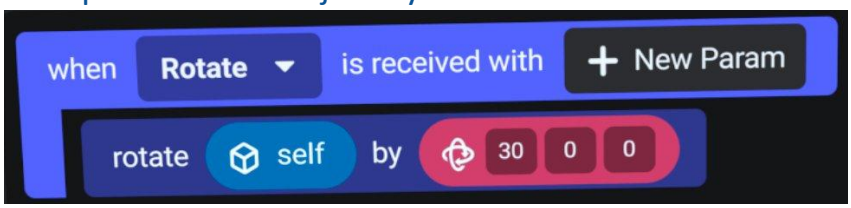
Instantly rotates the object by the amount provided along the object's local axis.

Parameters

object, rotation

Rotates the object by the rotation.

Example 1: Rotate object by



Rotates self by 30 degrees along the object's x axis.

See Also

- Values > Value Input > rotation input
- Motion > Instant Motion > rotate to

scale to

Motion > Instant Motion > scale to

Instantly scales the object to the value provided.

Appearance in Composition Pane

Description

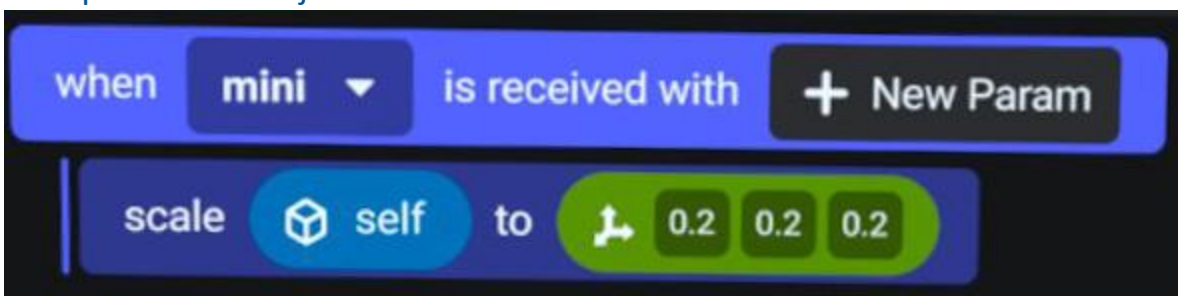
Instantly scales the object to the provided scale, expressed as a vector. If the object is a group, the scale is relative to its scale when the object was grouped. If the object is not a group, the scale is the size in meters of the object. Scaling may cause physics interactions with surrounding objects.

Parameters

object, vector

Scales the object to the vector.

Example 1: Scale object to



The *mini* event scales *self* to *0.2,0.2,0.2*.

See Also

- Values > Value Input > vector input
- Motion > Instant Motion > scale by

multiply scale by

Motion > Instant Motion > multiply scale by

Instantly scales the object by multiplying the supplied scale by its current size.

Appearance in Composition Pane

Description

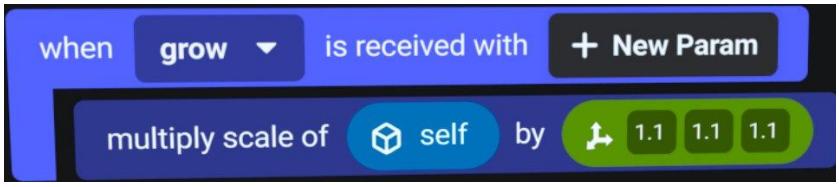
Instantly scales the object by the provided scale vector, growing or even shrinking. Scaling may cause physics interactions with surrounding objects.

Parameters

object, vector

Scales the object by the vector.

Example 1: Scale object by



The *grow* event increases *self*'s scale by *1.1,1.1,1.1* or 110% in all directions. Repeated calls to *grow* will scale *self* by an additional 110% of its new size.

See Also

- Values > Value Input > vector input
- Motion > Instant Motion > scale to

Motion Over Time

Motion > Motion Over Time

Cause changes to an object over time, the object needs to be set to "Interactive" or "Animated."

move to over time

Motion > Motion Over Time > move to over time

Moves the object to the coordinates provided over time.

Appearance in Composition Pane

Description

Moves the object to the vector coordinates over the period of time provided. The coordinates are relative to the world's grid. The movement can interact with collidable objects between the starting and ending positions.

Parameters

object, vector, number

Moves the object to a vector position over a number of seconds.

Example 1: Move object to an absolute position over time.

Moves self to the global position 1, 0, 0 over one second.

See Also

- Operations > Object Transform > position of object
- Motion > Instant Motion > move to

move by over time

Motion > Motion Over Time > move by over time

Moves the object relative to its current position over time.

Appearance in Composition Pane

Description

Moves the object by the vector over the time provided. Think of this like adding to the current position. The movement can interact with collidable objects between the starting and ending positions.

Parameters

object, vector, number

Moves the object to the vector position over a number of seconds

Example 1: Move object by over time

Move by over time moves *self* over a period of 3 seconds by the vector variable moveby. Creating a variable means we can run this on separate objects with different amounts for each.

See Also

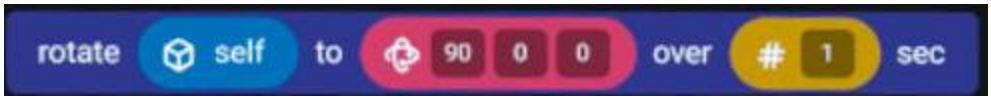
- Motion > Instant Motion > move by

rotate to over time

Motion > Motion Over Time > rotate to over time

Rotates the object from its current rotation to the provided rotation over time.

Appearance in Composition Pane



Description

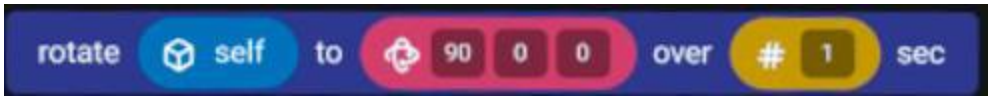
Rotates the object to the rotation over time. Rotations along one axis can affect the other axis, this makes understanding rotation amounts very difficult. Consider rotating the object into position, and getting the rotation value from the attributes tab of it's property panel. Rotating may cause physics interactions with surrounding objects.

Parameters

object, rotation, number

Rotates the object to a rotation over a number of seconds.

Example 1: Rotate to over time



Rotates self to 90,0,0 over one second.

See Also

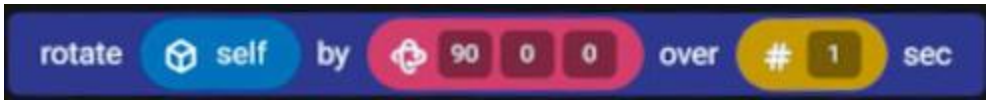
- Motion > Instant Motion > rotate by

rotate by over time

Motion > Motion Over Time > rotate by over time

Rotates the object from its current rotation by the given rotation over time.

Appearance in Composition Pane



Description

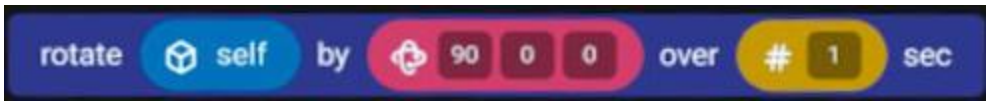
Rotates the object by the amount provided along the object's local axis over time. Rotating may cause physics interactions with surrounding objects.

Parameters

object, rotation, number

Rotates the object by a rotation over a number of seconds.

Example 1: Rotate by over time



Rotates self by 90,0,0 over one second.

See Also

- Motion > Instant Motion > rotate by

scale to over time

Motion > Motion Over Time > scale to over time

Scales the object from its current scale to the scale provided over time.

Appearance in Composition Pane

Description

Scales the object to the provided scale vector over time. If the object is a group, the scale is relative to its scale when the object was grouped. If the object is not a group, the scale is the size in meters of the object. Scaling may cause physics interactions with surrounding objects.

Parameters

object, vector, number

Scales the object to a scale vector of a number of seconds.

Example 1: Scale to over time

Scale to over time scales *self* to *1,1,1* over a period of one second.

See Also

- Values > Value Input > vector input
- Motion > Motion Over Time > scale by over time

scale by over time

Motion > Motion Over Time > scale by over time

Scales the object over time by adding the supplied scale to its current scale.

Appearance in Composition Pane

Description

Scales the object by the provided scale vector over time, adding or even subtracting. Scaling may cause physics interactions with surrounding objects.

Parameters

object, vector, number

Scales the object by the scale vector over a number of seconds.

Example 1: Scale an object by over time

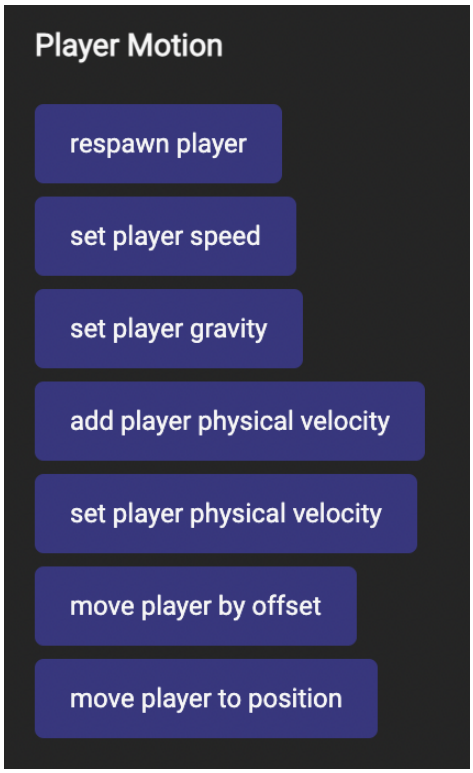
Here we increase *self*'s scale by *2.1* in all directions over two seconds.

See Also

- Motion > Motion Over Time > scale to over time

Player Motion

Motion > Player Motion



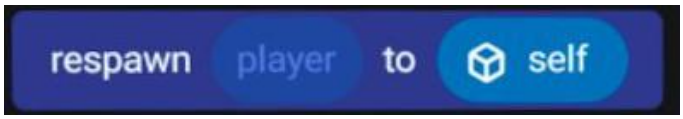
Codeblocks that move the player or change their speed and gravity.

respawn player

Motion > Player Motion > respawn player

Moves and rotates a player to a spawn gizmo.

Appearance in Composition Pane



Description

Respawns a player to a spawn point gizmo. If the spawn point is set to “set position only” it will not rotate the player. The spawn point can also change the player's speed and gravity.

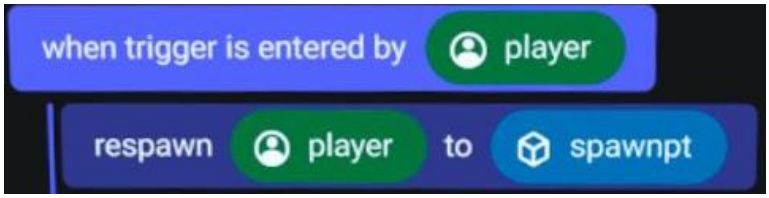
Parameters

player, object

Respawns the player to the spawn gizmo object.

Example 1: Water balloon that respawns a player when hit

Example 2: Respawn/teleport trigger



Placing a trigger under your world with this script allows you to catch any fallen players, or those who experience severe lag. You can also use this script to teleport players.

See Also

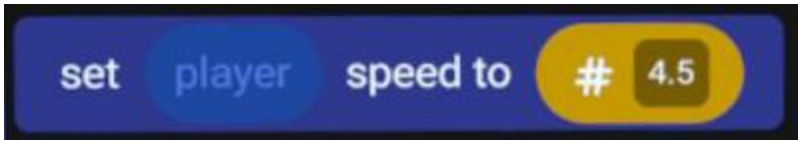
- Events > Collision Events > when colliding with player
- Events > Player Events > when trigger is entered by player

set player speed

Motion > Player Motion > set player speed

Change the speed at which the player moves in the world.

Appearance in Composition Pane



Description

Set player speed changes their movement speed to the number provided in meters per second.

Parameters

player, number

Set the player's speed to the number in meters per second.

Example 1: Change player speed when object is grabbed

Imagine slowing a player's speed when they grab a heavy object.

See Also

- Events > Grab Events > when object is grabbed by player

set player gravity

Motion > Player Motion > set player gravity

Change the gravity that is applied to a player.

Appearance in Composition Pane

Description

Set player gravity changes the downward acceleration of the player as they fall. The number unit is in meters per second squared. Smaller numbers allow higher jumping heights.

Parameters

player, number

Set the player's gravity to the number in meters per second squared.

Example 1: Decrease a player's gravity

Imagine this is applied to an astronaut entering a new planet. Or it could be in a bounce house!

See Also

- Events > Player Events > when trigger is entered by player

add player physical velocity

Operators > Player > add player physical velocity

Modifies the current physical velocity of the player. Physical velocity is velocity only caused by physics such as colliding, falling, applied forces, etc. It does not include locomotion, teleporting or sliding, and also does not include any velocity caused by a person actually moving (e.g. walking around in their guardian space).

Appearance in Library

Appearance in Composition Pane

Description

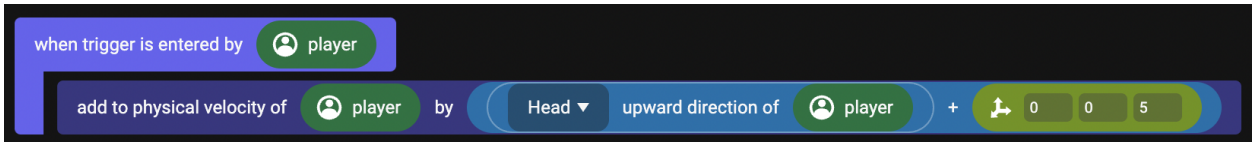
This code block instantly adds to the current physical velocity for the player coming only from physics. Any movement the person does in reality or avatar locomotion are not included. The player's previous velocity is overwritten by the new velocity. This code block will not take effect if the player does not have a physical velocity as defined above.

All code blocks affecting player velocity or instant movement must have the "Custom Player Manipulation" enabled. This setting can be found by navigating to the player settings section under the "World" tab in the build menu.

Parameters

player - the player whose velocity is being increased
vector - the amount a player's vector velocity is being increased by

Example: give a player a brief forward speed boost



What it does: When a player enters the trigger, their rate of forward motion is increased by 5m/s.

How it works: When a player enters the trigger, the player's vector velocity is increased by adding 5 to the z value of the vector. This creates a "speed boost" effect, increasing a player's forward motion by 5m/s.

See Also

- Operators > Player > physical velocity of player
- Operators > Player > set physical velocity of player

set player physical velocity

Operators > Player > set player physical velocity

Sets the physical velocity of the player. Physical velocity is velocity only caused by physics such as colliding, falling, applied forces, etc. It does not include locomotion, teleporting or sliding, and also does not include any velocity caused by a person actually moving (e.g. walking around in their guardian space).

Appearance in Library

Appearance in Composition Pane

Description

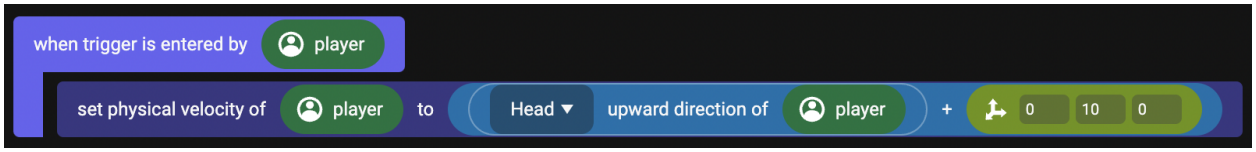
This code block instantly sets a new velocity for the player coming only from physics. Any movement the person does in reality or avatar locomotion are not included. The player's previous velocity is overwritten by the new velocity.

All code blocks affecting player velocity or instant movement must have the "Custom Player Manipulation" enabled. This setting can be found by navigating to the player settings section under the "World" tab in the build menu.

Parameters

player - the player whose velocity is being set
vector - the vector velocity being set

Example: boost a player into the air



What it does: When a player enters the trigger, they are ejected upwards at 10m/s into the air.

How it works: When a player enters the trigger, the desired vector velocity is calculated by finding the position of their head and adding 10 to the y value of the vector. This creates a "super-jump" effect, boosting a player straight up at a rate of at 10m/s.

See Also

- Operators > Player > physical velocity of player
- Operators > Player > add physical velocity of player

move player by offset

Operators > Player > move player by offset

Instantly moves a player to a vector offset from their current position. The player maintains their current physical velocity. Physical velocity is velocity only caused by physics such as colliding, falling, applied forces, etc. It does not include locomotion, teleporting or sliding, and also does not include any velocity caused by a person actually moving (e.g. walking around in their guardian space).

Appearance in Library

Appearance in Composition Pane

Description

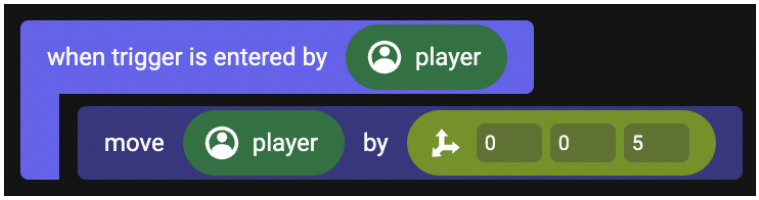
This code block instantly moves a player to a new position offset from their current position by a vector. The player's current vector velocity is unchanged by this movement (for example, if a player has an upward velocity of 10m/s, they will continue to have an upward velocity of 10m/s after their position is offset).

All code blocks affecting player velocity or instant movement must have the "Custom Player Manipulation" enabled. This setting can be found by navigating to the player settings section under the "World" tab in the build menu.

Parameters

player - the player whose position is being offset
vector - the vector added to a player's current position

Example: allow a player to teleport over a gap



What it does: When a player enters the trigger, they are instantly teleported to a vector 5m from their current position on the world's z axis.

How it works: When a player enters the trigger, the player's new vector position is calculated by adding their current vector and the vector 0,0,5. This instantly moves the player 5m along the world's z axis, circumventing any pitfall or obstacle in their path.

See Also

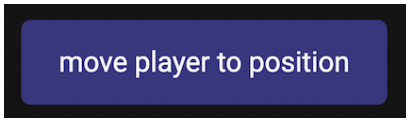
- Operators > Player > move player to position
-

move player to position

Operators > Player > move player to position

Instantly moves a player to a new position. The player maintains their current physical velocity. Physical velocity is velocity only caused by physics such as colliding, falling, applied forces, etc. It does not include locomotion, teleporting or sliding, and also does not include any velocity caused by a person actually moving (e.g. walking around in their guardian space).

Appearance in Library



Appearance in Composition Pane

Description

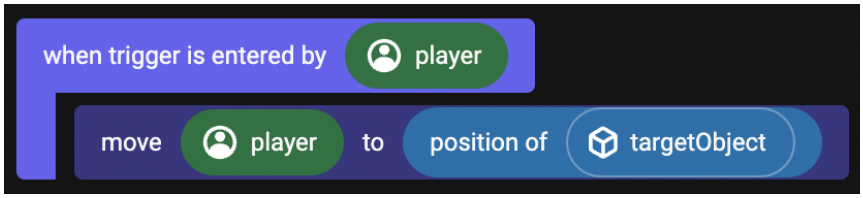
This code block instantly moves a player to a new vector. The player's avatar will be centered on the new position. Keep in mind that a player is approximately 2.5m in height. Configuring a target vector at the expected position of the player's feet may cause them to clip through the ground of your world. The player's current vector velocity is unchanged by this movement (For example, if a player has an upward velocity of 10m/s, they will continue to have an upward velocity of 10m/s after their position is offset).

All code blocks affecting player velocity or instant movement must have the "Custom Player Manipulation" enabled. This setting can be found by navigating to the player settings section under the "World" tab in the build menu.

Parameters

player - the player who is being moved to the new position
vector - the target vector a player will be moved to

Example: instantly teleport a player to a thrown object



What it does: When a player enters the trigger, they are instantly teleported to the vector of an object.

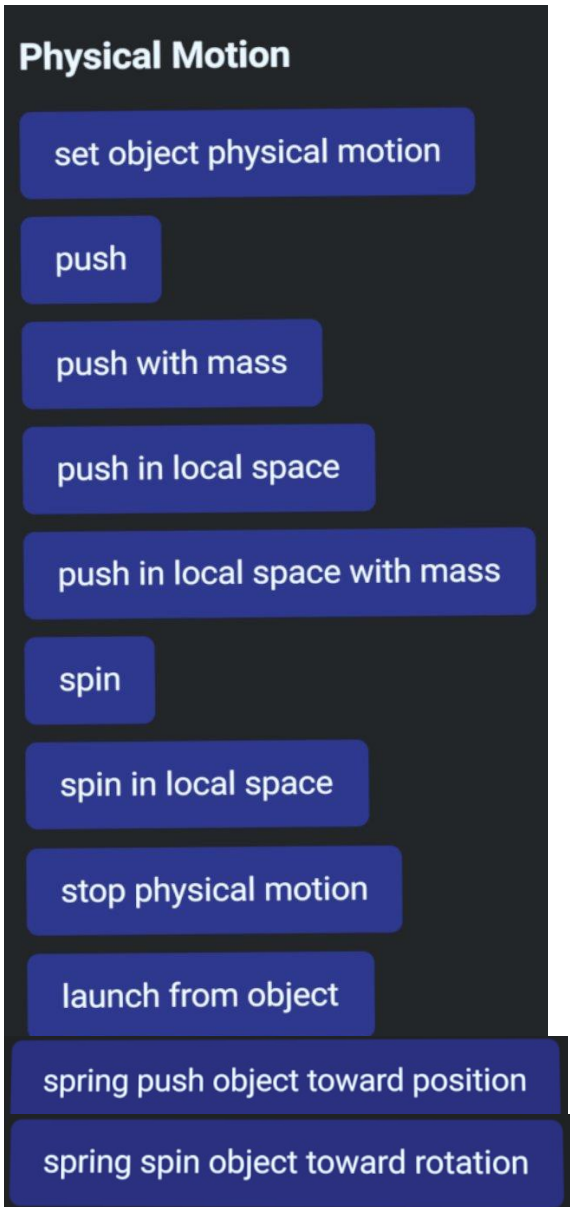
How it works: Make sure the target object's interaction is set to "both" (grabbable and physics enabled) in the target object's property panel. A player can throw the object and then enter the trigger to instantly teleport to the target object's vector. The player's physical velocity will not change, even if the object is still in motion.

See Also

- Operators > Player > move player by offset

Physical Motion

Motion > Physical Motion



Codeblocks that manipulate and affect an object's physics motion. For these codeblocks to work, an object must have interactive physics turned on in its properties panel.

set object physical motion

Motion > Physical Motion > set object physical motion

Locks and unlocks a physics object in place.

Appearance in Composition Pane

Description

When set to true, **set object physical motion** immediately enables an object's physics. When false, the object immediately halts any physics affecting the object and prevents additional physics from affecting the object. **Move to over time** and recorded animations can still be run.

Parameters

object, boolean

Set the object's physical motion true or false.

Example 1: freeze and unfreeze an object with a button press

Imagine a magic wand that can lock *ObjectVar* in place when *player* presses button 1. Then the *player* can unfreeze the *ObjectVar* by pressing button 2**.**

See Also

- Events > Controller Events > when button1 is pressed

push

Motion > Physical Motion > push

The object's velocity becomes equal to its current velocity plus the given velocity.

Appearance in Composition Pane

Description

Push adjusts an object's velocity by the vector in meters per second. If **push** is used multiple times in a single event, the object will be pushed by the sum of the pushes.

Parameters

object, vector

Pushes the object by the velocity vector in meters per second.

Example 1: Slow down over time

The event *airresistance* loops to itself every 0.1 seconds, each time slowing *self*'s velocity. Note the calculation for **push** using **velocity of object** *self*.

See Also

- Operators > Object Transform > velocity of object
- Operations > Basic Operations

push with mass

Motion > Physical Motion > push with mass

The object's velocity is adjusted by the given velocity divided by the object's mass.

Appearance in Composition Pane

Description

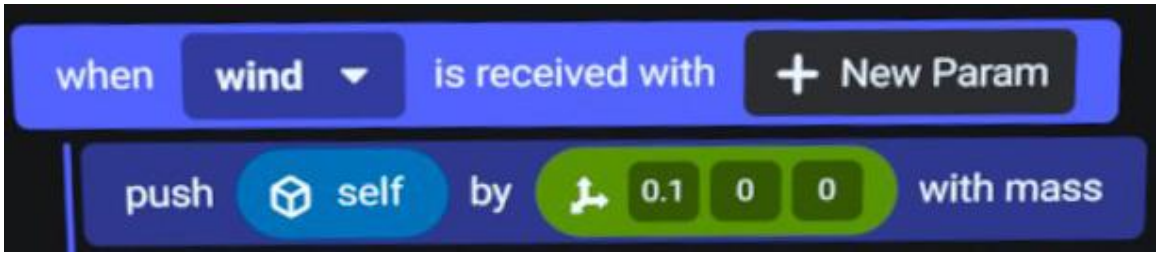
Push with mass increases an object's velocity by the vector based on the size and material of the object. The larger the object, the less it's velocity is affected. If **push** is used multiple times in a single event, the object will be pushed by the sum of the pushes.

Parameters

object, vector

Pushes the object by a velocity vector.

Example 1: Gently push an object



In this example wind slightly pushes the object *self* by *0.1,0,0*. If your object does not move, try using larger numbers.

See Also

- Motion > Physical Motion > push
- Motion > Physical Motion > push in local space with mass
- Motion > Physical Motion > push at position with mass

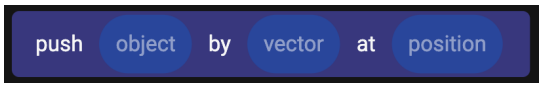
push at position with mass

Motion > Physical Motion > push at position with mass

Add to an object's velocity, applying the "force" at a specific location, meaning that the object will likely both *move* and *spin*.

Appearance in Library

Appearance in Composition Pane



Description

Push at position with mass modifies an object's linear velocity, and likely angular velocity too. This block applies a momentary force (an impulse) to an object. The other "push" code blocks apply the impulse to an object's center point. This block allows you to specify where to apply the impulse. For instance applying the force on the edge of an object will generate a momentary torque (an angular impulse) which will also modify the spin of the object.

Parameters

object - the object to push.
vector - the amount, and direction, to push the object.
position - a position, in world coordinate, to apply the push at.

Example: Push an object above its center

In this example we have a 2-meter tall block at the origin of the world. When the event "earthquake" is received we apply a force 1m below its center (which is its base point). This simulates the ground dragging the block from the bottom, which may be enough to topple it, depending on the physics material.

See Also

- Motion > Physical Motion > push

- Motion > Physical Motion > push in local space with mass

push in local space

Motion > Physical Motion > push in local space

Push in local space orients your velocity to the local object's axis rather than the world's axis.

Appearance in Composition Pane

Description

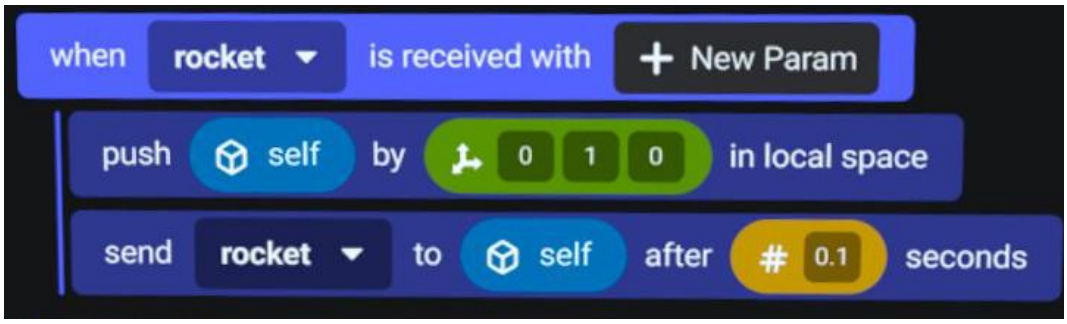
Push in local space adjusts the velocity of an object by a vector in meters per second. The direction of the push is based on the rotation of the object. Each coordinate of the push vector applies to the respective axis of the object. For instance, the X of the vector applies a push in the X direction of the object.

Parameters

object, vector

Pushes the object by a vector in meters per second, based on the object's rotation.

Example 1: Propulsive motion



Every 0.1 seconds, the rocket's velocity increases 1 meter per second in the direction of the rocket's y-axis. The y-axis of the rocket needs to point up for the rocket to accelerate upwards.

See Also

- Motion > Physical Motion > push
- Motion > Physical Motion > push at position with mass

push in local space with mass

Motion > Physical Motion > push in local space with mass

Push in local space orients your velocity to the local object's axis rather than the world's axis. The object is then pushed with mass, meaning smaller objects move further.

Appearance in Composition Pane

Description

Push in local space with mass adjusts the velocity of an object by a vector based on the size and material of the object. The larger the object, the less it's velocity is affected. The direction of the push is based on the rotation of the object. Each coordinate of the push vector applies to the respective axis of the object. For instance, the X of the vector applies a push in the X direction of the object.

Parameters

object, vector

Pushes the object by a vector that is based on the object's rotation, size, and material.

Example 1: Propulsive motion with mass

Every 0.1 seconds, the rocket's velocity increases in the direction of the rocket's y-axis. The y-axis of the rocket needs to point up for the rocket to accelerate upwards. Greater vectors may be necessary to overcome the object's mass.

See Also

- Motion > Physical Motion > push in local space
- Motion > Physical Motion > push

spin

Motion > Physical Motion > spin

The object's angular velocity becomes equal to its current angular velocity, plus the given angular velocity relative to its rotation in the world.

Appearance in Composition Pane

Description

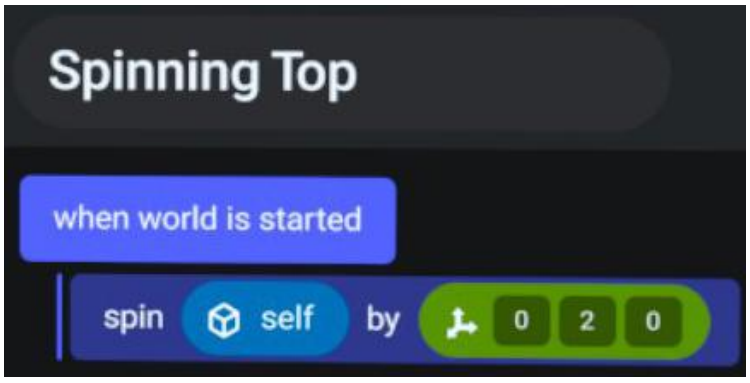
Spin increases an object's angular velocity by the vector in rotations per second relative to it's rotation in the world.

Parameters

object, vector

Spins an object by the angular velocity vector in rotations per second.

Example 1: Spinning top



When world is started a **spin** vector of *0,2,0* spins the object along the world's *y* axis. A greater *y* number will spin the object faster.

See Also

- Motion > Physical Motion > push

spin in local space

Motion > Physical Motion > spin in local space

The given angular velocity is added to the object's current rotation along its axis.

Appearance in Composition Pane

Description

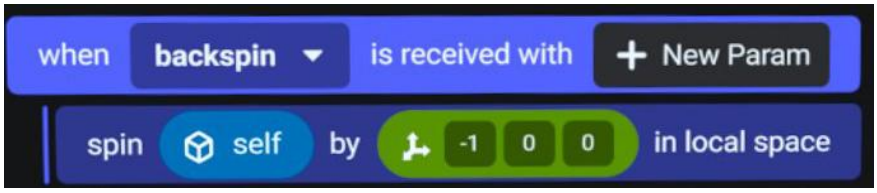
Spin in local space changes the angular velocity of an object by a vector in rotations per second. The axis of spin is based on the object's rotation. Each vector coordinate is applied to the object's respective axis. For instance, the x coordinate is applied to the x axis of the object.

Parameters

object, vector

Spins an object by the vector along the object's axis in rotations per second.

Example 1: Backspin



The *backspin* event here applies a **spin** vector to rotate the object backwards on its x axis.

See Also

- Motion > Physical Motion > spin

stop physical motion

Motion > Physical Motion > stop physical motion

The object's velocity and angular velocity are negated and become zero.

Appearance in Composition Pane

Description

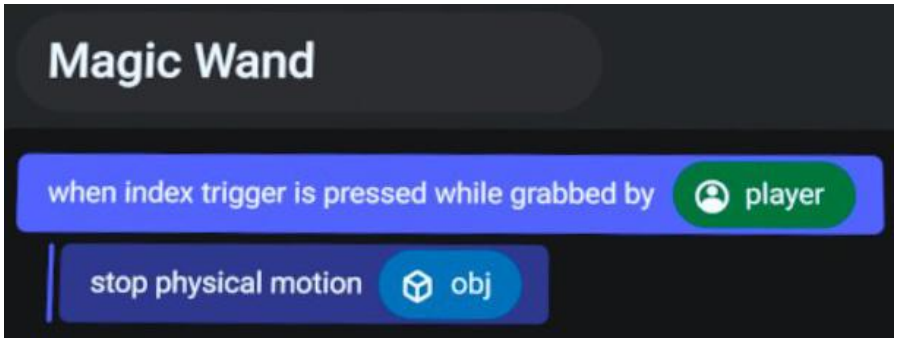
Stop physical motion sets the velocity and spin of the object to zero. This works as long as it is not grabbed by a player. It has no effect on Motion Over Time codeblocks**.**

Parameters

object

Stops the physical motion of the object.

Example 1: Stop an object in motion



If this script is attached to a wand, **when index trigger is pressed** the *obj*'s trajectory is halted.

See Also

- Motion > Physical Motion > spin

launch from object

Motion > Physical Motion > launch from object

Allows you to launch an object from a position with speed.

Appearance in Composition Pane

Description

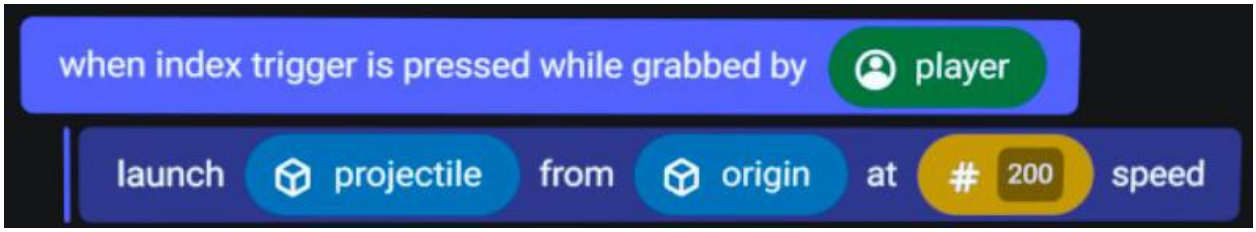
Launch from object sets *projectile's* position and rotation to match *origin's* position and rotation. Then, its speed is set to the number parameter, in meters per second. This can be set to negative to fire in the opposite direction. Make sure your origin object is non-collidable, you may also consider using a trigger gizmo for this. If the object is collidable your projectile may collide and not launch correctly.

Parameters

object, object, number

Launch a physics object from the origin at a speed.

Example 1: Fire a projectile



When a player is holding the object the script is attached to and presses their index trigger. The projectile will orient to the origin object's position and rotation, and then launch from there along the origin's z-axis at the speed in meters per second.

spring push object toward position

Motion > Physical Motion > spring push object toward position

Simulates *one frame* of an object being pulled towards a position by a spring.

Appearance in Composition Pane

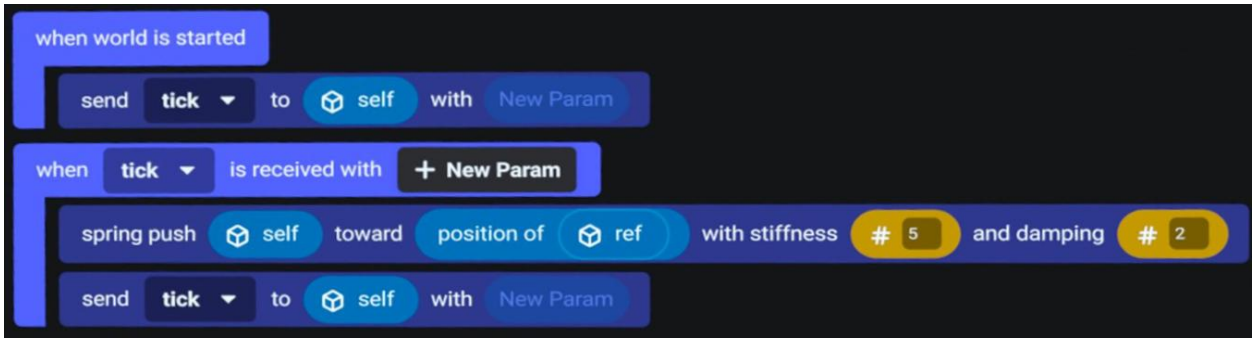
Description

Allows you to create a spring-like motion towards a position. Spring push applies a force that pushes the object towards “position”. It acts as if there were a spring between the object and “position” and calculates the spring force. Since this only applies the force for *one frame* you likely want to run it every frame to get a smooth spring-like bounce / oscillation. The **stiffness** parameter controls how strong the spring is; the bigger that value the harder and faster the object will move. You’ll get unintuitive behavior if you pass a negative stiffness value. The **damping** parameter controls how much “friction” there is. A low damping value will lead to a lot of bouncing back and forth whereas a high damping value will cause the object to slow down and settle more quickly (as if there was air resistance, friction, or other forces preventing the spring from oscillating forever). The damping value should also be non-negative for realistic behavior.

Parameters

object - the object to move
position - the vector position to push the object toward
stiffness - a number for how forceful the spring is
damping - a number for how quickly the spring will dissipate energy and slow down

Example: Release a stretched spring



What it does: This example acts like there is a spring between “self” and “ref” such that when the world starts it is as if you just let go of “self”; it will then move toward position and oscillate back and forth until it eventually stops at position.

How it works: When the world starts the object will send itself “tick”. In the “tick” event handler we send “tick” again, so this script will do “tick”, “tick”, “tick”, … every frame forever. Within the “tick” handler the spring-push code block is used to apply one frame of push force to “self” to move toward the position of another object, “ref”. “Self” will move toward ref, but overshoot, and then back-and-forth again until it finally matches the position of “ref”. Since stiffness is “5” it won’t oscillate too fast; the damping of “2” will have it slow down at a moderate pace.

See Also

- Motion > Physical Motion > spring spin object toward rotation

spring spin object toward rotation

Motion > Physical Motion > spring spin object toward rotation

Simulates *one frame* of an object being spun to a rotation / orientation by a *rotational* spring.

Appearance in Composition Pane

Description

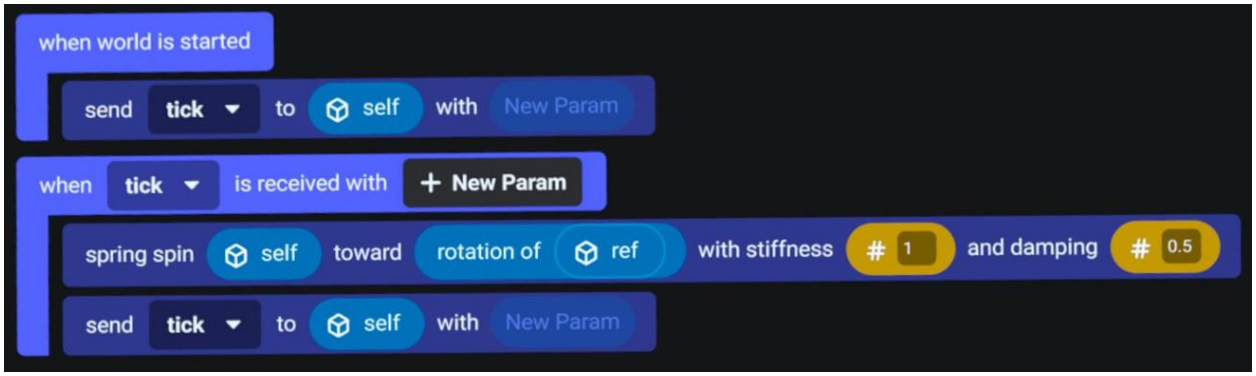
Spring spin applies a force that spins the object toward “rotation”. Just like “rotateTo” this spins the object toward the rotation, but will do so with a spring, meaning that it can rotate too far, then rotate back, and repeatedly oscillate as it slows down and settles into “rotation”.

Note that this code block only calculates the force for *one frame* so you likely want to run it every frame to get a smooth spring-like bounce / oscillation. The **stiffness** parameter controls how strong the spring is; the bigger that value the harder and faster the object will spin toward “rotation”. You’ll get unintuitive behavior if you pass a negative stiffness value. The **damping** parameter controls how much “friction” there is. A low damping value will lead to a lot of bouncing back and forth whereas a high damping value will cause the object to slow down and settle more quickly (as if there was air resistance, friction, or other forces preventing the spring from oscillating forever). The damping value should also be non-negative for realistic behavior.

Parameters

object - the object to move
rotation - the rotation / orientation to spin the object toward
stiffness - a number for how forceful the spring is
damping - a number for how quickly the spring will dissipate energy and slow down

Example: Release a stretched 3D spring



What it does: This example has “self” rotate to match the orientation of “ref”, in a spring-like manner. “Self” will rotate, but “overshoot”, then rotate back but overshoot again, rotate forward, and oscillate repeatedly as it slows down and finally matches the orientation of “ref”.

How it works: When the world starts the object will send itself “tick”. In the “tick” event handler we send “tick” again, so this script will do “tick”, “tick”, “tick”, … every frame forever. Within the “tick” handler the spring-spin codeblock is used to apply one frame of spin force to “self” to rotate toward the rotation / orientation of another object, “ref”. “Self” will rotate toward ref, but overshoot, and then rotate back-and-forth again until it finally matches the position of “ref”. Since stiffness is “1” it won’t oscillate too fast; the damping of “0.5” will have it slow down at a moderate pace.

See Also

- Motion > Physical Motion > spring push object toward position