12.4.4 Game Engine - Logic - Actuators

Actuators

- Introduction
- Actuator Editing
  - Column Heading
  - Object Heading
- Actuator Common Options
- Actuator Types
  - Filter 2D Actuator
  - Action Actuator
  - Camera Actuator
  - Constraints Actuator
  - Edit Object Actuator

- Game Actuator
- Message Actuator
- Motion Actuator
- Parent Actuator
- Property Actuator
- Random Actuator
- Scene Actuator
- Sound Actuator
- State Actuator
- Steering Actuator
- Visibility Actuator

Introduction

Actuators perform actions, such as move, create objects, play a sound. The actuators initiate their functions when they get a positive pulse from one (or more) of their controllers.

The logic blocks for all types of actuator may be constructed and changed using the *Logic Editor*; details of this process are given in the *Actuator Editing* page.

The following types of actuator are currently available:

Action
Handles armature actions. This is only visible if an armature is selected.

Camera
Has options to follow objects smoothly, primarily for camera objects, but any object can use this.

Constraint
Constraints are used to limit object's locations, distance, or rotation. These are useful for controlling the physics of the object in game.

Edit Object
Edits the object's mesh, adds objects, or destroys them. It can also change the mesh of an object (and soon also recreate the collision mesh).

Filter 2D
Filters for special effects like sepia colors or blur.

Game
Handles the entire game and can do things as restart, quit, load, and save.

Message
Sends messages, which can be received by other objects to activate them.

Motion
Sets object into motion and/or rotation. There are different options, from "teleporting" to physically push rotate objects.

Parent
Can set a parent to the object, or unparent it.

Property
Manipulates the object's properties, like assigning, adding, or copying.

Random
Creates random values which can be stored in properties.

Scene

Manage the scenes in your .blend file. These can be used as levels or for UI and background.

Sound

Used to play sounds in the game.

State

Changes states of the object.

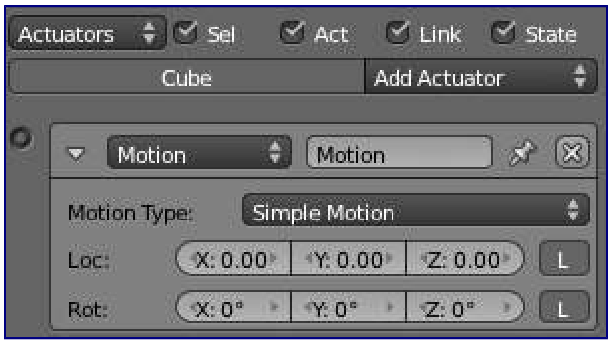
Steering

Provides pathfinding options for the object.

Visibility

Changes visibility of the object.

Actuator Editing

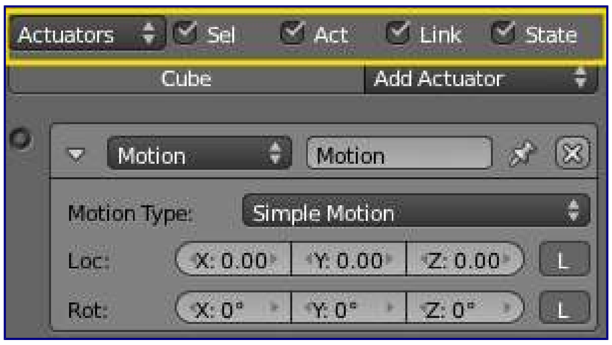


Actuator Column with Typical Actuator

Blender actuators can be set up and edited in the right-hand column of the Logic Panel. This page describes the general column controls, and also those parameters which are common to all individual actuator types.

The image shows a typical actuator column with a single example actuator. At the top of this column, the column heading includes menus and buttons to control which of all the actuators in the current Game Logic are displayed.

Column Heading



Actuator Column Heading

The column headings contain controls to set which actuators, and the level of detail given, in the actuator column. This is very useful for hiding unecessary actuators so that the necessary ones are visible and easier to reach. Both these can be controlled individually.

Actuators

Show Objects
: Expands all objects.

Hide Objects
: Collapses all objects to just a bar with their name.

Show Actuators
: Expands all actuators.

Hide Actuators
: Collapses all actuators to bars with their names.

It is also possible to filter which actuators are viewed using the four heading buttons:

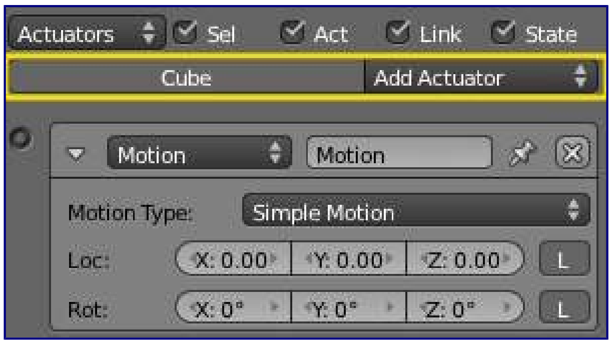
Sel
: Shows all actuators for selected objects.

Act
: Shows only actuators belonging to the active object.

Link
: Shows actuators which have a link to a controller.

State
: Only actuators connected to a controller with active states are shown.

Object Heading



Actuator Object Heading

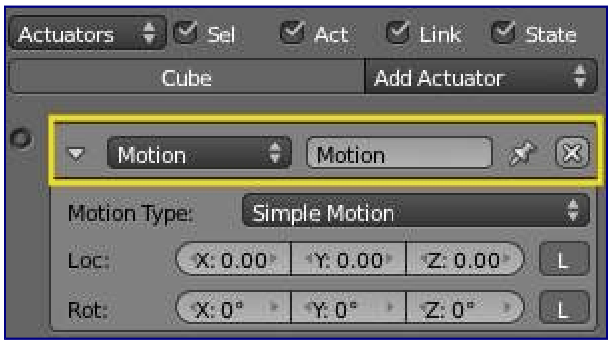
In the column list, actuators are grouped by object. By default, actuators for every selected object appear in the list, but this may be modified by the column heading filters.

At the head of each displayed object sensor list, two entries appear:

Name
: The name of the object.

Add
: When clicked, a menu appears with the available actuator types. Selecting an entry adds a new actuator to the object. See *Actuators* for list of available actuator types.

Actuator Common Options



Motion Type: Simple Motion
Loc: X: 0.00 Y: 0.00 Z: 0.00 L
Rot: X: 0° Y: 0° Z: 0° L

Common Actuator Options

All actuators have a set of common buttons, fields and menus. They are organized as follows:

Triangle button
: Collapses the sensor information to a single line (toggle).

Actuator type menu
: Specifies the type of the sensor.

Actuator name
: The name of the actuator. This can be selected by the user. It is used to access actuators with python; it needs to be unique among the selected objects.

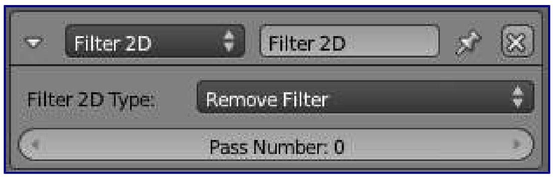
X **Button**
: Deletes the actuator.

Actuator Types

- Filter 2D Actuator
  - Motion Blur
  - Built-In 2D Filters
  - Custom Filters
- Action Actuator
- Camera Actuator
- Constraints Actuator
- Edit Object Actuator
- Game Actuator
- Message Actuator
- Motion Actuator
  - Simple Motion
  - Servo Control
- Parent Actuator
- Property Actuator
  - Example
- Random Actuator
- Scene Actuator
- Sound Actuator
- State Actuator
  - Usage Notes

Filter 2D Actuator

2D Filter s are image filtering actuators, that apply on final render of objects.



Edit Object actuator

Filter 2D Type Select the type of 2D Filter required.

> *Custom Filter Invert Sepia Gray Scale Prewitt Sobel Laplacian Erosion Dilation Sharpen Blur Motion Blur Remove Filter Disable Filter Enable Filter*

Only one parameter is required for all filters:

Pass Number
: The pass number for which this filter is to be used.

Details of the filters are given in the descriptive text below.

Motion Blur

Motion Blur is a *2D Filter* that needs previous rendering information to produce motion effect on objects. Below you can see *Motion Blur* filter in Blender window, along with its logic bricks:

2D Filters: Motion Blur.

You can enable Motion Blur filter using a *Python* controller: from bge import render render.enableMotionBlur(0.85)

And disable it: from bge import render render.disableMotionBlur()

Note

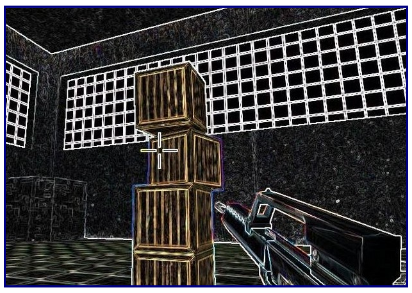
Your graphic hardware and OpenGL driver must support accumulation buffer (`glAccum` function).

Built-In 2D Filters

All 2D filters you can see in *2D Filter* actuator have the same architecture, all built-in filters use fragment shader to produce final render view, so your hardware must support shaders.

2D Filters: Motion Blur.

2D Filters: Sepia.



2D Filters: Sobel.

Blur, *Sharpen*, *Dilation*, *Erosion*, *Laplacian*, *Sobel*, *Prewitt*, *Gray Scale*, *Sepia* and *Invert* are built-in filters. These filters can be set to be available in some passes.

To use a filter you should:

- Create appropriate sensor(s) and controller(s).
- Create a *2D Filter* actuator.
- Select your filter, for example *Blur.*
- Set the pass number that the filter will be applied.

To remove a filter on a specific pass:

- Create appropriate sensor(s) and controller(s).
- Create a *2D Filter* actuator.
- Select *Remove Filter.*
- Set the pass number you want to remove the filter from it.

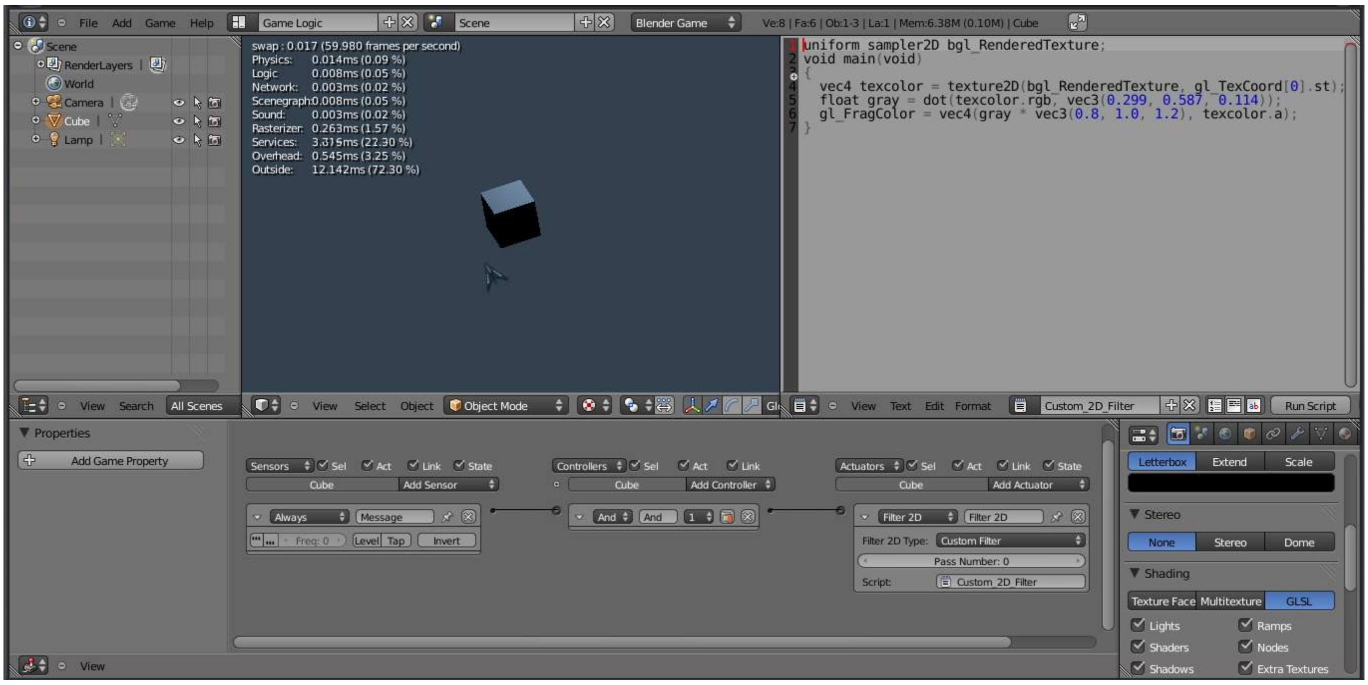
To disable a filter on a specific pass:

- Create appropriate sensor(s) and controller(s).
- Create a *2D Filter* actuator.
- Select *Disable Filter.*
- Set the pass number you want to disable the filter on it.

To enable a filter on a specific pass:

- Create appropriate sensor(s) and controller(s)
- Create a *2D Filter* actuator.
- Select *Enable Filter.*
- Set the pass number you want to enable the filter on it.

Custom Filters


```
swap : 0.017 (59.980 frames per second)
Physics:     0.014ms (0.09 %)
Logic:       0.008ms (0.05 %)
Network:     0.003ms (0.02 %)
Scenegraph:  0.008ms (0.05 %)
Sound:       0.003ms (0.02 %)
Rasterizer:  0.263ms (1.57 %)
Services:    3.315ms (22.30 %)
Overhead:    0.545ms (3.25 %)
Outside:     12.142ms (72.30 %)
```

```
uniform sampler2D bgl_RenderedTexture;
void main(void)
{
  vec4 texcolor = texture2D(bgl_RenderedTexture, gl_TexCoord[0].st);
  float gray = dot(texcolor.rgb, vec3(0.299, 0.587, 0.114));
  gl_FragColor = vec4(gray * vec3(0.8, 1.0, 1.2), texcolor.a);
}
```

Filter 2D Type: Custom Filter
Pass Number: 0
Script: Custom_2D_Filter

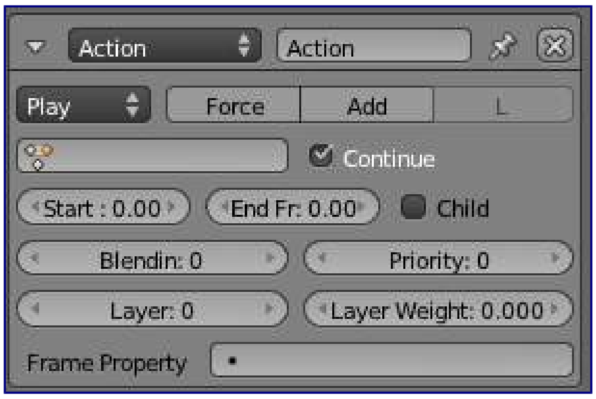
2D Filters: Custom Filter.

Custom filters give you the ability to define your own 2D filter using GLSL. Its usage is the same as built-in filters, but you must select *Custom Filter* in *2D Filter* actuator, then write shader program into the Text Editor, and then place shader script name on actuator.

Blue Sepia Example:

```
uniform sampler2D bgl_RenderedTexture;
void main(void)
{
  vec4 texcolor = texture2D(bgl_RenderedTexture, gl_TexCoord[0].st);
  float gray = dot(texcolor.rgb, vec3(0.299, 0.587, 0.114));
  gl_FragColor = vec4(gray * vec3(0.8, 1.0, 1.2), texcolor.a);
}
```

Action Actuator



Action Actuator

Actuates armature actions, and sets the playback method. The Action actuator is only visible when an armature is selected, because actions are stored in the armature.

See *Actuator Common Options* for common options.

Special Options: **Action Playback Type**

Play

Play ipo once from start to end when a TRUE pulse is received.

Ping Pong

Play ipo once from start to end when a TRUE pulse is received. When the end is reached play ipo once from end to start when a TRUE pulse is received.

Flipper

Play ipo once from start to end when a TRUE pulse is received. (Plays backwards when a FALSE pulse is received).

Loop End

Play ipo continuously from end to start when a TRUE pulse is received.

Loop Start

Play ipo continuously from start to end when a TRUE pulse is received.

Property

Uses a property to define what frame is displayed.

Action
Select the action to use

Continue
Restore last frame when switching on/off, otherwise play from the start each time.

Start Frame
Set the start frame of the action.

End Frame

Set the end frame of the action.

Child Button
: Update action on all children objects as well.

Blendin
: Number of frames of motion blending.

Priority
: Execution priority - lower numbers will override actions with higher numbers. With 2 or more actions at once, the overriding channels must be lower in the stack.

Frame Property
: Assign the action's current frame number to this property.

Property
: Use this property to define the Action position. Only for Property playback type.

Layer
: The animation layer to play the action on.

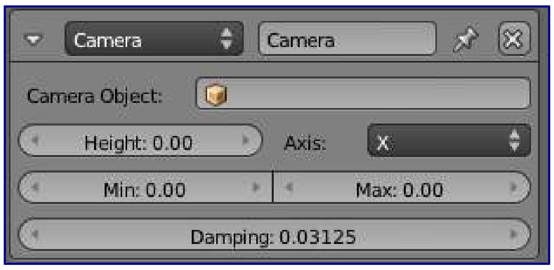
Layer Weight
: How much of the previous layer to blend into this one.

Camera Actuator

Makes the camera follow or track an object.

See *Actuator Common Options* for common options.

Special Options:



Camera Actuator

Camera Object
: Name of the Game Object that the camera follows/tracks.

Height
: Height the camera tries to stay above the Game Object's object center

Axis
: Axis in which the Camera follows (X or Y)

Min
: Minimum distance for the camera to follow the Game Object

Max
: Maximum distance for the camera to follow the Game Object

Damping
: Strength of the constraint that drives the camera behind the target. Range: 0 to 10. The higher the parameter, the quicker the camera will adjust to be inside the constrained range (of min, max and height).

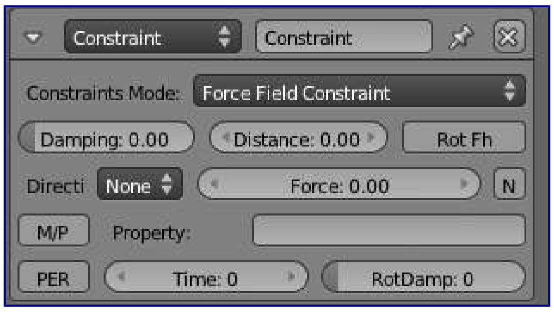
Constraints Actuator

Adds a constraint to the location, orientation

See *Actuator Common Options* for common options.

Special Options: **Constraint Mode** Menu specifying type of constraint required.

- Force Field Constraint
- Orientation Constraint
- Distance Constraint
- Location Constraint



Constraint actuator - Force Field

Force Field Constraint

Create a force field buffer zone along one axis of the object.

Damping
Damping factor of the Fh spring force (Range 0.0 - 1.0)

Distance
Height of Fh area

Rot Fh
Make game object axis parallel to the normal of trigger object.

Direction
Axis in which to create force field (can be + or -, or None)

Force
Force value to be used.

N
When on, use a horizontal spring force on slopes

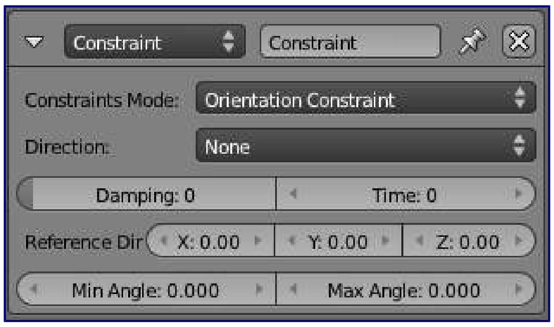
M/P
Trigger on another Object will be either Material (M) or Property (P)

Property
Property/Material that triggers the Force Field constraint (blank for ALL Properties/Materials)

Per
Persistence button When on, force field constraint always looks at Property/Material; when off, turns itself off if it can’t find the Property/Material.

Time
Number of frames for which constraint remains active

RotDamp
Damping factor for rotation



Constraint Actuator - Orientation

Orientation Constraint Constrain the specified axis in the Game to a specified direction in the World axis.

Direction

Game axis to be modified (X, Y, Z or none)

Damping

Delay (frames) of the constraint response (0 - 100)

Time

Time (frames) for the constraint to remain active (0 - 100)

ReferenceDir

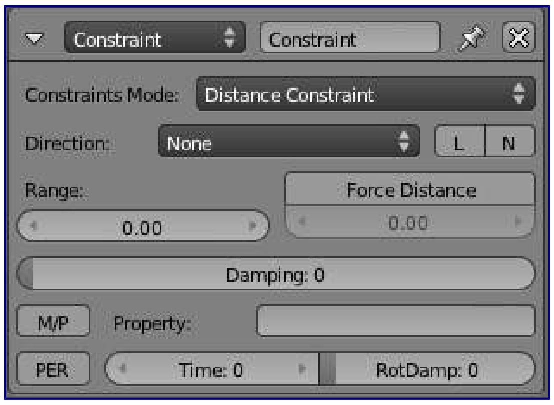
Reference direction (global coordinates) for the specified game axis.

MinAngle

Minimum angle for the axis modification;

MaxAngle

Maximum angle for the axis modification;



Constraint actuator - Distance

Distance Constraint Maintain the distance the Game Object has to be from a surface

Direction

Axis Direction (X, Y, Z, -X, -Y, -Z, or None)

L

If on, use local axis (otherwise use World axis)

N

If on, orient the Game Object axis with the mesh normal.

Range

Maximum length of ray used to check for Material/Property on another game object (0 - 2000 Blender Units)

Force Distance

Distance to be maintained between object and the Material/Property that triggers the Distance Constraint(-2000 to +2000 Blender Units).

Damping

Delay (frames) of the constraint response (0 - 100)

M/P

Trigger on another Object will be either Material (M) or Property (P)

Property

Property/Material that triggers the Force Field constraint (blank for ALL Properties/Materials)

Per

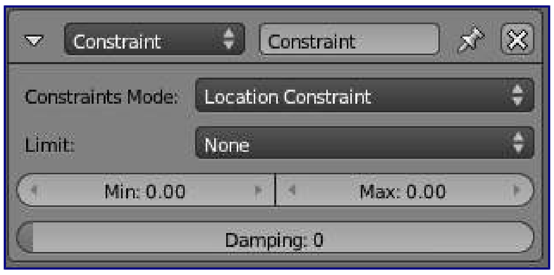
Persistence button: When on, force field constraint always looks at Property/Material; when off, turns itself off if it can’t find the Property/Material.

Time

Number of frames for which constraint remains active

RotDamp

Damping factor for rotation


Constraint
Constraint
Constraints Mode: Location Constraint
Limit: None
Min: 0.00
Max: 0.00
Damping: 0

Constraint actuator - Location

Location Constraint Limit the position of the Game Object within one World Axis direction. To limit movement within an area or volume, use two or three constraints.

Limit

Axis in which to apply limits (LocX, LocY, LocZ or none)

Min

Minimum limit in specified axis (Blender Units)

Max

Maximum limit in specified axis (Blender Units)

Damping

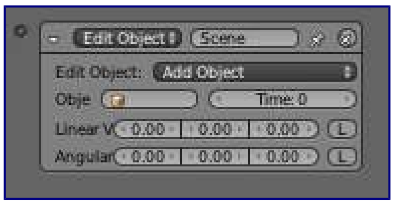
Delay (frames) of the constraint response (0 - 100)

Edit Object Actuator

The Edit Object actuator allows the user to edit settings of objects in game

See *Actuator Common Options* for common options.

Special Options:


Edit Object | Scene

Edit Object: Add Object

Obje | Time: 0

Linear V 0.00 0.00 0.00 L

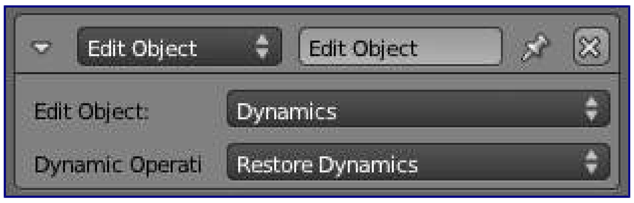
Angular 0.00 0.00 0.00 L

Edit Object actuator

Edit Object

Menu of options for Edit Object actuator

Dynamics Track To Replace Mesh End Object Add Object


Edit Object | Edit Object

Edit Object: Dynamics

Dynamic Operati Restore Dynamics

Edit Object actuator - Dynamics

Dynamics

Provides a menu of *Dynamic Operations* to set up dynamics options for object.

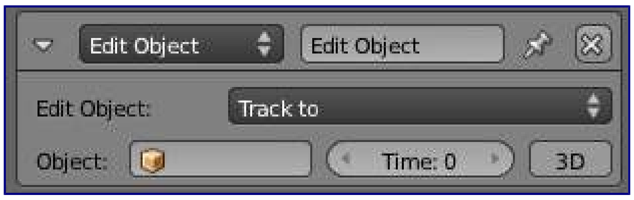
Set Mass
: Enables the user to set the mass of the current object for Physics (Range 0 - 10,000).

Disable Rigid Body
: Disables the Rigid Body state of the object - disables collision.

Enable Rigid Body
: Disables the Rigid Body state of the object - enables collision.

Suspend Dynamics
: Suspends the object dynamics (object velocity).

Restore Dynamics
: Resumes the object dynamics (object velocity).



Edit Object actuator - Track to

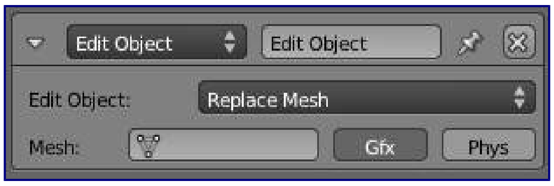
Track To

Makes the object "look at" another object, in 2D or 3D. The Y-axis is considered the front of the object.

Object
: Object to follow.

Time
: No. of frames it will take to turn towards the target object (Range 0-2000).

3D Button (toggle).
: Enable 2D (X,Y) or 3D (X,Y,Z) tracking.



Edit Object actuator - Replace Mesh

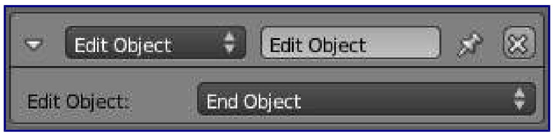
Replace Mesh

Replace mesh with another. Both the mesh and/or its physics can be replaced, together or independently.

Mesh
: name of mesh to replace the current mesh.

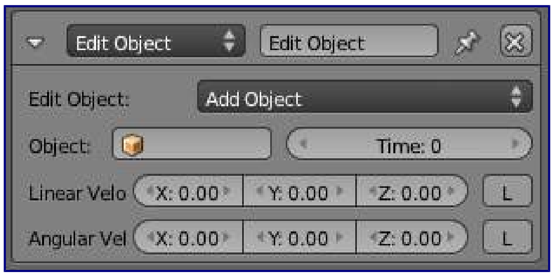
Gfx Button
: replace visible mesh.

Phys Button
: replace physics mesh (not compound shapes)



Edit Object actuator - End Object

End Object Destroy the current object (Note, debug properties will display error Zombie Object in console)


Edit Object | Edit Object
Edit Object: Add Object
Object: | Time: 0
Linear Velo X: 0.00 Y: 0.00 Z: 0.00 L
Angular Vel X: 0.00 Y: 0.00 Z: 0.00 L

Edit Object actuator - Add Object

Add Object

Adds an object at the centre of the current object.

The object that is added needs to be on another, hidden, layer.

Object
: The name of the object that is going to be added.:

Time
: The time (in frames) the object stays alive before it disappears. Zero makes it stay forever.

Linear Velocity
: Linear Velocity, works like in the motion actuator but on the created object instead of the object itself. Useful for shooting objects, create them with an initial speed.

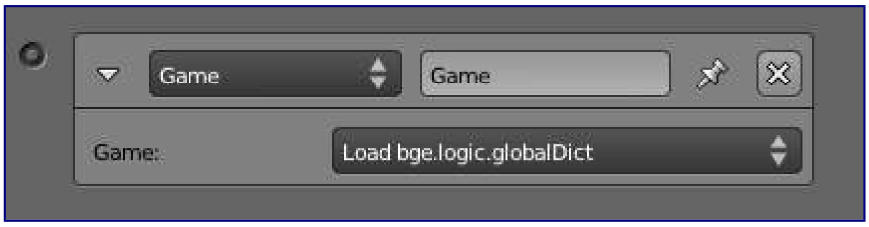
Angular Velocity
: Angular velocity, works like in the motion actuator but on the created object instead of the object itself.

Game Actuator

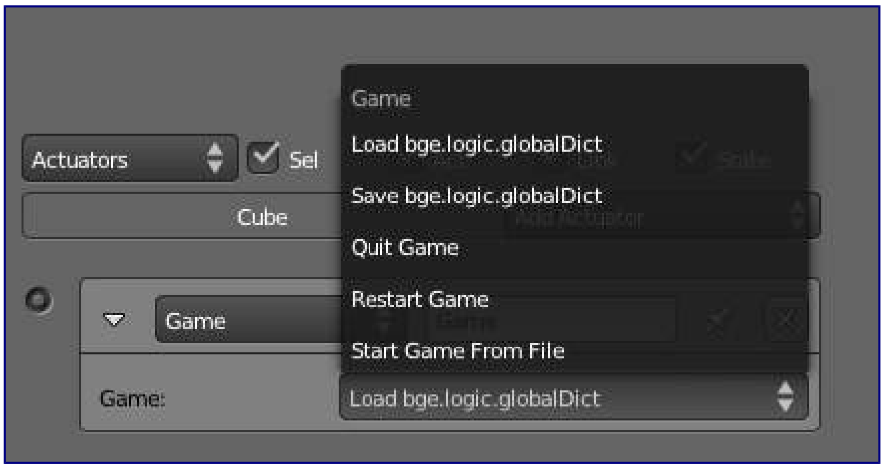
The Game actuator allows the user to perform Game-specific functions, such as Restart Game, Quit Game and Load Game.

See *Actuator Common Options* for common options.

Special Options:


Game | Game
Game: Load bge.logic.globalDict

Game actuator


Actuators
Sel
Cube
Game
Game:
Game
Load bge.logic.globalDict
Save bge.logic.globalDict
Quit Game
Restart Game
Start Game From File
Load bge.logic.globalDict

Game

Game

Load bge.logic.globalDict
Load *bge.logic.globalDict* from .bgeconf.

Save bge.logic.globalDict
Save *bge.logic.globalDict* to .bgeconf.

Quit Game
Once the actuator is activated, the blenderplayer exits the runtime.

Restart Game
Once the actuator is activated, the blenderplayer restarts the game (reloads from file).

Start Game From File

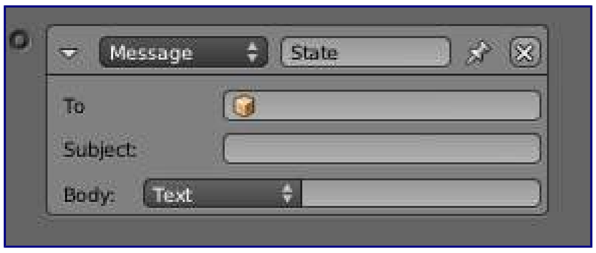
Once the actuator is activated, the blenderplayer starts the .blend file from the path specified.

File
Path to the .blend file to load.

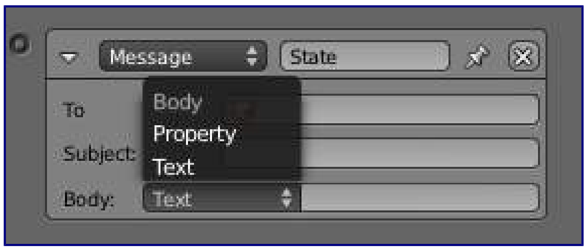
Notes If you use the keyboard sensor as a hook for the `Esc` key, in the event that the quit game actuator fails, such as an error in a python file, the game will be unable to close. Data may be recovered from quit.blend File ‣ Recover Last Session

Message Actuator

The Message actuator allows the user to send data across a scene, and between scenes themselves.



Message actuator



Message actuator Options

See *Actuator Common Options* for common options.

Special Options:

To
: Object to broadcast to. Leave blank if broadcast to all (or sending to another scene).

Subject
: Subject of message. Useful if sending certain types of message, such as “end-game”, to a message sensor listening for “end game”–>AND–>Quit Game actuator

Body
: Body of message sent (only read by Python*).

Text
: User specified text in body.

Property
: User specified property.

Usage Notes You can use the Message Actuator to send data, such as scores to other objects, or even across scenes! (alternatively use `bge.logic.globalDict`).

Motion Actuator

The Motion actuator sets an object into motion. There are two modes of operation, Simple or Servo, in which the object can either teleport & rotate, or dynamically move.

See *Actuator Common Options* for common options.

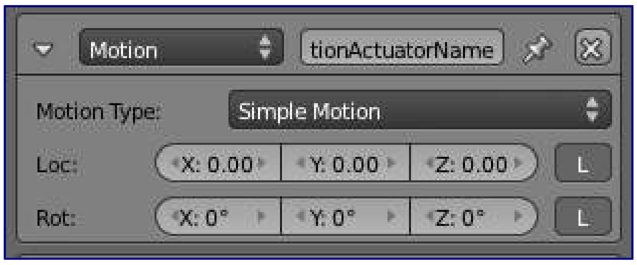
Special Options: **Motion Type**, which determines the type of motion:

Simple Motion
: Applies a change in location and/or rotation directly.

Servo Control
: Sets a target speed, and also how quickly it reaches that speed.

The Simple Motion actuator gives control over position and velocity, but does this as an instant displacement; the object never passes any of the coordinates between the start and end positions. This can interfere with the physical simulation of other objects, and can cause an object to go through another object. The Servo Control actuator does not suffer from this, since it produces physically correct velocities, and leaves updating the position to the physics simulation.

Simple Motion



Motion actuator for Simple Motion

Loc

The object jumps the number of blender units entered, each time a pulse is received.

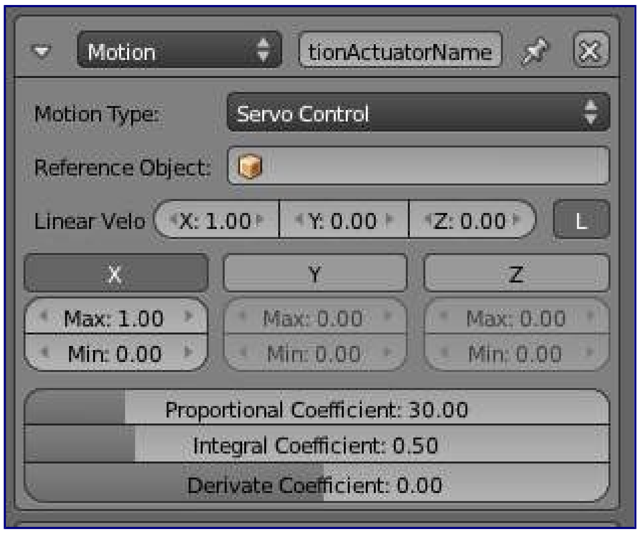
Rot

The object rotates by the specified amount, each time a pulse is received.

L

Coordinates specified are Global (gray) or Local (White).

Servo Control



Motion actuator set to *Servo Control*

The Servo Control actuator influences the velocity of a game object by applying forces, resulting in correct behavior when colliding with other objects controlled by the physics simulation. The amount of force necessary is determined by a PID controller, a type of controller that is often used in control systems. Only the positional velocity is influenced by this actuator; it does not control rotation at all, and it controls position only indirectly.

Controlling the position is not necessary in that respect; that is left to a player moving the object via direction-type controls (such as the WSAD keys in a first person shooter). In such a scenario, each direction-key sensor should be attached to a different Servo Control actuator setting a different target velocity.

Tip

To use the Servo Control actuator, it is necessary to set the object's Physics Type to "Dynamic" or "Rigid Body", and to mark the object as "Actor" in the same panel. This actuator does not work with the Character physics type.

Reference Object
: Specifies the object which the actuator uses as a reference for the velocity. When set, it will use a velocity relative to that object instead of absolute (i.e. world-relative) velocity. Use this for a player object standing on a moving platform.

Linear Velocity
: The target linear velocity for the object.

L
: Determines whether the Linear Velocity specified are in Local (button depressed) or Global (button released) coordinates.

X, Y, Z force limits
: Sets minimum and maximum limits for the force applied to the object. If disabled (i.e. X, Y or Z buttons are depressed) the force applied is unlimited.

The following three coefficients determine the response to the *velocity error*, which is the difference between the target velocity and the object's actual velocity.

Proportional Coefficient
: This controls the reaction proportional to the velocity error. Small values cause smooth (but possibly too slow) changes in velocity. Higher values cause rapid changes, but may cause overshooting.

Integral Coefficient
: This controls the reaction to the sum of errors so far. Using only the Proportional component results in a systematic velocity error if there is friction: some velocity delta is necessary to produce the force that compensates the friction. Using the Integral component suppresses this effect (the target velocity is achieved on average) but can create oscillations; the control will speed to compensate the initial velocity error. To avoid the oscillation, the Proportional component must be used with the Integral component (the Proportional component damps the control) This is why the GUI sets the Proportional Coefficient systematically when you change the Integral Coefficient.

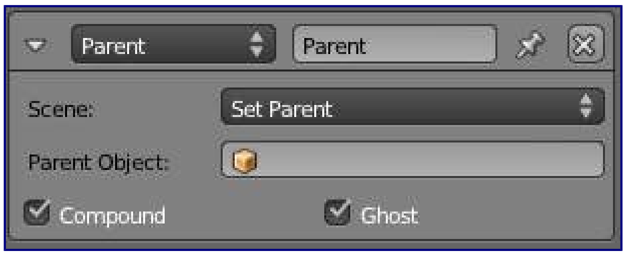
Derivative Coefficient
: Set the Derivative Coefficient. This dampens the acceleration when the target velocity is almost reached.

Parent Actuator

Enables you to change the parent relationships of the current object.

See *Actuator Common Options* for common options.

Special Options: **Scene** Menu for parenting operation required.


Parent Parent
Scene: Set Parent
Parent Object:
☑ Compound ☑ Ghost

Parent Actuator

Set Parent

Make this object to be current object’s parent

Parent Object
Name of parent object
Compound’
Add this object shape to the parent shape (only if the parent shape is already compound)
Ghost’
Make this object ghost while parented
Remove Parent

Remove all parents of current object

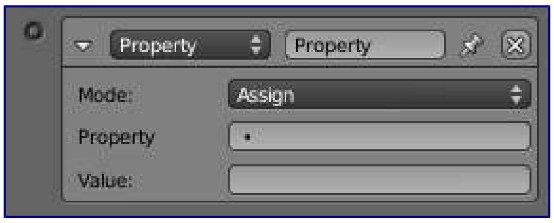
Parent Object
Name of parent object

Property Actuator

Using the Property actuator you can change the value of a given property once the actuator itself is activated.

See *Actuator Common Options* for common options.

Special Options:


Property
Property
Mode: Assign
Property
Value:

Property actuator

Mode

Assign
the *Property* target property will become equal to the set *Value* once the actuator is activated
Add
adds *Value* to the value of the property *Property* once the actuator is activated (enter a negative value to decrease). For *Bool*, a value other than 0 (also negative) is counted as True.
Copy
copies a property from another object to a property of the actuator owner once the actuator is activated.
Toggle
switches 0 to 1 and any other number than 0 to 0 once the actuator is activated. Useful for on/off switches.
Property
The target property that this actuator will change
Value
The value to be used to change the property

Example

You have a character, it has a property called “hp” (hit points) to determine when he has taken enough damage to die. hp is an int with the start value of 100.

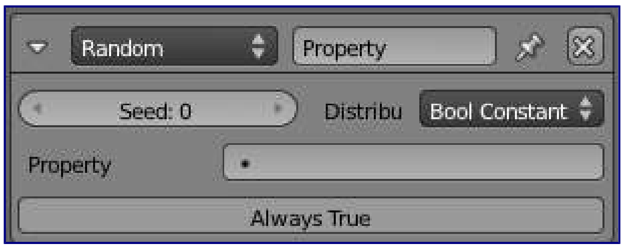
You set up two *Collision* sensors, one for enemy bullets, and one for picking up more health. The first one is connected (through an *AND* controller) to an *Add Property* actuator with the property hp and the value -10. Every time the player is hit by an enemy bullet he loses 10 hp. The other sensor is connected (through an *AND* controller) to an other *Add Property* actuator, this one with the value 50. So every time the player collides with a health item the hp increases by 50. Next you set up a *Property* sensor for an interval, greater than 100. This is connected (through an *AND* controller) to an *Assign Property* actuator which is set to 100. So if the players hp increases over 100 it is set to 100.

Random Actuator

Sets a random value into a property of the object

See *Actuator Common Options* for common options.

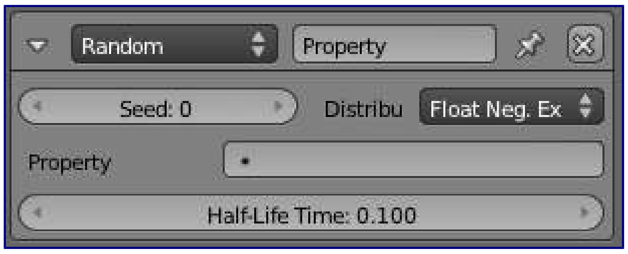
Special Options:



Camera Actuator

Seed Starting seed for random generator (range 1 - 1000)

Distribution Menu of distributions from which to select the random value. The default entry of Boolean Constant gives either True or False, which is useful for test purposes.



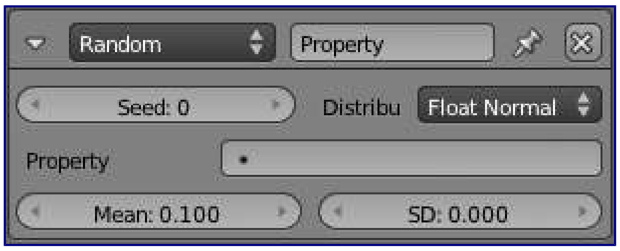
Float Neg. Exp.

Float Neg. Exp.

Values drop off exponentially with the specified half-life time.

Property
Float property to receive value

Half-Life Time
Half-life time (Range 0.00 -10000.00)



Float Normal

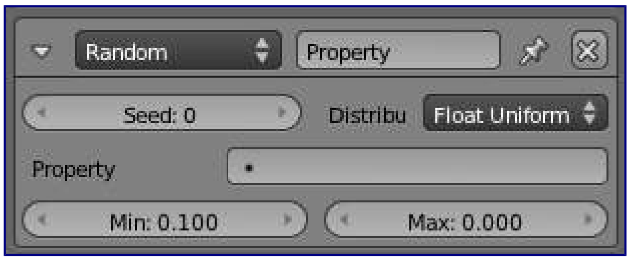
Float normal

Random numbers from a normal distribution.

Property
: Float property to receive value

Mean
: Mean of normal distribution (Range -10000.00 to +10000.00)

SD
: Standard deviation of normal distribution (Range 0.00 to +10000.00)



Float Uniform

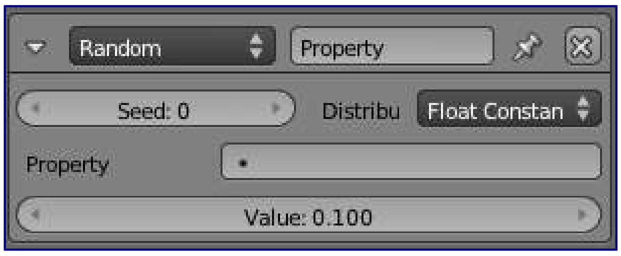
Float uniform

Random values selected uniformly between maximum and minimum.

Property
: Float property to receive value

Min
: Minimum value (Range -10000.00 to +10000.00)

Max
: Maximum value (Range -10000.00 to +10000.00)



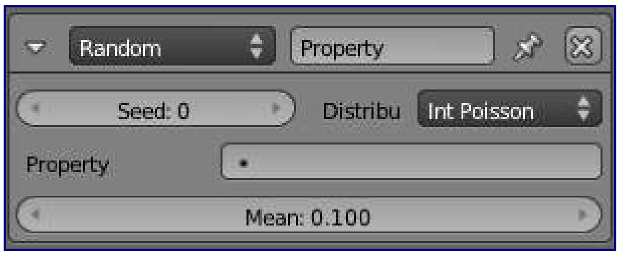
Float Constant

Float constant

Returns a constant value.

Property
: Float property to receive value

Value
: Value (Range 0.00 to +1.00)



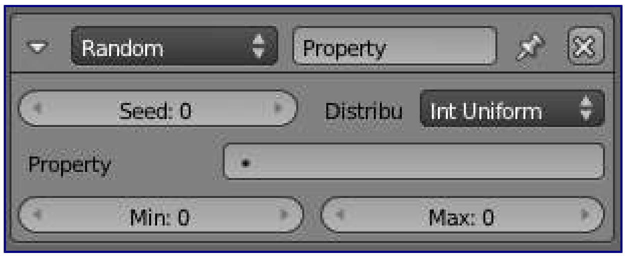
Random Integer Poisson

Int Poisson

Random numbers from a Poisson distribution.

Property
: Integer property to receive value

Mean
: Mean of Poisson distribution (Range 0.01 to +100.00)



Random Integer Uniform

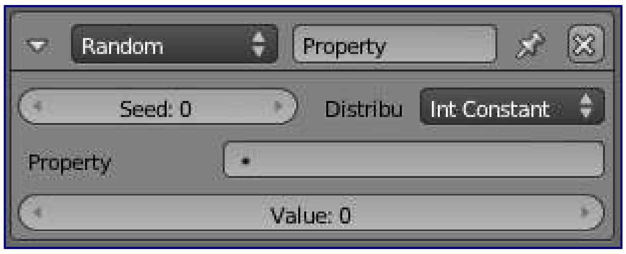
Int uniform

Random values selected uniformly between maximum and minimum.

Property
: Integer property to receive value

Min
: Minimum value (Range -1000 to +1000)

Max
: Maximum value (Range -1000 to +1000)

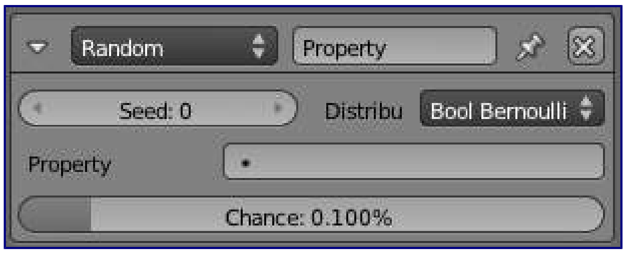


Random Integer Constant

Int constant

Returns a constant value.

Property
Integer property to receive value
Value
Value (Range 0.00 to +1.00)



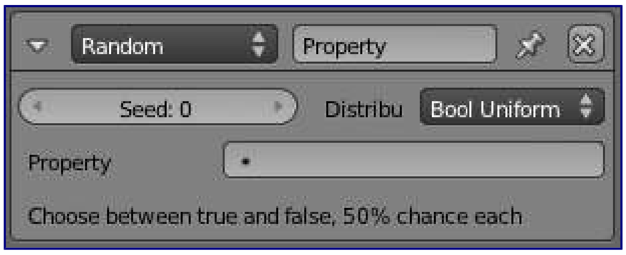
Seed: 0
Distribu: Bool Bernoulli
Property
Chance: 0.100%

Random Bool Bernoulli

Bool Bernoulli

Returns a random distribution with specified ratio of TRUE pulses.

Property
Boolean property to receive value
Chance
Proportion of TRUE responses required.



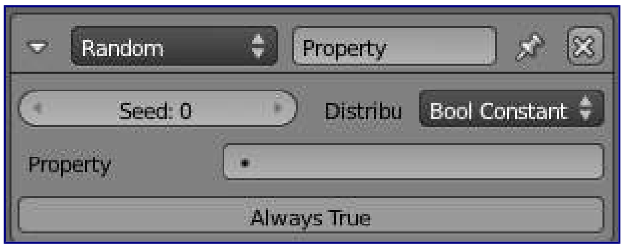
Seed: 0
Distribu: Bool Uniform
Property
Choose between true and false, 50% chance each

Random Bool Uniform

Bool uniform

A 50/50 chance of obtaining True/False.

Property
Boolean property to receive value



Seed: 0
Distribu: Bool Constant
Property
Always True

Random Bool Constant

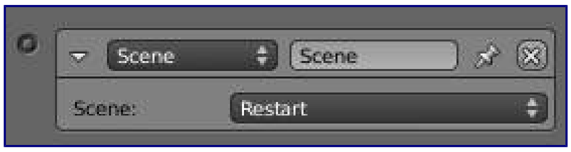
Bool constant

Returns a constant value.

Property
: Boolean property to receive value

Value
: Value (True or False)

Scene Actuator

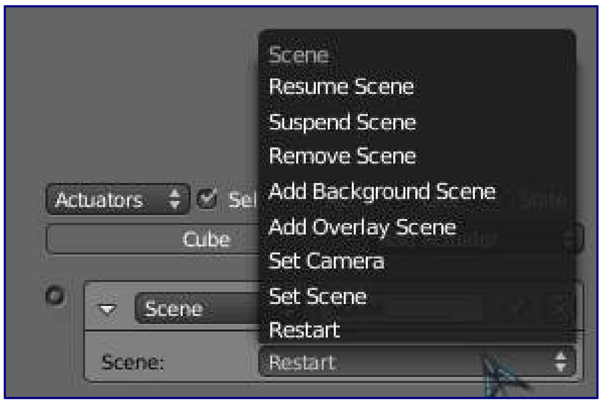


Scene actuator

The *Scene* actuator manages the scenes in your .blend file, these can be used as levels or for UI and background.

See *Actuator Common Options* for common options.

Special Options: The actuator has eight modes:



Scene actuator options

Restart
: Restarts the current scene, everything in the scene is reset

Set Scene
: Changes scene to selected one

Set Camera
: Changes which camera is used

Add OverlayScene
: This adds an other scene, and draws it on top of the current scene. It is good for interfacing: keeping the health bar, ammo meter, speed meter in an overlay scene makes them always visible.

Add BackgroundScene
: This is the opposite of an overlay scene, it is drawn behind the current scene

Remove Scene
: Removes a scene.

Suspend Scene
: Pauses a scene

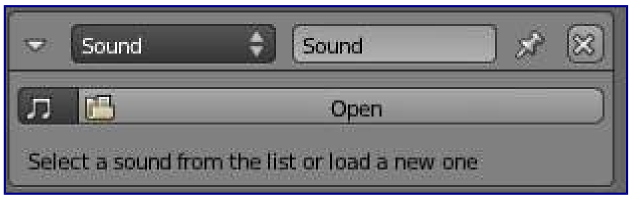
Resume Scene
: Resumes a paused scene.

Note

A scene that it is paused can not resume itself. You need an active scene to resume other scene that it is paused.

Sound Actuator

Select a sound file from the list or make a new one.



Select a sound from the list or load a new one

Sound Actuator

See *Actuator Common Options* for common options.

Special Options:

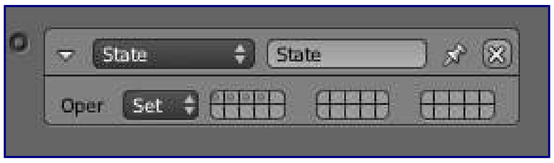
Music File title
Select music file from the list presented.

State Actuator

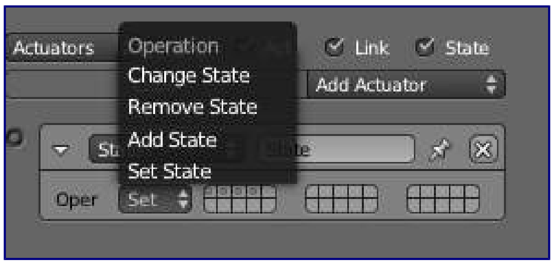
The State actuator allows the user to create complex logic, whilst retaining a clear user interface. It does this by having different states, and performing operations upon them

See *Actuator Common Options* for common options.

Special Options:



State actuator



State actuator options

Operation

Menu to select the state operation required.

Change State
: Change from the current state to the state specified.

Remove State
: Removes the specified states from the active states (deactivates them).

Add State
: Adds the specified states to the active states (activates them).

Set State
: Moves from the current state to the state specified, deactivating other added states.

Usage Notes

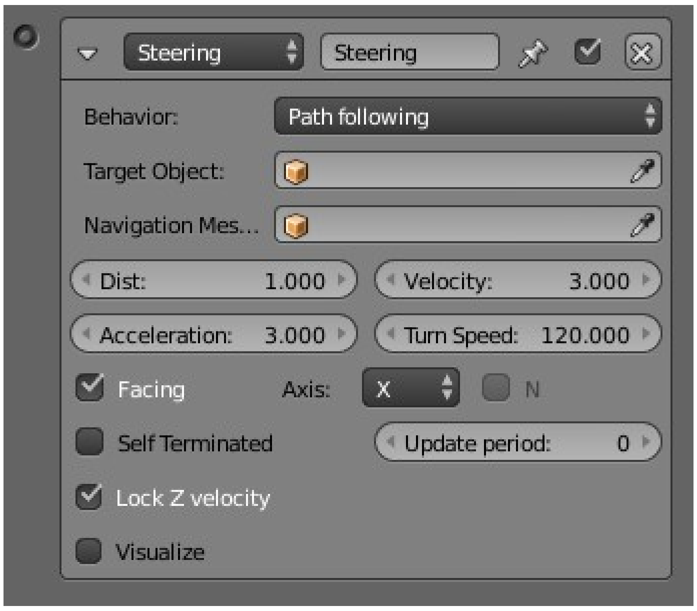
With the state actuator, you can create tiers of logic, without the need for hundreds of properties. Use it well, and you benefit greatly, but often problems may be circumvented by python.

Steering Actuator

The steering actuator moves an object towards a target object, with options to seek, flee, or follow a path. This actuator will not actually try to avoid obstacles by deviating the objects course.

See *Actuator Common Options* for common options.

Options



Steering | Steering

Behavior: Path following

Target Object:

Navigation Mes...

Dist: 1.000 | Velocity: 3.000

Acceleration: 3.000 | Turn Speed: 120.000

✓ Facing | Axis: X | N

Self Terminated | Update period: 0

✓ Lock Z velocity

Visualize

Steering Actuator Panel

Behavior:
: Seek, Flee or Path following

Target Object:
: The game object to seek.

Navigation Mesh Object:
The name of the navigation mesh object used by the Steering Actuator when in Path following behavior. The game object will use the Navigation Mesh to create a path to follow the Target Object.

Tip

You can create your own mesh to use for navigation and make it a Navigation Mesh in:

- Properties Window → Physics context → Physics panel → choosing Physics Type: Navigation Mesh

Or you can let Blender create a Navigation Mesh, then select a mesh. (Floor or ground or etc.)

- Properties Window → Scene context → Navigation mesh object panel → Build navigation mesh

Dist
The maximum distance for the game object approach the Target Object.

Velocity
The velocity used to seek the Target Object.

Acceleration
The maximum acceleration to use when seeking the Target Object.

Turn Speed
The maximum turning speed to use when seeking the Target Object.

Facing:
Set a game object axis that always faces the Target Object.

Axis
The game object axis that always faces the Target Object. Options are: Positive (X, Y, Z) and Negative (-X, - Y, -Z).

Axis N
Use the Normal of the Navigation Mesh to align the up vector of the game object.

Self Terminated:

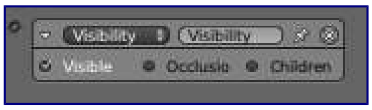
Disabled
Stops moving toward the Target Object once it reaches the maximum distance to approach the Target Object. Will follow the Target Object if it moves further away than the maximum distance.

Enabled
Stops moving toward the Target Object once it reaches the maximum distance to approach the Target Object. Won’t follow even if the Target Object moves further away than the maximum distance.

Visualize
This checkbox let the user specify whether to show or not the debug informations of the actuator. It is also necessary to enable Debug Properties in the Display menu of the render context.

Visibility Actuator

The Visibility actuator allows the user to change the visibility of objects during runtime.



Visibility actuator

See *Actuator Common Options* for common options.

Special Options:

Visible
: Toggle checkbox to toggle visibility

Occlusion
: Toggle checkbox to toggle occlusion. Must be initialized from the *Physics* tab.

Children
: Toggle checkbox to toggle recursive setting - will set visibility / occlusion state to all child objects, children of children (recursively)

Usage Notes

Using the visiblity actuator will save on Rasterizer usage, however not Physics, and so is limited in terms of Level of Detail (LOD). For LOD look at replace mesh, but be aware that the logic required can negate the effect of the LOD.