7.2.6 Rigging - Constraints - Relationship

Action Constraint

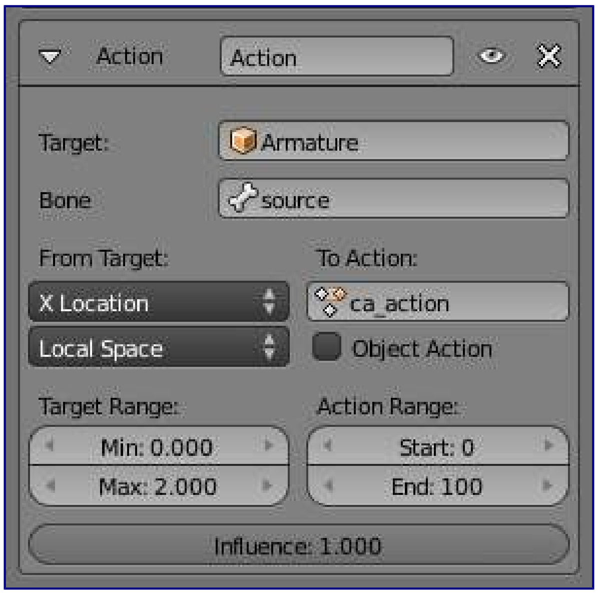
The *Action* constraint is powerful. It allows you control an *Action* using the transformations of another object.

The underlying idea of the *Action* constraint is very similar to the one behind the *Drivers*, except that the former uses a whole action (i.e. a bunch a Fcurves of the same type), while the latter controls a single Fcurve of their “owner”...

Note that even if the constraint accepts the *Mesh* action type, only the *Object*, *Pose* and *Constraint* types are really working, as constraints can only affect objects’ or bones’ transform properties, and not meshes’ shapes. Also note that only the object transformation (location, rotation, scale) is affected by the action, if the action contains keyframes for other properties they are ignored, as constraints do not influence those.

As an example, let’s assume you have defined an *Object* action (it can be assigned to any object, or even no object at all), and have mapped it on your owner through an *Action* constraint, so that moving the target in the `[0.0, 2.0]` range along its X axis maps the action content on the owner in the `[0, 100]` frame range. This will mean that when the target’s X property is `0.0` the owner will be as if in frame `0` of the linked action; with the target’s X property at `1.0` the owner will be as if in frame `50` of the linked action, etc.

Options



Action | Action

Target: Armature

Bone: source

From Target: X Location, Local Space

To Action: ca_action

Object Action

Target Range: Min: 0.000, Max: 2.000

Action Range: Start: 0, End: 100

Influence: 1.000

Action panel

Target
This constraint uses one target, and is not functional (red state) when it has none.

Bone:
When target is an armature object, use this field to select the target bone.

Transform Channel
This drop-down list controls which transform property (location, rotation or scale along/around one of its axes) from the target to use as "action driver".

Target Space
This constraint allows you to choose in which space to evaluate its target's transform properties.

To Action

Select the name of the action you want to use.

> **Warning**
>
> Even though it might not be in red state (UI refresh problems...), this constraint is obviously not functional when this field does not contain a valid action.

Object Action
Bones only, when enabled, this option will make the constrained bone use the "object" part of the linked action, instead of the "same-named pose" part. This allows you to apply the action of an object to a bone.

Target Range Min / Max

The lower and upper bounds of the driving transform property value. By default, both values are set to `0.0`

> **Warning**
>
> Unfortunately, here again we find the constraints limitations:
>
> - When using a rotation property as "driver", these values are "mapped back" to the `[-180.0- ,`

`180.0- [` range.

- When using a scale property as “driver”, these values are limited to null or positive values.

Action Range Start / End

The starting and ending frames of the action to be mapped. Note that:

- These values must be strictly positive.
- By default, both values are set to `0` which disables the mapping (i.e. the owner just gets the properties defined at frame `0` of the linked action...).

Notes

- When the linked action affects some location properties, the owner’s existing location is added to the result of evaluating this constraint (exactly as when the *Offset* button of the *Copy Location constraint* is enabled...).
- When the linked action affects some scale properties, the owner’s existing scale is multiplied with the result of evaluating this constraint.
- When the linked action affects some rotation properties, the owner’s existing rotation is overridden by the result of evaluating this constraint.
- Unlike usual, you can have a *Start* value higher than the *End* one, or a *Min* one higher than a *Max* one: this will reverse the mapping of the action (i.e. it will be “played” reversed...), unless you have both sets reversed, obviously!
- When using a *Constraint* action, it is the constraint *channel’s names* that are used to determine to which constraints of the owner apply the action. E.g. if you have a constraint channel named “trackto_empt1”, its keyed *Influence* and/or *Head/Tail* values (the only ones you can key) will be mapped to the ones of the owner’s constraint named “trackto_empt1”.
- Similarly, when using a *Pose* action (which is obviously only meaningful and working when constraining a bone!), it is the bone’s name that is used to determine which bone *channel’s names* from the action to use (e.g. if the constrained bone is named “arm”, it will use and only use the action’s bone channel named “arm”...). Unfortunately, using a *Pose* action on a whole armature object (to affect all the keyed bones in the action at once) won’t work...
- Note also that you can use the *pose library feature* to create/edit a *Pose* action data-block... just remember that in this situation, there’s one pose per frame!

Child Of Constraint

Child Of is the constraint version of the standard parent/children relationship between objects (the one established through the `Ctrl-P` shortcut, in the 3D views).

Parenting with a constraint has several advantages and enhancements, compared to the traditional method:

- You can have several different parents for the same object (weighting their respective influence with the

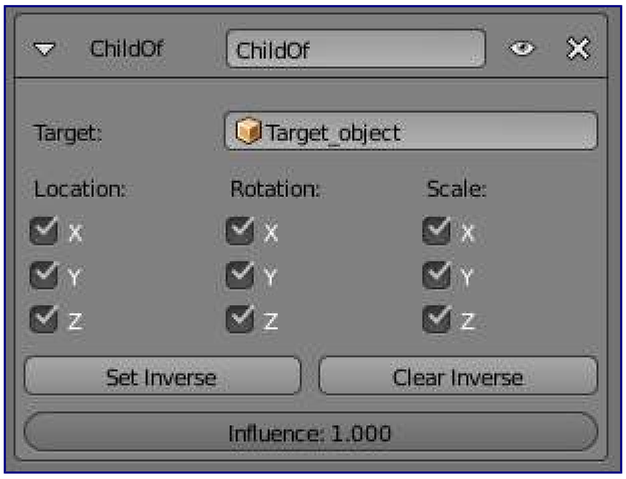
Influence slider).

- As with any constraint, you can key (i.e. animate) its Influence setting. This allows the object which has a Child Of constraint upon it to change over time which target object will be considered the parent, and therefore have influence over the Child Of constrained object.

Warning

Don’t confuse this “basic” object parenting with the one that defines the chains of bones inside of an armature. This constraint is used to parent an object to a bone (the so-called *object skinning*), or even bones to bones. But don’t try to use it to define chains of bones.

Options



Child Of panel

Target
: The target object that this object will act as a child of. This constraint uses one target, and is not functional (red state) when it has none. If *Target* is an armature or a mesh, a new name field appears where a name of a *Bone* or a *Vertex Group* can be selected.

Location X, Y, Z
: Each of these buttons will make the parent affect or not affect the location along the corresponding axis.

Rotation X, Y, Z
: Each of these buttons will make the parent affect or not affect the rotation around the corresponding axis.

Scale X, Y, Z
: Each of these buttons will make the parent affect or not affect the scale along the corresponding axis.

Set Inverse
: By default, when you parent your owner to your target, the target becomes the origin of the owner’s space. This means that the location, rotation and scale of the owner are offset by the same properties of the target. In other words, the owner is transformed when you parent it to your target. This might not be desired! So, if you want to restore your owner to its before-parenting state, click on the *Set Inverse* button.

Clear Inverse
: This button reverses (cancels) the effects of the above one, restoring the owner/child to its default state regarding its target/parent.

Tips

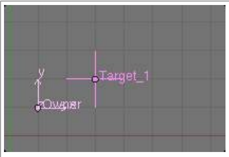
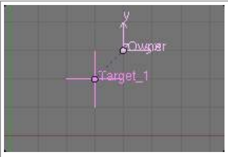
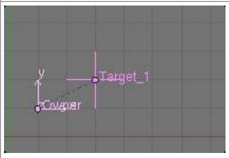
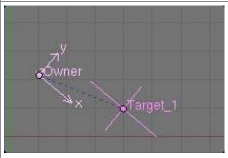
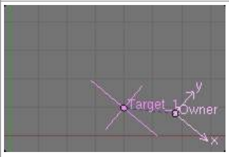
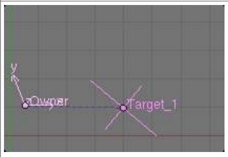
When creating a new parent relationship using this constraint, it is usually necessary to click on the *Set Inverse* button after assigning the parent. As noted above, this cancels out any unwanted transform from the parent, so that the owner returns to the location/rotation/scale it was in before the constraint was applied. Note that you should apply *Set Inverse* with all other constraints disabled (their *Influence* set to **0.0**) for a particular *Child Of* constraint, and before transforming the target/parent (see example below).

About the toggle buttons that control which target's (i.e. parent's) individual transform properties affect the owner, it is usually best to leave them all enabled, or to disable all three of the given Location, Rotation or Scale transforms.

Technical Note

If you use this constraint with all channels on, it will use a straight matrix multiplication for the parent relationship, not decomposing the parent matrix into loc/rot/size. This ensures any transformation correctly gets applied, also for combinations of rotated and non-uniform scaled parents.

Examples

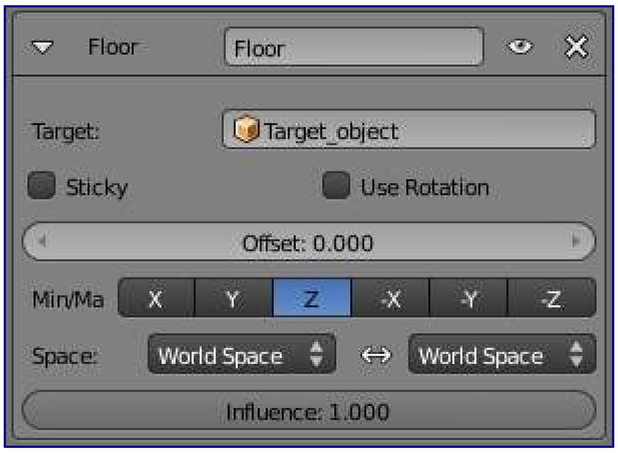
 **1. No constraint** Note the position of `Owner` empty - `1.0` BU along X and Y axes.	 **2.** ***Child Of*** **just added** Here you can see that `Owner` empty is now **1.0 BU** away from `Target_1` empty along X and Y axes.
 **3. Offset set** *Set Inverse* has been clicked, and `Owner` is back to its original position.	 **4. Target/parent transformed** `Target_1` has been translated in the XY plane, rotated around the Z axis, and scaled along its *local* X axis.
 **5. Offset cleared** *Clear Inverse* has been clicked - `Owner` is fully again controlled by `Target_1`.	 **6. Offset set again** *Set Offset* has been clicked again. As you can see, it does not gives the same result as in (*Target/parent transformed*). As noted above, use *Set Inverse* only once, before transforming your target/parent.

Floor Constraint

Description

The *Floor* constraint allows you to use its target position (and optionally rotation) to specify a plane with a "forbidden side", where the owner cannot go. This plane can have any orientation you like. In other words, it creates a floor (or a ceiling, or a wall)! Note that it is only capable of simulating entirely flat planes, even if you use the *Vertex Group* option. It cannot be used for uneven floors or walls.

Options



Floor panel

Targets

This constraint uses one target, and is not functional (red state) when it has none.

Bone
: When *Target* is an armature, a new field for a bone is displayed.

Vertex Group
: When *Target* is a mesh, a new field is display where a vertex group can be selected.

Sticky
: This button makes the owner immovable when touching the "floor" plane (it cannot slide around on the surface of the plane any more). This is fantastic for making walk and run animations!

Use Rotation
: This button forces the constraint to take the target's rotation into account. This allows you to have a "floor" plane of any orientation you like, not just the global XY, XZ and YZ ones...

Offset
: This numeric field allows you to offset the "floor" plane from the target's center, by the given number of Blender Units. Use it e.g. to account for the distance from a foot bone to the surface of the foot's mesh.

Max / Min
: This set of (mutually exclusive) buttons controls which plane will be the "floor". The buttons' names correspond indeed to the *normal* to this plane (e.g. enabling *Z* means "XY plane", etc.) By default, these normals are aligned with the *global* axes. However, if you enable *Use Rotation* (see above), they will be aligned with the *local target's axes*. As the constraint does not only define an uncrossable plane, but also a side of it which is forbidden to the owner, you can choose which side by enabling either the positive or negative normal axis... E.g, by default (*Z*), the owner is stuck in the positive Z coordinates.

Space

This constraint allows you to choose in which space to evaluate its owner’s and target’s transform properties.

Follow Path Constraint

The *Follow Path* constraint places its owner onto a *curve* target object, and makes it move along this curve (or path). It can also affect its owner’s rotation to follow the curve’s bends, when the *Follow Curve* option is enabled.

The owner is always evaluated in the global (world) space:

- Its location (as shown in the *Transform Properties* panel, `N`) is used as an offset from its normal position on the path. E.g. if you have an owner with the `(1.0, 1.0, 0.0)` location, it will be one BU away from its normal position on the curve, along the X and Y axis. Hence, if you want your owner *on* its target path, clear its location (`Alt-G`)!
- This location offset is also proportionally affected by the *scale of the target curve*. Taking the same `(1.0, 1.0, 0.0)` offset as above, if the curve has a scale of `(2.0, 1.0, 1.0)`, the owner will be offset *two* BU along the X axis (and one along the Y one)...
- When the *Curve Follow* option is enabled, its rotation is also offset to the one given by the curve (i.e. if you want the Y axis of your object to be aligned with the curve’s direction, it must be in rest, non-constrained state, aligned with the global Y axis). Here again, clearing your owner’s rotation (`Alt-R`) might be useful...

The movement of the owner along the target curve/path may be controlled in two different ways:

- The most simple is to define the number of frames of the movement, in the Path Animation panel of the Object Data context, via the numeric field Frames, and its start frame via the constraint’s Offset option (by default, start frame: 1 [= offset of 0)], duration: 100).
- The second way, much more precise and powerful, is to define a *Evaluation Time* interpolation curve for the *Target* path (in the *Graph Editor.* See the *animation chapter* to learn more about Fcurves.
- If you don’t want your owner to move along the path, you can give to the target curve a flat *Speed* FCurve (its value will control the position of the owner along the path).

Follow Path is another constraint that works well with the *Locked Track one*. One example is a flying camera on a path. To control the camera’s roll angle, you can use a *Locked Track* and a target object to specify the up direction, as the camera flies along the path.

Note

Follow Path and *Clamp To*

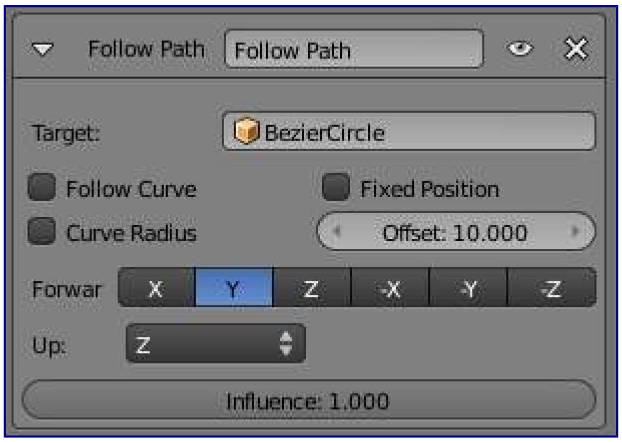
Do not confuse these two constraints. Both of them constraint the location of their owner along a curve, but *Follow Path* is an “animation-only” constraint, inasmuch that the position of the owner along the curve is determined by the time (i.e. current frame), whereas the *Clamp To constraint* determines the position of its owner along the curve using one of its location properties’ values.

Note

Note

Note that you also need to keyframe Evaluation Time for the Path. Select the path, go to the path properties, set the overall frame to the first frame of the path (e.g. frame 1), set the value of Evaluation time to the first frame of the path (e.g. 1), right click on Evaluation time, select create keyframe, set the overall frame to the last frame of the path (e.g. frame 100), set the value of Evaluation time to the last frame of the path (e.g. 100), right click on Evaluation time, select create keyframe. .. Comment: <!– from http://overshoot.tv/node/1123 paragraph needs cleanup but this definitely needs to be in the documentation –> .

Options



Follow Path panel

Target
: This constraint uses one target, which *must be a curve object*, and is not functional (red state) when it has none.

Curve Radius
: Objects scale by the curve radius. See *Curve Editing*

Fixed Position
: Object will stay locked to a single point somewhere along the length of the curve regardless of time

Offset
: The number of frames to offset from the “animation” defined by the path (by default, from frame **1**).

Follow Curve
: If this option is not activated, the owner’s rotation isn’t modified by the curve; otherwise, it’s affected depending on the following options:

    Forward
    : The axis of the object that has to be aligned with the forward direction of the path (i.e. tangent to the curve at the owner’s position).

    Up
    : The axis of the object that has to be aligned (as much as possible) with the world Z axis. In fact, with this option activated, the behavior of the owner shares some properties with the one caused by a *Locked Track constraint*, with the path as “axle”, and the world Z axis as “magnet”.

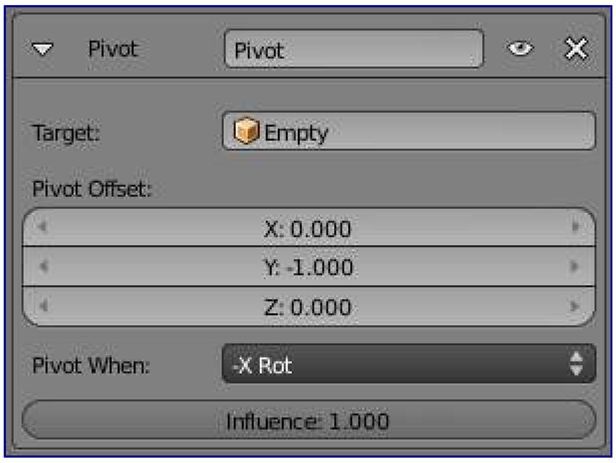
Pivot Constraint

Description

The *Pivot* constraint allows the owner to rotate around a target object.

It was originally intended for foot rigs.

Options



Target: Empty

Pivot Offset:

X: 0.000

Y: -1.000

Z: 0.000

Pivot When: -X Rot

Influence: 1.000

Pivot panel

Target

The object to be used as a pivot point

Bone
When *Target* is an armature, a new field for a bone is displayed.

Head/Tail
When using a bone target, you can choose where along this bone the target point lies.

Vertex Group
When *Target* is a mesh, a new field is display where a vertex group can be selected.

Pivot Offset
Offset of pivot from target

Pivot When
Always, Z Rot, Y Rot...

Example

https://www.youtube.com/watch?v=sncFstaycIo

Rigid Body Joint Constraint

Description

The *Rigid Body Joint* constraint is very special. Basically, it is used by the physical part of the Blender Game Engine to simulate a joint between its owner and its target. It offers four joint types: hinge type, ball-and-socket type, cone-twist, and generic six-DoF (degrees of freedom) type.

The joint point and axes are defined and fixed relative to the owner. The target moves as if it were stuck to the center point of a stick, the other end of the stick rotating around the joint/pivot point...

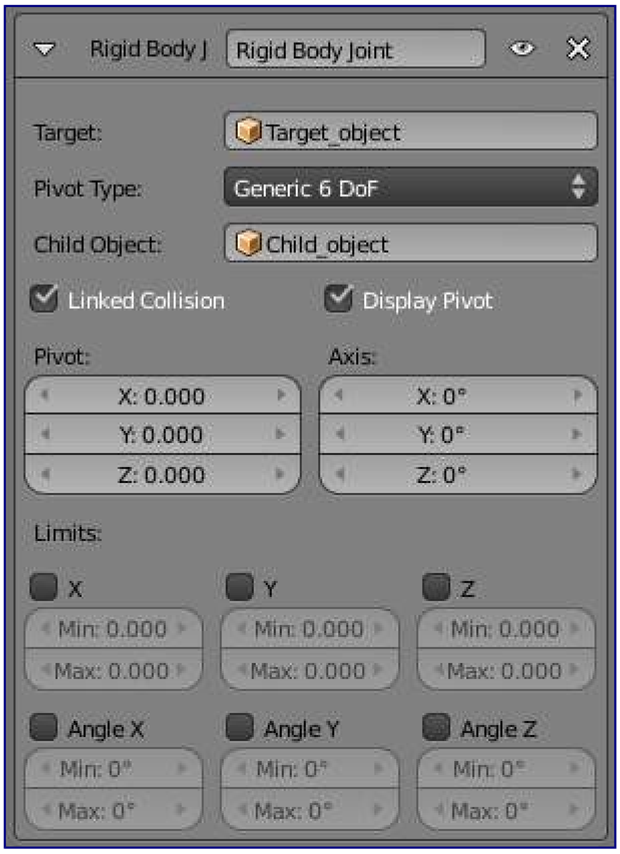
This constraint is of no use in most “standard” static or animated projects. However, you can use its results outside of the BGE, through the Game ‣ Record Animation menu entry (from the main menu of the *User Preferences* window, see *Rigid Bodies* for more info on this topic).

For a demo file that shows some of the different types, see: BGE-Physics-RigidBodyJoints.blend.

Note

In order for this constraint to work properly, both objects (so the owner and the target object) need to have “Collision Bounds” enabled.

Options


Rigid Body J: Rigid Body Joint
Target: Target_object
Pivot Type: Generic 6 DoF
Child Object: Child_object
Linked Collision ✓ Display Pivot ✓
Pivot: X: 0.000 Y: 0.000 Z: 0.000
Axis: X: 0° Y: 0° Z: 0°
Limits:
X Min: 0.000 Max: 0.000
Y Min: 0.000 Max: 0.000
Z Min: 0.000 Max: 0.000
Angle X Min: 0° Max: 0°
Angle Y Min: 0° Max: 0°
Angle Z Min: 0° Max: 0°

Rigid Body Joint panel

Target
: This constraint uses one target, and is not functional (red state) when it has none.

Joint Type:
: **Ball**

works like an ideal ball-and-socket joint, i.e. allows rotations around all axes like a shoulder joint.

Hinge

works in one plane, like an elbow: the owner and target can only rotate around the X axis of the pivot (joint point).

Limits
Angular limits for the X axis

Cone Twist

similar to *Ball*, this is a point-to-point joint with limits added for the cone and twist axis

Limits
Angular limits

Generic 6DOF

works like the *Ball* option, but the target is no longer constrained at a fixed distance from the pivot point, by default (hence the six degrees of freedom: rotation and translation around/along the three axes). In fact, there is no longer a joint by default, with this option, but it enables additional settings which allow you to restrict some of these DoF:

Limits
Linear and angular limits for a given axis (of the pivot) in Blender Units and degrees respectively.

Child Object
normally, leave this blank. You can reset it to blank by right clicking and selecting Reset to Default Value. Comment: <!– Is this right? 2.4 just had a 'to object'. Now we have a 'target' and a 'child object'. These are not documented. It seems that we recreate the behaviour of 2.4 by leaving the child object blank. The target seems to be the 2.4 'to object'. What is the child object? Please explain: m.e –> .

Linked Collision
When enabled, this will disable the collision detection between the owner and the target (in the physical engine of the BGE).

Display Pivot
When enabled, this will draw the pivot of the joint in the 3D views. Most useful, especially with the *Generic 6DOF* joint type!

Pivot
These three numeric fields allow you to relocate the pivot point, *in the owner's space*.

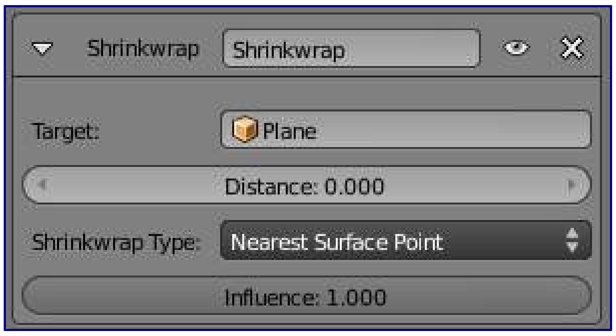
Axis
These three numeric fields allow you to rotate the pivot point, *in the owner's space*.

Shrinkwrap Constraint

The *Shrinkwrap* constraint is the "object counterpart" of the *Shrinkwrap modifier.* It moves the owner origin and therefore the owner object's location to the surface of its target.

This implies that the target *must* have a surface. In fact, the constraint is even more selective, as it can only use meshes as targets. Hence, the *Shrinkwrap* option is only shown in the *Add Constraint to Active Object* menu, `Ctrl-Alt-C`, (or its bone's equivalent), when the selected inactive object is a mesh.

Options


Shrinkwrap
Shrinkwrap
Target: Plane
Distance: 0.000
Shrinkwrap Type: Nearest Surface Point
Influence: 1.000

Shrinkwrap panel

Target

This constraint uses one target, which *must be a mesh object*, and is not functional (red state) when it has none.

Distance

This numeric field controls the offset of the owner from the shrunk computed position on the target’s surface. Positive values place the owner “outside” of the target, and negative ones, “inside” the target. This offset is applied along the straight line defined by the original (i.e. before constraint) position of the owner, and the computed one on the target’s surface.

Shrinkwrap Type

This drop-down list allows you to select which method to use to compute the point on the target’s surface to which to translate the owner’s center. You have three options:

Nearest Surface Point

The chosen target’s surface’s point will be the nearest one to the original owner’s location. This is the default and most commonly useful option.

Projection

The target’s surface’s point is determined by projecting the owner’s center along a given axis. This axis is controlled by the three *X*, *Y* and *Z* toggle buttons that show up when you select this type. This mean the projection axis can only be aligned with one of the global axes, median to both of them (XY, XZ or YZ), or to the three ones (XYZ). When the projection of the owner’s center along the selected direction does not hit the target’s surface, the owner’s location is left unchanged.

Nearest Vertex

This method is very similar to the *Nearest Surface Point* one, except that the owner’s possible shrink locations are limited to the target’s vertices.