8.8 Animation - Using Constraints in Animation

Using Constraints in Animation

Constraints are a way to control an object's properties (its location/rotation/scale), using either plain static values (like the *"limit" ones*), or (an)other object(s), called "targets" (like e.g. the *"copy" ones*).

Even though these constraints might be useful in static projects, their main usage is obviously in animation. There are two different aspects in constraints' animation:

- You can control an object's animation through the targets used by its constraints (this is a form of indirect animation).
- You can animate constraints' settings

Controlling Animation with Constraints

This applies only to constraints using target(s). Indeed, these targets can then control the constraint's owner's properties, and hence, animating the targets will indirectly animate the owner.

This indirect "constraint" animation can be very simple, like for example with the *Copy Location constraint*, where the owner object will simply copy the location of its target (with an optional constant offset). But you can also have very complex behaviors, like when using the *Action constraint*, which is a sort of *Animation Driver* for actions!

We should also mention the classical *Child Of constraint*, which creates parent/child relationship. These relationships indeed imply indirect animation (as transforming the parent affects by default all its children). But the *Child Of* constraint is also very important, as it allows you to parent your objects to bones, and hence use *Armatures* to animate them!

Back to our simple *Copy Location* example, you can have two different behaviors of this constraint:

- When its *Offset* button is disabled (the default), the location of the owner is "absolutely" controlled by the constraint's target, which means nothing (except other constraints below in the stack...) will be able to control the owner's position. Not even the object's animation curves.
- However, when the *Offset* button is enabled, the location of the owner is "relatively" controlled by the constraint's target. This means that location's properties of the owner are offset from the location of the target. And these owner's location properties can be controlled e.g. by its *Loc...* curves (or actions, or NLA...)!

Example

Let's use the *Copy Location* constraint and its *Offset* button. For example, you can make your owner (let's call it `moon`) describe perfect circles centered on the `(0.0, 0.0, 0.0)` point (using e.g. pydriven *LocX* / *LocY*

animation curves, see *this page*), and then make it copy the location of a target (called, I don't know... `earth`, for example) - with the *Offset* button enabled. Congratulation, you just modeled a satellite in a (simplified) orbit around its planet... Just do the same thing with its planet around its star (which you might call `sun`, what do you think?), and why not, for the star around its galaxy...

Here is a small animation of a "solar" system created using (among a few others) the technique described above:

https://vimeo.com/15187945

Note that the this "solar" system is not realistic at all (wrong scale, the "earth" is rotating in the wrong direction around the "sun", ...).

You can download the `.blend` file (download here) used to create this animation.

Animating Constraints Influence

More "classically", you can also animate a few properties of each constraint using animation curves.

You only have two animation curves (see also: *Graph Editor*):

- You can animate the *Influence* of a constraint. For example, in the Example above, I used it to first stick the camera to the "moon", then to the "earth", and finally to nothing, using two *Copy Location* constraints with *Offset* set, and their *Influence* cross-fading together...
- More anecdotal, you can also, for some constraints using an armature's bone as target, animate where along this bone (between root and tip) lays the real target point (`0.0` means "full-root", and `1.0`, "full-tip").