Issue 30 | Oct 2010

Blender learning made easy

blender art MAGAZINE

Once Upon an Image

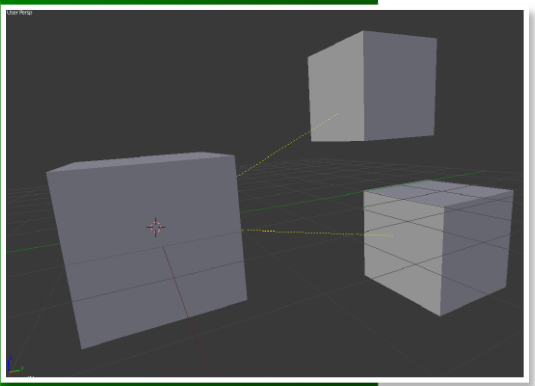
The Parent Inverse and the Origin of Children

Does Your Character Have Any Feelings

7 Elements of Digital Storytelling

The Importance Of Bodylanguage

COVERART Confinement - by reynante



By Pep Ribal

Introduction

This is not an article about where babies come from. Actually what I'm talking about is child objects, their location, rotation and scale in relation to their parent objects.

I want to write about this topic that drove me nuts when I started learning Blender. I'm talking about the obscure and terrifying Parent Inverse. The first thing that came to my mind when I read about it was my father hanging upside down. But I consider myself a good son, so I quickly wiped that image off my mind.

I found it really confusing at the beginning: it was very difficult to understand what exactly was the Parent Inverse, and why the location coordinates of the child object remained the same after aligning it to the position of its parent with Alt-O (Clear Origin).

At that time I had used other 3D applications, in which the origin of a child object was always the location of its parent, as plain as that. But in Blender things seemed to work a bit different, and I couldn't find out exactly how, even in all the documentation around. It is still difficult to find that explanation out there. Even when I posted the question in Blender forums a time long ago, all I got were replies of the type "I don't exactly know, but..." or "I'm also wondering how it works, but...". I was even asked "why on earth I wanted to know about all that stuff". So that's why I'd like to shed some light onto this obscure issue.

First, we will focus on location coordinates for this first example, so scale and rotation will be left apart for now.

Let's start with the following scenario: just one cube and a UV sphere. Let's say the cube is located at the global point (3,3,0), and the sphere at global point (5,5,0). A Top Ortho view will be very useful for this experiment.

What happens if we make the cube the parent of the sphere? RMB the sphere, then Shift-RMB the cube, press Ctrl-P, and select Object. Now the cube is the parent of the sphere. Neither of the objects have moved apparently. So, what are the coordinates of both objects now? Well, actually the same as before (you can check in the object transform properties, hotkey N). But the real question is: what is the origin of both objects now?

This is an easy question. Provided that no object moved, and there wasn't any change in location coordinates, the origin should be the same as before: the global origin (0,0,0).

Does it make sense?

Well, for the parent cube, it definitely does, as it's still a global object (it has no parent). But, shouldn't the origin of the sphere be the location of its parent? Actually it is, but with a small modification. If we now move the parent cube to (4,4,0), what happens to the sphere? It is apparently at global (6,6,0)... And its location (local) coordinates haven't changed; they still are (5,5,0), as the Properties sidebar shows. That means that its origin is now (1,1,0). Why?

by Pep Ribal

It's obvious that the sphere origin moves with the parent cube, but it's not exactly the location of the parent cube. When the cube was at (3,3,0), the child origin was at (0,0,0); now the cube is at (4,4,0) and the child origin is at (1,1,0)...

So, we can easily see that the child origin is the parent minus (3,3,0), or the parent origin plus (-3,-3,0), which will give the same result. What is this (-3,-3,0) value? Well, that is exactly the Parent Inverse...!

To speak properly, that is not exactly the Parent Inverse. So before we proceed, we need now to introduce briefly the concept of Transformation Matrices.

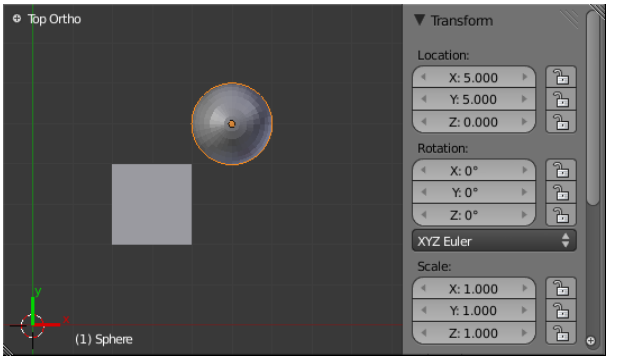
The Matrix

Even if you are not the One (and you are probably not, don't fool yourself), you deserve to know that location, rotation and scale of any object, in each of the 3 axes (X, Y and Z) are stored internally in a matrix of 4x4 numbers, called the Transformation Matrix of the object. The contents of the matrix of the active object are shown in the Transform panel of the Properties sidebar in the 3D View (hotkey N) while in Object Mode, translated to easily understandable coordinate numbers that show the transformation values of the active object in the 3 axes.

Every object has its own associated transformation matrix. To know the effective location, rotation and scale of a given object, we need two things: its transformation matrix, and its origin (the departing point of that transformation). For a global object (that has no parent), this origin is location (0,0,0), rotation (0,0,0), and scale (1,1,1). For a child object, this origin is its parent location, rotation and scale, but child objects also have an additional matrix applied: the Parent Inverse.

So we need to introduce the concept of "inverse matrix". What is an inverse transformation matrix? It's a matrix that when applied to an object takes it back to its origin. For instance, given a global object at its origin, if we apply a series of transformations on it (location, rotations and scaling), that global object ends up having a transformation matrix that shows all these transformations. The inverse of this matrix, applied to the same object undoes all of them, and the object rests again in its origin, with no rotations or scaling at all.

To make it simple, let's go back to the cube/sphere example. Let's focus on location only. At the very beginning, when the sphere wasn't related to the cube yet, the cube was at global (3,3,0). Let's simplify things, saying (though it's wrong) that the transformation matrix of the cube was "location (3,3,0)". The inverse transformation matrix, the one that would take the cube back to its origin, is naturally "location (-3,-3,0)", as (3,3,0) + (-3,-3,0) = (0,0,0), that is, matrix + inverse matrix = origin.



So, now we know what the inverse transformation matrix of the cube is. Blender is not using this inverse

matrix yet, but as soon as the sphere becomes the child of the cube, that inverse matrix (of the parent cube) is calculated and effectively applied to the child sphere. That's how we find the real origin of the sphere.

So to summarize: *at the moment of parenting, the matrix that would take the parent to its origin (the parent's inverse transformation matrix) is calculated and applied to the new child.*

As you might remember, the effective origin of the sphere was the location of the parent object, plus "location (-3,-3,0)". That is, the location of the parent plus the Parent Inverse. So why does Blender act this way? Why does it use the inverse transformation matrix of a parent object into child objects? Let's actually see what would happen if Blender didn't do so. There are 2 options:

a What if Blender parents the sphere to the cube without applying the Parent Inverse to the sphere, and without changing the transformation matrix of the sphere child? That is, without changing its location, rotation and scale values. This means that as the origin of the sphere is changing from the global origin to the parent object location, the actual sphere would change its position visibly on screen. Thus, with the position of the sphere being (5,5,0), and its new origin (3,3,0), we would automatically see the sphere jump to (3,3,0) + (5,5,0), that is, (8,8,0).

b What if Blender wants to avoid this jump at the moment of parenting, and still not apply the Parent Inverse? Then the actual location coordinates of the sphere would need to be changed. In this case, as the new origin is (3,3,0), the new location coordinates of the sphere should be changed to (2,2,0) so that it remains at global (5,5,0), as (3,3,0) + (2,2,0) = (5,5,0). But that wouldn't be too suitable, as we would be changing the object attributes (transformation matrix) for the sake of parenting, which is not justified. What would happen to such a sphere if we cleared the parenting relation with the cube, or we deleted the cube? It would jump to (2,2,0), which is bad. So this one is not an option.

That said, the only way to preserve the object attributes (its own transformation matrix) and yet avoid the jump when parenting, is to apply the Parent Inverse to the child.

So to summarize all this, the **global location of the child** is: *the location of the parent + the Parent Inverse as it was at the moment of parenting + the (local) location coordinates of the child*.

We speak of global coordinates when they are relative to the global origin; local coordinates are relative to some parent object. So the transformation properties of an object become local as soon as we parent the object, and they are global as soon as we unparent it.

So, to make it easier to understand we have focused exclusively on location, but as I mentioned earlier, the parent inverse transformation matrix (the Parent Inverse) is made up of all three types of transformations: location, rotation and scale. There is no need to go over the example again focusing on these other transformation types, as they work in a similar way. If the parent has a rotation of (10,-40,90), the Parent Inverse matrix will have a rotation of (-10,40,-90); if the parent's scale is (2,1,2), the scale of the Parent Inverse applied to the child will be (0.5,1,0.5), and so on...

Keep in mind that as soon as the parenting is done, the Parent Inverse matrix applied to the child is the inverse of the parent's transformation matrix at the very moment of parenting; that is, that matrix is never changed

by Pep Ribal

afterwards, even if we apply one thousand transformations to the parent. There is a way to change that matrix applied to the child as you will see next.

Another question to consider is that a global object, an object that has no parent, doesn't have a Parent Inverse applied, naturally. This only affects child objects.

Clear Origin (Alt-O)

In the sphere/cube example, what will happen if we clear the sphere location (**Alt-G**)? Its location coordinates will go (0,0,0), and so it will jump to its origin. If you remember, that origin is exactly its parent cube location plus the cube inverse matrix at the moment of parenting. This means that the sphere will jump to the position of the parent plus (-3,-3,0).

In Blender, with the child object active, you can press **Alt-P** and select Clear Parent Inverse. What does that do? Well, it clears the Parent Inverse... Surprised? Me too.

In this case (with location already cleared) that means that the sphere jumps to the same location of its parent. When the Parent Inverse is cleared, the origin of the child is actually the parent location, rotation and scaling, as simple as that. The Parent Inverse is ignored from then on, unless we modify it.

Clear Parent Inverse doesn't have any effect on non-child objects.

However, there is another way to make the child jump directly to the location of the parent without clearing the child location and the Parent Inverse. It's the command Clear Origin (**Alt-O**). This command only affects location (not rotation or scale). It makes the child object jump to the same global position of its parent, so that we see them placed in the same global coordinate.

And it does it without changing the child's attributes: it actually changes the values of the child's Parent Inverse matrix accordingly which will no longer have the value calculated at the time of parenting. This is the only way to change the Parent Inverse matrix of a child object in Blender (unless you use a Python script to change its values, naturally).

Unparenting (Alt-P)

We have already seen the **Clear Parent Inverse** (**Alt-P**) command. There are other uses for Alt-P. None of these have effect on global objects:

The first one is **Clear Parent**, which will cancel the parenting relationship between the selected object(s) and his (their) parents. The effect of the transformations of the parent will be discarded, and so the formerly child object will jump according to its new (global) origin, in relation to location, rotation and scale. Its local coordinates will become global but unchanged in value.

The other one is **Clear and Keep Transformation**. This one also cancels the parenting relationship, but it changes the child object attributes (location, rotation, scale) so that the transformations of the parent are applied to the child and when the child becomes a global object, no apparent change is seen on screen. In other words, it translates its local coordinates to the corresponding global ones.

Parenting methods

Besides the usual **Ctrl-P** parenting command, there is another one: Make Parent without inverse (**Shift-Ctrl-P**). This command is equivalent to making a usual parenting (**Ctrl-P**), then clearing the Parent Inverse (**Alt-P**), and finally clearing the child's location

by Pep Ribal

(just the location, with **Alt-G**). The child jumps into the same global point where its parent is, without Parent Inverse matrix applied, and with its transform properties intact, except for its location coordinates which become (0,0,0).

And that's pretty much all, folks. I hope to have made this topic a bit more clear, as it was a difficult thing to grasp, in my case. But it's known that things between parents and children are always very difficult ●

Be good!

Pep Ribal works as an IT Engineer. However he is very interested in the audiovisual and multimedia world, and he has worked in a few TV productions and short films as a director, actor, screenplay writer and video editor. He has made a few small 3D works for TV using Blender.