
Subject: Re: adding a 4th dimension to 3D array during concatenation

Posted by [Jim Pendleton](#) on Thu, 17 Mar 2016 01:46:25 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Wednesday, March 16, 2016 at 9:00:35 AM UTC-6, Paul van Delst wrote:

> Hello,

>

> On 03/15/16 18:37, Wayana Dolan wrote:

>> So outside a loop I start out with an empty array (ex. array=[]).

>> Then each time through the loop, I make an array with 3 dimensions

>> (for example, array x has dimensions[91, 41, 33]), and then

>> concatenate it to the previous array. (ex. array=[array, x]).

>>

>> Lets say we run through the loop 16 times. What I'd like as a result

>> is something that has dimensions like this [16, 91, 41, 33].

>>

>> I'm not sure how to do this... I've looked at IDL coyote's

>> concatenation tutorial, and still am having trouble.

>>

>> I'm pretty new to coding period, so this is a challenge. Any ideas?

>>

>

> It depends on what you want to do with your monster array after

> concatenation, and will each array be the same shape? (Even if the

> answer right now is yes, will they always be?)

>

> Concatenation is a slow operation in IDL, and I have always found

> multi-dimensional concatenation similar to dealing with regular

> expression - counting all the [[[s and]]]'s to make sure they match

> up, etc. This is not a fault with IDL, IMO it's just that arrays,

> really, are not meant to have those sorts of things done to them.

>

> So, why use an array?

>

> Why not, say, a list?

>

> IDL> array=list()

> IDL> help, array

> ARRAY LIST <ID=1 NELEMENTS=0>

> IDL> x=findgen(91,43,33)

> IDL> array.Add, x

> IDL> x=findgen(14,17,36)

> IDL> array.Add, x

> IDL> help, array

> ARRAY LIST <ID=1 NELEMENTS=2>

> IDL> help, array[0]

> <Expression> FLOAT = Array[91, 43, 33]

> IDL> help, array[1]

> <Expression> FLOAT = Array[14, 17, 36]
>
> Or a hash? Works similarly.
>
> Lists and hashes are data constructs that are designed to be added to
> and extended. Arrays, not so much.
>
> Anyhoo...
>
> cheers,
>
> paulv

To add to Paul's excellent suggestion about lists and the perils of array concatenation, I'll add the following. If you actually need an array at the end of your accumulation, you can use the List.ToArray() method. Use the /NO_COPY keyword to get extra points for efficiency.

```
IDL> a = fltarr(91, 41, 33)
IDL> b = a
IDL> c = list()
IDL> c.add, a, /NO_COPY
IDL> c.add, b, /NO_COPY
IDL> d = c.toArray(/NO_COPY)
IDL> help, d
D          FLOAT    = Array[2, 91, 41, 33]
```

Jim P.
