Subject: Array Subscripting Puzzle
Posted by David Fanning on Fri, 17 May 2002 17:53:08 GMT
View Forum Message <> Reply to Message

Folks,

I have a 24-bit image. You can interleave it anyway
you like that will make the problem described below
trackable. At the moment it is 800 by 600 by 3.

I have the indices of something I want to draw on
the image. Say they are the indices of the outlines
of some continents. For example, like this:

```
   window, xsize=800, ysize=600
   map_set, /Cylindrical, position=[0,0,1,1]
   map_continents, /fill
   a = tvrd()
   indices = where(a GT 0)
```

I want to make all the outline pixels yellow.
I *could* do this:

```
   r = Reform((image[*,*,0]))
   g = Reform((image[*,*,1]))
   b = Reform((image[*,*,2]))
   r[indices] = 255
   g[indices] = 255
   b[indices] = 0
   image[*,*,0] = r
   image[*,*,1] = g
   image[*,*,2] = b
```

That seems wasteful and inelegant. There must be
a way to do this in one go. I'm sure it uses REBIN
and REFORM, but I'm not sure in which order. :-(

Can anyone help?

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Array Subscripting Puzzle
Posted by JD Smith on Wed, 22 May 2002 21:00:16 GMT
View Forum Message <> Reply to Message

On Fri, 17 May 2002 10:53:08 -0700, David Fanning wrote:

```
> Folks,
>
> I have a 24-bit image. You can interleave it anyway you like that will
> make the problem described below trackable. At the moment it is 800 by
> 600 by 3.
>
> I have the indices of something I want to draw on the image. Say they
> are the indices of the outlines of some continents. For example, like
> this:
>
>   window, xsize=800, ysize=600
>   map_set, /Cylindrical, position=[0,0,1,1] map_continents, /fill a =
>   tvrd()
>   indices = where(a GT 0)
>
> I want to make all the outline pixels yellow. I *could* do this:
>
>   r = Reform((image[*,*,0]))
>   g = Reform((image[*,*,1]))
>   b = Reform((image[*,*,2]))
>   r[indices] = 255
>   g[indices] = 255
>   b[indices] = 0
>   image[*,*,0] = r
>   image[*,*,1] = g
>   image[*,*,2] = b
>
> That seems wasteful and inelegant. There must be a way to do this in one
> go. I'm sure it uses REBIN and REFORM, but I'm not sure in which order.
> :-(
>
> Can anyone help?
```

An excellent exercise for the reader of the rebin/reform tutorial ;)

Here's what I used:

```
inds=where(a GT 0,n)  & s=size(a,/DIMENSIONS)
```

```
image[rebin(inds,n,3)+rebin(1#(s[0]*s[1]*lindgen(3)),n,3)]= $
   rebin(1#[255,255,0],n,3)
```

As you can see, I employed the:

```
1#x = transpose(x) = reform(x,1,n_elements(x))
```

shortcut described in the tutorial.

So what's it doing? On the LHS, inside the brackets, we're generating an nx3 list of the matched indices "drilled-down" through the 3 image color planes (where n is the number of image mask pixels which matched). The "(s[0]*s[1]*lindgen(3))" is just the index offset of each plane to add -- indices into the 3d array are simple to compute. To this nx3 list on the LHS, we assign a special nx3 list, created by filling the columns with the 3-element vector [255,255,0].

Now, you might ask yourself, why did I choose an nx3 format for the index and assignment arrays? Mostly for the convenience of a short RHS, but you could do it however you like: the only thing to remember is that the indices on the LHS, and values on the RHS need to match up one to one (so that, e.g., pixel 25,25 in plane 1,2, & 3 line up with RHS values 255,255,0, respectively). Once you have this basic mapping in mind, it doesn't matter at all how you format the interim product, as long as it's the same on each side of the assignment. Note the format doesn't even need to correspond to the initial array layout at all; use any convenient mapping.

What if you had 3x800x600 interleaving? It's a simple change:

```
image[rebin(1#(3*(inds mod s[0]+inds/s[0]*s[0])),3,n)+ $
   rebin(indgen(3),3,n)]= rebin([255,255,0],3,n)
```

Here, computing the index of a given image pixel into the 3xNXM array is more difficult (since the image is "on it's side"), but threading over the "short" colors dimension is easier (just add 0, 1 or 2!). Notice that I used the natural 3xn intermediary index format for assignment.

So, to sum up, the usual procedure is:

1. Figure out the indices of the values in the array you want to change.
2. Figure out any grouping of contiguous indices inside the array you're changing. This, along with the overall array dimensions, usually dictates an appropriate intermediary index format.
3. Generate the indices of #1, using rebin and reform to manipulate them into the chosen intermediary format.
4. Manipulate the assigning data into the intermediary format, and assign.

Sometimes this is an iterative procedure. I find the new "Examine" commands in IDLWAVE indispensable for this process: I can check the dimensions or values for any sub-expressions while building the overall assignment.

Good luck,

JD

Subject: Re: Array Subscripting Puzzle
Posted by David Fanning on Wed, 22 May 2002 21:57:53 GMT
View Forum Message <> Reply to Message

JD Smith (jdsmith@as.arizona.edu) writes:

> An excellent exercise for the reader of the rebin/reform tutorial ;)

Right, I'll add this.

And the users should submit their exams to...oh, look,
JD has a new e-mail address! :-)

Cheers,

David

P.S. Let's just say I'm gratified to see so many
people reading (and understanding) the REBIN/REFORM
tutorial. I'm going to move my name off the "Read It 50
Times" list and add it to the "Finally Got It" list
real soon now. :-)

--
David W. Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Array Subscripting Puzzle
Posted by Paul Van Delst[1] on Thu, 23 May 2002 13:07:40 GMT
View Forum Message <> Reply to Message

JD Smith wrote:

```
>
> On Fri, 17 May 2002 10:53:08 -0700, David Fanning wrote:
>
>> Folks,
>>
>> I have a 24-bit image. You can interleave it anyway you like that will
>> make the problem described below trackable. At the moment it is 800 by
>> 600 by 3.
>>
>> I have the indices of something I want to draw on the image. Say they
>> are the indices of the outlines of some continents. For example, like
>> this:
>>
>>    window, xsize=800, ysize=600
>>    map_set, /Cylindrical, position=[0,0,1,1] map_continents, /fill a =
>>    tvrd()
>>    indices = where(a GT 0)
>>
>> I want to make all the outline pixels yellow. I *could* do this:
>>
>>    r = Reform((image[*,*,0]))
>>    g = Reform((image[*,*,1]))
>>    b = Reform((image[*,*,2]))
>>    r[indices] = 255
>>    g[indices] = 255
>>    b[indices] = 0
>>    image[*,*,0] = r
>>    image[*,*,1] = g
>>    image[*,*,2] = b
>>
>> That seems wasteful and inelegant. There must be a way to do this in one
>> go. I'm sure it uses REBIN and REFORM, but I'm not sure in which order.
>> :-(
>>
>> Can anyone help?
>
> An excellent exercise for the reader of the rebin/reform tutorial ;)
>
> Here's what I used:
>
> inds=where(a GT 0,n)  & s=size(a,/DIMENSIONS)
> image[rebin(inds,n,3)+rebin(1#(s[0]*s[1]*lindgen(3)),n,3)]= $
>    rebin(1#[255,255,0],n,3)
```

Holy cow!

For my (relatively pokey) PC:

DF's method execution time: 0.65045297seconds
JD's method execution time: 0.28429806seconds

For my (_extremely_ pokey) sultana:

DF's method understanding time: oh, about 5-10seconds or so
JD's method understanding time: uh... still working on it.

And now, for some more syntactical gymnastics, I'm off to comp.lang.c..... :o)

paulv

--
Paul van Delst Religious and cultural
CIMSS @ NOAA/NCEP/EMC purity is a fundamentalist
Ph: (301)763-8000 x7274 fantasy
Fax:(301)763-8545 V.S.Naipaul