Subject: Re: Please help me avoid loops and conditionals
Posted by Chris Lee on Fri, 19 Sep 2003 09:40:46 GMT
View Forum Message <> Reply to Message

In article <c857619b.0309131258.541e73de@posting.google.com>, "Patrick Ford" <pford@bcm.tmc.edu> wrote:

> "Chris Lee" <cl@127.0.0.1> wrote in message
>> <big snip>
...

> First, thank you to all who responded. Chris, I did not get your email.

Damn :) if you really want to make the function as fast as possible and know that you will resuse the mask, pass the mask into the function as an optional keyword, then check for the value of the keyword, if the masks are available, resuse them, this gets you a huge increase in speed but you need to be careful when changing the values of the ellipse (i.e the mask).

> I fiddled around with this trying to do this and I came up with the
> following:
> pro test
> a_d = 6
> b_d = 4
> a = 15
> b = 21
> box_dim = 64
> box = intarr(box_dim,box_dim)
> bxd2 = box_dim*2
> view = intarr(bxd2,bxd2)
>
> x = float((indgen(box_dim) - box_dim/2) # replicate(1, box_dim)) x1 =
> x
> y = (transpose(x) / b)^2
> x = (x / a)^2
>
> y1 = (transpose(x1) / (b+b_d))^2
> x1 = (x1 / (a_d+a))^2
>
> r0 = x + y
> r1 = x1 + y1
>
> mask0 = where(r0 LE 1.0)
> mask1 = where(r1 LE 1.0)
> mask2 = where((r1 LE 1.0) AND NOT(r0 LE 1.0))

>
> box[mask1] = 255
> box[mask0] = 128
> view[0:box_dim-1, 0:box_dim-1] = box
>
> box[mask2] = 255
> box[mask0] = 128
> view[box_dim:bxd2-1, box_dim:bxd2-1] = box tv, view
>
> end
> Notice no loops or if statements.(yea!) Now the next question is it
> more efficient to use the mask1 method where I double assign or mask2
> where I expand a logical expression?

My guess would be that they are both the same, when I tried a similar
thing I had

mask2= where(r0[mask1] gt 1.0)
or something , you might not want to use that without checking
it... also, count your where results to make sure you have valid masks if
you do that. (I think the mask1 and mask2 would need swapping).

This only produced a 0.5 second improvement in 30 seconds, but gets
better as the outer mask gets smaller and the array size gets larger (you
only look at the previously masked data with this)

> Any significant time improvement in changing NOT(r0 LE 1.0) to (r0 GT
> 1.0)?

I doubt it, if there is an improvement it would be small, my assembly knowledge
is 'decaying' but I think there is a GT and LT in assembly so you may get a
2x speedup for this line only, but's it's O(n) so why worry :) Unrolling
that 3 layer FOR loop you wrote last year will save you much more time :)

> I was trying to eliminate the values in mask1 that corresponded to
> mask0, but at this point I don't see how to do that efficiently. My
> attempts using the where command were a disaster

The following pseudo-code snippet (from elp4) is how I did the mask
within a mask thing

w1=where(outer mask le 1.0,c1)
 if(c1 gt 0) then w2=where((inner mask of subset)[w1] le 1.0,c2)

 if(c1 gt 0) then box[w1]=v
 if(c2 gt 0) then box[w1[w2]]=o_val
                              ^^^^^^^^
                             the 'magic'

If I understand what you wrote, you want to remove anything inside the outer mask if it's inside the inner mask, David Fanning and others have written functions which can tell you which values are in one array and not another. Using the where command as you did for mask2 should work (it does, right?) but you are repeating the same logical expression twice (not exactly a bottleneck though), have a look on David's website or search the newsgroup for the "A and not B" thing when both are arrays.

You might get a speed up from that, but I'm not convinced of it, if your doing some complicated 3d polygon drawing, then yes, but if you plotting/contouring, IDL probably doesn't care if you put a 255 in a box or a 0.
