Subject: Recursive Function Program in IDL
Posted by David Fanning on Wed, 14 Dec 2005 16:02:17 GMT
View Forum Message <> Reply to Message

Folks,

Does anyone have a handy recursive function that does something neat?
Someone is asking, and I don't have time to work on this. He
(apparently) can't get to the newsgroup.

Thanks,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Recursive Function Program in IDL
Posted by b_gom on Wed, 14 Dec 2005 19:43:35 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Does anyone have a handy recursive function that does something neat?
> Someone is asking, and I don't have time to work on this.

Here is a recursive algorithm for simplifying polyline vertices. See
'poly_simplify.pro' in the IDL user contributed library.
http://www.rsinc.com/codebank/search.asp?FID=307

Brad

```
;*************************************
pro simplifyDP,tol,vertices,j,k,mk
; This is the Douglas-Peucker recursive simplification routine
; It just marks vertices that are part of the simplified polyline
; for approximating the polyline subchain vertices[j] to vertices[k].
; Input: tol = approximation tolerance
; vertices[] = polyline array of vertex points
; j,k = indices for the subchain vertices[j] to vertices[k]
; Output: mk[] = array of markers matching vertex array vertices[]

 if (k le j+1) then return ; there is nothing to simplify

 ; check for adequate approximation by segment S from vertices[j] to
vertices[k]
```

```
 maxi = j ; index of vertex farthest from S
 maxd2 = 0. ; distance squared of farthest vertex
 S = [[vertices[*,j]], [vertices[*,k]]]  ; segment from vertices[j] to
vertices[k]
 u = S[*,1]-S[*,0] ; segment direction vector
 cu = ps_dot(u,u); segment length squared

 ;test each vertex vertices[i] for max distance from S
 ;compute using the Feb 2001 Algorithm's dist_Point_to_Segment()
 ;Note: this works in any dimension (2D, 3D, ...)

 ;Pb = base of perpendicular from vertices[i] to S
 ;dv2 = distance vertices[i] to S squared

 for i=j+1,k-1 do begin
  ;compute distance squared
  w = vertices[*,i] - S[*,0]
  cw = ps_dot(w,u)
  if cw le 0 then begin
   dv2 = ps_d2(vertices[*,i], S[*,0]);
   endif else begin
   if cu le cw then begin
    dv2 = ps_d2(vertices[*,i], S[*,1])
    endif else begin
    b = cw / cu;
    Pb = S[*,0] + b * u;
    dv2 = ps_d2(vertices[*,i], Pb);
    endelse
   endelse
  ;test with current max distance squared
  if dv2 le maxd2 then continue
  ;vertices[i] is a new max vertex
  maxi = i
  maxd2 = dv2
  endfor

 if (maxd2 gt tol^2) then begin ;// error is worse than the tolerance
;  split the polyline at the farthest vertex from S
  mk[maxi] = 1 ; mark vertices[maxi] for the simplified polyline
;  recursively simplify the two subpolylines at vertices[*,maxi]
  simplifyDP, tol, vertices, j, maxi, mk ; // polyline vertices[j] to
vertices[maxi]
  simplifyDP, tol, vertices, maxi, k, mk ; // polyline vertices[maxi]
to vertices[k]
  endif
;  else the approximation is OK, so ignore intermediate vertices
 return
end
```

Subject: Re: Recursive Function Program in IDL
Posted by JD Smith on Wed, 14 Dec 2005 21:06:08 GMT
View Forum Message <> Reply to Message

On Wed, 14 Dec 2005 09:02:17 -0700, David Fanning wrote:

> Folks,
>
> Does anyone have a handy recursive function that does something neat?
> Someone is asking, and I don't have time to work on this. He (apparently)
> can't get to the newsgroup.

How about one to get all the widget ID's on a given widget tree:

```
;; decend heirarchy of tlb .. return entire tree's widget_ids in 'list'
pro treedesc, curin,list
  cur=curin                  ;ensure local copy of current widget
  if cur ne 0 then begin
    if n_elements(list) eq 0 then list=cur else list=[list,cur]
    cur=widget_info(cur,/CHILD) ;descend one level
    while cur ne 0 do begin ; find siblings... descend their subtrees
      treedesc,cur,list      ; recurs on sibling
      cur=widget_info(cur,/SIBLING)
    endwhile
  endif
end
```

I think Mark rewrote this and stuck it in his library too.

JD

Subject: Re: Recursive Function Program in IDL
Posted by peter.albert@gmx.de on Thu, 15 Dec 2005 07:53:42 GMT
View Forum Message <> Reply to Message

Here is my recursive treasure: interpolate_n, extending IDL's
INTERPOLATE routine to up to 8 dimensions. I have to admit that is has
been years since I wrote it and I am not completely sure any more how
the routine actually works, but it still seems to give the right
results ... :-) The recursive part is about getting the neighbouring
values for each dimension, I guess.

http://wew.met.fu-berlin.de/idl/interpolate_n.pro

Cheers,

Peter

Subject: Re: Recursive Function Program in IDL
Posted by David Fanning on Thu, 15 Dec 2005 08:36:20 GMT
View Forum Message <> Reply to Message

Peter Albert writes:

> Here is my recursive treasure: interpolate_n, extending IDL's
> INTERPOLATE routine to up to 8 dimensions. I have to admit that is has
> been years since I wrote it and I am not completely sure any more how
> the routine actually works, but it still seems to give the right
> results ... :-) The recursive part is about getting the neighbouring
> values for each dimension, I guess.
>
> http://wew.met.fu-berlin.de/idl/interpolate_n.pro

Uh, sorry. I wasn't clear. I was looking for simple examples
I could understand. :-)

Cheers,

David

P.S. It does qualify as "neat", though. I always love programs
(I have a few myself) that you can never understand after you
write them. They are proof you are a decent programmer. Or,
I guess, used to be. :-)

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Recursive Function Program in IDL
Posted by peter.albert@gmx.de on Thu, 15 Dec 2005 09:06:47 GMT
View Forum Message <> Reply to Message

Oh, well, it is just a good reason for "crossing their eyes
and sticking their tongues out the corner of their mouths" when
reading, isn't it ? :-)

Ah, uhm, well, now I see, you wanted examples for *preventing* this,
didn't you? :-)

Cheers,

Peter
