Subject: Re: connectivity array
Posted by Rick Towler on Wed, 11 Feb 2004 19:06:11 GMT
View Forum Message <> Reply to Message

"Amar Nayegandhi" wrote in message...

> I need to define a connectivity array for a regular grid so that I can
> use the mesh_surfacearea routine. I created the regular grid using
> triangulate/trigrid. Is there any function out there that creates the
> connectivity array from a regular grid?
> Thanks,
> -Amar

I would think that IDLgrSurface would do this... hummm. Forget that. It doesn't expose its polygon connectivity property. It should (at least as read-only).

Plan B, meshing a regularly gridded surface is pretty easy. The pro below accepts a 2xn array (depth_data) and returns the vertex (data) and polygon connectivity.

The 2xn surface is meshed as a function of the array indices but you can control the location of vertices using the LOCATION keyword by specifying [startX, startY, widthX, widthY]. A little awkward maybe but it works for my application.

NTRIANGLES returns the number of tris in the mesh.

Set the USE_TRIANGLES keyword to mesh using tri strips. If unset it will mesh using quad strips. You should try both and compare your results since I don't know if it will make a difference in your case.

-Rick

```
pro MakeB3dFile_MeshSurface,    depth_data, $
                        data, $
                        polygons, $
                        location=location, $
                        nTriangles=nTriangles, $
                        use_triangles=useTriangles

   compile_opt idl2

   dims = SIZE(depth_data)

   useTriangles = (N_ELEMENTS(useTriangles) eq 0) ? 0 : $
      KEYWORD_SET(useTriangles)
```

```
if (N_ELEMENTS(location) ne 4) then location=[0,0,dims[1],dims[2]]

;convert 2xN data to 3xN
data = FLTARR(3,dims[4], /nozero)
for n = 0L, dims[2] - 1L do begin
   s = n * dims[1]
   e = s + dims[1] - 1L
   data[0,s:e] = FINDGEN(dims[1]) * ((location[2]-location[0]) / $
      (dims[1] - 1)) + location[0]
   data[1,s:e] = depth_data[*,n]
   data[2,s:e] = n * ((location[3]-location[1]) / $
      (dims[2] - 1)) + location[1]
endfor

depth_data = 0B

;create the surface mesh
if (useTriangles) then begin
   ;create triangle strip mesh
   nconn = (dims[2] - 1L) * (8L * (dims[1] - 1L))
   nstrip = (8L * (dims[1] - 1L))
   polygons = FLTARR(nconn, /nozero)
   v1 = 0L
   v2 = LONG(dims[1])
   np = 0L
   while (np lt nconn-1) do begin
      for ns = 0L, nstrip-1, 8L do begin
         n = ns + np
         polygons[n:n+3L] = [3, v1, v1+1L, v2]
         polygons[n+4L:n+7L] = [3, v2, v1+1L, v2+1L]
         v1 = v1 + 1L
         v2 = v2 + 1L
      endfor
      np = np + nstrip
      if (np lt nconn-1) then begin
         ;mesh back row
         v1 = v1 + (2L * dims[1])
         for ns = 0L, nstrip-1, 8L do begin
            n = ns + np
            polygons[n:n+3L] = [3, v2-1L, v2, v1]
            polygons[n+4L:n+7L] = [3, v2-1L, v1, v1-1L]
            v1 = v1 - 1L
            v2 = v2 - 1L
         endfor
         v2 = v2 + (2L * dims[1])
         np = np + nstrip
      endif
   endwhile
```

```
    endif else begin
      ;  Create quad strip mesh
      nconn = (dims[2] - 1) * (5 * (dims[1] -1))
      nstrip = (5L * (dims[1] - 1L))
      polygons = FLTARR(nconn, /nozero)
      v1 = 0L
      v2 = LONG(dims[1])
      np = 0L
      while (np lt nconn-1) do begin
        for ns = 0L, nstrip-1, 5L do begin
          n = ns + np
          polygons[n:n+4L] = [4, v1, v1+1L, v2+1, v2]
          v1 = v1 + 1L
          v2 = v2 + 1L
        endfor
        np = np + nstrip
        if (np lt nconn-1) then begin
          ;  Mesh back row
          v1 = v1 + (2L * dims[1])
          for ns = 0L, nstrip-1, 5L do begin
            n = ns + np
            polygons[n:n+4L] = [4, v2-1L, v2, v1, v1-1L]
            v1 = v1 - 1L
            v2 = v2 - 1L
          endfor
          v2 = v2 + (2L * dims[1])
          np = np + nstrip
        endif
      endwhile
    endelse

    nTriangles = MESH_NUMTRIANGLES(polygons)

end
```