Subject: about DXF format
Posted by airy.jiang on Tue, 12 Jun 2007 10:36:53 GMT
View Forum Message <> Reply to Message

Hi,everybody.
Recently,i'm troubled in with loading DXF files into my program.
When I loading some simple DXF files ,it can be displayed very well.I
used IDLffDXF object,and I just checked whether the file has the
IDL_DXF_POLYGON entity acoording the GetContents and GetEntity
methods.Then load the vertex data and connectivities into new
IDLgrPolygon objects.But once I load some complex DXF files,the
trouble has coming:first,the position of some IDLgrPolygons are not
precision.Second,we know,sometimes,it just need one IDLgrPolygon
object to display a polygon which be composed by a lot of
triangles.But when I loading some DXF files, the GetEntity method
shows a very bad result.It produed many many IDLgrPolygons,so much as
decomposed some polygon entitys.That makes my program became very
slow ,and need to wait a long time for it .How could I promote my
loading speed?Is there any better way to avoid making too many
IDLgrPolygons through GetEntity method?
I don't know whether I made my question clear,please parden me for my
poor english^_*,I'll pracitse more.
Thanks.

Subject: Re: about DXF format
Posted by Rick Towler on Tue, 12 Jun 2007 23:24:28 GMT
View Forum Message <> Reply to Message

> Second,we know,sometimes,it just need one IDLgrPolygon
> object to display a polygon which be composed by a lot of
> triangles.
> But when I loading some DXF files, the GetEntity method
> shows a very bad result.It produed many many IDLgrPolygons,so much as
> decomposed some polygon entitys.That makes my program became very
> slow ,and need to wait a long time for it .

I could be wrong, I have never actually looked, but I thought that the
entities are defined in the DXF file. So there isn't a problem with
GetEntity. Your issue is that your DXF files are more complicated than
you would like.

> Is there any better way to avoid making too many
> IDLgrPolygons through GetEntity method?

Ideally you fix this at the source, whatever program you are exporting
your DXF models from. Alternatively you could combine entities into a
single IDLgrPolygon by concatenating the vertex arrays and applying the

proper offsets to the polygon arrays. It's hard to say how much this
would speed things up since it depends on where your bottleneck lies.
If your program is burdened by the overly complex graphics hierarchy
(too many IDLgrPolygon objects) this will help. But if you simply have
too many polygons on your screen this will do little in terms of
improving performance.

What hardware and OS are you running IDL on? How many polygons are you
trying to display?

-Rick

Subject: Re: about DXF format
Posted by Andrew Cool on Tue, 12 Jun 2007 23:45:18 GMT

View Forum Message <> Reply to Message

On Jun 12, 7:36 pm, airy.ji...@gmail.com wrote:
> Hi,everybody.
> Recently,i'm troubled in with loading DXF files into my program.
> When I loading some simple DXF files ,it can be displayed very well.I
> used IDLffDXF object,and I just checked whether the file has the
> IDL_DXF_POLYGON entity acoording the GetContents and GetEntity
> methods.Then load the vertex data and connectivities into new
> IDLgrPolygon objects.But once I load some complex DXF files,the
> trouble has coming:first,the position of some IDLgrPolygons are not
> precision.Second,we know,sometimes,it just need one IDLgrPolygon
> object to display a polygon which be composed by a lot of
> triangles.But when I loading some DXF files, the GetEntity method
> shows a very bad result.It produed many many IDLgrPolygons,so much as
> decomposed some polygon entitys.That makes my program became very
> slow ,and need to wait a long time for it .How could I promote my
> loading speed?Is there any better way to avoid making too many
> IDLgrPolygons through GetEntity method?
> I don't know whether I made my question clear,please parden me for my
> poor english^_*,I'll pracitse more.
> Thanks.

How about trying to load your DXF file in REVOLUTION IDL, to see
whether the
expert's program can handle your file?

Download REVOLUTION IDL from : -

http://www.ittvis.com/codebank/search.asp?FID=473

Cheers,

Andrew

Subject: Re: about DXF format
Posted by airy.jiang on Thu, 14 Jun 2007 01:51:49 GMT
View Forum Message <> Reply to Message

Rick,Maybe you are right.
The DXF format is indeed complicated,I knew it and I have read some
documents about that.The key is a DXF file is not only has "Entity"
section,but also include many other sections,like "Table"
section,"Object" section,"Class" setction and so on.The IDLffDXF
object can get a lot information from one DXF file by the GetEntity
method.Just like I've wrote,we can get the most basic vertex data and
vertex connectivities,that is enough information to make a full
polygon.But that's all we can do until now.The truth is,I use
GetContent method get the entity type of the file,the result is
[9,18,19,20],that represents the types of the entities which including
in the
file :IDL_DXF_POLYGON,IDL_DXF_INSERT,IDL_DXF_BLOCK,IDL_DXF_LAYER. The
vertex data and connectivity can be found in the IDL_DXF_POLYGON ,and
we can use them to make a polygon.But what the use of the other types
of entity?IDL_DXF_INSERT,IDL_DXF_BLOCK,IDL_DXF_LAYER,Are they useless?
I use 3DMAX and AutoCAD read the same DXF file,they all showed a exact
result.My program which used IDLffDXF object has two errors.
First,the shape of the polygon are all right,but the polygon are all
composed of the triangles.The 3DMAX and AutoCAD shows that the right
result is not only the triangles,many polygons are composed of the
rectangles.And the number of polygons of the DXF file which made by my
program are more than the number which made by the 3DMAX and AutoCAD.
Second,the position of some parts of polygons are not precision.I
think that's maybe some offset information contained in the DXF
file,but I'm not very sure.

My computer is good,that' not the problem.The number of polygons is
nearly 10000,but infact the right result showed by the 3DMAX and
AutoCAD tell us that's not actual number,it just needs 1000 or more
less polygons to show that.Anyway,let me list my hardware:
cpu:AMD 3600+
memory: 1G DDR2 667
Graphic Card: Unika 1650XT (600MHz/1200)
HardDisk:Hitachi 160G SATA2 8M buffer memory

Subject: Re: about DXF format

Posted by airy.jiang on Thu, 14 Jun 2007 02:07:53 GMT
View Forum Message <> Reply to Message

On 6 13 ,   7 45 , Andrew Cool <andrew.c...@dsto.defence.gov.au>
wrote:
> On Jun 12, 7:36 pm, airy.ji...@gmail.com wrote:
>
>
>
>
>
>> Hi,everybody.
>> Recently,i'm troubled in with loading DXF files into my program.
>> When I loading some simple DXF files ,it can be displayed very well.I
>> used IDLffDXF object,and I just checked whether the file has the
>> IDL_DXF_POLYGON entity acoording the GetContents and GetEntity
>> methods.Then load the vertex data and connectivities into new
>> IDLgrPolygon objects.But once I load some complex DXF files,the
>> trouble has coming:first,the position of some IDLgrPolygons are not
>> precision.Second,we know,sometimes,it just need one IDLgrPolygon
>> object to display a polygon which be composed by a lot of
>> triangles.But when I loading some DXF files, the GetEntity method
>> shows a very bad result.It produed many many IDLgrPolygons,so much as
>> decomposed some polygon entitys.That makes my program became very
>> slow ,and need to wait a long time for it .How could I promote my
>> loading speed?Is there any better way to avoid making too many
>> IDLgrPolygons through GetEntity method?
>> I don't know whether I made my question clear,please parden me for my
>> poor english^_*,I'll pracitse more.
>> Thanks.
>
> How about trying to load your DXF file in REVOLUTION IDL, to see
> whether the
> expert's program can handle your file?
>
> Download REVOLUTION IDL from : -
>
> http://www.ittvis.com/codebank/search.asp?FID=473
>
> Cheers,
>
> Andrew-   -
>
> -

Thanks for your suggestion,Andrew Cool.
I've used that program read that DXF file,the result is expert's
program has made the same mistakes.And it made whole a file into one
polygon,that's not the answer I want.The different entities should be

detached.One entity correspond one polygon,that is the right
result,and that relationship should be wrote in the DXF file.We have
no need to do it by ourselves,we just need to read it from the
file,that's all.

Hope more people could join here to discuss this subject.Specailly
about IDL_DXF_INSERT,IDL_DXF_BLOCK,IDL_DXF_LAYER these structures,is
there anybody know their's use?

thanks.

Subject: Re: about DXF format
Posted by airy.jiang on Sun, 17 Jun 2007 02:20:52 GMT
View Forum Message <> Reply to Message

no more people would like to discuss this topic?what a pity!

Subject: Re: about DXF format
Posted by JMZawodny on Mon, 18 Jun 2007 12:23:25 GMT
View Forum Message <> Reply to Message

On Jun 16, 10:20 pm, airy.ji...@gmail.com wrote:
> no more people would like to discuss this topic?what a pity!

I have had nothing but trouble trying to export DXF files from AutoCAD
and read them with IDLffDXF. Some (most) objects never appear and
others are improperly positioned or rotated. My limited investigations
led me to conclude that while DXF may be an open standard to exchange
CAD models, it also allows for the inclusion of proprietary formatting
and objects. True, IDL does not support all object types that may
occur in DXF files, but this is not the primary problem. If you read
the DXF file directly (it's ASCII) you'll note a lot of AutoCAD
specific stuff in there that I gather tells AutoCAD more about how to
position and orient objects in the model. It would be much more useful
to me if IDL could read/write either IGES or STEP files as these are
really designed to exchange model geometries. I currently export these
types from AutoCAD and translate them to IDL-compatible DXF files
using 3rd party software from TechnoSoft (AML).

Subject: Re: about DXF format
Posted by Vince Hradil on Mon, 18 Jun 2007 14:55:57 GMT
View Forum Message <> Reply to Message

On Jun 18, 7:23 am, JMZawo...@gmail.com wrote:

> On Jun 16, 10:20 pm, airy.ji...@gmail.com wrote:
>
>> no more people would like to discuss this topic?what a pity!
>
> I have had nothing but trouble trying to export DXF files from AutoCAD
> and read them with IDLffDXF. Some (most) objects never appear and
> others are improperly positioned or rotated. My limited investigations
> led me to conclude that while DXF may be an open standard to exchange
> CAD models, it also allows for the inclusion of proprietary formatting
> and objects. True, IDL does not support all object types that may
> occur in DXF files, but this is not the primary problem. If you read
> the DXF file directly (it's ASCII) you'll note a lot of AutoCAD
> specific stuff in there that I gather tells AutoCAD more about how to
> position and orient objects in the model. It would be much more useful
> to me if IDL could read/write either IGES or STEP files as these are
> really designed to exchange model geometries. I currently export these
> types from AutoCAD and translate them to IDL-compatible DXF files
> using 3rd party software from TechnoSoft (AML).

In my experience, it IS possible to parse a dxf file. You just have
to read the docs that describe the format, then parse the file
correctly. The trick is that some entities contain other entities and
lines and they all have different local and global origins and scale
factors. Yeah, it complicated, but I've written a parser to parse a
few dxf files, and works (most of the time).

Here's my very crude code. Just try >plot_dxf, "file.dxf"

```
function resolve_inserts, innow, inserts, plines

  ;; plines
  nbdx = where( plines.blockname ne '', nbcnt )
  if nbcnt gt 0 then begin
     nbplines = plines[nbdx]
     bdx = where( nbplines.blockname eq innow.blockref, bcnt )
     for b=0l, bcnt-1 do begin
        pnow = plines[nbdx[bdx[b]]]
        *(pnow.vertices) = *(pnow.vertices) + rebin(innow.coord,
3,pnow.nvert,/sample)
        plines[nbdx[bdx[b]]] = pnow
     endfor
  endif

  ;; inserts
  nbdx = where( inserts.blockname ne '', nbcnt )
  if nbcnt gt 0 then begin
     nbinserts = inserts[nbdx]
     bdx = where( nbinserts.blockname eq innow.blockref, bcnt )
```

```
    for b=0l, bcnt-1 do begin
      ininside = inserts[nbdx[bdx[b]]]
      ininside.coord = ininside.coord + innow.coord
      if size(plines,/type) eq 8 then plines =
resolve_inserts(ininside,inserts, plines)
    endfor

    return, plines
  endif
end

function eval_spline, ncp, controlpts, nsegs

  tarray = findgen(nsegs)/(nsegs)
  np = (ncp-1)/3

  sval = fltarr(2,nsegs*np+1)
  for i=0l, np-1 do begin
    p0 = controlpts[*,3*i]
    p1 = controlpts[*,3*i+1]
    p2 = controlpts[*,3*i+2]
    p3 = controlpts[*,3*i+3]

    sval[*,nsegs*i] = p0
    for j=1l, nsegs-1 do begin
      t = tarray[j]
      vert = p0*(1-t)*(1-t)*(1-t) + p1*3.0*t*(1-t)*(1-t) +
p2*3.0*t*t*(1-t) + p3*t*t*t
      sval[*,nsegs*i+j] = vert
    endfor
  endfor
  sval[*,nsegs*np] = controlpts[*,ncp-1]

  return, sval
end

function decodetext, instring

  upos = strpos(instring,'\U+')
  if upos lt 0 then return, ''

  usplit = strsplit( strmid( instring,upos ), '\U+', /extract )
  outstring = bytarr(n_elements(usplit))
  reads, usplit, outstring, format='(Z)'

  return, string(outstring)
end
```

```
function read_dxf, fname

  nlines = file_lines(fname)
  print, 'NLINES: ', nlines

  openr, lun, fname, /get_lun
  line = ''
  adxf = strarr(nlines)
  for i=0l, nlines-1 do begin
     readf, lun, line
     adxf[i] = line
  endfor
  free_lun, lun

  return, adxf
end

function dxf_plines, fname, layer=layer, nsegs=nsegs

  if n_elements(layer) eq 0 then layer='ALL'
  if n_elements(nsegs) eq 0 then nsegs=4L

  adxf = read_dxf(fname)

  hdx = where( adxf eq 'HEADER')
  cdx = where( adxf eq 'CLASSES')
  tdx = where( adxf eq 'TABLES')
  bdx = where( adxf eq 'BLOCKS')
  edx = where( adxf eq 'ENTITIES')
  odx = where( adxf eq 'OBJECTS')
  eofdx = where( adxf eq 'EOF' )

  header = adxf[hdx:cdx-2]
  classes = adxf[cdx:tdx-2]
  tables = adxf[tdx:bdx-2]
  blocks = adxf[bdx:edx-2]
  entities = adxf[edx:odx-2]
  objects = adxf[odx:eofdx-2]

;   hfdx = where(header eq layer, hfcnt)
;   cfdx = where(classes eq layer, cfcnt)
;   tfdx = where(tables eq layer, tfcnt)
;   bfdx = where(blocks eq layer, bfcnt)
;   efdx = where(entities eq layer, efcnt)
;   ofdx = where(objects eq layer, ofcnt)

;   print, 'HEADER: ', hfcnt
;   print, 'CLASSES: ', cfcnt
;   print, 'TABLES: ', tfcnt
;   print, 'BLOCKS: ', bfcnt
;   print, 'ENTITIES: ', efcnt
;   print, 'OBJECTS: ', ofcnt

  ;; blocks
  iblocks = where( blocks eq 'BLOCK', nblocks )
  eblocks = where( blocks eq 'ENDBLK', neblocks )
  print, 'NBLOCKS: ', nblocks

  inserts = {blockname:'', blockref:'', coord:fltarr(3)}
  mtexts = {blockname:'', coord:fltarr(3), height:0.0, mstring:''}
  plines = {blockname:'', nvert:0L, vertices:ptr_new() }

  for i=0l, nblocks-1 do begin
     ibnow = iblocks[i]
     ebnow = eblocks[i]
     bnow = blocks[ibnow:ebnow]
     boffset = ( where( bnow eq 'AcDbBlockBegin' ) )[0]
     blockptr = boffset + 2l
     blockname = bnow[blockptr]
     blockptr = boffset + 16l
     if layer ne 'ALL' then begin
        bfdx = where( bnow eq layer, bfcnt )
     endif else begin
        bfcnt = 1l
     endelse
     if bfcnt gt 0 then begin
        print, 'BLOCK: ', blockname, i+1, format="(8A,12A,8I)"
        blen = ebnow-ibnow+1
        bent = bnow[blockptr]
        while blockptr lt blen and bent ne 'ENDBLK' do begin
           bent = bnow[blockptr]
           case bent of
              'INSERT': begin
                 boffset = ( where( bnow[blockptr:blen-1] eq
'AcDbBlockReference' ) )[0]
                 blockptr = blockptr + boffset + 2l
                 blockref = bnow[blockptr]
                 blockptr = blockptr + 2l
                 xinsert = float(bnow[blockptr])
                 blockptr = blockptr + 2l
                 yinsert = float(bnow[blockptr])
                 blockptr = blockptr + 2l
                 zinsert = float(bnow[blockptr])
                 print, ' INSERT: ', blockref, xinsert, yinsert,
zinsert
                 inserts = [ inserts, {blockname:blockname,
blockref:blockref, coord:[xinsert,yinsert,zinsert]} ]
                 blockptr = blockptr + 2l
              end
              'LINE': begin
                 boffset = ( where( bnow[blockptr:blen-1] eq
'AcDbLine' ) )[0]
                 blockptr = blockptr + boffset + 2l
                 startpt = fltarr(3)
                 endpt = fltarr(3)
                 for p=0l, 2 do begin
                    startpt[p] = float(bnow[blockptr])
                    blockptr = blockptr+2l
                 endfor
                 for p=0l, 2 do begin
                    endpt[p] = float(bnow[blockptr])
                    blockptr = blockptr+2l
                 endfor
                 print, ' LINE: ', strtrim(startpt) ;, strtrim(endpt)
                 plines = [ plines, {blockname:blockname, nvert:2l,
vertices:ptr_new([[startpt],[endpt]])} ]
              end
              'SPLINE': begin
                 boffset = ( where( bnow[blockptr:blen-1] eq
'AcDbSpline' ) )[0]
                 blockptr = blockptr + boffset + 2l
                 splineflag = long(bnow[blockptr])

                 blockptr = blockptr + 8l
                 degree = long(bnow[blockptr])

                 blockptr = blockptr + 2l
                 nknots = long(bnow[blockptr])
                 knots = fltarr(nknots)

                 blockptr = blockptr + 2l
                 ncontrolpts = long(bnow[blockptr])
;                 controlpts = fltarr(3,ncontrolpts)
                 controlpts = fltarr(2,ncontrolpts)

                 blockptr = blockptr + 2l
                 nfitpoints = long(bnow[blockptr])
                 if nfitpoints gt 0 then fitpoints =
fltarr(3,nfitpoints)

                 print, ' SPLINE: ', splineflag, degree, nknots,
ncontrolpts, nfitpoints

                 blockptr = blockptr + 2l
                 blockptr = blockptr +
( where( strtrim(bnow[blockptr:blen-1],2) eq '40' ) )[0] + 1
                 for p=0l, nknots-1 do begin
                   knots[p] = float(bnow[blockptr])
                   blockptr = blockptr + 2l
                 endfor
                 for p=0l, ncontrolpts-1 do begin
                   controlpts[0,p] = float(bnow[blockptr])
                   blockptr = blockptr + 2l
                   controlpts[1,p] = float(bnow[blockptr])
                   blockptr = blockptr + 2l
;                   controlpts[2,p] = float(bnow[blockptr])
                   blockptr = blockptr + 2l
                 endfor
;                 sx = sort( controlpts[0,*] )
;                 controlpts = controlpts[*,sx]
                 for p=0l, nfitpoints-1 do begin
                   fitpoints[0,p] = float(bnow[blockptr])
                   blockptr = blockptr + 2l
                   fitpoints[1,p] = float(bnow[blockptr])
                   blockptr = blockptr + 2l
                   fitpoints[2,p] = float(bnow[blockptr])
                   blockptr = blockptr + 2l
                 endfor
;                 print, 'DONE SPLINE'
                 sval = eval_spline(ncontrolpts,controlpts,nsegs)
                 np = nsegs*(ncontrolpts-1)/3+1
                 sval = transpose( [ [transpose([sval])],
[replicate(0.0,np)] ] )
                 plines = [ plines,
{blockname:blockname,nvert:ncontrolpts*nsegs
+1L,vertices:ptr_new(sval)} ]
               end
               'HATCH': begin
                 boffset = ( where( strtrim(bnow[blockptr:blen-1],2)
eq '2' ) )[0]
                 blockptr = blockptr + boffset + 1L
                 pattern = bnow[blockptr]

                 blockptr = blockptr + 6l
                 nloops = long(bnow[blockptr])
                 print, ' HATCH: ', pattern, nloops, format="(8A,12A,
8I,$)"
                 for p=0l, nloops-1 do begin
                   blockptr = blockptr + 2l
                   btype = long(bnow[blockptr])

                   blockptr = blockptr + 2l
```

```
              nedge = long(bnow[blockptr])

              blockptr = blockptr + 2l
              edgetype = long(bnow[blockptr])
              case edgetype of
                 1: begin ; line
                 end
                 2: begin ; circular arc
                 end
                 3: begin ; elliptical arc
                 end
                 4: begin ; spline
                    blockptr = blockptr + 2l
                    degree = long(bnow[blockptr])

                    blockptr = blockptr + 6l
                    nknots = long(bnow[blockptr])
                    knots = fltarr(nknots)

                    blockptr = blockptr + 2l
                    ncontrolpts = long(bnow[blockptr])
                    controlpts = fltarr(2,ncontrolpts)

                    blockptr = blockptr + 2l

                    print, ' SPLINE: ', degree, nknots,
ncontrolpts
                    for p=0l, nknots-1 do begin
                       knots[p] = float(bnow[blockptr])
                       blockptr = blockptr + 2l
                    endfor
                    for p=0l, ncontrolpts-1 do begin
                       controlpts[0,p] = float(bnow[blockptr])
                       blockptr = blockptr + 2l
                       controlpts[1,p] = float(bnow[blockptr])
                       blockptr = blockptr + 2l
                    endfor
;                    print, 'DONE HATCH SPLINE'
                    sval =
eval_spline(ncontrolpts,controlpts,nsegs)
                    np = nsegs*(ncontrolpts-1)/3+1
                    sval = transpose( [ [transpose([sval])],
[replicate(0.0,np)] ] )
                    plines = [ plines,
{blockname:blockname,nvert:ncontrolpts*nsegs
+1L,vertices:ptr_new(sval)} ]
                 end
              endcase
```

```
          endfor
          blockptr = blockptr + 12l
        end
        'MTEXT' : begin
          boffset = ( where( bnow[blockptr:blen-1] eq
'AcDbMText' ) )[0]
          blockptr = blockptr + boffset + 2l
          position = fltarr(3)
          for p=0l, 2 do begin
            position[p] = float(bnow[blockptr])
            blockptr = blockptr+2l
          endfor
          height = float(bnow[blockptr])
          blockptr = blockptr+8l
          tstring = bnow[blockptr]
          mstring = decodetext(tstring)
          print, ' MTEXT: ', mstring
          mtexts = [ mtexts, {blockname:blockname,
coord:position, height:height, mstring:mstring} ]
          blockptr = blockptr + 14l
        end
        'POLYLINE' : begin
          seqend = ( where( bnow[blockptr:blen-1] eq
'SEQEND' ) )[0]
          vdx = where( bnow[blockptr:blockptr+seqend] eq
'AcDb2dVertex', vcnt )
          if vcnt gt 0 then begin
            vertices = fltarr(3,vcnt)
            for v=0l, vcnt-1 do begin
              boffset = ( where( bnow[blockptr:blen-1] eq
'AcDb2dVertex' ) )[0]
              blockptr = blockptr + boffset + 2l
              for p=0l, 2 do begin
                vertices[p,v] = float(bnow[blockptr])
                blockptr = blockptr+2l
              endfor
            endfor
            print, ' PLINE: ', strtrim(vertices[*,0])
            plines = [ plines, {blockname:blockname,
nvert:vcnt, vertices:ptr_new(vertices)} ]
          endif
          blockptr = blockptr + 10l
        end
        'ENDBLK': break
        else: begin       ; unknown block
          print, 'Unknown Block: ', bent
          blockptr = blockptr + 1l
        end
```

```
          endcase
        endwhile
      endif else begin
;        print, 'BLOCK: ', blockname, i+1, format="(8A,12A,8I,$)"
;        print, ' (NOT FIGURE)'
      endelse
    endfor

    bnow = entities
    blen = odx-2-edx

    blockptr = 2l
    if layer ne 'ALL' then begin
      bfdx = where( bnow eq layer, bfcnt )
    endif else begin
      bfcnt = 1l
    endelse
    if bfcnt gt 0 then begin
      bent = bnow[blockptr]
      repeat begin
        bent = bnow[blockptr]
        case bent of
          'INSERT': begin
            boffset = ( where( bnow[blockptr:blen-1] eq
'AcDbEntity' ) )[0]
            blockptr = blockptr + boffset + 2l
            if layer eq 'ALL' or layer eq bnow[blockptr] then begin
              boffset = ( where( bnow[blockptr:blen-1] eq
'AcDbBlockReference' ) )[0]
              blockptr = blockptr + boffset + 2l
              blockref = bnow[blockptr]
              blockptr = blockptr + 2l
              xinsert = float(bnow[blockptr])
              blockptr = blockptr + 2l
              yinsert = float(bnow[blockptr])
              blockptr = blockptr + 2l
              zinsert = float(bnow[blockptr])
              print, ' INSERT: ', blockref, xinsert, yinsert,
zinsert
              inserts = [ inserts, {blockname:'',
blockref:blockref, coord:[xinsert,yinsert,zinsert]} ]
              blockptr = blockptr + 2l
            endif else begin
              blockptr = blockptr + ( where( bnow[blockptr:blen-1]
eq '  0' ) )[0] + 1
            endelse
          end
          'LINE': begin
```

```
             boffset = ( where( bnow[blockptr:blen-1] eq
'AcDbEntity' ) )[0]
             blockptr = blockptr + boffset + 2l
             if layer eq 'ALL' or layer eq bnow[blockptr] then begin
               boffset = ( where( bnow[blockptr:blen-1] eq
'AcDbLine' ) )[0]
               blockptr = blockptr + boffset + 2l
               startpt = fltarr(3)
               endpt = fltarr(3)
               for p=0l, 2 do begin
                 startpt[p] = float(bnow[blockptr])
                 blockptr = blockptr+2l
               endfor
               for p=0l, 2 do begin
                 endpt[p] = float(bnow[blockptr])
                 blockptr = blockptr+2l
               endfor
               print, ' LINE: ', strtrim(startpt) ;, strtrim(endpt)
               plines = [ plines, {blockname:'', nvert:2l,
vertices:ptr_new([[startpt],[endpt]])} ]
             endif else begin
               blockptr = blockptr + ( where( bnow[blockptr:blen-1]
eq '  0' ) )[0] + 1
             endelse
           end
           'SPLINE': begin
             boffset = ( where( bnow[blockptr:blen-1] eq
'AcDbEntity' ) )[0]
             blockptr = blockptr + boffset + 2l
             if layer eq 'ALL' or layer eq bnow[blockptr] then begin
               boffset = ( where( bnow[blockptr:blen-1] eq
'AcDbSpline' ) )[0]
               blockptr = blockptr + boffset + 2l
               splineflag = long(bnow[blockptr])

               blockptr = blockptr + 8l
               degree = long(bnow[blockptr])

               blockptr = blockptr + 2l
               nknots = long(bnow[blockptr])
               knots = fltarr(nknots)

               blockptr = blockptr + 2l
               ncontrolpts = long(bnow[blockptr])
;               controlpts = fltarr(3,ncontrolpts)
               controlpts = fltarr(2,ncontrolpts)

               blockptr = blockptr + 2l
```

```
                nfitpoints = long(bnow[blockptr])
                if nfitpoints gt 0 then fitpoints =
fltarr(3,nfitpoints)

                print, ' SPLINE: ', splineflag, degree, nknots,
ncontrolpts, nfitpoints

                blockptr = blockptr + 2l
                blockptr = blockptr +
( where( strtrim(bnow[blockptr:blen-1],2) eq '40' ) )[0] + 1
                for p=0l, nknots-1 do begin
                   knots[p] = float(bnow[blockptr])
                   blockptr = blockptr + 2l
                endfor
                for p=0l, ncontrolpts-1 do begin
                   controlpts[0,p] = float(bnow[blockptr])
                   blockptr = blockptr + 2l
                   controlpts[1,p] = float(bnow[blockptr])
                   blockptr = blockptr + 2l
;                   controlpts[2,p] = float(bnow[blockptr])
                   blockptr = blockptr + 2l
                endfor
;                sx = sort( controlpts[0,*] )
;                controlpts = controlpts[*,sx]
                for p=0l, nfitpoints-1 do begin
                   fitpoints[0,p] = float(bnow[blockptr])
                   blockptr = blockptr + 2l
                   fitpoints[1,p] = float(bnow[blockptr])
                   blockptr = blockptr + 2l
                   fitpoints[2,p] = float(bnow[blockptr])
                   blockptr = blockptr + 2l
                endfor
;                print, 'DONE SPLINE'
                sval = eval_spline(ncontrolpts,controlpts,nsegs)
                np = nsegs*(ncontrolpts-1)/3+1
                sval = transpose( [ [transpose([sval])],
[replicate(0.0,np)] ] )
                plines = [ plines,
 {blockname:'',nvert:ncontrolpts*nsegs+1L,vertices:ptr_new(sv al)} ]
             endif else begin
                blockptr = blockptr + ( where( bnow[blockptr:blen-1]
eq '  0' ) )[0] + 1
             endelse
          end
          'HATCH': begin
             boffset = ( where( bnow[blockptr:blen-1] eq
'AcDbEntity' ) )[0]
             blockptr = blockptr + boffset + 2l
```

```
         if layer eq 'ALL' or layer eq bnow[blockptr] then begin
            boffset = ( where( strtrim(bnow[blockptr:blen-1],2)
eq '2' ) )[0]
            blockptr = blockptr + boffset + 1L
            pattern = bnow[blockptr]

            blockptr = blockptr + 6l
            nloops = long(bnow[blockptr])
            print, ' HATCH: ', pattern, nloops, format="(8A,12A,
8I,$)"
            for p=0l, nloops-1 do begin
               blockptr = blockptr + 2l
               btype = long(bnow[blockptr])

               blockptr = blockptr + 2l
               nedge = long(bnow[blockptr])

               blockptr = blockptr + 2l
               edgetype = long(bnow[blockptr])
               case edgetype of
                  1: begin ; line
                  end
                  2: begin ; circular arc
                  end
                  3: begin ; elliptical arc
                  end
                  4: begin ; spline
                     blockptr = blockptr + 2l
                     degree = long(bnow[blockptr])

                     blockptr = blockptr + 6l
                     nknots = long(bnow[blockptr])
                     knots = fltarr(nknots)

                     blockptr = blockptr + 2l
                     ncontrolpts = long(bnow[blockptr])
                     controlpts = fltarr(2,ncontrolpts)

                     blockptr = blockptr + 2l

                     print, ' SPLINE: ', degree, nknots,
ncontrolpts
                     for p=0l, nknots-1 do begin
                        knots[p] = float(bnow[blockptr])
                        blockptr = blockptr + 2l
                     endfor
                     for p=0l, ncontrolpts-1 do begin
                        controlpts[0,p] = float(bnow[blockptr])
```

```
                          blockptr = blockptr + 2l
                          controlpts[1,p] = float(bnow[blockptr])
                          blockptr = blockptr + 2l
                        endfor
;                        print, 'DONE HATCH SPLINE'
                        sval =
eval_spline(ncontrolpts,controlpts,nsegs)
                        np = nsegs*(ncontrolpts-1)/3+1
                        sval = transpose( [ [transpose([sval])],
[replicate(0.0,np)] ] )
                        plines = [ plines,
 {blockname:'',nvert:ncontrolpts*nsegs+1L,vertices:ptr_new(sv al)} ]
                      end
                    endcase
                  endfor
                  blockptr = blockptr + 12l
                endif else begin
                  blockptr = blockptr + ( where( bnow[blockptr:blen-1]
eq '  0' ) )[0] + 1
                endelse
              end
              'MTEXT' : begin
                boffset = ( where( bnow[blockptr:blen-1] eq
'AcDbEntity' ) )[0]
                blockptr = blockptr + boffset + 2l
                if layer eq 'ALL' or layer eq bnow[blockptr] then begin
                  boffset = ( where( bnow[blockptr:blen-1] eq
'AcDbMText' ) )[0]
                  blockptr = blockptr + boffset + 2l
                  position = fltarr(3)
                  for p=0l, 2 do begin
                    position[p] = float(bnow[blockptr])
                    blockptr = blockptr+2l
                  endfor
                  height = float(bnow[blockptr])
                  blockptr = blockptr+8l
                  tstring = bnow[blockptr]
                  mstring = decodetext(tstring)
                  print, ' MTEXT: ', mstring
                  mtexts = [ mtexts, {blockname:'', coord:position,
height:height, mstring:mstring} ]
                  blockptr = blockptr + 14l
                endif else begin
                  blockptr = blockptr + ( where( bnow[blockptr:blen-1]
eq '  0' ) )[0] + 1
                endelse
              end
              'POLYLINE' : begin
              boffset = ( where( bnow[blockptr:blen-1] eq
'AcDbEntity' ) )[0]
              blockptr = blockptr + boffset + 2l
              if layer eq 'ALL' or layer eq bnow[blockptr] then begin
                 seqend = ( where( bnow[blockptr:blen-1] eq
'SEQEND' ) )[0]
                 vdx = where( bnow[blockptr:blockptr+seqend] eq
'AcDb2dVertex', vcnt )
                 if vcnt gt 0 then begin
                    vertices = fltarr(3,vcnt)
                    for v=0l, vcnt-1 do begin
                       boffset = ( where( bnow[blockptr:blen-1] eq
'AcDb2dVertex' ) )[0]
                       blockptr = blockptr + boffset + 2l
                       for p=0l, 2 do begin
                          vertices[p,v] = float(bnow[blockptr])
                          blockptr = blockptr+2l
                       endfor
                    endfor
                    print, ' PLINE: ', strtrim(vertices[*,0])
                    plines = [ plines, {blockname:'', nvert:vcnt,
vertices:ptr_new(vertices)} ]
                 endif
                 blockptr = blockptr + 10l
              endif else begin
                 blockptr = blockptr + ( where( bnow[blockptr:blen-1]
eq '  0' ) )[0] + 1
              endelse
           end
           'ENDSEC': break
           else: begin          ; unknown block
              print, 'Unknown Block: ', bent
              blockptr = blockptr + ( where( bnow[blockptr:blen-1] eq
'  0' ) )[0] + 1
           end
        endcase
     endrep until blockptr ge blen or bent eq 'ENDSEC'
  endif else begin
;       print, 'BLOCK: ', blockname, i+1, format="(8A,12A,8I,$)"
;       print, ' (NOT FIGURE)'
  endelse

  ninserts = ( size(inserts,/dimensions) )[0]
  if ninserts gt 1 then begin
     ninserts = ninserts - 1
     inserts = inserts[1:ninserts]
  endif else begin
     inserts = (-1)
  endelse

  nmtexts = ( size(mtexts,/dimensions) )[0]
  if nmtexts gt 1 then begin
     nmtexts = nmtexts - 1
     mtexts = mtexts[1:nmtexts]
  endif else begin
     mtexts = (-1)
  endelse

  nplines = ( size(plines,/dimensions) )[0]
  ptr_free, (plines[0]).vertices
  if nplines gt 1 then begin
     nplines = nplines - 1
     plines = plines[1:nplines]
  endif else begin
     plines = (-1)
  endelse

  ;; resolve inserts
  if size(inserts,/type) eq 8 then begin
     idx = where( inserts.blockname eq '', cnt )
     for i=0L, cnt-1 do begin
;        print, i+1, ' of ', cnt
;      ans = ''
;      read, ans, prompt='Enter something:'
        innow = inserts[idx[i]]
        if size(plines,/type) eq 8 then plines =
resolve_inserts(innow,inserts, plines)
     endfor
  endif

  return, plines
end

pro plot_dxf, fname, nsegs=nsegs, layer=layer

  colors
  bignum = 9999999.9
  minx = bignum
  maxx = -bignum
  miny = bignum
  maxy = -bignum

  if n_elements(nsegs) eq 0 then nsegs=10L

  plines = dxf_plines(fname, layer=layer, nsegs=nsegs)
  if size(plines,/type) ne 8 then return
  nplines = n_elements(plines)
  for i=0L, nplines-1 do begin
    minx = min( (*(plines[i].vertices))[0,*] ) < minx
    miny = min( (*(plines[i].vertices))[1,*] ) < miny
    maxx = max( (*(plines[i].vertices))[0,*] ) > maxx
    maxy = max( (*(plines[i].vertices))[1,*] ) > maxy
  endfor

  mmx = [minx,maxx]
  mmy = [miny,maxy]

  print, mmx
  print, mmy

  mmx = [0,300]
  mmy = [0,300]

  plot, [0,0], [1,1], /nodata, /noerase, xstyle=5, ystyle=5, /
isotropic, xrange=mmx, yrange=mmy

  for j=0l, nplines-1 do plots, *(plines[j].vertices)

  return
end
```

Subject: Re: about DXF format
Posted by Vince Hradil on Mon, 18 Jun 2007 15:14:19 GMT
View Forum Message <> Reply to Message

Subject: Re: about DXF format
Posted by airy.jiang on Tue, 19 Jun 2007 15:20:48 GMT
View Forum Message <> Reply to Message

Thanks everyone to join here and give me so much help,specially thanks
hradilv who showed a lot of useful sourece code,though I didn't read
this code completely yet.I'll spend more time on it until when a nice
result to be made.Later I'll came back here to continue discuss this
topic.Thank you very much!

Subject: Re: about DXF format
Posted by JMZawodny on Thu, 21 Jun 2007 13:51:34 GMT

Thanks for offering the code. Unfortunately it did not work and it was
not because of the missing COLORS procedure. My DXF files are composed
primarily of 3DBLOCKs. These appear to go unparsed in your routine.
What I really need is something that will read the DXf files and
output a set of mesh_objects, one for each layer. This what IDLffDXf
should do, but does not.

Thanks again for the effort.

Joe

On Jun 18, 10:55 am, hradilv <hrad...@yahoo.com> wrote:
> On Jun 18, 7:23 am, JMZawo...@gmail.com wrote:
>
>
>
>> On Jun 16, 10:20 pm, airy.ji...@gmail.com wrote:
>
>>> no more people would like to discuss this topic?what a pity!
>
>> I have had nothing but trouble trying to export DXF files from AutoCAD
>> and read them with IDLffDXF. Some (most) objects never appear and
>> others are improperly positioned or rotated. My limited investigations
>> led me to conclude that while DXF may be an open standard to exchange
>> CAD models, it also allows for the inclusion of proprietary formatting
>> and objects. True, IDL does not support all object types that may
>> occur in DXF files, but this is not the primary problem. If you read
>> the DXF file directly (it's ASCII) you'll note a lot of AutoCAD
>> specific stuff in there that I gather tells AutoCAD more about how to
>> position and orient objects in the model. It would be much more useful
>> to me if IDL could read/write either IGES or STEP files as these are
>> really designed to exchange model geometries. I currently export these
>> types from AutoCAD and translate them to IDL-compatible DXF files
>> using 3rd party software from TechnoSoft (AML).
>
> In my experience, it IS possible to parse a dxf file. You just have
> to read the docs that describe the format, then parse the file
> correctly. The trick is that some entities contain other entities and
> lines and they all have different local and global origins and scale
> factors. Yeah, it complicated, but I've written a parser to parse a
> few dxf files, and works (most of the time).
>
> Here's my very crude code. Just try >plot_dxf, "file.dxf"
>
> function resolve_inserts, innow, inserts, plines
> ... snip ...

Subject: Re: about DXF format
Posted by Vince Hradil on Thu, 21 Jun 2007 14:50:32 GMT
View Forum Message <> Reply to Message

```
On Jun 21, 8:51 am, JMZawo...@gmail.com wrote:
>  Thanks for offering the code. Unfortunately it did not work and it was
>  not because of the missing COLORS procedure. My DXF files are composed
>  primarily of 3DBLOCKs. These appear to go unparsed in your routine.
>  What I really need is something that will read the DXf files and
>  output a set of mesh_objects, one for each layer. This what IDLffDXf
>  should do, but does not.
>
>    Thanks again for the effort.
>
>       Joe
>
>  On Jun 18, 10:55 am, hradilv <hrad...@yahoo.com> wrote:
>
>>  On Jun 18, 7:23 am, JMZawo...@gmail.com wrote:
>
>>>  On Jun 16, 10:20 pm, airy.ji...@gmail.com wrote:
>
>>>>  no more people would like to discuss this topic?what a pity!
>
>>>  I have had nothing but trouble trying to export DXF files from AutoCAD
>>>  and read them with IDLffDXF. Some (most) objects never appear and
>>>  others are improperly positioned or rotated. My limited investigations
>>>  led me to conclude that while DXF may be an open standard to exchange
>>>  CAD models, it also allows for the inclusion of proprietary formatting
>>>  and objects. True, IDL does not support all object types that may
>>>  occur in DXF files, but this is not the primary problem. If you read
>>>  the DXF file directly (it's ASCII) you'll note a lot of AutoCAD
>>>  specific stuff in there that I gather tells AutoCAD more about how to
>>>  position and orient objects in the model. It would be much more useful
>>>  to me if IDL could read/write either IGES or STEP files as these are
>>>  really designed to exchange model geometries. I currently export these
>>>  types from AutoCAD and translate them to IDL-compatible DXF files
>>>  using 3rd party software from TechnoSoft (AML).
>
>>  In my experience, it IS possible to parse a dxf file.  You just have
>>  to read the docs that describe the format, then parse the file
>>  correctly.  The trick is that some entities contain other entities and
>>  lines and they all have different local and global origins and scale
>>  factors.  Yeah, it complicated, but I've written a parser to parse a
>>  few dxf files, and works (most of the time).
>
>>  Here's my very crude code.  Just try >plot_dxf, "file.dxf"
>
>>  function resolve_inserts, innow, inserts, plines
```

>> ... snip ...

That's true, I wrote it for one particular purpose - to parse lines
and polylines.

I guess you meant 3DFACE or 3DSOLID (everything is a block?? isn't
it. [it's been a while since I looked at this]).

Anyway, maybe you can use mine as a start. Here's a link to the DXF
format, if you haven't found it yet: http://tinyurl.com/232tsa

---

Subject: Re: about DXF format
Posted by Rick Towler on Thu, 21 Jun 2007 16:26:27 GMT
View Forum Message <> Reply to Message

JMZawodny wrote:
> Thanks for offering the code. Unfortunately it did not work and it was
> not because of the missing COLORS procedure. My DXF files are composed
> primarily of 3DBLOCKs. These appear to go unparsed in your routine.
> What I really need is something that will read the DXf files and
> output a set of mesh_objects, one for each layer. This what IDLffDXf
> should do, but does not.

What specifically is the problem with IDLffDXF? It is hard to offer
advice or suggestions if you don't specify the problem. Is it similar
to the OP's issue that some specific coordinate transformation
information is lost? Or is it something else?

-Rick

Subject: Re: about DXF format
Posted by airy.jiang on Wed, 27 Jun 2007 04:12:59 GMT
View Forum Message <> Reply to Message

On 6 22 ,   12 26 , Rick Towler <rick.tow...@nomail.noaa.gov> wrote:
> JMZawodny wrote:
>> Thanks for offering the code. Unfortunately it did not work and it was
>> not because of the missing COLORS procedure. My DXF files are composed
>> primarily of 3DBLOCKs. These appear to go unparsed in your routine.
>> What I really need is something that will read the DXf files and
>> output a set of mesh_objects, one for each layer. This what IDLffDXf
>> should do, but does not.
>
> What specifically is the problem with IDLffDXF? It is hard to offer
> advice or suggestions if you don't specify the problem. Is it similar

> to the OP's issue that some specific coordinate transformation
> information is lost? Or is it something else?
>
> -Rick

The IDLffDXF objects have a lot of subobjects,but I just know how to
use IDL_DXF_POLYGON,IDL_DXF_POLYLINE,which I learned from the example
code.But in fact,there are many other subobjects,like
IDL_DXF_BLOCK,IDL_DXF_LAYER,IDL_DXF_INSERT,and so on.It won't be
useless,but until now,I still don't know how to use them.

Subject: Re: about DXF format
Posted by Vince Hradil on Wed, 27 Jun 2007 13:39:02 GMT
View Forum Message <> Reply to Message

On Jun 26, 11:12 pm, airy.ji...@gmail.com wrote:
> On 6 22 , 12 26 , Rick Towler <rick.tow...@nomail.noaa.gov> wrote:
>
>> JMZawodny wrote:
>>> Thanks for offering the code. Unfortunately it did not work and it was
>>> not because of the missing COLORS procedure. My DXF files are composed
>>> primarily of 3DBLOCKs. These appear to go unparsed in your routine.
>>> What I really need is something that will read the DXf files and
>>> output a set of mesh_objects, one for each layer. This what IDLffDXf
>>> should do, but does not.
>
>> What specifically is the problem with IDLffDXF? It is hard to offer
>> advice or suggestions if you don't specify the problem. Is it similar
>> to the OP's issue that some specific coordinate transformation
>> information is lost? Or is it something else?
>
>> -Rick
>
> The IDLffDXF objects have a lot of subobjects,but I just know how to
> use IDL_DXF_POLYGON,IDL_DXF_POLYLINE,which I learned from the example
> code.But in fact,there are many other subobjects,like
> IDL_DXF_BLOCK,IDL_DXF_LAYER,IDL_DXF_INSERT,and so on.It won't be
> useless,but until now,I still don't know how to use them.

Did you see my post earlier:

qoute:
Anyway, maybe you can use mine as a start. Here's a link to the DXF
format, if you haven't found it yet: http://tinyurl.com/232tsa

I know this isn't "idl specific", but it does explain the dxf format
quite well.

Subject: Re: about DXF format
Posted by JMZawodny on Thu, 28 Jun 2007 17:58:10 GMT
View Forum Message <> Reply to Message

On Jun 21, 12:26 pm, Rick Towler <rick.tow...@nomail.noaa.gov> wrote:
> JMZawodny wrote:
>> Thanks for offering the code. Unfortunately it did not work and it was
>> not because of the missing COLORS procedure. My DXF files are composed
>> primarily of 3DBLOCKs. These appear to go unparsed in your routine.
>> What I really need is something that will read the DXf files and
>> output a set of mesh_objects, one for each layer. This what IDLffDXf
>> should do, but does not.
>
> What specifically is the problem with IDLffDXF? It is hard to offer
> advice or suggestions if you don't specify the problem. Is it similar
> to the OP's issue that some specific coordinate transformation
> information is lost? Or is it something else?
>
> -Rick

After reading the dxf file into my object IDLffDXF->GetContents tells
me that I have 3 blocks, 9 inserts, and 6 layers. The layers appear to
have the correct names. The blocks appear to be generic/default
AutoCAD 'things'. I have no idea what the Inserts are all about. They
are all the same except for the .color field. There are no other
Entities present and nothing that would represent my model (types
1-17). The dxf file is good in that AutoCAD generates the proper model
from the file. So, I really cannot track down what is going wrong
beyond what I've just told you.

Subject: Re: about DXF format
Posted by JMZawodny on Thu, 28 Jun 2007 18:37:31 GMT
View Forum Message <> Reply to Message

One more thing about the dxf file. The bulk of the file is comprised
of several groups of lines like this ...

```
  0
3DSOLID
  5
63
330
1F
100
AcDbEntity
  8
TELESCOPE
```

```
100
AcDbModelerGeometry
 70
    1
  1
noi io n o
  1
=0;& {n {m {rn {rn |
  1
-:9@)+r3(;r>++-6= {rn {rn {rn {o {l {k |
  1
3*2/ {j {rn {i {o |
  1
-:961:2:1+ {rn o o o o oqomknolinhmfjimmgmi lo o o n o |
  1
):-+:'@+:2/3>+: {rn l o n g |
  1
-:9@)+r3(;r>++-6= {rn {rn {rn {m {l {k |
  1
,7:33 {h {rn {rn {g {m |
  1
-:9@)+r3(;r>++-6= {rn {rn {rn {i {l {k |
  1
9><: {f {no {nn {i {rn {nm -:):-,:; ,6183: |
  1
92:,7r3(;r>++-6= {rn {nl {rn {g |
  1
9><: {nk {nj {ni {i {rn {nh -:):-,:; ,6183: |
```

...

I have not had the time to dig into the DXF file format specifications
to figure out exactly what these lines define, but I strongly suspect
they are my missing model objects.

Joe

Subject: Re: about DXF format
Posted by Vince Hradil on Thu, 28 Jun 2007 19:23:35 GMT
View Forum Message <> Reply to Message

On Jun 28, 1:37 pm, JMZawo...@gmail.com wrote:
> One more thing about the dxf file. The bulk of the file is comprised
> of several groups of lines like this ...
>
>   0
> 3DSOLID

```
>   5
> 63
> 330
> 1F
> 100
> AcDbEntity
>   8
> TELESCOPE
> 100
> AcDbModelerGeometry
>  70
>      1
>   1
> noi io n o
>   1
> =0;& {n {m {rn {rn |
>   1
> -:9@)+r3(;r>++-6= {rn {rn {rn {o {l {k |
>   1
> 3*2/ {j {rn {i {o |
>   1
> -:961:2:1+ {rn o o o o oqomknolinhmfjimmgmi lo o o n o |
>   1
> ):-+:'@+:2/3>+: {rn l o n g |
>   1
> -:9@)+r3(;r>++-6= {rn {rn {rn {m {l {k |
>   1
> ,7:33 {h {rn {rn {g {m |
>   1
> -:9@)+r3(;r>++-6= {rn {rn {rn {i {l {k |
>   1
> 9><: {f {no {nn {i {rn {nm -:):-,:; ,6183: |
>   1
> 92:,7r3(;r>++-6= {rn {nl {rn {g |
>   1
> 9><: {nk {nj {ni {i {rn {nh -:):-,:; ,6183: |
>
> ...
>
```

> I have not had the time to dig into the DXF file format specifications
> to figure out exactly what these lines define, but I strongly suspect
> they are my missing model objects.
>
> Joe

> From the specs:

3dsolid group codes

Group codes Description
100
Subclass marker (AcDbModelerGeometry)

70
Modeler format version number (currently = 1)

1
Proprietary data (multiple lines < 255 characters each)

3
Additional lines of proprietary data (if previous group 1 string is
greater than 255 characters) (optional)

So I'm guessing these are "proprietary data" - maybe depending on the
APP that wrote the DXF. Bummer...

Subject: Re: about DXF format
Posted by JMZawodny on Fri, 29 Jun 2007 12:30:53 GMT

View Forum Message <> Reply to Message

On Jun 28, 3:23 pm, hradilv <hrad...@yahoo.com> wrote:
> On Jun 28, 1:37 pm, JMZawo...@gmail.com wrote:
>
>
>
>> One more thing about the dxf file. The bulk of the file is comprised
>> of several groups of lines like this ...
>
```
>>    0
>>  3DSOLID
>>    5
>>  63
>>  330
>>  1F
>>  100
>>  AcDbEntity
>>    8
>>  TELESCOPE
>>  100
>>  AcDbModelerGeometry
>>   70
>>       1
>>    1
>>  noi io n o
>>    1
```

>
>> Joe
>> From the specs:
>
> 3dsolid group codes
>
> Group codes   Description
> 100
> Subclass marker (AcDbModelerGeometry)
>
> 70
> Modeler format version number (currently = 1)
>
> 1
> Proprietary data (multiple lines < 255 characters each)
>
> 3
> Additional lines of proprietary data (if previous group 1 string is
> greater than 255 characters) (optional)
>
> So I'm guessing these are "proprietary data" - maybe depending on the
> APP that wrote the DXF. Bummer...

Bingo! That was the thought behind my first post to this thread. The "standard" supports the use of proprietary data. Things like that generally make for useless standards. My only success in getting AutoCAD-generated models into IDL via IDLffDXF is to export the model from AutoCAD as an IGES or STEP file. Read it in with 3rd party software which can tesselate the model and export the points and connectivity arrays in a DXF. Hence my original request for IDL to support reading IGES or STEP files.

I do appreciate everyone's ideas and input on this.

Joe