Subject: Re: access to vercities & connectivity data
Posted by Rick Towler on Mon, 10 Nov 2003 22:33:24 GMT
View Forum Message <> Reply to Message

"Neil" wrote...
> I would like to read in a DXF file and get access to the numerical
> values of the vertices and the connectivity. That's all i wish to do,
> which IDL routines should i use for this task. From the IDL help it is
> not immediately obvious how this can be done.

There are a couple of ways:

You can use the IDLffDXF object. There is an example close to what you are looking for in in the docs under IDLffDXF::Read. Also of interest will be IDLffDXF::GetEntity which describes how to pull out the different entity types. The types are described in the IDLffDXF::GetContents docs.

Another way, which is a bit of a hack but might be easier (especially if you don't know what is in your DXF file) is to use the undocumented GET_DXF_OBJECTS function (found in $RSI_DIR/lib/utilities). There is some documentation in the file header.

Yet another way, which I had completely forgot about, is to use my dxfmodel object which reads a dxf file (only extracts types 4,8,9, and 11) and creates a model containing the primitives. It also gives you access to the primitive objects (IDLgrPolyline and IDLgrPolygon) from which you could get at the vertex and connectivity data. It requires David Fanning's linkedlist object (linkedlist__define.pro) available from
http://www.dfanning.com/documents/programs.html

Using my dxfmodel object would go something like:

```
IDL> dxfmod=OBJ_NEW('dxfmodel', 'stonehenge.dxf')
% Loaded DLM: DXF.
IDL> p=dxfmod->getprimitive(0)
IDL> help, p, /struct
** Structure <1760450>, 4 tags, length=32, data length=30, refs=2:
   BLOCK           STRING    ''
   LAYER           STRING    'biglent02'
   TYPE            INT              9
   OBJECT          OBJREF    <ObjHeapVar252(IDLGRPOLYGON)>
IDL> p.object->getproperty, data=verts, polygons=conn
IDL> help,verts
VERTS           DOUBLE    = Array[3, 216]
IDL> help, conn
CONN            LONG      = Array[1200]
```

Note that dxfmodel__define isn't "finished" but unless you are in need of
extracting some odd dxf types it should work fine for what you want to do.

you can find my object here:

 http://www.acoustics.washington.edu/~towler/dxfmodel__define .pro

Enjoy!

-Rick

Subject: Re: access to vercities & connectivity data
Posted by nasalmon on Tue, 11 Nov 2003 21:12:31 GMT

View Forum Message <> Reply to Message

"Rick Towler" <rtowler@u.washington.edu> wrote in message
news:<bop3rj$20ts$1@nntp6.u.washington.edu>...
> "Neil" wrote...
>> I would like to read in a DXF file and get access to the numerical
>> values of the vertices and the connectivity. That's all i wish to do,
>> which IDL routines should i use for this task. From the IDL help it is
>> not immediately obvious how this can be done.
>
> There are a couple of ways:
>
> You can use the IDLffDXF object. There is an example close to what you are
> looking for in in the docs under IDLffDXF::Read. Also of interest will be
> IDLffDXF::GetEntity which describes how to pull out the different entity
> types. The types are described in the IDLffDXF::GetContents docs.
>
> Another way, which is a bit of a hack but might be easier (especially if you
> don't know what is in your DXF file) is to use the undocumented
> GET_DXF_OBJECTS function (found in $RSI_DIR/lib/utilities). There is some
> documentation in the file header.
>
> Yet another way, which I had completely forgot about, is to use my dxfmodel
> object which reads a dxf file (only extracts types 4,8,9, and 11) and
> creates a model containing the primitives. It also gives you access to the
> primitive objects (IDLgrPolyline and IDLgrPolygon) from which you could get
> at the vertex and connectivity data. It requires David Fanning's linkedlist
> object (linkedlist__define.pro) available from
> http://www.dfanning.com/documents/programs.html
>
> Using my dxfmodel object would go something like:
>
> IDL> dxfmod=OBJ_NEW('dxfmodel', 'stonehenge.dxf')
> % Loaded DLM: DXF.

```
>  IDL> p=dxfmod->getprimitive(0)
>  IDL> help, p, /struct
>  ** Structure <1760450>, 4 tags, length=32, data length=30, refs=2:
>     BLOCK          STRING    ''
>     LAYER          STRING    'biglent02'
>     TYPE           INT              9
>     OBJECT         OBJREF    <ObjHeapVar252(IDLGRPOLYGON)>
>  IDL> p.object->getproperty, data=verts, polygons=conn
>  IDL> help,verts
>  VERTS           DOUBLE    = Array[3, 216]
>  IDL> help, conn
>  CONN            LONG      = Array[1200]
>
>
```
> Note that dxfmodel__define isn't "finished" but unless you are in need of
> extracting some odd dxf types it should work fine for what you want to do.
>
> you can find my object here:
>
>   http://www.acoustics.washington.edu/~towler/dxfmodel__define .pro
>
> Enjoy!
>
> -Rick

Thank you Rick for you comments. Well i decided on option 1 of yours
from above.
So i wrote this simple programme, trying to access the "verticies" and
"connectivity" from the DXF file in the examples from the heart file:

```
filename = filepath('heart.dxf', subdir=['examples', 'data'])
oTest = OBJ_NEW('IDLffDXF')   ; Initialize DXF data access object
status = oTest -> Read(filename); Read data within DXF file into
access object.
TestTypes = oTest -> GetContents(COUNT = TestCounts);Determine what
type entity (and how many of each entity) exist in the file.
PRINT, 'Entity Types:   ', TestTypes
PRINT, 'Count of Types: ', TestCounts
oModel = OBJ_NEW('IDLgrModel'); Initialize a model for displaying
tissue = oTest -> GetEntity(TestTypes[1]); Obtain the tissue data.
HELP, tissue
HELP, tissue, /STRUCTURE
print, 'vertices',tissue.vertices
print, 'connectivity',tissue.connectivity
```

This seems to be getting part of the way, the information looks the right stuff, but i dont know if all the routine calls are necessary.

However, i do not yet seem to be able to access the real numberical
values of the "verticies" and "connectivities". I can see that
tissues.verticies and tissue.connectivity are pointers, but to what?
How do i get the one dimensional array with all the connectivity and
the 3 by n array of verticies? Which variable holds the data?

many thanks,

Neil

Subject: Re: access to vercities & connectivity data
Posted by Rick Towler on Wed, 12 Nov 2003 18:01:51 GMT
View Forum Message <> Reply to Message

"Neil" wrote...
> "Rick Towler" wrote ...
>> "Neil" wrote...
>>> I would like to read in a DXF file and get access to the numerical
>>> values of the vertices and the connectivity. That's all i wish to do,
>>> which IDL routines should i use for this task. From the IDL help it is
>>> not immediately obvious how this can be done.

>> There are a couple of ways:
>>
>> You can use the IDLffDXF object. There is an example close to what you
are
>> looking for in in the docs under IDLffDXF::Read. Also of interest will
be
>> IDLffDXF::GetEntity which describes how to pull out the different entity
>> types. The types are described in the IDLffDXF::GetContents docs.

> Thank you Rick for you comments. Well i decided on option 1 of yours
> from above.

> This seems to be getting part of the way, the information looks the
> right stuff, but i dont know if all the routine calls are necessary.
>
> However, i do not yet seem to be able to access the real numberical
> values of the "verticies" and "connectivities". I can see that
> tissues.verticies and tissue.connectivity are pointers, but to what?
> How do i get the one dimensional array with all the connectivity and
> the 3 by n array of verticies? Which variable holds the data?

Neil,

You are sooooo close. You probably have already figured this out but try:

```
print, *tissue.connectivity
print, *tissue.vertices
```

As you discovered, the GetEntity method returns a structure which contains 3 pointers. Without really getting into it, pointers in IDL are lightweight tokens that "point to" some data. The pointer variable only contains a reference to that data which is what you have discovered. What you need to do is to dereference your pointer using the "*" character.

Pointers are persistent heap variables. That means that they will exist until you explicitly destroy them or exit IDL. What may not be obvious is that when using the IDLffDXF object the pointers that are returned to you are your responsibility. You must keep track of them and free them using PTR_FREE() when you are finished. Something as simple as the following would work:

```
tissue = oTest -> GetEntity(TestTypes[1])

verts = *tissue.vertices
conn = *tissue.connectivity

PTR_FREE, [tissue.vertices, tissue.connectivity, $
   tissue.vertex_colors]
```

-Rick