Subject: putting data into .kml (google earth) from IDL
Posted by james-a-roo on Wed, 09 Jan 2008 19:48:59 GMT
View Forum Message <> Reply to Message

Hello-

I'm thinking about undertaking some work to be able to take data structures or arrays in IDL and visualize them in google earth. The idea is to build an object orient set of codes which will write the data into .kml files.

Has anyone begun this or thought about it? I'm going over the .kml specs and making some headway, but i clearly dont have then entire picture yet. I have a small goal of getting some (extruded to the ground) polygons working in the very near future.

thanks,

jlm

Subject: Re: putting data into .kml (google earth) from IDL
Posted by mccreigh on Mon, 14 Jan 2008 19:53:51 GMT
View Forum Message <> Reply to Message

(I'm not really sure of the best way to do this so I put all the IDL codes referenced here and a few more into http://atoc.colorado.edu/~mccreigh/KML_obj/jan_14_08 . note i also tarred (jan_14_08.tar.gz) them so you could download as well as browse.)

I decided that I needed to use the XML tools in IDL. So, this is kinda a crash course for myself (and could be for you as well) as I try to develop some tools for writing kml more efficiently in IDL. If you go through this, you may note that things are a bit rough, i have some questions lingering in the code here and there and details have not been fully implemented. But you should be able to make it through the examples OK. (see pentagon2.pro and its output IDL_pentagon2.kml if you want to see how far i got for now).

Since it was easier to first understand how to parse an existing XML document than how to create one, I first wrote a routine which will show the tag structure for *most* (sometimes it wouldnt define the document object, may have been anamoulous behavior and I'm sure there are exceptions) XML documents. This is one of the very few recursive routines which i've ever written, this is a cool example of recursive programming. Try it on any KML file you might have, or just on the

IDL_pentagon.kml files in with the code.

A KML file has a basic structure. The XML tag is refered to as the document object and is the root node (in the IDLffXMLDOM). The firstchild of that root is the KML tag, in which all the action happens. So, i wrote KML_init.pro to set up the XML and KML tags as document object and firstchild node,respectively.

From here, new elements of the document are created and appended to a parent node object (the parent can be the KML tag node object or its children) by the routine kml_new_element.pro. Their TagNames and attributes can be set in this routine. If the node contains data values these can be specified by NodeValue, which only does textnodes for now. This is where all the action happens, just append new nodes to old ones and create your KML!

Finally, you want to write the xml document (which contains your kml) to a file (note that IDL does not save XML objects) and then you destroy that xml object. This is the routine kml_write_destroy.pro.

Since I'm interested in drawing polygons, I thought I'd recreate the google earth example kml of rendering the pentagon:
http://code.google.com/apis/kml/documentation/kml_tut.html#p olygons

I did this using the 3 above routines in the file pentagon.pro. Of course, you ccan see, this takes a tremendous amount of work, probably more than just writing the kml by hand (of course i am not yet looping over multiple polygons...). So the next step was to take the KML object definition to a new level. I did this in kml_polygon.pro . The file pentagon2.pro shows how the pentagon example is written using this new, KML_polygon object procedure. Much more sleek. I've even implemented it so that multiple innerBoundaryIs nodes may be implied by an added dimension in the data. In pentagon2.pro i hacked the inner points to define 2 triagles to remove from the pentagon, instead of an inner pentagon. you can comment or uncomment this to remove the regular, inner pentagon or the 2 triangular inner boundaries.

Next, i was quickly realizing that it is desirable to build routines akin to kml_polygon.pro for to do this with all the KML objects seen in the diagram:

http://code.google.com/apis/kml/documentation/kml_tags_beta1 .html

In particular, in pentagon2.pro, i am still using the rudimentary routines to define the placemark and name nodes (though this wasnt problematic since they are very basic definitions). So it would be nice to have routines to streamline these definitions.

While i was looking at the specs for each KML object, it occured to me that if i could simply parse the specs and create code based on these, then my work is done. While there clearly isnt enough in the specs to generate the level of sophistication in my kml_polygon.pro, i think most of the stuff could be gleaned. Parsing the spec with the kml_show_structure works but I need to return more information, basically one needs a vector of child/parent relations for each node (i started on ths but i'm out of time for now...). At least in the polygon example, nodes with ("only children") chlidren only need to be set at one level, the children look to be cloned for all such nodes. So, determining which children do and do not need to be set is important. Also, gleaning default values from the spec would be nice. In light of the specs being updated, it would be nice to be able to do this in a way which might interface with any routines built on top of what parsing the specs might give you. So that updating the code with new specs means a minimial amount of rewrite. (I also looked into trying to parse the specs from the html document, instead of having to copy them out of my browser. It looked like a nightmare to me, but this would be the most efficient way of doing it. Maybe extracting all the <pre> </pre> tags and their contents from the html and writing the contents to a file. - this would be the prefered, long-term way of doing this.) But this is probably better suited to PERL than IDL programming.

Anyway, I am out of time to work on this for at least another week now. But I thought I'd post this for anyone who is interested. Since I was not at all familiar with the IDLffXMLDOM before this, I would appreciate feedback on if i'm doing this correctly or not. Of course, i would appreciate people building more routines like kml_polygon.pro, but i think the foundation of this still needs to be examined. One thing worth mentioning is that i've come at this entirely from the perspective of writing a KML document via an IDL routine. I dont really care to read or edit existing KML documents, I'm primarily concerned with generating KML from scratch. So, parsing and editing existing KML docuements has not even been considered in what i've done so far.

Cheers.

Subject: Re: putting data into .kml (google earth) from IDL
Posted by Karl[1] on Tue, 15 Jan 2008 00:56:29 GMT
View Forum Message <> Reply to Message

Interesting project. More comments inserted below.

On Jan 14, 1:53 pm, mccre...@gmail.com wrote:
> (I'm not really sure of the best way to do this so I put all the IDL

> codes referenced here and a few more
intohttp://atoc.colorado.edu/~mccreigh/KML_obj/jan_14_08
> . note i also tarred (jan_14_08.tar.gz) them so you could download as
> well as browse.)
>
> I decided that I needed to use the XML tools in IDL. So, this is kinda

snip

> Since it was easier to first understand how to parse an existing XML
> document than how to create one, I first wrote a routine which will
> show the tag structure for *most* (sometimes it wouldnt define the
> document object, may have been anamoulous behavior and I'm sure there
> are exceptions) XML documents. This is one of the very few recursive
> routines which i've ever written, this is a cool example of recursive
> programming. Try it on any KML file you might have, or just on the
> IDL_pentagon.kml files in with the code.
>

I noted a place in your code where you had to skip every other
element. These elements are probably text nodes that contain newline
characters or other whitespace. XML parsers (correctly) insist on
including nodes that contain whitespace, because without a schema, the
parser cannot tell whether or not whitespace is important to the
user. In this case, they are probably just newlines to make the data
file easier to read.

Note that if you call Document::Save without the pretty print option,
you'll probably get a single line file, unless you do have text nodes
with newlines in them.

I think that there are some options on Document::Read to cause IDL to
not create ignorable whitespace nodes. If you need a schema, google
has one for kml on their site.

snip

> I did this using the 3 above routines in the file pentagon.pro. Of
> course, you ccan see, this takes a tremendous amount of work, probably
> more than just writing the kml by hand (of course i am not yet looping

It might seem like it would be easier to just spit out text at this
point, but the XMLDOM library is handling all the syntax issues for
you, and you don't really appreciate that until you really do try it
on your own :-).

And yeah, as things get more complex, there's no doubt that the
library is the way to go.

> over multiple polygons...). So the next step was to take the KML
> object definition to a new level. I did this in kml_polygon.pro . The
> file pentagon2.pro shows how the pentagon example is written using
> this new, KML_polygon object procedure. Much more sleek. I've even
> implemented it so that multiple innerBoundaryIs nodes may be implied
> by an added dimension in the data. In pentagon2.pro i hacked the inner
> points to define 2 triagles to remove from the pentagon, instead of an
> inner pentagon. you can comment or uncomment this to remove the
> regular, inner pentagon or the 2 triangular inner boundaries.
>
> Next, i was quickly realizing that it is desirable to build routines
> akin to kml_polygon.pro for to do this with all the KML objects seen
> in the diagram:
>
> http://code.google.com/apis/kml/documentation/kml_tags_beta1 .html
>
> In particular, in pentagon2.pro, i am still using the rudimentary
> routines to define the placemark and name nodes (though this wasnt
> problematic since they are very basic definitions). So it would be
> nice to have routines to streamline these definitions.
>

I'm still deciding if the following idea is worthwhile or not. You have evolved from a set of "rudimentary" routines to more structured routines like kml_polygon.pro. The next logical step is to implement all this with IDL objects. You would suck your data out from wherever it comes from and build a tree of objects that represents your data.

For example, one object would be a KML polygon object. When you create an instance, it would also create the appropriate XMLDOM objects, setting all the attributes to the defaults, etc. You would have methods to set things like coordinate data. The object would essentially "edit" the XMLDOM objects that you have created to represent the polygon. If someone uses a method to set an inner ring, the object would create the new XMLDOM element objects to represent this data. The object would stitch all this together as well as link with parents and children, etc. It would just be a more organized framework for what you are already doing.

When you've got the entire tree built, call Document::Save to write the KML file out.

> While i was looking at the specs for each KML object, it occured to me
> that if i could simply parse the specs and create code based on these,
> then my work is done. While there clearly isnt enough in the specs to
> generate the level of sophistication in my kml_polygon.pro, i think
> most of the stuff could be gleaned. Parsing the spec with the

> kml_show_structure works but I need to return more information,
> basically one needs a vector of child/parent relations for each node
> (i started on ths but i'm out of time for now...). At least in the
> polygon example, nodes with ("only children") chlidren only need to be
> set at one level, the children look to be cloned for all such nodes.
> So, determining which children do and do not need to be set is
> important. Also, gleaning default values from the spec would be nice.
> In light of the specs being updated, it would be nice to be able to do
> this in a way which might interface with any routines built on top of
> what parsing the specs might give you. So that updating the code with
> new specs means a minimial amount of rewrite. (I also looked into
> trying to parse the specs from the html document, instead of having to
> copy them out of my browser. It looked like a nightmare to me, but
> this would be the most efficient way of doing it. Maybe extracting all
> the <pre> </pre> tags and their contents from the html and writing the
> contents to a file. - this would be the prefered, long-term way of
> doing this.) But this is probably better suited to PERL than IDL
> programming.

Would the schema file give you enough information to do this?
It is a lot easier to parse.

>
> Anyway, I am out of time to work on this for at least another week
> now. But I thought I'd post this for anyone who is interested. Since I
> was not at all familiar with the IDLffXMLDOM before this, I would
> appreciate feedback on if i'm doing this correctly or not. Of course,
> i would appreciate people building more routines like kml_polygon.pro,
> but i think the foundation of this still needs to be examined. One
> thing worth mentioning is that i've come at this entirely from the
> perspective of writing a KML document via an IDL routine. I dont
> really care to read or edit existing KML documents, I'm primarily
> concerned with generating KML from scratch. So, parsing and editing
> existing KML docuements has not even been considered in what i've done
> so far.
>
> Cheers.

One final thought. You might try getting the schema file and use IDL
XMLDOM to read back your KML output file with schema checking turned
on. This would be a way to check your work in that you are putting
the right elements in the right places, etc.

Karl