Subject: GDAL/OGR library 'import' now in sandbox
Posted by mirek on Mon, 23 Nov 2015 20:39:28 GMT
View Forum Message <> Reply to Message

svn://www.ultimatepp.org/upp/sandbox/se

Subject: Re: GDAL/OGR library 'import' now in sandbox
Posted by mirek on Mon, 07 Dec 2015 13:37:41 GMT
View Forum Message <> Reply to Message

Moved to bazaar/plugin. Light encaplulation added (practically copied form plugin/geotiff)

Subject: Re: GDAL/OGR library 'import' now in sandbox
Posted by Tom1 on Mon, 07 Dec 2015 20:48:18 GMT
View Forum Message <> Reply to Message

Hi Mirek,

I can't just now read the source, but just out of curiosity, how did you handle the TIFF dependency which gets satisfied twice (once in gdal/geotiff and second in plugin/tiff)? This is actually one thing that I could not solve immediately in a clean way... (The linker complained about double definitions of TIFF functions when I tried the sandbox version briefly.)

Best regards,

Tom

Subject: Re: GDAL/OGR library 'import' now in sandbox
Posted by Tom1 on Wed, 09 Dec 2015 11:55:09 GMT
View Forum Message <> Reply to Message

Hi Mirek,

After tinkering around for a while, I managed to get both GDAL and OGR side working beautifully for all the drivers I needed. The key here was to call the appropriate driver registration functions separately instead of relying on the default GDALAllRegister(), which I guess remains unconfigured.

Thank you very much for this excellent addition to bazaar!

Best regards,

Tom

Subject: Re: GDAL/OGR library 'import' now in sandbox
Posted by mirek on Wed, 09 Dec 2015 15:29:41 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 07 December 2015 21:48Hi Mirek,

I can't just now read the source, but just out of curiosity, how did you handle the TIFF dependency which gets satisfied twice (once in gdal/geotiff and second in plugin/tiff)? This is actually one thing that I could not solve immediately in a clean way... (The linker complained about double definitions of TIFF functions when I tried the sandbox version briefly.)

Best regards,

Tom

Well, I have solved in unclean way :) A lot #defines to rename gdal tiff routines...

Subject: Re: GDAL/OGR library 'import' now in sandbox
Posted by mirek on Wed, 09 Dec 2015 15:32:48 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 09 December 2015 12:55Hi Mirek,

After tinkering around for a while, I managed to get both GDAL and OGR side working beautifully for all the drivers I needed. The key here was to call the appropriate driver registration functions separately instead of relying on the default GDALAllRegister(), which I guess remains unconfigured.

Thank you very much for this excellent addition to bazaar!

Best regards,

Tom

Well, there is some light encapsulation provided, which should call all registres on Open.

Check bool Gdal::Open(const char *fn)

I would like to know whether encapsulation provided is usable...

Mirek

Subject: Re: GDAL/OGR library 'import' now in sandbox
Posted by Tom1 on Thu, 10 Dec 2015 14:10:51 GMT
View Forum Message <> Reply to Message

Hi Mirek,

I did not try the encapsulation, since my code (since 2008) references GDAL directly. I just replaced the external GDAL library with the Upp GDAL plugin to get rid of the bloated DLL and to ease compilation of GDAL. (To be more specific, the executable grew by about 2 MB while I was able to drop the 7 MB DLL. :)

--

It seems the encapsulation is for the raster part, while I also use the vector part (OGR). When I get some time, I could look at the raster encapsulation and test how it works.

The vector part (OGR) obviously needs the corresponding driver registrations. Encapsulation for OGR would be quite different.

Best regards,

Tom

Subject: Re: GDAL/OGR library 'import' now in sandbox
Posted by Tom1 on Mon, 21 Dec 2015 15:01:24 GMT
View Forum Message <> Reply to Message

Hi Mirek,

I have initially tried to connect to your GDAL encapsulation for reading raster maps.

I'm missing at least access to:

```
Band::GetColorTable();
Band::GetColorTable()->GetColorEntryCount();
Band::GetColorTable()->GetColorEntry();
Band::GetColorTable()->GetPaletteInterpretation();
```

and also a way to read the data block directly in the native buffer format. Some rasters are really big and in one byte per pixel format. If these are readily converted to int or double, it is easy to run out of memory on 32 bit systems.

--

Before we proceed with this any further, may I ask why the encapsulation? While rewriting my calls to match Gdal:: instead of original GDAL interface, I found similar complexity but with partially unfamiliar interface. I could better understand this encapsulation if we had e.g. a GeoImage:: class and even a TiledGeoImage:: class, which were automatically loaded from raster files using Gdal:: and representing the raster content as an Image:: that has georeferencing data directly available. (Of course the 32 bits per pixel RGBA data storage of Image makes this less

than optimal in memory for large files, but a very nice solution in principle.) Maybe storing the raster data in original pixel depth plus a palette in a GeoImage:: and then adding Image generation for selected area with desired transform would be a better solution.

Best regards,

Tom
