Subject: Re: Bug in IDLgrPolygon ?
Posted by David Fanning on Thu, 12 Dec 2002 17:55:13 GMT
View Forum Message <> Reply to Message

Thomas Gutzler (tgutzler@ee.uwa.edu.au) writes:

> If I just leave the last (or any other) line away there's no error and
> the result is, what I expeced, an open cube. But this can't be the
> solution, because I want to interactively permit polygons being drawn.

I don't know. That looks like the solution to me.

> My question is: Is this a bug ?

Looks more like bad documentation to me. I wouldn't
have expected it to work the way it is described.

> or: what did I wrong ?

I don't think you did anything wrong. I just think you
need to do more coding to discover a way to build the
proper poly_array on the fly according to user specifications.

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Bug in IDLgrPolygon ?
Posted by Karl Schultz on Thu, 12 Dec 2002 19:19:37 GMT
View Forum Message <> Reply to Message

"Thomas Gutzler" <tgutzler@ee.uwa.edu.au> wrote in message
news:3DF86288.D1D90638@ee.uwa.edu.au...
> Hi,
>
> I just detected some weird behaviour in IDLgrPolygon. I generated an
> Object with this data:
> arr = Transpose([[-1, 1,-1, 1,-1, 1,-1, 1], $
> [-1,-1,-1,-1, 1, 1, 1, 1], $
> [-1,-1, 1, 1,-1,-1, 1, 1]])

> ;vertex: 1 2 3 4 5 6 7 8
> arr_colors = Transpose([[255,255, 0, 0,255,127, 0,127], $
>   [190, 0, 65,255, 0, 0,255,255], $
>   [0,190,255, 65, 0,255,255, 0]])
> arr_polys = [[4, 0, 1, 3, 2], $; vertices 1243
>   [4, 0, 2, 6, 4], $; vertices 1375
>   [4, 2, 3, 7, 6], $; vertices 3487
>   [4, 1, 3, 7, 5], $; vertices 2486
>   [4, 0, 1, 5, 4], $; vertices 1265
>   [4, 4, 5, 7, 6]]   ; vertices 5687
> poly = OBJ_NEW('IDLgrPolygon', arr, polygons=arr_polys,
> vert_color=arr_colors, SHADING=1)
>
> which is a colored cube.
>
> The onlinehelp to IDLgrPolygon sais:
>
> POLYGONS (Get, Set)
> [...] To ignore an entry in the POLYGONS array, set the vertex count,
> n, to 0.
>
> Fine - I set Element 25 (the first of the last line) from 4 to 0 an
> looked forward to see an "open cube" beacuse the last polygon shouldn't
> be drawn.
> What I saw was an open cube ... and an error:
>
> % OBJ_NEW: Error, invalid connectivity list detected (invalid final
> polygon).
>
> If I just leave the last (or any other) line away there's no error and
> the result is, what I expeced, an open cube. But this can't be the
> solution, because I want to interactively permit polygons being drawn.
>
> My question is: Is this a bug ?

The docs are misleading.

> or: what did I wrong ?

What you want to do is replace:

```
             [4, 4, 5, 7, 6]]   ; vertices 5687
```

with:

```
             [0, 0, 0, 0, 0]]   ; vertices 5687
```

If you try:

[0, 4, 5, 7, 6]]   ; vertices 5687

then IDL skips over the 0 and thinks that the 4 is the vertex count for the next polygon.
If you replace the just count with 0, there is no way for IDL to know how far to skip to get to the next polygon. The 0 in the POLYGONS list is a special value, which really just means "skip over me". It is really only useful when wanting to rub out a complete [n,v0,v1,...,v(n-1)] polygon definition without moving data around. -1 is the other special value which means "stop here, even if there is more data after me" and really only makes sense to put in a list after a polygon definition.

The docs should say:

To ignore an entry in the POLYGONS array, set the entry to 0. (and maybe some of the discussion above)

Karl

Subject: Re: Bug in IDLgrPolygon ?
Posted by Thomas Gutzler on Fri, 13 Dec 2002 02:36:46 GMT
View Forum Message <> Reply to Message

Karl Schultz wrote:
>
> What you want to do is replace:
>
> [4, 4, 5, 7, 6]]   ; vertices 5687
>
> with:
>
> [0, 0, 0, 0, 0]]   ; vertices 5687
>
> If you try:
>
> [0, 4, 5, 7, 6]]   ; vertices 5687
>
> then IDL skips over the 0 and thinks that the 4 is the vertex count for the
> next polygon.

Yeah, I should have noticed this. doh!

> -1 is the other special value which
> means "stop here, even if there is more data after me" and really only makes
> sense to put in a list after a polygon definition.

Hm, this was my last idea. Just move the 5 elements describing the
polygon I want not be shown to the end and put a -1 in front. But I
thought thats bad programming style, isn't it ?

Thomas

Subject: Re: Bug in IDLgrPolygon ?
Posted by Karl Schultz on Fri, 13 Dec 2002 14:30:49 GMT
View Forum Message <> Reply to Message

"Thomas Gutzler" <tgutzler@ee.uwa.edu.au> wrote in message
news:3DF947BE.F454E7E@ee.uwa.edu.au...
> Karl Schultz wrote:
>>
>> What you want to do is replace:
>>
>> [4, 4, 5, 7, 6]] ; vertices 5687
>>
>> with:
>>
>> [0, 0, 0, 0, 0]] ; vertices 5687
>>
>> If you try:
>>
>> [0, 4, 5, 7, 6]] ; vertices 5687
>>
>> then IDL skips over the 0 and thinks that the 4 is the vertex count for
the
>> next polygon.
>
> Yeah, I should have noticed this. doh!
>
>> -1 is the other special value which
>> means "stop here, even if there is more data after me" and really only
makes
>> sense to put in a list after a polygon definition.
>
> Hm, this was my last idea. Just move the 5 elements describing the
> polygon I want not be shown to the end and put a -1 in front. But I
> thought thats bad programming style, isn't it ?

I'm not sure about style, but appending the "disabled" polygon and the -1 to
the end of your polygon list is going to cause an alloc/free and a copy of
the entire list. This might be bad if the list is really long. If you
needed to put the disabled polygon back, you would also have to store
someplace the position of the disabled polygon in the main list. If you

accomplished this by also adding this index to your list after the -1, then you're starting to store odd things in this list, which might make it confusing to someone reading the code.

Another way to reenable the polygon would be to just change the -1 to a 0. So you could disable a polygon by copying -1 and that polygon to the end, and set the original polygon indices to 0. Change the -1 to 0 to turn the polygon back on. After you do this a few times, you'll have a bunch of 0's in the list that you can clean up with a periodic call to MESH_VALIDATE. This approach would work best for short lists and if you disabled only one polygon at a time and/or were OK with reenabling all of them at once.

If it were me, I'd think about just copying the disabled polygon out to another data structure, along with its starting index. Then, I zap the poly in the original list to zero. I copy the polygon back to the same place to reenable it.

Hopefully one of these approaches will work well for your application.

I don't think that the polygon list was designed to get very sophisticated with editing polygon lists like we're discussing. I think that the '0' feature is there to allow the easy disabling of polygons that contain vertices with bad data or NaN's. The -1 is used by some of the MESH_* functions to indicate the the result is empty - there are no polygons in the function's result.

One cool thing I thought of is that a negative value for a vertex count would mean skip over the (positive) number of vertices. That is, a -3 in a list would mean skip the next three elements. This would let you skip a polygon by just changing the sign in the count. -1 would still mean "end of list", since a polygon with 1 vertex isn't valid anyway. I don't know what the utility/confusion ratio would be here...

HTH,
Karl

Subject: Re: Bug in IDLgrPolygon ?
Posted by Thomas Gutzler on Mon, 16 Dec 2002 03:35:07 GMT
View Forum Message <> Reply to Message

Karl Schultz wrote:
>
> "Thomas Gutzler" <tgutzler@ee.uwa.edu.au> wrote in message
> news:3DF947BE.F454E7E@ee.uwa.edu.au...
>>
>> [..] Just move the 5 elements describing the
>> polygon I want not be shown to the end and put a -1 in front. But I

>> thought thats bad programming style, isn't it ?
>
> I'm not sure about style, but appending the "disabled" polygon and the -1 to
> the end of your polygon list is going to cause an alloc/free and a copy of
> the entire list.

Argh, you're right. I forgot that moving the elements to the end is fine
but they cannot be reenabled.

> If it were me, I'd think about just copying the disabled polygon out to
> another data structure, along with its starting index. Then, I zap the poly
> in the original list to zero. I copy the polygon back to the same place to
> reenable it.

I'm going to do this.

> One cool thing I thought of is that a negative value for a vertex count
> would mean skip over the (positive) number of vertices. That is, a -3 in a
> list would mean skip the next three elements. This would let you skip a
> polygon by just changing the sign in the count. -1 would still mean "end of
> list", since a polygon with 1 vertex isn't valid anyway. I don't know what
> the utility/confusion ratio would be here...

nice toy :)

thanks,
Tom