Subject: Re: 3d device coordinates from a 3D polyline....
Posted by Karl[1] on Tue, 02 Oct 2012 21:02:26 GMT
View Forum Message <> Reply to Message

On Tuesday, October 2, 2012 10:46:16 AM UTC-6, (unknown) wrote:
> On Tuesday, October 2, 2012 3:22:12 AM UTC-6, alx wrote:
>
>> Le lundi 1 octobre 2012 19:07:22 UTC+2, (inconnu) a écrit :
>
>>
>
>>> On Monday, October 1, 2012 6:32:00 AM UTC-6, David Fanning wrote:
>
>>
>
>>>
>
>>
>
>>>> > I have am idlgrpolyline which I can rotate in a 3D view (with the trackball).
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> > I want to know the 2D coordinates of this line in the device (ie,the 2D the projection in the window). Can't figure it out.
>
>>
>
>>>
>
>>
>
>>>>
>
>>

>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> I'm no expert in this area, but I think the 3D to 2D
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> conversions of the transformation matrix (which you
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>

>>>
>
>>
>
>>>> can recover from the trackball) are well known. You
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> can read the answer at the bottom of this article,
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> for example:
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>

>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> http://math.stackexchange.com/questions/336/why-are-3d-
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> transformation-matrices-4x4-instead-of-3x3
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>

>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> Cheers,
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>

>>>
>
>>
>
>>>> David
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>

>>>
>
>>
>
>>>> --
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> David Fanning, Ph.D.
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> Fanning Software Consulting, Inc.
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>

>>>
>
>>
>
>>>> Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> Sepore ma de ni thui. ("Perhaps thou speakest truth.")
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> Hmm,
>
>>
>

>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> So there is nothing in the object graphics system like the 'CONVERT_COORD' routine ?
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> George.
>
>>
>
>>
>
>>
>
>> In Object Graphics and New Graphics you can use "[XYZ]COORD_CONV" and
>
>>
>
>> "ConvertCoord" methods, respectively. The last one being quite similar to the
"Convert_Coord" function in Direct Graphics.
>
>>
>
>> Alain.
>
>

>
> Alain,
>
>
>
> Thanks for your help.....I'm not using new graphics - this is all object graphics. I've been trying to understand how [XYZ]COORD_CONV in object graphics relates to all of this - but it's somewhat confusing.
>
> I understand how XCOORD_CONV and YCOORD_CONV are used to map a 2D line to 2D normalized space - but I'm wanting the same for 3D...
>
>
>
> Cheers
>
>
>
> George.

[XYZ]COORD_CONV is a PROPERTY, not a method, for many IDLGr* Object Graphics classes. It is used to apply a transform to the raw vertex data stored in the object instance as the first part of the overall object-to-window transform that IDL performs when drawing the scene.

This is a simple scale and translate transform, so the values for all three ([XYZ]COORD_CONV) can be put into a 4x4 matrix and used to multiply all your 3D points stored in the object.

You'll also have to multiply the points by each 4x4 transform matrix stored in the TRANSFORM property in each IDLgrModel in your scene graph, working your way up to the view.

Then you need to apply a view transform using some of the properties (like VIEWPLANE_RECT) from the IDLgrView. Finally apply a view-to-window transform to get your 2D device coordinates.

Figuring out the last two are perhaps the trickiest, especially if the view projection is perspective. But it is possible.

It should also be possible to write a general-purpose function that takes a "leaf" graphics object and walks up the scene graph, computing the single 4x4 combined matrix and returns it. You would then use that single matrix to transform your points.

In a way, you are duplicating the entire transform that IDL applies to the points via the underlying graphics system (OpenGL). I don't remember if there is a way to get this transform directly from IDL - don't think so. And someone out there may have already written an IDL function to do this. But, I don't know of any.

Karl