Subject: Re: Displaying 3-D vector fields
Posted by Rick Towler on Fri, 08 Nov 2002 19:31:12 GMT
View Forum Message <> Reply to Message

"Jim" <jim.blackwell@gsfc.nasa.gov> wrote
> "Rick Towler" <rtowler@u.washington.edu>
>> This sounds like a job for object graphics.
>>
>> Someone has to have written a vector object which consists of a few
>> polylines that make up the body and head in a model. Use would be as
simple
>> as defining the location and magnitude.
>>
>> Once you have that, something as simple as this would work:
>>
>> ; Your vector locations - XYZ (empty array used as example)
>> location=FLTARR(100,3)
>>
>> ; Your vector magnitudes - ABC (empty array used as example)
>> magnitude=FLTARR(100,3)
>>
>> ; Create a model to put all of our vectors in
>> model = OBJ_NEW('IDLgrModel')
>>
>> ; Fill it up with vector objects
>> vectors = OBJARR(100)
>> for n=0, 99 do $
>> vectors[n] = OBJ_NEW('vector', LOCATION=location[n,*], $
>> MAGNITUDE=magnitude[n,*])
>>
>> ; Add the array of vectors to our model
>> model -> Add, vectors
>>
>> ; Display the contents of the model using xobjview
>> xobjview, model, /BLOCK
>>
>> ; Destroy the objects
>> OBJ_DESTROY, model
>>
>>
>> If you want to animate the vectors you'll have to do a little more work
but
>> it would be simple.
>>
>>
>> The trick is finding the "vector" object. Someone on this list has to
have
>> written something similar. I was giving this a day hoping someone with

such
>> an object would step up... Try searching the usual code archives. I
>> thought Mark Hadfield had something like this but his webpage isn't up
>> anymore.
>>
>> If you want to try and write the vector object yourself left me know and
I
>> can help get you started.
>>
>> -Rick
>

> Thanks for the advice. As far as a vector object, I presume one could
> take the program offered in another reply to this posting and make it
> an object ? Not being familiar with Object Graphics other than for
> some examples I've tried to figure out, I need some help here.

Well let me introduce you to the wonderful world of Object graphics. :)
Actually, let Ronn Kling do that with his book "Power Graphics with IDL".
You can get it from his website (www.kilvarock.com). You'll need it if you
want to go beyond the basics I outlined above.

I saw your other post too. I haven't looked at show_stream.pro so I can't
help you there. What I can do is provide you with a vector object. I just
whipped this up because I was trying to avoid other work so test it a bit
first to verify it does what it should. There are no guarantees...

Let me know how you make out.

-Rick

```
;+
; NAME:
;       VECTOR__DEFINE
;
; PURPOSE:
;
;       This is an example of a 3D vector class for plotting
;       vector fields.  This object is a subclass of IDLgrModel
;       which contains a polyline object representing a vector
;       provided a given location and magnitude.
;
; AUTHOR:
;       Rick Towler
;       School of Aquatic and Fishery Sciences
;       University of Washington
```

```
;       Box 355020
;       Seattle, WA 98195-5020
;       rtowler@u.washington.edu
;       www.acoustics.washington.edu
;
;
; CATEGORY: Object Graphics
;
;
; CALLING SEQUENCE:
;
;       vectorObject = OBJ_NEW('vector')
;
;
; KEYWORDS:
;
;   This object inherits keywords from it's superclass, IDLgrModel, and
;   passes keywords to IDLgrPolyline.
;
;       location:   A 3 element vector defining the X, Y and Z
;                   coordinates of the vector's location.
;
;       magnitude:  A 3 element vector defining the X, Y and Z
;                   magnitude of the vector.
;
;
; METHODS:
;
;       GetProperty:
;
;       SetProperty:
;
;
; DEPENDENCIES: None.
;
; EXAMPLE:
;
;       vecObj = OBJ_NEW('vector', LOCATION=[0,0,0], MAGNITUDE=[3,2,1], $
;          COLOR=[255,0,0], THICK=2.0)
;
;       xobjview, vecObj
;
;
; MODIFICATION HISTORY:
;       Written by: Rick Towler, 8 November 2002.
;
;-
```

```
function Vector::Init,  location=location, $
                        magnitude=magnitude, $
                        _ref_extra=extra


    ;  Check the keywords.
    self.location = (N_ELEMENTS(location) eq 0) ? [0,0,0] : location
    self.magnitude = (N_ELEMENTS(magnitude) eq 0) ? [0,0,-1] : magnitude

    ;  Initialize the superclass.
    ok = self->IDLgrModel::init(/SELECT_TARGET, _EXTRA=extra)
    if (not ok) then return, 0

    ;  Define the unit vector vertices.
    vertices = [[-0.1,0.0,-0.85], $
                [0.0,0.0,-1.0], $
                [0.1,0.0,-0.85], $
                [0,0,0]]

    ;  Connect the dots to form our vector
    polylines = [3,0,1,2,2,1,3]

    ;  Create the vector body
    self.oBody = OBJ_NEW('IDLgrPolyline', vertices, POLYLINES=polylines, $
        _EXTRA=extra)

    ;  Add the polyline to self.
    self -> Add, self.oBody

    ;  "Update" the vector to orient/translate/scale it correctly.
    self -> Update

    RETURN, 1

end


pro Vector::Update

    compile_opt idl2

    ;  Reset our transform.
    self -> Reset

    ;  Rotate the vector.
    lvn = TOTAL(self.magnitude^2)
    if (lvn eq 0.) then begin
```

```
      ;  Hide the vector if magnitude=0
      self -> SetProperty, /HIDE
      RETURN
   endif
   self -> SetProperty, HIDE=0
   lMag = SQRT(lvn)
   lvector = self.magnitude / lMag

   yaw = 180. + ATAN(lvector[0],lvector[2]) * !RADEG
   pitch = ATAN(lvector[1], SQRT(lvector[2]^2 + lvector[0]^2)) * !RADEG

   self -> Rotate, [1,0,0], pitch
   self -> Rotate, [0,1,0], yaw

   ;  Scale according to magnitude
   self -> Scale, lMag, lMag, lMag

   ;  Move the vector into place.
   self -> Translate, self.location[0], self.location[1], $
      self.location[2]

   RETURN

end


pro Vector::SetProperty,    location=location, $
                     magnitude=magnitude, $
                     _extra=extra

   compile_opt idl2

   update = 0B

   if (N_ELEMENTS(location) eq 3) then begin
      self.location = location
      update = 1B
   endif

   if (N_ELEMENTS(magnitude) eq 3) then begin
      self.magnitude = magnitude
      update = 1B
   endif

   if (update) then self -> Update

   self->IDLgrModel::SetProperty, _EXTRA=extra
   self.oBody->SetProperty, _EXTRA=extra
```

```
end


pro Vector::GetProperty,    location=location, $
                        magnitude=magnitude, $
                        _ref_extra=extra

   compile_opt idl2

   location = self.location
   magnitude = self.magnitude

   self->IDLgrModel::GetProperty, _EXTRA=extra
   self.oBody->GetProperty, _EXTRA=extra

end


pro Vector::Cleanup

   compile_opt idl2

   OBJ_DESTROY, self.oBody

   ;  Call our parents cleanup method
   self->IDLgrModel::Cleanup

end


pro Vector__Define

   struct={Vector, $
         inherits IDLgrModel, $
         oBody:OBJ_NEW(), $

         location:FLTARR(3), $
         magnitude:FLTARR(3) $
         }

end
```