Subject: Re: arbitrary rotation of 3-d arrays
Posted by steinhh on Fri, 11 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

> has anybody out there in idl-land written or seen code to apply arbitrary
> rotations to 3-d arrays???

Although others have posted solutions using the t3d style
rotations, you might want to look at the procedures below.
I don't really like the idea of using a (global) system
variable designed for 3d *graphics* as "the" temporary
variable to accumulate all kinds of 3d manipulations...

I'm sorry for the complete lack of documentation ... this
was something I did to experiment myself towards an understanding
of such rotations.. ROT3DMATRIX returns an array that to
be applied like this

```
ROTATED_XYZ = ROT3DMATRIX([alphax,alphay,alphaz]) ## XYZ_ARRAY
```

(Note that [alphax,alphay,alphaz] is wrt. a fixed coordinate
system, the axes don't move after each partial rotation...
this may be different from the way t3d applies its input..)

Also, I wrote a test application (ROTATE_SCREW) that uses
direct graphics to manipulate 3D objects on screen with the
help of a trackball object.

Try clicking either left *and* middle buttons to rotate the
box, and the (two!) corkscrews inside the box. Double-clicking
the middle button changes the "sense" of how the "loose"
corkscrew is rotated - wrt the *box* coordinate system or wrt
the *screen* (sort of) coordinate system.... You may get the
difference if you e.g. turn the box 180 degrees around and
try manipulating the loose corkscrew...

Ok, here goes,

Stein Vidar

```
;;
;; Return rotation matrix such that
;; rot3dmatrix([alphax,alphay,alphaz]) ## [X,Y,Z] gives your [X,Y,Z]
;; rotated alphax radians about the x axis, then alphay radians about
;; the y axis, and finally alphaz radians about the z axis. Note that
;; the axes are kept fixed. This allows an inverse operation to to
```

```
;; return alphax, alphay, alphaz from a given rotation matrix.

FUNCTION rot3dmatrix_angles,m

  m = double(m)

;  From mathematica:
;
;  mm = [[Cy Cz, Cz Sx Sy - Cx Sz, Cx Cz Sy + Sx Sz   ], $
;        [Cy Sz, Cx Cz + Sx Sy Sz, -(Cz Sx) + Cx Sy Sz],$
;        [-Sy  , Sx Cy           ,     Cx Cy         ]]


  cosyzero = (m(1,2) EQ 0 AND m(2,2) EQ 0)

  IF NOT cosyzero THEN BEGIN

     cys = 1

     xfind = atan(cys*m(1,2),cys*m(2,2)) ;; May be wrong quadrants!
     zfind = atan(cys*m(0,1),cys*m(0,0))

     sinx = sin(xfind) & sxgood = (abs(sinx) GT 0.05)
     cosx = cos(xfind) & cxgood = (abs(cosx) GT 0.05)

     wt = double(sxgood+cxgood)
     cosy = ((sxgood ? m(1,2)/sinx : 0)+(cxgood ? m(2,2)/cosx : 0))/wt

     yfind = atan(-m(0,2),cosy)

     return,[xfind,yfind,zfind]
  END

  ;; Cos[y] == 0 => yfind = +/- Pi/2

  yfind = atan(-m(0,2),0)

  ;; From mathematica we find that

  ;; MatrixForm[TrigFactor[r[x,+/-Pi/2,z]]]
  ;;
  ;; = [[ 0,  +/-Sin[x-z], +/-Cos[x+z]  ],$
  ;;    [ 0,     Cos[x-z],    -Sin[x-z] ],$
  ;;    [ -/+1,         0,           0  ]]

  ;; We thus arbitrarily set z = 0 and get:

  zfind = 0
```

```
  xfind = atan(-m(2,1),m(1,1))

  return,[xfind,yfind,zfind]
END


FUNCTION rot3dmatrix,alpha,inverse=inverse

  IF keyword_set(inverse) THEN return,rot3dmatrix_angles(alpha)

  alpha = double(alpha)
  ca = cos(alpha)
  sa = sin(alpha)

  mx = [[     1,     0,      0],$
        [     0, ca(0), -sa(0)],$
        [     0, sa(0),  ca(0)]]

  my = [[ ca(1),     0,  sa(1)],$
        [     0,     1,      0],$
        [-sa(1),     0,  ca(1)]]

  mz = [[ ca(2),-sa(2),      0],$
        [ sa(2), ca(2),      0],$
        [     0,     0,      1]]

  return,mz ## (my ## mx)
END

 ;------------------------------------------------------------ -


PRO plotcube,xr,yr,zr

  xi = [0,1,1,0,0]
  yi = [0,0,1,1,0]
  zi = [0,0,0,0,0]

  plots,xr(xi),yr(yi),zr(zi),/t3d,/data
  plots,xr(xi),yr(yi),zr(zi+1),/t3d,/data
  FOR i=0,4 DO plots,xr(xi([i,i])),yr(yi([i,i])),zr([0,1]),/t3d,/data

END


PRO rotate_screw,rotation

  ;; Create widget draw window
```

```
xs = (ys=512) ;; Size of draw window

id = widget_base()
dummy = widget_draw(id,xsize=xs,ysize=ys,/button_ev,/motion)
widget_control,id,/realize
widget_control,dummy,get_value=win
wset,win

xrange = [(xmin=-10), (xmax=10)]
yrange = [(ymin=-10), (ymax=10)]
zrange = [(zmin=-10), (zmax=10)]

!x.s = [-xmin,1.0]/(xmax-xmin)
!y.s = [-ymin,1.0]/(ymax-ymin)
!z.s = [-zmin,1.0]/(zmax-zmin)

theta = findgen(120)/199.0*2*!PI
x = cos(20*theta)
y = sin(20*theta)
z = 5*theta

xyz = [[x],[y],[z]]
t3d,/reset,translate=-[.5,.5,.5] & xyzform = !P.t

;; X/Y/Z "axis" vectors for easy drawing

xa = 8*[[0,1],[0,0],[0,0]]
ya = 8*[[0,0],[0,1],[0,0]]
za = 8*[[0,0],[0,0],[0,1]]

h = [.5,.5,.5]
persp = 5
scale = .6

;; Build the initial viewing matrix

t3d,/reset,trans=-h
t3d,scale=.6*[1,1,1]
IF n_elements(rotation) EQ 3 THEN BEGIN
  t3d,rotate=rotation*!radeg
  print,"Rotation"
END

t3d,trans=h

track = obj_new('trackball',[xs/2.,ys/2.0],0.25*xs)
track2 = obj_new('trackball',[xs/2.,ys/2.0],0.25*xs)
```

```
inspace = 1

REPEAT BEGIN
  t = !P.t
  t3d,trans=-h
  angles = rot3dmatrix(!p.t(0:2,0:2),/inverse)
  t3d,perspect=persp
  t3d,trans=h

  erase

  plotcube,xrange,yrange,zrange

  plots,transpose(xa),/t3d,/data
  plots,transpose(ya),/t3d,/data,color=140
  plots,transpose(za),/t3d,/data,color=100

  xyouts,xa(1,0),xa(1,1),z=xa(1,2),"X",/t3d,/data
  xyouts,ya(1,0),ya(1,1),z=ya(1,2),"Y",/t3d,/data
  xyouts,za(1,0),za(1,1),z=za(1,2),"Z",/t3d,/data

  plots,x,y,z,/t3d,/data
  plots,transpose(xyz),/t3d,/data

  xyouts,.1,.15,string(angles(0))+"!c"+string(angles(1))+$
    "!c"+string(angles(2))+"!c"+"insp:"+string(inspace),/normal

  !P.t = t
  empty
  ev = widget_event(id)
  IF ev.press EQ 2 AND ev.clicks EQ 2 THEN inspace = (inspace + 1) MOD 3

  xformq = track->update(ev,transform=xform,mouse=1b)
  IF xformq THEN BEGIN
    t3d,translate=-h
    !P.t = xform ## !p.t
    t3d,translate=h
  END

  xformq = track2->update(ev,transform=xform,mouse=2b)
  IF xformq THEN BEGIN
    IF inspace EQ 1 THEN BEGIN

      ;; Now - rotate/shift the screw into the orientation it has on the
      ;; screen, then apply trackball rotation, then rotate/shift back
      ;; into it's own space.
```

```
      xform = invert(!p.t) ## xform ## !p.t
    END ELSE IF inspace EQ 2 THEN BEGIN
      ;; Rotate the screw "in its own data space".
      ;; Apply inverse transform on xyz,
      ;;
      xform = xyzform ## xform ## invert(xyzform)
    END

    ;; Apply the resulting transform on the screw. The xform really
    ;; includes a shift (since we placed it centered on the screen, not on
    ;; the coordinate axes), but this is not taken into account since
    ;; we're using only (0:2,0:2) of the resulting transform

    xyz = xform(0:2,0:2) ## xyz

    ;; We need to keep track of the transform that has been applied
    ;; in order to do transformations in
    xyzform = xform ## xyzform
  END
END UNTIL ev.press EQ 4

widget_control,id

rotation = angles
END
```