Subject: help with transformations and scaling
Posted by Richard Tyc on Fri, 25 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

IDL Experts,

I have been busting my head over this one for some time now. I have been getting help from RSI tech support (BIG thanks to Jim Howell! - I know your busy) but I thought I'd throw it out here and see if anyone has solved it before ?

I will pose the question with a simple example. I volume render a 3-D cube, with dimensions [20,20,20]. I draw a line from the origin to a point P (xc,yc,zc). I then create a cutting plane through the cube perpendicular to the line. I then want to draw a local xy coord system on the cutting plane centered at the point P. The problem arises when the dimensions of the cube are not equal which effectively sets !x.s, !y.s and !z.s to be different. It seems that the transformation angles used do NOT reflect this change properly (IDL stretching th Z-axis filling the screen) yet the cutting plane normal (which is used to define the transformation angles) IS dependent on the scaling in each direction:

I have included a sample program showing a X (blue) and Y (Green) local axis on the cutting plane. When DataDim = [20,20,20] everything looks fine. Try setting
DataDim[20,20,10] - The blue Xaxis cuts through the volume indicting it does not lie parallel to the plane.

If anyone has any ideas , I would appreciate any help. I have some experience with Transformation matrices from my robotic/kinematic background but a am an IDL newbie!

Thanks in Advance

Richard Tyc
Project Engineer
St. Boniface Hospital Research Center
richt@sbrc.umanitoba.ca

Here's some snippet of code illustrating the problem (note i have left some code out to simplify - see example attached):

DataDim = [20,20,20]
xfac = 0.5 & yfac = 0.5 & zfac = 0.4

```
; define Point P
xc = xfac*DataDim[0] & yc = yfac*DataDim[1] & zc = zfac*DataDim[2]

; create data representing cube - it will become cube[20,20,20] array
cube = CubeData(DataDim[0],DataDim[1],DataDim[2])
s=size(cube) & xs=s[1]-1 & ys=s[2]-1 & zs=s[3]-1

;create cutting plane

LenVox = sqrt((xc*xs/DataDim[0])^2 + (yc*ys/DataDim[1])^2 + $
              (zc*zs/DataDim[2])^2)

cutplnVox = [(xc*xs/DataDim[0])/LenVox, (yc*ys/DataDim[1])/LenVox, $
            (zc*zs/DataDim[2])/LenVox, -LenVox ]

cpNorm     = cutplnVox[0:2]  ; normal vector for line from origin to P

; normalized distance to point P
cpNormCtr  = [xc/s[1],yc/s[2],zc/s[3]]

; setup scaling
Scale3, XRange=[0,xs], YRange=[0,ys], ZRange=[0,zs]
.................
.................
; create volume
oVolCube  = OBJ_NEW('IDLgrVolume', cube, $
        xcoord_conv=!x.s,ycoord_conv=!y.s, zcoord_conv=!z.s )
oVolCube->setproperty,cutting_planes=[cutplnVox]
oVolCube->setproperty,/zbuffer
oCannula = OBJ_NEW('IDLgrPolyLine', color=[0,255,0], $
    [[0,0,0],[cpNormCtr]], thick=5)

; create local coord frame axis this will be transformed
oXCaxis = OBJ_NEW('IDLgrAxis',0,color=[0,0,255],range=[0,CanAxLen], $
         major=0,minor=0,xcoord_conv=!x.s,ycoord_conv=!y.s, $
        zcoord_conv=!z.s,/notext)
oYCaxis = OBJ_NEW('IDLgrAxis',1,color=[0,255,0],range=[0,CanAxLen], $
         major=0,minor=0,xcoord_conv=!x.s,ycoord_conv=!y.s, $
        zcoord_conv=!z.s,/notext)
oModel->Add,oVolCube
oModel->Add,oCannula

oModelC->Add,oXCaxis
oModelC->Add,oYCaxis

; reset global view  (cube data)  to fit and display
t3d,/reset
t3d,trans=[-.5,-.5,-.5],scale=[.577,.577,.577]
```

```
t3d,rot=[-90, 300, 0]
t3d,rot=[ 220, 0, 0 ]
t3d,trans=[ .5, .5, .5]

oModel->SetProperty,transform=!p.t ;set for cube here only

; Now setup transformation to bring local coord system
; out to Point P This is where the heart of the problem is I think !

Rxy = sqrt(cpNorm[0]^2 + cpNorm[1]^2 )
if ((cpNorm[0] eq 0.0) and (cpNorm[1] eq 0.0)) then Az = 0.0 $
else Az = ATAN(cpNorm[1], cpNorm[0]) * !radeg

if ((Rxy eq 0.0) and (cpNorm[2] eq 0.0)) then Ay = 0.0 $
else Ay = ATAN(cpNorm[2], Rxy) * !radeg

t3d,/reset,rot=[0,90-Ay,Az]
t3d,trans=[cpNormctr[0],cpNormctr[1], cpNormctr[2]]
ctrans = !p.t

; reformat image view area same as above for cube data
t3d, /reset, trans=[-0.5,-0.5,-0.5],scale=[.577,.577,.577]
!p.t = ctrans # !p.t   ;reflect previous transformation
t3d,rot=[-90,300,0]
t3d,rot=[220,0,0]
t3d,trans=[.5,.5,.5]
oModelC->SetProperty,transform=!p.t

.... and draw data
```

File Attachments

1) richt.vcf, downloaded 68 times