Application of Tilt, Slope, Rotation, and Offset in NISP_Win

December 7, 2005

Objects are defined as being bounded by general quadratic surfaces, represented as

$$\mathrm{A}\, x^2 + \mathrm{B}\, x + \mathrm{C}\, y^2 + \mathrm{D}\, y + \mathrm{E}\, z^2 + \mathrm{F}\, z + \mathrm{G} + \mathrm{P}\, x\, y + \mathrm{Q}\, y\, z + \mathrm{R}\, z\, x = 0$$

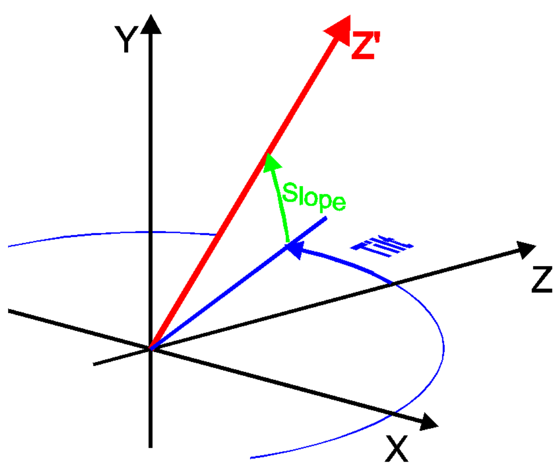
Suppose an object is defined in its own intrinsic coordinate system, (x', y', z'), such that the origin is a reference point used as a center of rotation and translation for the object. We will start by considering surfaces which are invariant under rotation about the z'-axis. For instance, a plane perpendicular to the z'-axis at a distance L from the origin is

$$z' - L = 0$$

and a cylinder of radius r about the z'-axis is

$$x'^2 + y'^2 - r^2 = 0$$

We need to express these surfaces in "world" coordinates (x, y, z) after the object *and its intrinsic coordinate system* have been rotated and translated. The "Tilt" and "Slope" angles are illustrated in the figure, and the z'-axis indicates the resulting orientation of the object. (Note that the coordinate system used in NISP is left-handed.)



The first rotation has to be Slope by an angle S about the x-axis (which is initially also the x'-axis). The second rotation is Tilt by an angle T about the y-axis. The order of matrix operations is first (**S**) and then (**T**), where

$$(\mathbf{T}) = \begin{pmatrix} \cos T & 0 & -\sin T \\ 0 & 1 & 0 \\ \sin T & 0 & \cos t \end{pmatrix}, \qquad (\mathbf{S}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos S & \sin S \\ 0 & -\sin S & \cos S \end{pmatrix}$$

Finally, the object is translated so that its reference origin is at (X_0, Y_0, Z_0):

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos T & \sin T \sin S & -\sin T \cos S \\ 0 & \cos S & \sin S \\ \sin T & -\cos T \sin S & \cos T \cos S \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} + \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix}$$

The determinant of this rotation matrix is 1, and its inverse is equal to its transpose. The transform from world coordinates back to the frame in which the surfaces are defined is

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos T & 0 & \sin T \\ \sin T \sin S & \cos S & -\cos T \sin S \\ -\sin T \cos S & \sin S & \cos T \cos S \end{pmatrix} \begin{pmatrix} x - X_0 \\ y - Y_0 \\ z - Z_0 \end{pmatrix}$$

Applying the rotation to the translation, we define

$$\begin{aligned} XX &= X_0 \cos T + Z_0 \sin T \\ YY &= X_0 \sin T \sin S + Y_0 \cos S - Z_0 \cos T \sin S \\ ZZ &= -X_0 \sin T \cos S + Y_0 \sin S + Z_0 \cos T \cos S \end{aligned}$$

Then

$$\begin{aligned} x' &= x \cos T + z \sin T - XX \\ y' &= x \sin T \sin S + y \cos S - z \cos T \sin S - YY \\ z' &= -x \sin T \cos S + y \sin S + z \cos T \cos S - ZZ \end{aligned}$$

Now we can look at the two example surfaces. The plane ($z' = L$) becomes

$$(-\sin T \cos S)\, x + (\sin S)\, y + (\cos T \cos S)\, z - (ZZ + L) = 0$$

For the cylinder ($x'^2 + y'^2 = r^2$), we square the polynomials for x' and y', and gather like terms:

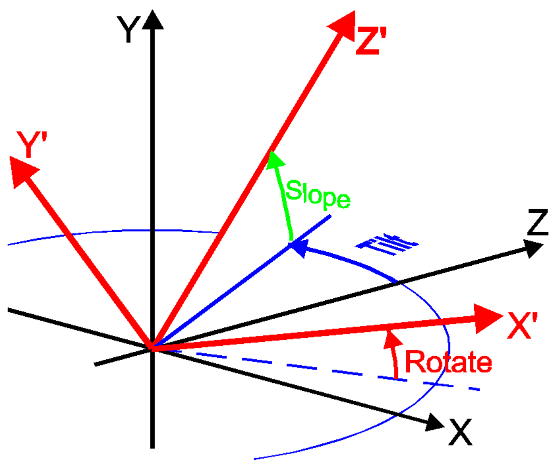
$$\begin{aligned} &(\cos^2 T + \sin^2 T \sin^2 S)\, x^2 - 2\,(XX \cos T + YY \sin T \sin S)\, x + (\cos^2 S)\, y^2 - 2\,(YY \cos S)\, y \\ &\quad + (\sin^2 T + \cos^2 \mathrm{T} \sin^2 S)\, z^2 - 2\,(XX \sin T - YY \cos \mathrm{T} \sin \mathrm{S})\, z + (XX^2 + YY^2 - r^2) \\ &\quad + 2\,(\sin T \sin S \cos S)\, x\, y - 2\,(\cos T \sin S \cos S)\, y\, z + 2\,(\cos T \sin T \cos^2 S)\, z\, x \\ &= 0 \end{aligned}$$

Here is a code excerpt to implement these two surface definitions, which allows a cylindrical region with perpendicular end surfaces to be defined:

```
!      Express offset in rotated coordinates
       XX = cosT*X0/1.d3 + sinT*Z0
       ZZ = cosT*Z0      - sinT*X0/1.d3
       YY = cosS*Y0/1.d3 - sinS*ZZ
       ZZ = cosS*ZZ      + sinS*Y0/1.d3
!
!      Surface 1: entrance plane
       SURFACE(1).B = -sinT*cosS
       SURFACE(1).D =       sinS
       SURFACE(1).F =  cosT*cosS
       SURFACE(1).G = -ZZ
!
!      Surface 2: general cylinder
       SURFACE(2).A =  cosT**2 + (sinT*sinS)**2
       SURFACE(2).B = -2.d0*(cosT*XX +(sinT*sinS)*YY)
       SURFACE(2).C =  cosS**2
       SURFACE(2).D = -2.d0*cosS*YY
       SURFACE(2).E =  sinT**2 + (cosT*sinS)**2
```

```
SURFACE(2).F = -2.d0*(sinT*XX - (cosT*sinS)*YY)
SURFACE(2).G =  XX**2 + YY**2 - (Radius/1.d3)**2
SURFACE(2).P =  2.d0*sinT*sinS*cosS
SURFACE(2).Q = -2.d0*cosT*sinS*cosS
SURFACE(2).R =  2.d0*cosT*sinT*cosS**2
```

Next, we consider the case when rotation about the *z´*-axis is significant. The rotation matrix (**R**) is applied *before* (**S**) and (**T**), while the *z*- and *z´*-axes are coincident. The rotation angle is defined to be positive in the usual sense in the *x*-*y* plane, so because of the left-handed coordinate system it is about the –*z*-axis.



$$(\mathbf{R}) = \begin{pmatrix} \cos R & -\sin R & 0 \\ \sin R & \cos R & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Multiplying the previous inverse matrix on the left by $(\mathbf{R})^T$,

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos T \cos R + \sin T \sin S \sin R & \cos S \sin R & \sin T \cos R - \cos T \sin S \sin R \\ \sin T \sin S \cos R - \cos T \sin R & \cos S \cos R & -\cos T \sin S \cos R - \sin T \sin R \\ -\sin T \cos S & \sin S & \cos T \cos S \end{pmatrix} \begin{pmatrix} x - X_0 \\ y - Y_0 \\ z - Z_0 \end{pmatrix}$$

We can now consider a tapered rectangular region, such as a guide or a block of material, bounded on the sides by two vertical planes. Let the distance between the planes be given by

$$W(z') = W_0 + 2 \tan\phi \; z'$$

where ϕ is the dihedral angle each surface makes with the *y´*-*z´* plane, with positive angle being divergence. Then the two sides can be represented by planes

$$x' + \tan\phi \; z' + W_0/2 = 0, \qquad \text{and} \qquad x' - \tan\phi \; z' - W_0/2 = 0$$

Note that there are no “mixed” terms in between ϕ and *R*. That is because the taper only adds terms in *z´*, and the (**R**) operation leaves *z´* unchanged.

The following code segment represents a symmetric block with varying rectangular cross section, with separate horizontal and vertical divergence angles. While this code appears complicated, note that it is only performed *once* when the element is “built” in the user interface NISP_Win. The surface parameters are recorded in the geometry file for further use, allowing the Monte Carlo code to operate efficiently in world coordinates.

```
!     Express offset in rotated coordinates and convert from mm to m
      XX = ((cosT*cosR + sinT*sinS*sinR)*X0 + cosS*sinR*Y0)/1.d3 +
     &      (sinT*cosR - cosT*sinS*sinR)*Z0
      YY = ((sinT*sinS*cosR - cosT*sinR)*X0 + cosS*cosR*Y0)/1.d3 -
     &      (sinT*sinR + cosT*sinS*cosR)*Z0
      ZZ = (-sinT*cosS*X0 + sinS*Y0)/1.d3 + cosT*cosS*Z0
!
!     Surface 1:  Entrance Plane
      SURFACE(1).B = -sinT*cosS
      SURFACE(1).D =       sinS
      SURFACE(1).F =  cosT*cosS
      SURFACE(1).G = -ZZ
!
!     Surface 2:  Left Side
      TX = -tanH*sinT*cosS
      TY =  tanH*     sinS
      TZ =  tanH*cosT*cosS
      TT = Width/2.d3 - tanH*ZZ
      SURFACE(2).B =  cosT*cosR + sinT*sinS*sinR + TX
      SURFACE(2).D =                   cosS*sinR + TY
      SURFACE(2).F =  sinT*cosR - cosT*sinS*sinR + TZ
      SURFACE(2).G = -(XX - TT)
!
!     Surface 3:  Right Side
      SURFACE(3).B = SURFACE(2).B - 2.d0*TX
      SURFACE(3).D = SURFACE(2).D - 2.d0*TY
      SURFACE(3).F = SURFACE(2).F - 2.d0*TZ
      SURFACE(3).G = -(XX + TT)
!
!     Surface 4:  Bottom Side
      TX = -tanV*sinT*cosS
      TY =  tanV*     sinS
      TZ =  tanV*cosT*cosS
      TT = Height/2.d3 - tanV*ZZ
      SURFACE(4).B =  sinT*sinS*cosR - cosT*sinR + TX
      SURFACE(4).D =       cosS*cosR             + TY
      SURFACE(4).F = -cosT*sinS*cosR - sinT*sinR + TZ
      SURFACE(4).G = -(YY - TT)
!
!     Surface 5:  Top side
      SURFACE(5).B = SURFACE(4).B - 2.d0*TX
      SURFACE(5).D = SURFACE(4).D - 2.d0*TY
      SURFACE(5).F = SURFACE(4).F - 2.d0*TZ
      SURFACE(5).G = -(YY + TT)
!
!     Surface 6:  Exit Plane
      SURFACE(6).B = SURFACE(1).B
      SURFACE(6).D = SURFACE(1).D
      SURFACE(6).F = SURFACE(1).F
      SURFACE(6).G = -(ZZ + Length)
```