The text below is extracted from source code posted in a .zip file at:
http://ftp.setterholm.com/3DEnvC .

… in unzipped file '3DEnv.c' in function 'SeeFrustums()'.

The text is visualized in application:
3DEnv.exe, Version 0.6, dated July 10, 2016
by entering: App-F8 'V' 'b' and using the '+-' keys to scroll the live text.

The explicit realization of the transforms below as 'C'source code
is achieved by '3DEnv.h' -&-
'3DEnv.c' in function 'HindSight()' - lines 1515-2007.

Author email: jeff.setterholm@gmail.com

Use these results at your own risk.

```
---------------------------------------------------------------------------------
---------------------------------------------------------------------------------

2D/3D Visualization Transforms:      (Public Domain)

Projection Matrix \"Concatenation\" Sequence (six matrices) :

#1. Offset model by X=X+D & Y=Y-E*LR :                    <-- Welcome  to
| +1  ,  0  ,  0  , +D    |      LR =-1. for left eye         HindSight!!
|  0  , +1  ,  0  , -E*LR |         = 0. for perspective
|  0  ,  0  , +1  ,  0    |         =+1. for right eye
|  0  ,  0  ,  0  , +1    |

#2. Project YZ onto plane X=D & map X->[-1,+1] :
| +Bnf,  0  ,  0  , +Mnf  |
|  0  , +D  ,  0  ,  0    |
|  0  ,  0  , +D  ,  0    |
| +1  ,  0  ,  0  ,  0    |

#3. Offset model by Y=Y+E*LR :
| +1  ,  0  ,  0  ,  0    |
|  0  , +1  ,  0  , +E*LR |
|  0  ,  0  , +1  ,  0    |
|  0  ,  0  ,  0  , +1    |

#4.  Y->[-1,+1], Z->[-1,+1] :
| +1  ,  0  ,  0  ,  0    |
|  0  , +Mlr,  0  , +Blr  |
|  0  ,  0  , +Mtb, +Btb  |
|  0  ,  0  ,  0  , +1    |
```

```
#5. Screen subsetting for split-screen 3D modes :
| +1  ,  0  ,  0  ,  0    |
|  0  , +Ms ,  0  , +Bsh  |  <-- Screen- horizontal- scale factors
|  0  ,  0  , +Ms , +Bsv  |  <--       -  vertical
|  0  ,  0  ,  0  , +1    | Adjusting L,R,T,& B turns this into
                              an identity matrix... the factors go away.

#6. Convert to OpenGL(Y,-Z,X) left-handed coordinates :
|  0  , +1  ,  0  ,  0    |
|  0  ,  0  , -1  ,  0    |
| +1  ,  0  ,  0  ,  0    |
|  0  ,  0  ,  0  , +1    |

The concatenated projection matrix is #6<#5<#4<#3<#2<#1 :
|+Ms*(+Mlr*E*LR+Blr)+Bsh ,+Ms*Mlr*D , 0        ,+Ms *D*(Blr+Bsh)|
|-Ms*           Btb -Bsv , 0        ,-Ms*Mtb*D ,-Ms *D(*Btb-Bsv)|
|+Bnf                    , 0        , 0        ,    +D *Bnf+Mnf |
|+1                      , 0        , 0        ,    +D          |
       ... per my symbolic matrix concatenator & inverter;
               such a tool saves a lot of time & error.
            (Permutations can be evaluated symbolically.)

As implemented in function 'HindSight':
 Substitute: 'e'=LR*E
'd'= D, but in the orthographic chase 'd'=infinity ...very large is close.
 In split screen 3D,the left and right sides of the viewing frustums
 are fudged. L,R,T,& B are modified to l,r,t,& b, centering & shrinking the
 individual eye views onto their respective sides of the screen.
     Peripheral dissimilarities, large or small, are clipped away.

#2/#4's bias & scaling factors map the viewing volume into a +-1 unit cube:

 Mnf = 2.*(N+d)*(F+d)/(N-F);Bnf=-((N+F)+2.*d)/(N-F);//~y=Mx+B for depth **
 Mlr = 2.            /(r-l);Blr=- (r+l)      /(r-l);// y=Mx+B for lateral
 Mtb = 2.            /(b-t);Btb=- (b+t)      /(b-t);// y=Mx+B for vertical

    ** Note: depth isn't really linear. If N=-D/2. and F=+infinity,
   XS0 would still exactly coincide with the modelspace origin, & the
ability to resolve depth in the near field of view would remain excellent.
  For zero screen depth at X=0.,  use:
    N = -D*F/(D+2.*F)     when: F & D are predefined.
    F = -D*N/(D+2.*N)     when: N & D  \"      \"        & if: N= -D/ a
    D = -2._16*F*N/(F+N)  when: F & N  \"      \"        then: F= +D/(a-2.)
        Thus 'Depth Selfie's become a little easier to interpret.'

 Each eye's frustum (a homogeneous matrix) is defined
  in the next five lines of code:
h44Fill(   h44      ,                             //Visually: this is
       (+Mlr*e+Blr),(+d*Mlr),( 0    ),(+d*Blr    ),  // the exact
       (      -Btb),( 0    ),(-d*Mtb),(-d*Btb    ),  //  symbolic
       (      +Bnf),( 0    ),( 0    ),(+d*Bnf+Mnf),  //   homogeneous
       (+1         ),( 0    ),( 0    ),(+d        ) );//    solution.

... except in the ORTHOGRAPHIC Case...
               when D=Infinity, the matrix terms blow up.
Algebraically dividing all the terms of the above matrix by 'd' yields:
h44Fill(   h44      ,
       ( 0.e0      ),( 2.e0/(R-L)),( 0.e0      ),(-(R+L)/(R-L)),
       ( 0.e0      ),( 0.e0      ),(-2.e0/(B-T)),( (B+T)/(B-T)),
       (-2.e0/(N-F)),( 0.e0      ),( 0.e0      ),( 0.e0       ),
```

```
      ( 0.e0      ),( 0.e0      ),( 0.e0      ),( 1.e0       ) );
```

The projection matrix is 'Zoomed' by variable S.FovYZzoom which divides L,R,T,& B at the outset; zooming does not affect the screen depth range.

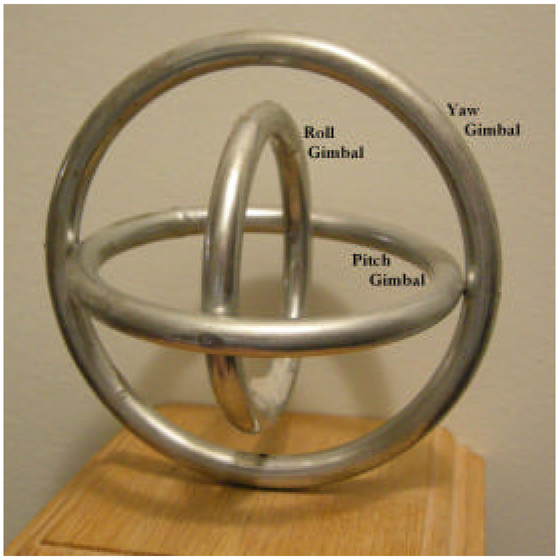
Use function 'PAhXR' for 6dof control of your model, which HindSight uses - with position zeroed - to generate the model rotation matrix (3dof).

The rotation concatenation sequence in Flight Simulation coordinates is:

```
#1. Roll:    positive  Roll rotates +Y toward +Z ,   ~inner gimbal
| +1  ,  0  ,  0  ,  0    |
|  0  , +cR , -sR ,  0    |          sR=  sine(Roll)
|  0  , +sR , +cR ,  0    |          cR=cosine(Roll)
|  0  ,  0  ,  0  , +1    |

#2. Pitch:   positive pitch rotates +Z toward +X,    ~middle gimbal
| +cP ,  0  , +sP ,  0    |
|  0  , +1  ,  0  ,  0    |          sP=  sine(Pitch)
| -sP ,  0  , +cP ,  0    |          cP=cosine(Pitch)
|  0  ,  0  ,  0  , +1    |

#3. Yaw:     positive  yaw  rotates +X toward +Y     ~outer gimbal
| +cY , -sY ,  0  ,  0    |
| +sY , +cY ,  0  ,  0    |          sY=  sine(Yaw)
|  0  ,  0  , +1  ,  0    |          cY=cosine(Yaw)
|  0  ,  0  ,  0  , +1    |
```




```
The concatenated pure rotation matrix #3<#2<#1 is:
| cy*cp,  cy*sp*sr-sy*cr,  cy*sp*cr+sy*sr,  0 |
| sy*cp,  sy*sp*sr+cy*cr,  sy*sp*cr-cy*sr,  0 |
|   -sp,     cp*sr      ,     cp*cr      ,  0 |
|    0 ,        0       ,        0       , +1 |
```

Each of the upper-left 3x3 sub-matrices in the four matrices above is a 'Direction Cosine Matrix', because the numerical values are the cosines of the projection of each input axis onto each output axe... which is why the result is a rigid rotation rather than a warp/'morph'.
For pure rotation matrices- the transpose is the inverse.
Direction Cosine matrices, once populated with numbers, are independent of the 'angles' used to compute them. But if you don't know in which directions +X,+Y,& +Z are point, you've got a problem!

In real-world processes - like manufacturing or navigating - not knowing the Six Degree-Of-Freedom (6dof) coordinate frame you're working in puts you on perilous ground. Questions to ask:

    Where is the origin?
    Where do +X, +Y, & +Z point, & what is the unit of measurement? Meters?
    How are rotations defined & what is the unit of measurement? Degrees?

HindSight's 3D viewer uses: **"Standard Flight Simulation Coordinates"** in ModelView space, described at/in:

www.setterholm.com in the /Geodesy subdirectory:
'qVPMath12-AppendixA-20091211.pdf'

Note: 'Quaternions' provide another way of implementing model rotation which has no specific gimbal sequence, but instead exactly rotates around an arbitrarily-chosen axis. (The math is complex.)

The Model Translation & Scaling Sequence (two matrices,4dof):

```
#1. ReCenter on (i.e. translate to) the 'Point of interest':
| +1    ,  0    ,  0    , -PoIX |
|  0    , +1    ,  0    , -PoIY |
|  0    ,  0    , +1    , -PoIZ |
|  0    ,  0    ,  0    , +1    |

#2.\"Scale\" (i.e. 3D Magnify):
| +Scale,  0    ,  0    ,  0    |
|  0    , +Scale,  0    ,  0    |
|  0    ,  0    , +Scale,  0    |
|  0    ,  0    ,  0    , +1    |

Which concatinates to:
    h44Fill(PoIScaleh44,
      ( Scale), ( 0.e0 ), ( 0.e0 ), (-Scale*PoIX),
      ( 0.e0 ), ( Scale), ( 0.e0 ), (-Scale*PoIY),
      ( 0.e0 ), ( 0.e0 ), ( Scale), (-Scale*PoIZ),
      ( 0.e0 ), ( 0.e0 ), ( 0.e0 ), ( 1.e0      )  );
```

The Clipping Planes:

Geometric planes are defined by a surface 'normal' (= 'perpendicular') vector and a distance. In the frustum(s) viewer - the directions of the four clipping plane normals are displayed & scaled to exactly touch their respective clipped planes.

Clipping plane algebraic coefficients are shown on screen App-F8 'v'.

Press 'v' to see the numerical values live.

Press 'b' to view: the eye viewpoints(s) -&-
...from here the projection frustum(s) -mapped by inversion -&-
& 's' the clipping plane normals.

3DEnv/HindSight, Version 0.6, July 10,2016 Author: Jeffrey M. Setterholm
8095 230th St. E.
Lakeville, Minnesota 55044

Use these results at your own risk.