Source Code for ScreenX() and ScreenY()

Tomas P Nally -
steelweaver52

Return to Easy Functions for Plotting 3D Objects

The source code for **ScreenX()** is shown just below. The source code for ScreenY() is further down.

Source Code for ScreenX()

```
Function
 ScreenX(XX, YY, ZZ, CamX, CamY, CamZ, CtrX, CtrY, CtrZ, ScrCtrX, ScrC
trY, Scale)
'Note: This is version 1.1 of ScreenX()

    'Establish the Vector Components of the Camera-to-Center Vector
    Cam2CtrX = (CtrX - CamX)
    Cam2CtrY = (CtrY - CamY)
    Cam2CtrZ = (CtrZ - CamZ)

    LenCam2Ctr = sqr(Cam2CtrX^2 + Cam2CtrY^2 + Cam2CtrZ^2)

    'The vector equation for the virtual image plane can be written
    'as follows:
    '

'Cam2CtrX*(x - CtrX) + Cam2CtrY*(y - CtrY) + Cam2CtrZ*(z - CtrZ) = 0

    'Imagine a unit vector pointing in the -Y direction.  The
    'components of this vector are 0i -1j + 0k.  When we take
    'the cross product of this vector with the Camera-to-Center
    'vector, the result is the virtual x-axis vector imposed
    'upon the virtual image plane.  These are the vector
    'components of the virtual x-axis vector.

    virtualXi = (-1)*(Cam2CtrZ)
    virtualXj = 0
    virtualXk = Cam2CtrX

    'Find the length of this vector, then divide components
```

```
'by this length.

LenVirtualX = sqr(virtualXi^2 + virtualXj^2 + virtualXk^2)
virtualXi = virtualXi / LenVirtualX
virtualXj = virtualXj / LenVirtualX
virtualXk = virtualXk / LenVirtualX


'In order to find the virtual y-axis on the image plane,
'we need to take the cross product of the virtual x-axis
'vector with the Camera-to-Center Vector.  Given below is
'the result of that cross product.

virtualYi = (-1)*(virtualXk * Cam2CtrY)
virtualYj = (virtualXk * Cam2CtrX) - (virtualXi * Cam2CtrZ)
virtualYk = (virtualXi * Cam2CtrY)

'Find the length of this vector, then divide the
'components by this length

LenVirtualY = sqr(virtualYi^2 + virtualYj^2 + virtualYk^2)
virtualYi = virtualYi / LenVirtualY
virtualYj = virtualYj / LenVirtualY
virtualYk = virtualYk / LenVirtualY

'Divide the unit virtual X-axis vector and the
'virtual Y-axis vector by LenCam2Ctr.  This transformation
'of these two vectors will allow the objects to get
'smaller as the camera moves away from the viewing
'center, or get larger as the camera moves toward
'the viewing center.

virtualXi = virtualXi / LenCam2Ctr
virtualXj = virtualXj / LenCam2Ctr
virtualXk = virtualXk / LenCam2Ctr

virtualYi = virtualYi / LenCam2Ctr
virtualYj = virtualYj / LenCam2Ctr
virtualYk = virtualYk / LenCam2Ctr

'Establish the vector components of the Camera-to-Node Vector

Cam2NodeX = (XX - CamX)
Cam2NodeY = (YY - CamY)
Cam2NodeZ = (ZZ - CamZ)
```

```
    'The parametric equations for the Camera-to-Node Vector
    'can be written as follows:
    '
    'x = CamX + Cam2NodeX*t
    'y = CamY + Cam2NodeY*t
    'z = CamZ + Cam2NodeZ*t
    '
    'Plug these three equations into the vector equation
    'for the virtual image plane...
    '

'Cam2CtrX*(x - CtrX) + Cam2CtrY*(y - CtrY) + Cam2CtrZ*(z - CtrZ) = 0
    '
    '...and then solve for t

    numerator   = Cam2CtrX^2 + Cam2CtrY^2 + Cam2CtrZ^2
    denominator = (Cam2NodeX * Cam2CtrX) + (Cam2NodeY * Cam2CtrY) + (C
am2NodeZ * Cam2CtrZ)
    t = (numerator / denominator)

    'Having solved for t, determine the point in space
    '(ipx, ipy, ipz) where 'the Camera-to-Node vector intersects
    'the virtual image plane.

    ipX = CamX + Cam2NodeX*t
    ipY = CamY + Cam2NodeY*t
    ipZ = CamZ + Cam2NodeZ*t

    'Establish the vector components of the vector from the
    'center to (ipx, ipy, ipz).

    Ctr2ipX = (ipX - CtrX)
    Ctr2ipY = (ipY - CtrY)
    Ctr2ipZ = (ipZ - CtrZ)

    'The projection of this vector along the virtual X-axis
    'is the dot product of this vector with the unit vector
    'along the virtual X-Axis"

    PX = (Ctr2ipX*virtualXi) + (Ctr2ipY*virtualXj) + (Ctr2ipZ*virtualX
k)

    SCM = 500
'Note: SCM is an acronym for "Secondary Scale Multiplier".

'      This value was found by experimentation when it was
```

```
                '      observed that the Scale factor by itself was
                '      too small without a multiplier.

    ScreenX = ScrCtrX + (SCM*Scale * PX)

    'The projection of this vector along the virtual Y-axis
    'is the dot product of this vector with the unit vector
    'along the virtual Y-Axis"


'PY = (Ctr2ipX*virtualYi) + (Ctr2ipY*virtualYj) + (Ctr2ipZ*virtualYk)

    'ScreenY = ScrCtrY - (SCM*Scale * PY)

end function
```

Top of Page

Return to the companion article for these functions, Easy Functions For Plotting 3D Objects.

Source Code for ScreenY()

```
Function
 ScreenY(XX, YY, ZZ, CamX, CamY, CamZ, CtrX, CtrY, CtrZ, ScrCtrX, ScrC
trY, Scale)
'Note: This is version 1.1 of ScreenY()

    'Establish the Vector Components of the Camera-to-Center Vector
    Cam2CtrX = (CtrX - CamX)
    Cam2CtrY = (CtrY - CamY)
    Cam2CtrZ = (CtrZ - CamZ)

    LenCam2Ctr = sqr(Cam2CtrX^2 + Cam2CtrY^2 + Cam2CtrZ^2)


    'The vector equation for the virtual image plane can be written
    'as follows:
    '

'Cam2CtrX*(x - CtrX) + Cam2CtrY*(y - CtrY) + Cam2CtrZ*(z - CtrZ) = 0

    'Imagine a unit vector pointing in the -Y direction.  The
```

```
'components of this vector are 0i -1j + 0k.  When we take
'the cross product of this vector with the Camera-to-Center
'vector, the result is the virtual x-axis vector imposed
'upon the virtual image plane.  These are the vector
'components of the virtual x-axis vector.

virtualXi = (-1)*(Cam2CtrZ)
virtualXj = 0
virtualXk = Cam2CtrX

'Find the length of this vector, then divide components
'by this length.

LenVirtualX = sqr(virtualXi^2 + virtualXj^2 + virtualXk^2)
virtualXi = virtualXi / LenVirtualX
virtualXj = virtualXj / LenVirtualX
virtualXk = virtualXk / LenVirtualX


'In order to find the virtual y-axis on the image plane,
'we need to take the cross product of the virtual x-axis
'vector with the Camera-to-Center Vector.  Given below is
'the result of that cross product.

virtualYi = (-1)*(virtualXk * Cam2CtrY)
virtualYj = (virtualXk * Cam2CtrX) - (virtualXi * Cam2CtrZ)
virtualYk = (virtualXi * Cam2CtrY)

'Find the length of this vector, then divide the
'components by this length

LenVirtualY = sqr(virtualYi^2 + virtualYj^2 + virtualYk^2)
virtualYi = virtualYi / LenVirtualY
virtualYj = virtualYj / LenVirtualY
virtualYk = virtualYk / LenVirtualY

'Divide the unit virtual X-axis vector and the
'virtual Y-axis vector by LenCam2Ctr.  This transformation
'of these two vectors will allow the objects to get
'smaller as the camera moves away from the viewing
'center, or get larger as the camera moves toward
'the viewing center.

virtualXi = virtualXi / LenCam2Ctr
virtualXj = virtualXj / LenCam2Ctr
virtualXk = virtualXk / LenCam2Ctr
```

```
    virtualYi = virtualYi / LenCam2Ctr
    virtualYj = virtualYj / LenCam2Ctr
    virtualYk = virtualYk / LenCam2Ctr

    'Establish the vector components of the Camera-to-Node Vector

    Cam2NodeX = (XX - CamX)
    Cam2NodeY = (YY - CamY)
    Cam2NodeZ = (ZZ - CamZ)

    'The parametric equations for the Camera-to-Node Vector
    'can be written as follows:
    '
    'x = CamX + Cam2NodeX*t
    'y = CamY + Cam2NodeY*t
    'z = CamZ + Cam2NodeZ*t
    '
    'Plug these three equations into the vector equation
    'for the virtual image plane...
    '

'Cam2CtrX*(x - CtrX) + Cam2CtrY*(y - CtrY) + Cam2CtrZ*(z - CtrZ) = 0
    '
    '...and then solve for t

    numerator   = Cam2CtrX^2 + Cam2CtrY^2 + Cam2CtrZ^2
    denominator = (Cam2NodeX * Cam2CtrX) + (Cam2NodeY * Cam2CtrY) + (C
am2NodeZ * Cam2CtrZ)
    t = (numerator / denominator)

    'Having solved for t, determine the point in space
    '(ipx, ipy, ipz) where 'the Camera-to-Node vector intersects
    'the virtual image plane.

    ipX = CamX + Cam2NodeX*t
    ipY = CamY + Cam2NodeY*t
    ipZ = CamZ + Cam2NodeZ*t

    'Establish the vector components of the vector from the
    'center to (ipx, ipy, ipz).

    Ctr2ipX = (ipX - CtrX)
    Ctr2ipY = (ipY - CtrY)
    Ctr2ipZ = (ipZ - CtrZ)
```

```
    'The projection of this vector along the virtual X-axis
    'is the dot product of this vector with the unit vector
    'along the virtual X-Axis"


'PX = (Ctr2ipX*virtualXi) + (Ctr2ipY*virtualXj) + (Ctr2ipZ*virtualXk)

    SCM = 500
'Note: SCM is an acronym for "Secondary Scale Multiplier".

'      This value was found by experimentation when it was
                '      observed that the Scale factor by itself was
                '      too small without a multiplier.


    'ScreenX = ScrCtrX + (SCM*Scale * PX)

    'The projection of this vector along the virtual Y-axis
    'is the dot product of this vector with the unit vector
    'along the virtual Y-Axis"

    PY = (Ctr2ipX*virtualYi) + (Ctr2ipY*virtualYj) + (Ctr2ipZ*virtualY
k)

    ScreenY = ScrCtrY - (SCM*Scale * PY)

end function
```

Top of Page

Return to the companion article for these functions, Easy Functions For Plotting 3D Objects.

Tom Nally
Steelweaver52@aol.com

Note: This linked source code accompanies Easy Functions for Plotting 3D Objects, which originally appeared in the Liberty BASIC Newsletter, Issue #113. It is reprinted here with the permission of the author.

-
JanetTerra