Source Code for Plot3D

Tom Nally -
steelweaver52

If you prefer, you may download the zipped file

Plot3D.zip

- Details
- Download
- 4 KB

.

Return to Easy Functions for Plotting 3D Objects.

Source Code for Plot3D

```
''''''''''''''''''''''''''''''''''''
'  Plot3D.bas                      '
'  Version 1.1                     '
'  by Tomas J. Nally               '
'  Steelweaver52@aol.com           '
'  Copyright September 2003        '
''''''''''''''''''''''''''''''''''''
'  Released as Open Source         '
''''''''''''''''''''''''''''''''''''
'  Made with the following:        '
'                                  '
'  Liberty BASIC                   '
'  by Carl Gundel                  '
'  http://www.libertybasic.com     '
'                                  '
'  Liberty BASIC Workshop          '
'  by Alyce Watson                 '
'  http://alycesrestaurant.com     '
''''''''''''''''''''''''''''''''''''
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Version 1.1 Changes to ScreenX() and ScreenY():                      '
'                                                                     '
'   The virtual X-axis vector and the virtual                         '
'   Y-axis vector is divided by the length of                         '
'   the Camera-to-Center vector.  This change                         '
'   enables the objects to look smaller as                            '
'   the camera moves away from the viewing                            '
'   center, or look larger as the camera moves                        '
'   toward the viewing center.                                        '
'                                                                     '
'   Additionally, the change noted above causes                       '
'   the objects to appear very small.  So, within                     '
'   ScreenX() and ScreenY(), the scale is multiplied                  '
'   by another number called SCM, for "Secondary                      '
'   Scale Multiplier".  This operations makes the                     '
'   objects appear about as large as they appeared                    '
'   in version 1.0 of ScreenX() and ScreenY().                        '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

            True = 1
           False = 0
        NumNodes = 42
        NumLines = 47
            CamX = 225
            CamY = 225
            CamZ = 325
            CtrX = 0
            CtrY = 0
            CtrZ = 0
         ScrCtrX = 200
         ScrCtrY = 200

      ScaleFactor = 1


Dim XX(100)
Dim YY(100)
Dim ZZ(100)
Dim SX(100)
Dim SY(100)
Dim inode(100)
Dim jnode(100)


'Node Data
```

```
'------X-----Y----Z----------
Data  25,    0, -25   'Node 1
Data  25,    0,  25   'Node 2
Data -25,    0,  25   'Node 3
Data -25,    0, -25   'Node 4
Data  25,   50, -25   'Node 5
Data  25,   50,  25   'Node 6
Data -25,   50,  25   'Node 7
Data -25,   50, -25   'Node 8
''''''''''''''''''''''''''''''
Data -40,    0, -25   'Node 9
Data -40,    0,  25   'Node 10
Data -90,    0,  25   'Node 11
Data -90,    0, -25   'Node 12
Data -50,   40, -15   'Node 13
Data -50,   40,  15   'Node 14
Data -80,   40,  15   'Node 15
Data -80,   40, -15   'Node 16
''''''''''''''''''''''''''''''
Data -105,   0, -05   'Node 17
Data -105,   0,  45   'Node 18
Data -155,   0,  45   'Node 19
Data -155,   0, -05   'Node 20
Data -105,  80, -05   'Node 21
Data -105,  80,  45   'Node 22
Data -155,  80,  45   'Node 23
Data -155,  80, -05   'Node 24
''''''' X - Axis ''''''''''''
Data    0,    0,   0   'Node 25
Data   50,    0,   0   'Node 26
Data   50,    0,  -5   'Node 27
Data   60,    0,   5   'Node 28
Data   60,    0,  -5   'Node 29
Data   50,    0,   5   'Node 30
''''''' Y - Axis ''''''''''''''
Data    0,    0,   0   'Node 31
Data    0,   70,   0   'Node 32
Data   -5,   85,   0   'Node 33
Data    5,   85,   0   'Node 34
Data    0,   80,   0   'Node 35
Data    0,   75,   0   'Node 36
''''''' Z - Axis ''''''''''''''
Data    0,    0,   0   'Node 37
Data    0,    0,  50   'Node 38
Data   -5,    0,  55   'Node 39
Data    5,    0,  55   'Node 40
```

```
Data   -5,    0,  65   'Node 41
Data    5,    0,  65   'Node 42



'Line Data
'-----inode----jnode--------
Data      1,   2     'Line 1
Data      2,   3     'Line 2
Data      3,   4     'Line 3
Data      4,   1     'Line 4
Data      5,   6     'Line 5
Data      6,   7     'Line 6
Data      7,   8     'Line 7
Data      8,   5     'Line 8
Data      1,   5     'Line 9
Data      2,   6     'Line 10
Data      3,   7     'Line 11
Data      4,   8     'Line 12
''''''''''''''''''''''''''''''
Data      9,   10    'Line 13
Data     10,   11    'Line 14
Data     11,   12    'Line 15
Data     12,   9     'Line 16
Data     13,   14    'Line 17
Data     14,   15    'Line 18
Data     15,   16    'Line 19
Data     16,   13    'Line 20
Data      9,   13    'Line 21
Data     10,   14    'Line 22
Data     11,   15    'Line 23
Data     12,   16    'Line 24
''''''''''''''''''''''''''''''
Data     17,  18     'Line 25
Data     18,  19     'Line 26
Data     19,  20     'Line 27
Data     20,  17     'Line 28
Data     21,  22     'Line 29
Data     22,  23     'Line 30
Data     23,  24     'Line 31
Data     24,  21     'Line 32
Data     17,  21     'Line 33
Data     18,  22     'Line 34
Data     19,  23     'Line 35
Data     20,  24     'Line 36
''' X - Axis Lines '''''''''
```

```
Data     25,  26     'Line 37
Data     27,  28     'Line 38
Data     29,  30     'Line 39
''' Y - Axis Lines '''''''''
Data     31,  32     'Line 40
Data     33,  35     'Line 41
Data     34,  35     'Line 42
Data     35,  36     'Line 43
''' Z - Axis Lines '''''''''
Data     37,  38     'Line 44
Data     39,  40     'Line 45
Data     40,  41     'Line 46
Data     41,  42     'Line 47


For i = 1 to NumNodes
    Read QQ$, RR$, SS$
    XX(i) = val(QQ$)
    YY(i) = val(RR$)
    ZZ(i) = val(SS$)
next i

For i = 1 to NumLines
    Read QQ$, RR$
    inode(i) = val(QQ$)
    jnode(i) = val(RR$)

next i

[InitColors]
    'It is suggested that default colors be used when possible.
    'Activate the following color statements only if
    'they are really necessary in the program.
    'ForegroundColor$ = "Black"
    'BackgroundColor$ = "Buttonface"
    'TexteditorColor$ = "White"
    'TextboxColor$    = "White"
    'ComboboxColor$   = "White"
    'ListboxColor$    = "White"

[WindowSetup]
    NOMAINWIN
    WindowWidth = 653 : WindowHeight = 485
    UpperLeftX = INT((DisplayWidth-WindowWidth)/2)
    UpperLeftY = INT((DisplayHeight-WindowHeight)/2)
```

```
[ControlSetup]

Menu        #Plot3D, "&File" , _
                     "E&xit", [btnQuit.click]

Menu        #Plot3D, "Help" , _

"Open Plot3D.txt with Notepad...", [Open.Plot3D.txt.With
.Notepad], |, _
                     "About Plot 3D...", [File.About]

graphicbox  #Plot3D.gbox1, 230, 5, 400, 400
statictext  #Plot3D.stat01, "Cam:",    15,  45, 45, 20
statictext  #Plot3D.stat02, "Ctr:",    15,  75, 45, 20
statictext  #Plot3D.stat03, "ScrCtr:", 15, 105, 45, 20
statictext  #Plot3D.stat04, "Scale:",  15, 135, 45, 20
statictext  #Plot3D.stat05,
"X ---------- Y --------- Z", 85, 15, 120, 20
button      #Plot3D.btnRedraw, "RePlot Objects"
,[btnRedraw.click],UL, 60, 185, 150, 35
button      #Plot3D.btnClear, "Clear Screen"
,[btnClear.click],UL, 60, 225, 150, 35
button      #Plot3D.btnQuit, "Quit"
,[btnQuit.click],UL, 60, 370, 150, 35
button      #Plot3D.btnReset,
"Reset Original Values",[btnReset.click],UL, 60, 265, 150, 35
textbox     #Plot3D.txtCamX,     60,  40, 50, 25
textbox     #Plot3D.txtCamY,    115,  40, 50, 25
textbox     #Plot3D.txtCamZ,    170,  40, 50, 25
textbox     #Plot3D.txtCtrX,     60,  70, 50, 25
textbox     #Plot3D.txtCtrY,    115,  70, 50, 25
textbox     #Plot3D.txtCtrZ,    170,  70, 50, 25
textbox     #Plot3D.txtScrCtrX,  60, 100, 50, 25
textbox     #Plot3D.txtScrCtrY, 115, 100, 50, 25
textbox     #Plot3D.txtScale,    60, 130, 50, 25

Open "Functions for Plotting 3D Objects" for Window as #Plot3D

    print #Plot3D, "trapclose [btnQuit.click]"
    print #Plot3D.gbox1, "down; fill White; flush"
    print #Plot3D.gbox1, "setfocus; when mouseMove [MouseChange1]"
    print #Plot3D, "font ms_sans_serif 10"

    gosub [Initialize.All.Controls]

[loop]
```

```
    Wait

[Initialize.All.Controls]

    print #Plot3D.txtCamX, CamX
    print #Plot3D.txtCamY, CamY
    print #Plot3D.txtCamZ, CamZ

    print #Plot3D.txtCtrX, CtrX
    print #Plot3D.txtCtrY, CtrY
    print #Plot3D.txtCtrZ, CtrZ

    print #Plot3D.txtScrCtrX, ScrCtrX
    print #Plot3D.txtScrCtrY, ScrCtrX
    print #Plot3D.txtScale,   ScaleFactor


    For i = 1 to NumNodes
        SX(i) = ScreenX(XX(i), YY(i), ZZ(i), CamX, CamY, CamZ, CtrX, C
trY, CtrZ, ScrCtrX, ScrCtrY, ScaleFactor)
        SY(i) = ScreenY(XX(i), YY(i), ZZ(i), CamX, CamY, CamZ, CtrX, C
trY, CtrZ, ScrCtrX, ScrCtrY, ScaleFactor)
    next i

    print #Plot3D.gbox1, "cls"

    For i = 1 to NumLines
        print #Plot3D.gbox1, "line "; SX(inode(i)); " "
; SY(inode(i)); " "; SX(jnode(i)); " "; SY(jnode(i))
    next i

    return



[MouseChange1]
    'MouseX and MouseY contain mouse coordinates
    Wait

[btnRedraw.click]

    'Get the new values for the camera coordinates

    print #Plot3D.txtCamX, "!contents? CamX$"
    print #Plot3D.txtCamY, "!contents? CamY$"
    print #Plot3D.txtCamZ, "!contents? CamZ$"
```

```
    CamX = val(CamX$)
    CamY = val(CamY$)
    CamZ = val(CamZ$)

    'Get the new values for the "Center"

    print #Plot3D.txtCtrX, "!contents? CtrX$"
    print #Plot3D.txtCtrY, "!contents? CtrY$"
    print #Plot3D.txtCtrZ, "!contents? CtrZ$"

    CtrX = val(CtrX$)
    CtrY = val(CtrY$)
    CtrZ = val(CtrZ$)

    'Get the new values for the Screen Center

    print #Plot3D.txtScrCtrX, "!contents? ScrCtrX$"
    print #Plot3D.txtScrCtrY, "!contents? ScrCtrY$"

    ScrCtrX = val(ScrCtrX$)
    ScrCtrY = val(ScrCtrY$)

    'Get the new values for the Scale Factor

    print #Plot3D.txtScale, "!contents? ScaleFactor$"

    ScaleFactor = val(ScaleFactor$)


    For i = 1 to NumNodes
        SX(i) = ScreenX(XX(i), YY(i), ZZ(i), CamX, CamY, CamZ, CtrX, C
trY, CtrZ, ScrCtrX, ScrCtrY, ScaleFactor)
        SY(i) = ScreenY(XX(i), YY(i), ZZ(i), CamX, CamY, CamZ, CtrX, C
trY, CtrZ, ScrCtrX, ScrCtrY, ScaleFactor)
    next i

    print #Plot3D.gbox1, "cls"

    For i = 1 to NumLines
        print #Plot3D.gbox1, "line "; SX(inode(i)); " "
; SY(inode(i)); " "; SX(jnode(i)); " "; SY(jnode(i))
    next i

    Wait
```

```
[btnClear.click]

    print #Plot3D.gbox1, "cls"

    Wait

[btnQuit.click]
     close #Plot3D : END
    Wait

[btnReset.click]

    CamX = 225
    CamY = 225
    CamZ = 325
    CtrX = 0
    CtrY = 0
    CtrZ = 0
    ScrCtrX = 200
    ScrCtrY = 200
    ScaleFactor = 1

    gosub [Initialize.All.Controls]

    Wait

[Open.Plot3D.txt.With.Notepad]

    RUN "NOTEPAD Plot3D.txt"

    Wait

[File.About]

    Notice "About Plot 3D Version 1.1        " + chr$(13) + _
           "Copyright Tomas J. Nally         " + chr$(13) + _
           "September 2003                   " + chr$(13) + _
           "Steelweaver52@aol.com            " + chr$(13) + _
           "                                 " + chr$(13) + _
           "**Released as open source**      " + chr$(13) + _
           "                                 " + chr$(13) + _
           "Made with the following:         " + chr$(13) + _
           "                                 " + chr$(13) + _
           "Liberty BASIC by Carl Gundel     " + chr$(13) + _
           "http://www.libertybasic.com      " + chr$(13) + _
           "                                 " + chr$(13) + _
```

```
            "Liberty BASIC Workshop           " + chr$(13) + _
            "by Alyce Watson                  " + chr$(13) + _
            "http://alycesrestaurant.com      "

    Wait



''''''''''''''''''''''''''''''''''''

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

'''''''''''''''''''''''''''''''''''''

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

'''''''''''''''''''''''''''''''''''''''
```

Return to 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