Instant 3D-2D Perspective Mapping: the Navigator Functions

– *by George Lungu*

<excelunusual.com>

Australian supermaxi "Wild Oats XI"

Introduction:

This tutorial explains a pair of important user defined functions, the "Navigator_u" and the "Navigator_v". These functions save the user nine columns of formulas by calculating the effects of: 3-dimenssional shift, rotations around the three axes of coordinates and 3D-2D perspective transforms.

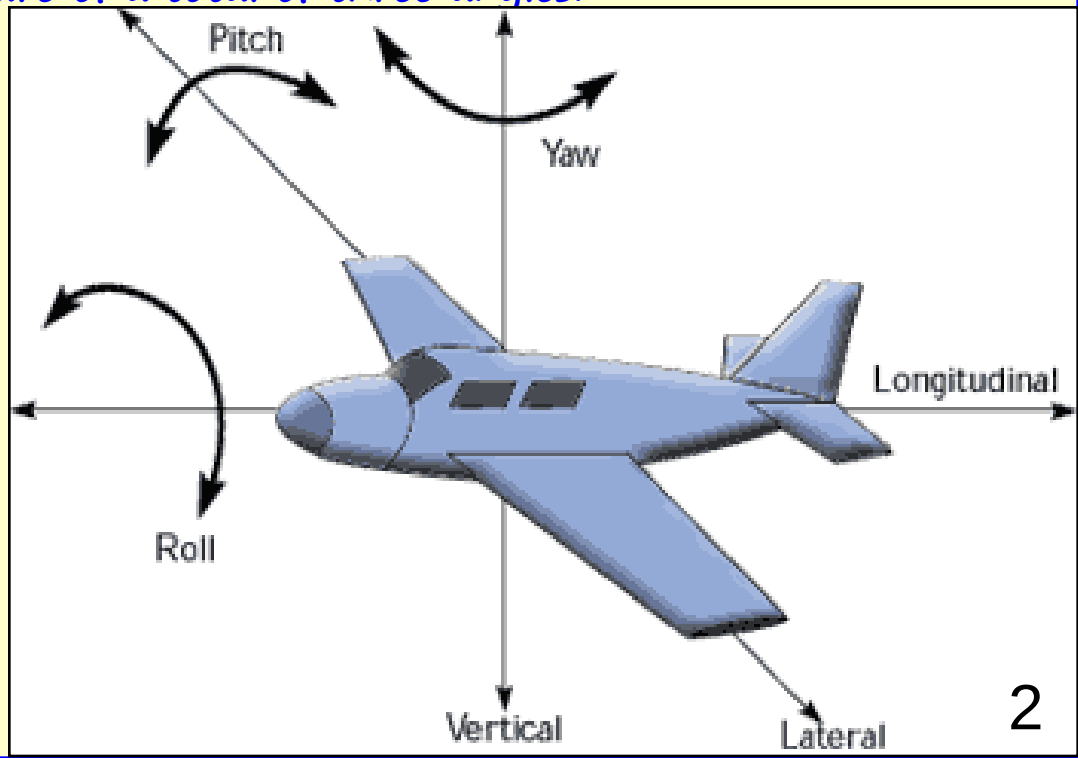
In the previous 3D perspective tutorials we took in consideration two angles of rotation, the azimuth and the altitude. In artillery, radar, astronomy or wherever we have a ground based platform with an aiming function we need only two angles (azimuth and altitude) to define the aiming direction.

The situation gets a bit more complicated in the case of a vehicle (ship, aircraft, or even a car) where we need to take care of a total of three angles:

- the yaw angle (formerly known as azimuth) defines rotation around the vertical - z axis

-the pitch angle or attitude (formerly known as altitude) defines rotation around the lateral - x axis

-the roll angle (new) defines rotation around longitudinal - y axis



The user defined function below is the first of a pair of functions which calculate spatial shift, three different types of rotations and the 3D-2D perspective conversion

User Defined Function declaration

Function name

Native point coordinate (object definition)

Spatial shift values

The three rotation angles

Eye-screen distance

Screen-origin distance

```
Function Navigator_u (x0, y0, z0, dx, dy, dz, Yaw, Pitch, Roll, eye_scr, scr_orig) As Double
```

Function return type

```
Dim x, y, z, x1, y1, z1, x2, y2, z2, x3, y3, z3 As Double
```

Variable type

<excelunusual.com>

Intermediate variables used inside the function to calculate intermediate results

```
x = x0 + dx
y = y0 + dy
z = z0 + dz
```

New coordinates after translation

```
x1 = x * Sin(Yaw / 57.29578) + y * Cos(Yaw / 57.29578)
y1 = x * Cos(Yaw / 57.29578) - y * Sin(Yaw / 57.29578)
z1 = z
```

New coordinates after yaw rotation

Division by 57.29578 converts degrees in radians

```
x2 = x1
y2 = y1 * Cos(Pitch / 57.29578) - z1 * Sin(Pitch / 57.29578)
z2 = y1 * Sin(Pitch / 57.29578) + z1 * Cos(Pitch / 57.29578)
```

New coordinates after pitch rotation

```
x3 = z2 * Sin(Roll / 57.29578) + x2 * Cos(Roll / 57.29578)
y3 = y2
z3 = z2 * Cos(Roll / 57.29578) - x2 * Sin(Roll / 57.29578)
```

New coordinates after roll rotation

```
Navigator_u = x3 * eye_scr / (eye_scr + scr_orig + y3)
```

("u") calculation as function output

$$u = \frac{x \cdot ES}{ES + SO + y}$$

```
End Function
```

End of function declaration

The user defined function below is the last of a pair of functions which calculate spatial shift, three different types of rotations and the 3D-2D perspective conversion

User Defined Function declaration

Function name

Native point coordinate (object definition)

Spatial shift values

The three rotation angles

Eye-screen distance

Screen-origin distance

Function return type

```
Function Navigator_v (x0, y0, z0, dx, dy, dz, Yaw, Pitch, Roll, eye_scr, scr_orig, enable) As Double
Dim x, y, z, x1, y1, z1, x2, y2, z2, x3, y3, z3 As Double
x = x0 + dx
y = y0 + dy
z = z0 + dz
x1 = x * Sin(Yaw / 57.29578) + y * Cos(Yaw / 57.29578)
y1 = x * Cos(Yaw / 57.29578) - y * Sin(Yaw / 57.29578)
z1 = z
x2 = x1
y2 = y1 * Cos(Pitch / 57.29578) - z1 * Sin(Pitch / 57.29578)
z2 = y1 * Sin(Pitch / 57.29578) + z1 * Cos(Pitch / 57.29578)
x3 = z2 * Sin(Roll / 57.29578) + x2 * Cos(Roll / 57.29578)
y3 = y2
z3 = z2 * Cos(Roll / 57.29578) - x2 * Sin(Roll / 57.29578)
If enable = 1 Then
    Navigator_v = z3 * eye_scr / (eye_scr + scr_orig + y3)
Else
    Navigator_v = 9999
End If
End Function
```

Enable – show or hide point from the chart

Variable type

Enable – show or hide point from the chart

New coordinates after translation

Intermediate variables used inside the function to calculate intermediate results

New coordinates after yaw rotation

New coordinates after pitch rotation

New coordinates after roll rotation

("u") calculation as function output

$$u = \frac{x \cdot ES}{ES + SO + y}$$

Enable – show or hide point from the chart

End of function declaration

Here is a final, cleaned up version of the two functions – comments are in green

```
Function Navigator_u(x0, y0, z0, dx, dy, dz, Yaw, Pitch, Roll, eye_scr, scr_orig) As Double
Dim x, y, z, x1, y1, z1, x2, y2, z2, x3, y3, z3 As Double
Yaw = Yaw / 57.29578        'convert degrees to radians and assign result back to same argument
Pitch = Pitch / 57.29578    'convert degrees to radians and assign result back to same argument
Roll = Roll / 57.29578      'convert degrees to radians and assign result back to same argument
x1 = (x0 + dx) * Sin(Yaw) + (y0 + dy) * Cos(Yaw)
y1 = (x0 + dx) * Cos(Yaw) - (y0 + dy) * Sin(Yaw)          ‘ z1 = (z0 + dz)
y2 = y1 * Cos(Pitch) - (z0 + dz) * Sin(Pitch)             ‘ x2 = x1
z2 = y1 * Sin(Pitch) + (z0 + dz) * Cos(Pitch)
x3 = z2 * Sin(Roll) + x1 * Cos(Roll)                      ' y3 = y2
z3 = z2 * Cos(Roll) - x1 * Sin(Roll)
Navigator_u = x3 * eye_scr / (eye_scr + scr_orig + y2) ' since y3 = y2
End Function
```

```
Function Navigator_v(x0, y0, z0, dx, dy, dz, Yaw, Pitch, Roll, eye_scr, scr_orig, enable) As Double
Dim x, y, z, x1, y1, z1, x2, y2, z2, x3, y3, z3 As Double
Yaw = Yaw / 57.29578          'convert degrees to radians
Pitch = Pitch / 57.29578      'convert degrees to radians
Roll = Roll / 57.29578        'convert degrees to radians
x1 = (x0 + dx) * Sin(Yaw) + (y0 + dy) * Cos(Yaw)
y1 = (x0 + dx) * Cos(Yaw) - (y0 + dy) * Sin(Yaw)          ‘ z1 = (z0 + dz)
y2 = y1 * Cos(Pitch) - (z0 + dz) * Sin(Pitch)             ‘ x2 = x1
z2 = y1 * Sin(Pitch) + (z0 + dz) * Cos(Pitch)
x3 = z2 * Sin(Roll) + x1 * Cos(Roll)                      ' y3 = y2
z3 = z2 * Cos(Roll) - x1 * Sin(Roll)
If enable = 1 Then
   Navigator_v = z3 * eye_scr / (eye_scr + scr_orig + y2) ' since y3 = y2
Else
   Navigator_v = 9999         ‘ we use argument "enable" different than 1 to make the point invisible
End If
End Function
```

Implementation:

The sample workbook contains three worksheets, the first two are identical, the only difference is that the latter is trimmed to have fast button response. The third worksheet is a demo and you must hit the "Run-Pause" button to make it work.

Thanks to John Kerr for his ideas of demo automation in VBA.