Reset Component Position

Submitted by NXJournaling on Fri, 05/20/2016 - 21:17

Reset Component Position

When working with assembly files, moving components around in space, there may come a time when you want to reset a component to its absolute starting point; the position where it would appear when adding it with the "absolute position" option. There are various work-arounds in NX to get a component back to its absolute position, but out-of-the-box there is no one click solution - we'll have to make our own.

Component Translations And Rotations

The assembly file keeps track of each component's location and orientation. You can see how a component has been moved around at any time by using the "information -> object" command and selecting the component; in the information window, the translation and rotation will be reported. When you add a component with the "absolute origin" positioning option, the absolute coordinate system of the component part will align to the absolute coordinate system of the assembly part. If the component has not been moved, the component translation delta X, Y, and Z values will be zero, the X-axis vector rotation will be (1,0,0), the Y-axis vector rotation will be (0,1,0), and the Z-axis vector rotation will be (0,0,1) as shown below.

```
Component Translations:

Delta X              =       9.00000000000
Delta Y              =       0.00000000000
Delta Z              =       0.00000000000

Component Rotations:

X-axis Vector        XC =    1.00000000000        X =    1.00000000000
                     YC =    0.00000000000        Y =    0.00000000000
                     ZC =    0.00000000000        Z =    0.00000000000

Y-axis Vector        XC =    0.00000000000        X =    0.00000000000
                     YC =    1.00000000000        Y =    1.00000000000
                     ZC =    0.00000000000        Z =    0.00000000000

Z-axis Vector        XC =    0.00000000000        X =    0.00000000000
                     YC =    0.00000000000        Y =    0.00000000000
                     ZC =    1.00000000000        Z =    1.00000000000
```

If you rotate the component by 30° about the X axis, the rotation vectors will be reported as shown below.

```
Component Rotations:

X-axis Vector        XC =    1.00000000000        X =    1.00000000000
                     YC =    0.00000000000        Y =    0.00000000000
                     ZC =    0.00000000000        Z =    0.00000000000

Y-axis Vector        XC =    0.00000000000        X =    0.00000000000
                     YC =    0.86602540378        Y =    0.86602540378
                     ZC =   -0.50000000000        Z =   -0.50000000000

Z-axis Vector        XC =    0.00000000000        X =    0.00000000000
                     YC =    0.50000000000        Y =    0.50000000000
                     ZC =    0.86602540378        Z =    0.86602540378
```

Search

User Login

Username *

Password *

- Create new account
- Request new password

Log in

RSS Feed

Main Menu

- Home
- Forums
- Tutorials
  - Getting started with journaling in NX
  - Create Geometry
  - User Interaction
  - Export Parasolid
  - Expressions: create an expression with units
  - Expressions: creating expressions
  - Expressions: query existing expressions
  - NXOpen.Features.Feature and the Feature Collection
  - Process file in directory
  - Read a text file
  - Report length of Tube Feature
  - Reset Component Position
  - Sort a list of NX objects
  - Start journal from custom button
  - Units
  - Using VectorArithmetic.Vector3
  - Wrangling the WCS
  - Write to a text file
  - Processing an Assembly
- Journals
- Submit
- Resources

The delta X, Y, and Z values are easy enough to understand, but what's going on with the rotation vectors? The mathematicians in the crowd may recognize the previous values as the sine and cosine of a 30° angle. When you put the rotation angle vectors together, you get a rotation matrix. This rotation matrix is defined in terms of the sine and cosine values of the rotation angles. Defining the rotations in matrix notation this way has several advantages, most of which are beyond the scope of this article. For more information, the rotation matrix article on wikipedia is a great place to start.

The component's location and rotation information is easily accessible through the NXOpen API. With this information, it isn't too much trouble to move the component back to the absolute position. All we need to do is find a matrix that is the inverse of the current rotation matrix, rotate the component according to this new rotation matrix, then translate the component back to the origin. You may be thinking "matrix inverse? I'm outta here...", but wait a moment, because I see the mathematician smiling. Why is he smiling, does the thought of L-U decomposition get his blood racing? Well, possibly; but I think he knows a trick that will make this problem easier than it sounds...

Mathematician

> Your rotation matrix is an orthogonal matrix. One of the useful properties of an orthogonal matrix is that the inverse is the same as the transpose.

If the term "matrix inverse" made you instinctively reach for your favorite headache medicine, rest assured that performing a matrix transpose is much easier than finding the inverse. This property of square matrices will really come in handy for this journal. The transpose really is the trickiest part of this journal; it is not tricky in the sense of being difficult, but rather it is easy to make an error while typing which would result in a difficult to track down bug in our code.

To perform the transpose, we simply have to switch the row and column values in the matrix. So the following matrix:

```
1 2 3
4 5 6
7 8 9
```

becomes:

```
1 4 7
2 5 8
3 6 9
```

You can see how the values in the first column (1, 4, 7) become the values in the first row. Repeat for the other two columns and you end up with the transpose, which in our case is also the matrix inverse.

The Code

```
Option Strict Off

Imports System
Imports NXOpen
Imports NXOpen.UF
Imports NXOpenUI

Module componentOriginalPosition

        Dim theSession As Session = Session.GetSession()
        Dim workPart As Part = theSession.Parts.Work

        Sub Main()

        Dim selComp As NXOpen.Assemblies.Component = Nothing
        If SelectComponent(selComp) = Selection.Response.Cancel Then
            Exit Sub
        End If
```

Recent Comments

- Whether the assembly is fully 3 days 5 hours ago
- An alternative way could be 4 days 10 hours ago
- Our initial plan was to use 4 days 11 hours ago
- re: edit part attribute 4 days 12 hours ago
- re: sketch plane 4 days 12 hours ago
- re: long run time 4 days 12 hours ago
- run journal in batch mode 6 days 5 hours ago
- re: run journal in batch mode 6 days 11 hours ago
- Thank you for your response. 6 days 15 hours ago
- re: PDF export 1 week 2 days ago

```
        Dim myInterpartDelay As Boolean = theSession.UpdateManager.InterpartDe

        Dim pt As Point3d
        Dim RotMat As Matrix3x3

        Dim pt2 As Vector3d
        Dim RotMat2 As Matrix3x3

        selComp.GetPosition(pt, RotMat)
        'msgbox("Translation: " & pt.x & ", " & pt.y & ", " & pt.z)
        'msgbox("Rotation: " & vbcrlf & _
        '       RotMat.xx & ", " & RotMat.xy & ", " & RotMat.xz & vbcrlf & _
        '       RotMat.yx & ", " & RotMat.yy & ", " & RotMat.yz & vbcrlf & _
        '       RotMat.zx & ", " & RotMat.zy & ", " & RotMat.zz)

        'no translation of component on first pass
        pt2.X = 0
        pt2.Y = 0
        pt2.Z = 0

        'transpose of the rotation matrix, undo any rotations on the component
        RotMat2.Xx = RotMat.Xx
        RotMat2.Xy = RotMat.Yx
        RotMat2.Xz = RotMat.Zx
        RotMat2.Yx = RotMat.Xy
        RotMat2.Yy = RotMat.Yy
        RotMat2.Yz = RotMat.Zy
        RotMat2.Zx = RotMat.Xz
        RotMat2.Zy = RotMat.Yz
        RotMat2.Zz = RotMat.Zz

        'avoid problems if the part contains wave links or other interpart dat
        theSession.UpdateManager.InterpartDelay = True

        'move the component back to the original rotation
        workPart.ComponentAssembly.MoveComponent(selComp, pt2, RotMat2)

        'get the translation information again, now that the component has bee
        selComp.GetPosition(pt, RotMat)

        'negate the translations that have been applied to the component
        pt2.X = -pt.X
        pt2.Y = -pt.Y
        pt2.Z = -pt.Z

        'set rotation matrix to identity matrix, we want no new rotations on t
        'or simply use RotMat returned from GetPosition, as it will be the ide
        'after the component has been rotated back to its original position
        RotMat2.Xx = 1
        RotMat2.Xy = 0
        RotMat2.Xz = 0
        RotMat2.Yx = 0
        RotMat2.Yy = 1
        RotMat2.Yz = 0
        RotMat2.Zx = 0
        RotMat2.Zy = 0
        RotMat2.Zz = 1

        'translate component back to 0,0,0
        workPart.ComponentAssembly.MoveComponent(selComp, pt2, RotMat2)

        'reset interpart delay to original value
        theSession.UpdateManager.InterpartDelay = myInterpartDelay

        End Sub

    Function SelectComponent(ByRef selObj As TaggedObject) As Selection.Respor

        Dim theUI As UI = UI.GetUI
        Dim message As String = "Select component to reset position"
        Dim title As String = "Select a Component"
        Dim includeFeatures As Boolean = False
        Dim keepHighlighted As Boolean = False
```

```
        Dim selAction As Selection.SelectionAction = Selection.SelectionActior
        Dim cursor As Point3d
        Dim scope As Selection.SelectionScope = Selection.SelectionScope.AnyIr
        Dim selectionMask_array(0) As Selection.MaskTriple

        With selectionMask_array(0)
            .Type = UFConstants.UF_component_type
            .Subtype = UFConstants.UF_all_subtype
        End With

        Dim resp As Selection.Response = theUI.SelectionManager.SelectTaggedOb
         title, scope, selAction, _
         includeFeatures, keepHighlighted, selectionMask_array, _
         selObj, cursor)
        If resp = Selection.Response.ObjectSelected OrElse resp = Selection.Re
            Return Selection.Response.Ok
        Else
            Return Selection.Response.Cancel
        End If

    End Function

    Public Function GetUnloadOption(ByVal dummy As String) As Integer
        GetUnloadOption = NXOpen.UF.UFConstants.UF_UNLOAD_IMMEDIATELY
    End Function

End Module
```

Conclusion

In this article we looked at how to get the component's location and orientation information. We then used that information to move the component back to the absolute origin. You've also seen how to use the .MoveComponent method and how to temporarily turn off the interpart update option. Hopefully, these methods will come in handy when creating your own journals.

Notes:
1) The code above is meant to be used on components that do not have constraints applied to them. If the journal is run on a component that has constraints applied to it, the component will move only insofar as the constraints allow.
2) The interpart update is turned off while moving the component because otherwise errors may occur during the move process if the component is involved in a wave link operation.

Comments

Reset Position

Iriastine - Wed, 10/12/2016 - 08:18

Is there a way of keeping the previous position of the part and then apply a new transfor from that position? Every time I try to move the object again, the object would move according to its origion position rather than from the previous one.

Recent Content