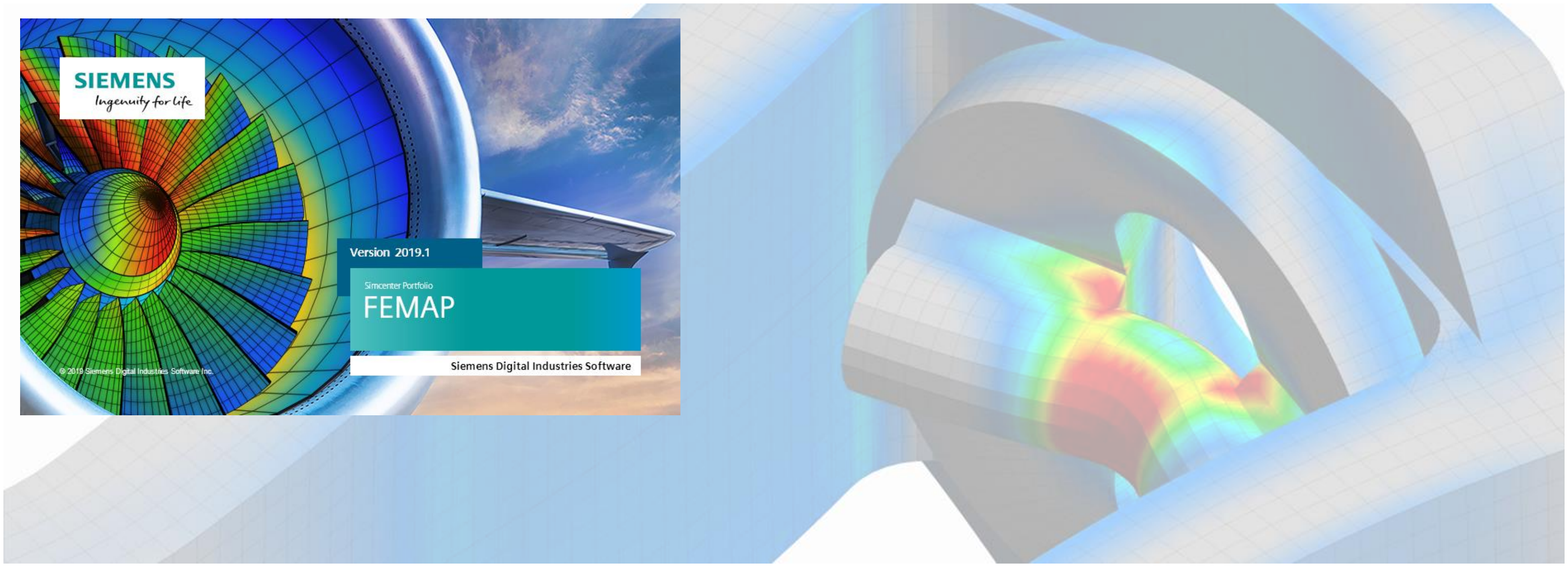

SIEMENS
Ingenuity for life
Version 2019.1
Simcenter Portfolio
FEMAP
Siemens Digital Industries Software
© 2019 Siemens Digital Industries Software Inc.

Automation and Optimization with the FEMAP API

A Beginner's Guide for FEMAP and NX Nastran Users

Adrian Jensen, PE – Senior Application Engineer

Finite Element Analysis
Predictive Engineering
LS-DYNA Sales, Support & Consulting
www.PredictiveEngineering.com

AppliedCAx
Siemens PLM Software Sales & Support
CAD I CAM I CAE I Teamcenter I STAR-CCM+
www.AppliedCAx.com

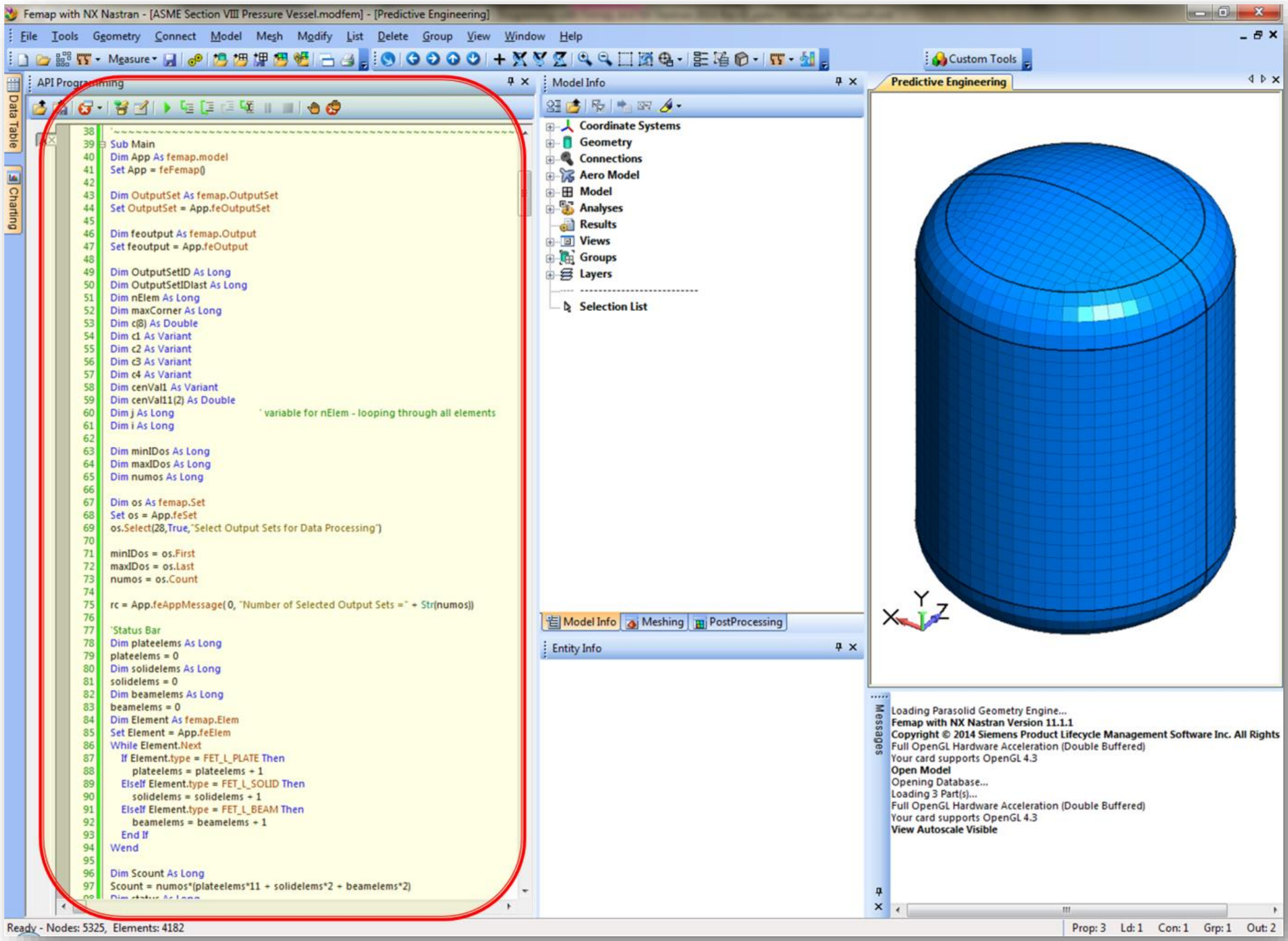
TABLE OF CONTENTS

1. THE FEMAP API AND OBJECT ORIENTED PROGRAMMING

1.1 WHAT IS AN API?

- The FEMAP API is an OLE/COM-based programming interface and is object oriented programming. If you have never programmed in an object oriented code, it can seem quite different and foreign.
- API means "Application Programming Interface". It is important to understand that the API script you write is **not part of FEMAP**, but is a stand-alone program that is interacting with FEMAP.
- There are a number of codes that can call FEMAP through the API: Visual Basic, VBA (Excel, Word, Access, etc.), C, or C++.
- The most commonly used codes used are Visual Basic, VBA, and WinWrap.
- WinWrap is a flavor of Visual Basic that is included with FEMAP. In the FEMAP interface, WinWrap is **noncompilable**, for this reason many choose not to use it, but it is a very convenient way to program if your specific application does not need to be compiled.
- This seminar will focus on using WinWrap via the FEMAP API window.

This is the optional FEMAP API editing window. Although the window appears to be part of your FEMAP session, it is not. It is merely a code editing tool.



```
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Sub Main
Dim App As femap.model
Set App = feFemap()

Dim OutputSet As femap.OutputSet
Set OutputSet = App.feOutputSet

Dim feoutput As femap.Output
Set feoutput = App.feOutput

Dim OutputSetID As Long
Dim OutputSetIDlast As Long
Dim nElem As Long
Dim maxCorner As Long
Dim c(8) As Double
Dim c1 As Variant
Dim c2 As Variant
Dim c3 As Variant
Dim c4 As Variant
Dim cenVal1 As Variant
Dim cenVal11(2) As Double
Dim j As Long                    ' variable for nElem - looping through all elements
Dim i As Long

Dim minIDos As Long
Dim maxIDos As Long
Dim numos As Long

Dim os As femap.Set
Set os = App.feSet
os.Select(28,True,"Select Output Sets for Data Processing")

minIDos = os.First
maxIDos = os.Last
numos = os.Count

rc = App.feAppMessage( 0, "Number of Selected Output Sets =" + Str(numos))

'Status Bar
Dim plateelems As Long
plateelems = 0
Dim solidelems As Long
solidelems = 0
Dim beamelems As Long
beamelems = 0
Dim Element As femap.Elem
Set Element = App.feElem
While Element.Next
   If Element.type = FET_L_PLATE Then
      plateelems = plateelems + 1
   ElseIf Element.type = FET_L_SOLID Then
      solidelems = solidelems + 1
   ElseIf Element.type = FET_L_BEAM Then
      beamelems = beamelems + 1
   End If
Wend

Dim Scount As Long
Scount = numos*(plateelems*11 + solidelems*2 + beamelems*2)
```

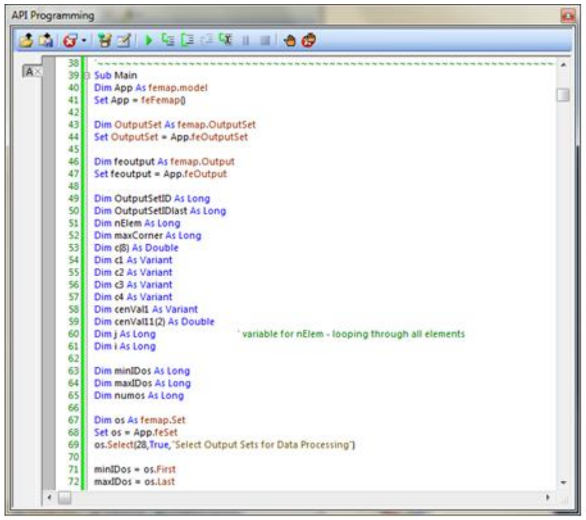
```
Loading Parasolid Geometry Engine...
Femap with NX Nastran Version 11.1.1
Copyright © 2014 Siemens Product Lifecycle Management Software Inc. All Rights
Full OpenGL Hardware Acceleration (Double Buffered)
Your card supports OpenGL 4.3
Open Model
Opening Database...
Loading 3 Part(s)...
Full OpenGL Hardware Acceleration (Double Buffered)
Your card supports OpenGL 4.3
View Autoscale Visible
```

```
Ready - Nodes: 5325, Elements: 4182                Prop: 3   Ld: 1   Con: 1   Grp: 1   Out: 2
```

1.2 What is Object Oriented Programming?

Traditional programming is usually seen as being a set of functions, or simply as a list of instructions.

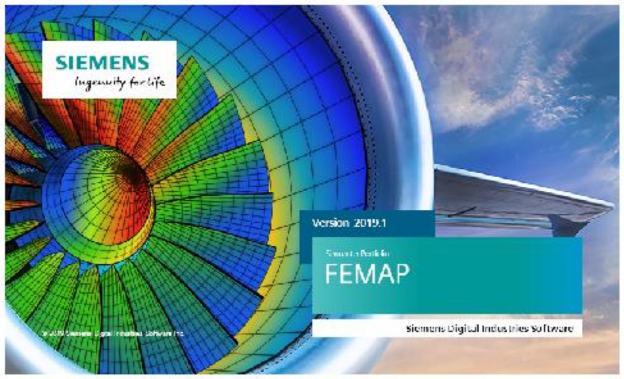
- Object Oriented Programming (or OOP) can be seen as a group of objects that cooperate with each other. Each of the objects has their own distinct set of capabilities.



```
'---------------------------------------------------------------
Sub Main
Dim App As femap.model
Set App = feFemap()

Dim OutputSet As femap.OutputSet
Set OutputSet = App.feOutputSet

Dim feoutput As femap.Output
Set feoutput = App.feOutput

Dim OutputSetID As Long
Dim OutputSetIDlast As Long
Dim nElem As Long
Dim maxCorner As Long
Dim c(8) As Double
Dim c1 As Variant
Dim c2 As Variant
Dim c3 As Variant
Dim c4 As Variant
Dim cenVal1 As Variant
Dim cenVal11(2) As Double
Dim j As Long                    ' variable for nElem - looping through all elements
Dim i As Long

Dim minIDos As Long
Dim maxIDos As Long
Dim numos As Long

Dim os As femap.Set
Set os = App.feSet
os.Select(28,True,"Select Output Sets for Data Processing")

minIDos = os.First
maxIDos = os.Last
```

Visual Basic Code

API



Femap

It is helpful to think of each of the entities as being separate.

- Your Visual Basic code acts like a traditional code, i.e. as a set of instructions.
- The VB code makes requests of the API, which then acts upon those requests either by retrieving from and putting things into the FEMAP database.
- Remember, FEMAP is a database, which only holds and displays data.

1.3 FEMAP API Objects

The objects found in the FEMAP API fall into three categories:

- ***FEMAP Application Object***. This is the "king of the objects" within the FEMAP API. It is needed to create other objects. It is also the object that will be used to perform operations, measure, calculate, display messages, etc.
  - For example, create the ***FEMAP Application Object***:
    ```
    Dim App As femap.model
    Set App = feFemap()
    ```
  - Then using the ***feNode*** Method, the ***FEMAP Application Object*** can create a ***Node Object***:
    ```
    Dim nd As femap.Node
    Set nd = App.feNode
    ```

- ***FEMAP Tool Objects***. These objects allow access to the various windows available in the user interface, others provide entity selection, and general data and file functionality. The ***Set Object*** is one of the most common.
  - Use the ***feSet*** Method of the ***FEMAP Application Object*** to create Set objects:
    ```
    Dim ndSet As femap.Set
    Set ndSet = App.feSet
    ```
  - Then use the ***Select*** Method of the ***Set Object*** to prompt the user to pick nodes:
    ```
    ndSet.Select(FT_NODE, True, "Pick Nodes")
    ```

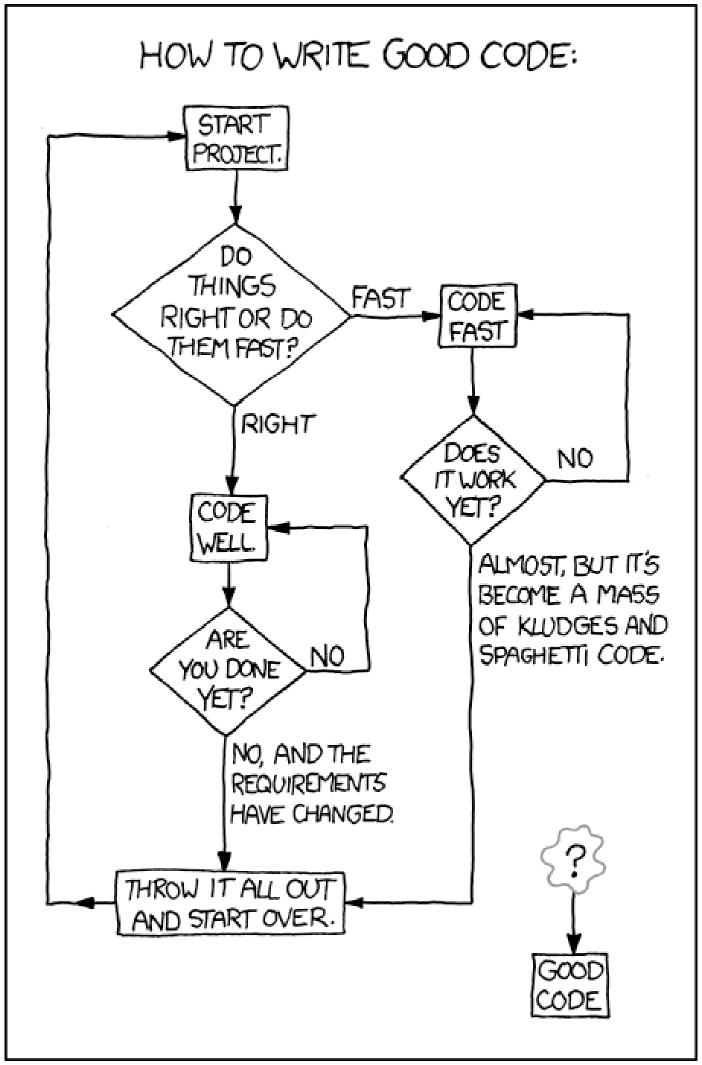
- ***FEMAP Entity Objects***. Objects. For every entity within FEMAP (nodes, elements materials, loads, etc.) there is a corresponding object within the API. These objects are used to manipulate entities.
  - The ***Node Object*** can now manipulate the coordinates of nodes:
    ```
    nd.x = 100
    nd.y = 26.1
    nd.z = 5.23
    ```

2. INTRODUCTION TO FEMAP'S API

At the end of this seminar, we will walk through an existing program, line-by-line, and identify the building blocks of an API.

That can be a bit boring, so before we get to that, let's briefly talk about some of the key concepts for working with the FEMAP API:

- FEMAP Application Object
- Set Object
- Object Methods
- Object Properties
- Variables
- Data Types
- Return Codes

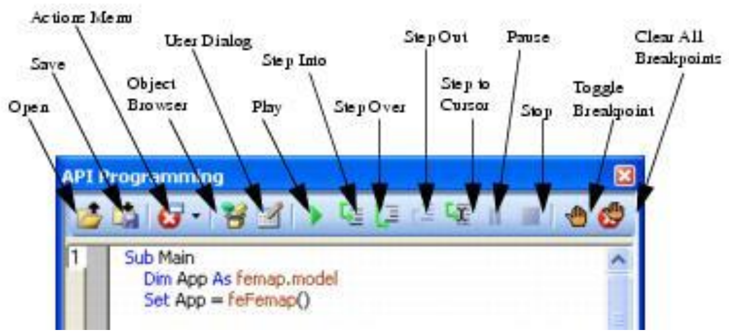

HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
DOES IT WORK YET?
NO
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
RIGHT
CODE WELL
ARE YOU DONE YET?
NO
NO, AND THE REQUIREMENTS HAVE CHANGED.
THROW IT ALL OUT AND START OVER.
?
GOOD CODE

2.1 "Hello World" – Your First FEMAP API

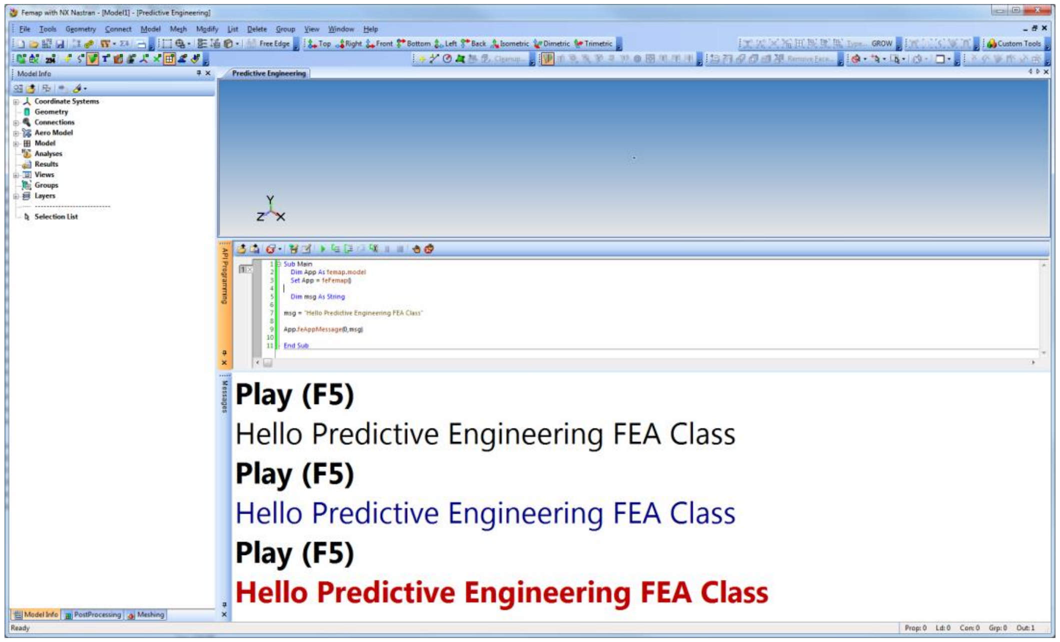
Programming can be very intimidating. For the uninitiated, the code sitting within a FEMAP API file is confusing and overwhelming. In this example, we will take the first step in programming and simply prove to ourselves that connection to the FEMAP interface through the API can be accomplished by "non-programmers".

Topics Covered:

- API Files vs. Program Files
- Connecting to FEMAP
- Auto Complete within the API Programming Window
- Data Types
- Parameter Info
- Hunting the FEMAP help Files



```
Sub Main
    Dim App As femap.model
    Set App = feFemap()
```



```
Sub Main
    Dim App As femap.model
    Set App = feFemap()

    Dim msg As String

msg = "Hello Predictive Engineering FEA Class"

App.feAppMessage(0,msg)

End Sub
```

Play (F5)
Hello Predictive Engineering FEA Class
Play (F5)
Hello Predictive Engineering FEA Class
Play (F5)
Hello Predictive Engineering FEA Class

Dimensioning the FEMAP Application Object, see FEMAP API Help Section Section 3.1.2:

Method 1**

```
Dim App As femap.model
Set App = feFemap()
```

Method 2*

```
Dim App As femap.model
Set App = GetObject(,"femap.model")
```

Method 3

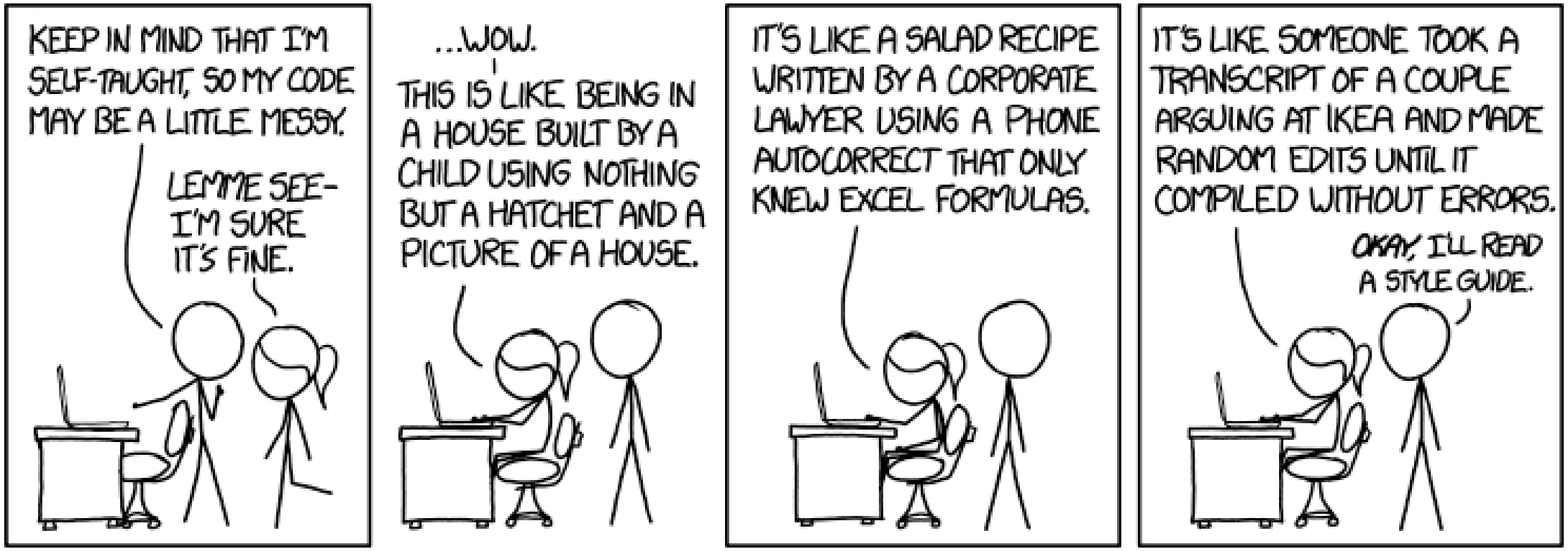
```
Dim App As Object
Set App = GetObject(, "femap.model")
```

***(Does not work from VB in Excel)*

(Does not* ***automatically* *work from VB in Excel)*

3. COLLECTING INFORMATION FROM THE USER

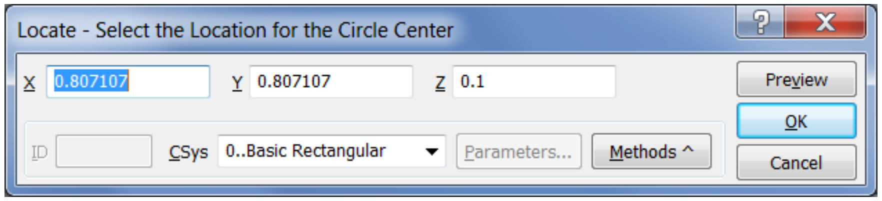
If you're just interested in automating a repetitive task without a lot of user interaction, you might be better of using a Program File (recorded macro). However, if you need some user interaction, you'll need a way to collect information from them.


KEEP IN MIND THAT I'M SELF-TAUGHT, SO MY CODE MAY BE A LITTLE MESSY.
LEMME SEE– I'M SURE IT'S FINE.
...WOW.
THIS IS LIKE BEING IN A HOUSE BUILT BY A CHILD USING NOTHING BUT A HATCHET AND A PICTURE OF A HOUSE.
IT'S LIKE A SALAD RECIPE WRITTEN BY A CORPORATE LAWYER USING A PHONE AUTOCORRECT THAT ONLY KNEW EXCEL FORMULAS.
IT'S LIKE SOMEONE TOOK A TRANSCRIPT OF A COUPLE ARGUING AT IKEA AND MADE RANDOM EDITS UNTIL IT COMPILED WITHOUT ERRORS.
OKAY, I'LL READ A STYLE GUIDE.

3.1 Standard Dialog Boxes

Summoning the “Standard Dialog Boxes” involves a variety of FEMAP Application Object Methods to define coordinates, vectors, planes or other types of information. These dialog boxes are very useful, nonspecialized methods to prompt the user for information.

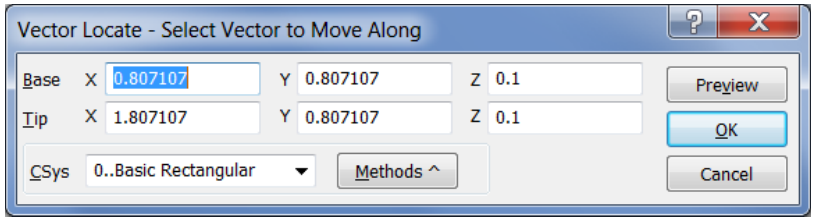
```
App.feCoordPick("Select the Location for the Circle Center", PointXYZ)
```


Locate - Select the Location for the Circle Center

X 0.807107 Y 0.807107 Z 0.1

ID CSys 0..Basic Rectangular

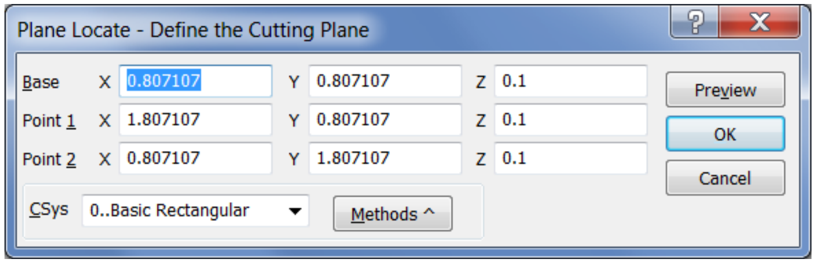
```
App.feVectorPick("Select Vector to Move Along", False, vecLength, vecBase, vecDir)
```


Vector Locate - Select Vector to Move Along

Base	X	0.807107	Y	0.807107	Z	0.1
Tip	X	1.807107	Y	0.807107	Z	0.1

CSys 0..Basic Rectangular

```
App.fePlanePick("Define the Cutting Plane", plBase, plNormal, plAxis)
```


Plane Locate - Define the Cutting Plane

Base	X	0.807107	Y	0.807107	Z	0.1
Point 1	X	1.807107	Y	0.807107	Z	0.1
Point 2	X	0.807107	Y	1.807107	Z	0.1

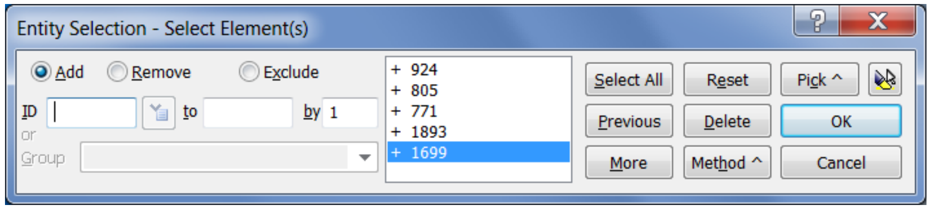
CSys 0..Basic Rectangular

3.2 Set Objects

Set objects are the most common of the tool objects and any program that requires the user to select entities is going to utilize Set objects.

The ***Select Method*** brings up the standard entity selection dialog box with all the bells and whistles to facilitate picking what you want.

```
entSET.Select(FT_ELEM, True, "Select Element(s)")
```


Entity Selection - Select Element(s)

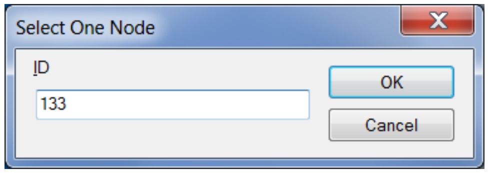
[✓ Add] Remove Exclude

ID | to | by 1

+ 924
+ 805
+ 771
+ 1893
+ 1699

The ***SelectID Method*** brings up an entity selection dialog box that only allows one entity to be picked.

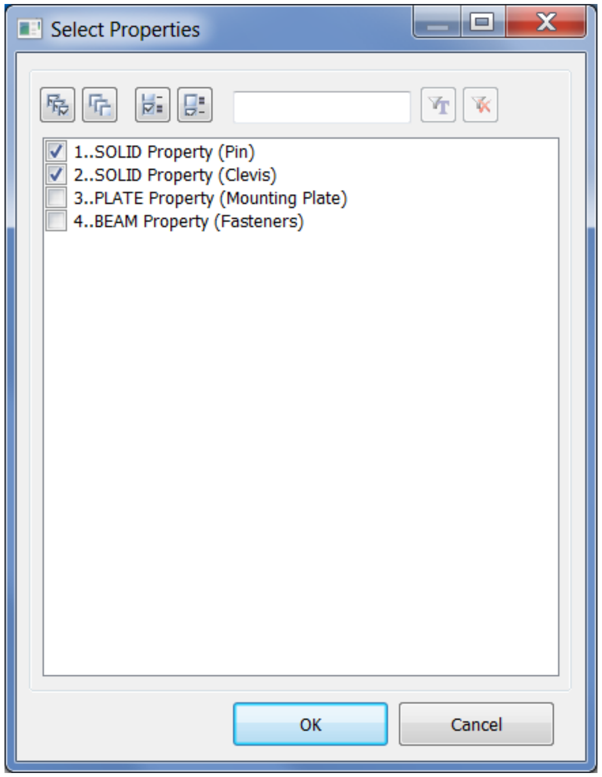
```
entSET.SelectID(FT_NODE, "Select One Node", ndID)
```


Select One Node

ID
133

The ***SelectMultiID Method*** brings up dialog box allowing the user to select entities from a list. In most cases you should only use it for entities with titles.

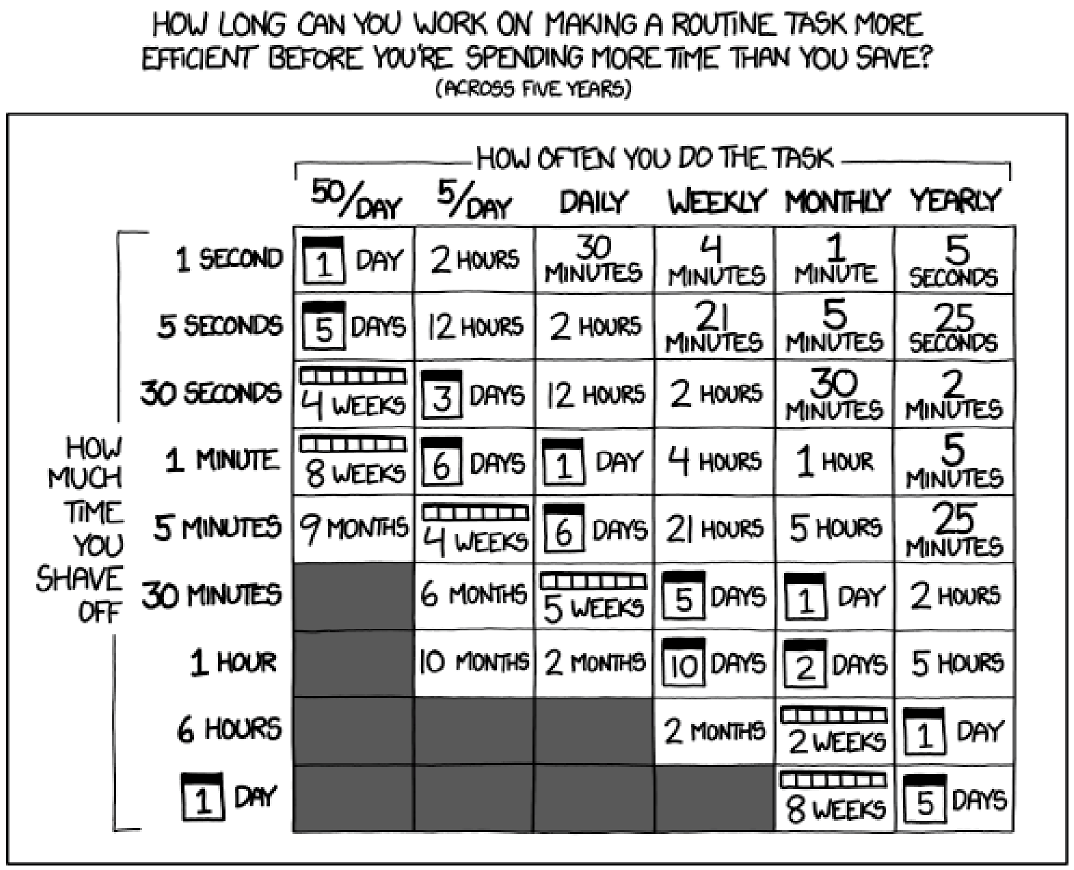
```
entSET.SelectMultiID(FT_PROP, 1, "Select Properties")
```


Select Properties

- [✓] 1..SOLID Property (Pin)
- [✓] 2..SOLID Property (Clevis)
- [ ] 3..PLATE Property (Mounting Plate)
- [ ] 4..BEAM Property (Fasteners)

4. WORKING WITH FEMAP ENTITIES

Incorporating materials and properties into our models is really where we start to differentiate FEA from CAD. It might seem like there isn't a lot of time to be saved by automating the creation or modification of materials and properties, but the subsequent programs will show that that these skills are important to have in your programming toolbox.

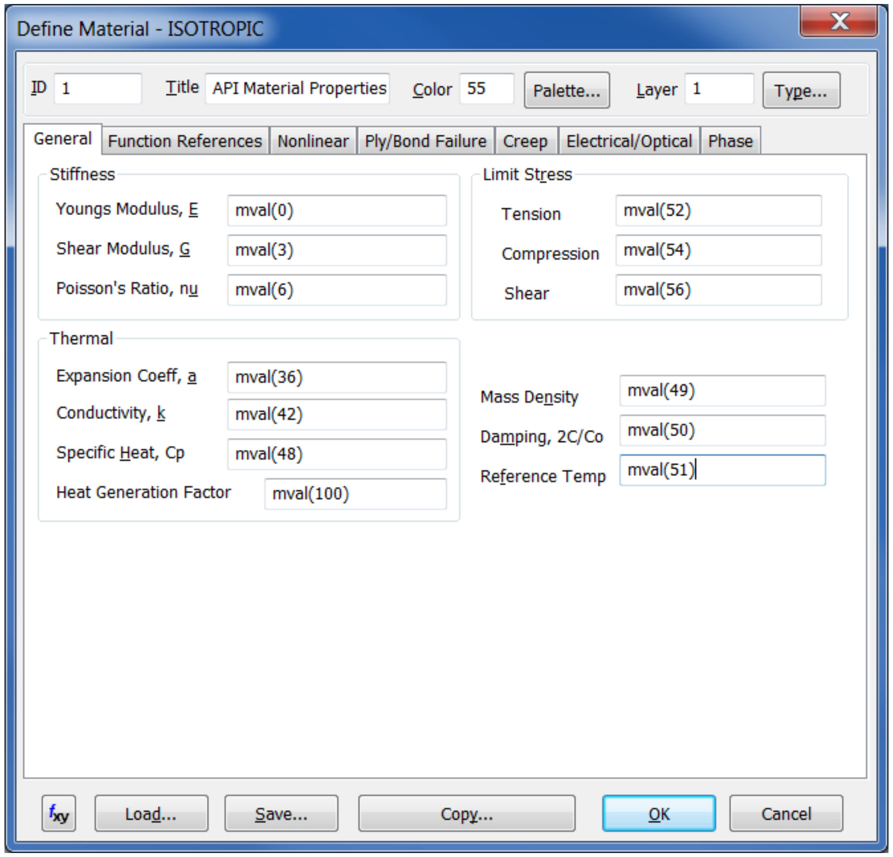


HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)

	HOW OFTEN YOU DO THE TASK					
HOW MUCH TIME YOU SHAVE OFF	50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
30 MINUTES	■	6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
1 HOUR	■	10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
6 HOURS	■	■	■	2 MONTHS	2 WEEKS	1 DAY
1 DAY	■	■	■	■	8 WEEKS	5 DAYS

4.1 Accessing Material Properties

Here's a guide to accessing the various properties of an isotropic material card.



ID	1
Title	API Material Properties
Color	55
Layer	1

Stiffness

Youngs Modulus, E	mval(0)
Shear Modulus, G	mval(3)
Poisson's Ratio, nu	mval(6)

Limit Stress

Tension	mval(52)
Compression	mval(54)
Shear	mval(56)

Thermal

Expansion Coeff, a	mval(36)
Conductivity, k	mval(42)
Specific Heat, Cp	mval(48)
Heat Generation Factor	mval(100)

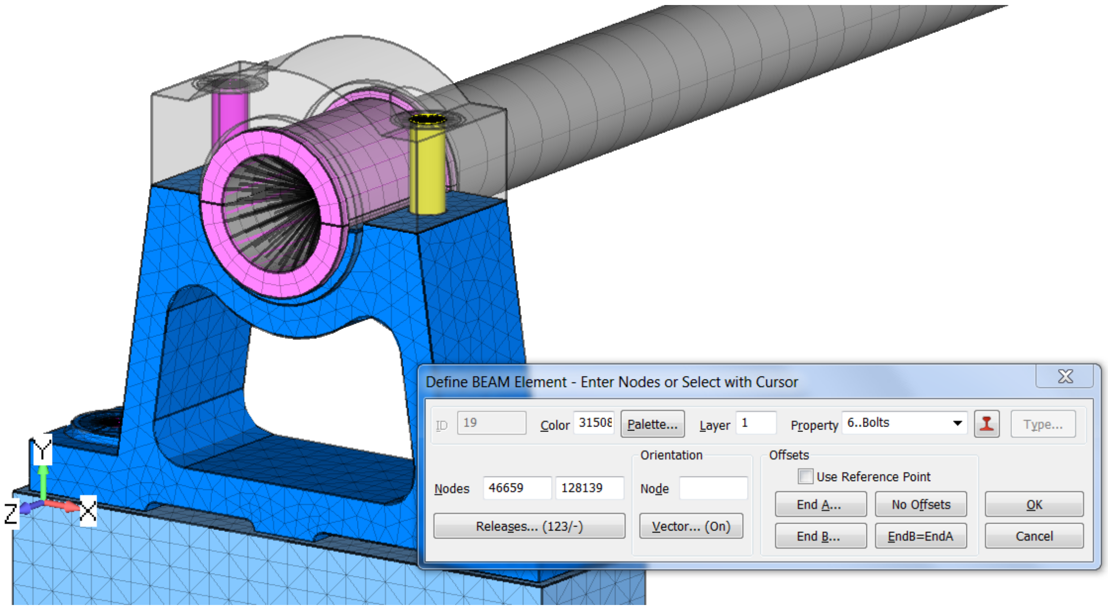
Mass Density	mval(49)
Damping, 2C/Co	mval(50)
Reference Temp	mval(51)

```
Sub Main
 Dim App As femap.model
 Set App = feFemap()

 Dim Mat As femap.Matl
 Set Mat = App.feMatl

 Dim MatSet As femap.Set
 Set MatSet = App.feSet
 Dim MatID As Long

 MatSet.Select(FT_MATL, True, "Select Materials")
 MatID = MatSet.First

 While MatID > 0
  Mat.Get(MatID)
  App.feAppMessage(4, "Mat ID = " + CStr(MatID))
  App.feAppMessage(2, "E = " + CStr(Mat.mval(0)))
  App.feAppMessage(2, "G = " + CStr(Mat.mval(3)))
  App.feAppMessage(2, "v = " + CStr(Mat.mval(6)))
  App.feAppMessage(2, "CTE = " + CStr(Mat.mval(36)))
  App.feAppMessage(2, "k = " + CStr(Mat.mval(42)))
  App.feAppMessage(2, "Cp = " + CStr(Mat.mval(48)))
  App.feAppMessage(2, "Density = " + CStr(Mat.mval(49)))
  App.feAppMessage(2, "Damping = " + CStr(Mat.mval(50)))
  App.feAppMessage(2, "Temp = " + CStr(Mat.mval(51)))
  App.feAppMessage(2, "Tens = " + CStr(Mat.mval(52)))
  App.feAppMessage(2, "Comp = " + CStr(Mat.mval(54)))
  App.feAppMessage(2, "Shear = " + CStr(Mat.mval(56)))
  App.feAppMessage(2, "Heat = " + CStr(Mat.mval(100)))

  MatID = MatSet.Next
 Wend
End Sub
```

4.2 Accessing Element Properties

Fortunately, *element* Properties aren't as complicated and numerous as *property or material* Properties. That is, the majority of the data is held in the FEMAP materials and properties. The most complicated part of defining an element is managing the nodes.

Let's take a look at some of the more straightforward properties like ID, color, layer, etc.:



Define BEAM Element - Enter Nodes or Select with Cursor

ID	19	Color	31508	Layer	1	Property	6..Bolts

Nodes 46659 128139

Orientation: Node (empty), Vector... (On)

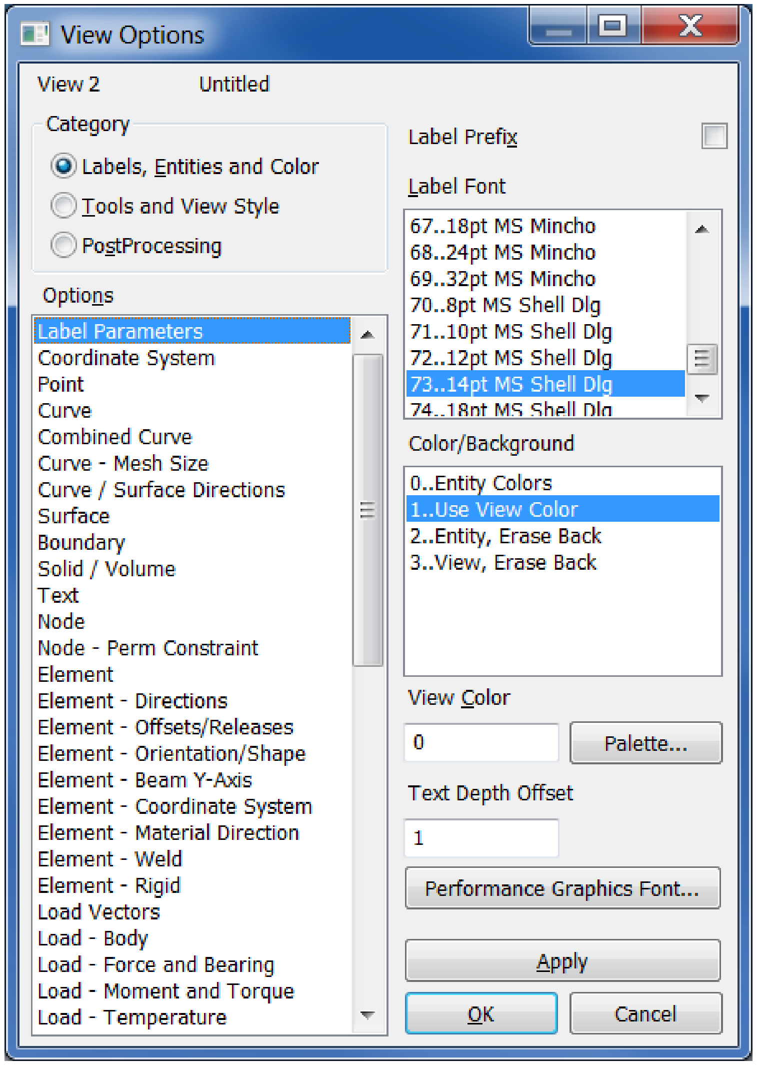
Offsets: Use Reference Point ☐

Releases... (123/-)

```
elemOBJ.ID = 19
elemOBJ.type = FET_L_BEAM
elemOBJ.topology = FTO_LINE2
elemOBJ.color = 31508
elemOBJ.layer = 1
elemOBJ.propID = 6
elemOBJ.Node(0) = 46659
elemOBJ.Node(1) = 128139
elemOBJ.release(0, 0) = 1
elemOBJ.release(0, 1) = 1
elemOBJ.release(0, 2) = 1
elemOBJ.orient(0) = 0
elemOBJ.orient(1) = 0
elemOBJ.orient(2) = 1
```

4.3 ACCESSING VIEW PROPERTIES

This can be a bit confusing because the syntax of View Object Properties differs slightly from other object Properties. Let's look at a few examples


View Options

View 2 Untitled

Category: [✓ Labels, Entities and Color] Tools and View Style PostProcessing

Options: Label Parameters (selected), Coordinate System, Point, Curve, Combined Curve, Curve - Mesh Size, Curve / Surface Directions, Surface, Boundary, Solid / Volume, Text, Node, Node - Perm Constraint, Element, Element - Directions, Element - Offsets/Releases, Element - Orientation/Shape, Element - Beam Y-Axis, Element - Coordinate System, Element - Material Direction, Element - Weld, Element - Rigid, Load Vectors, Load - Body, Load - Force and Bearing, Load - Moment and Torque, Load - Temperature

Label Prefix

Label Font: 67..18pt MS Mincho, 68..24pt MS Mincho, 69..32pt MS Mincho, 70..8pt MS Shell Dlg, 71..10pt MS Shell Dlg, 72..12pt MS Shell Dlg, 73..14pt MS Shell Dlg (selected), 74..18pt MS Shell Dlg

Color/Background: 0..Entity Colors, 1..Use View Color (selected), 2..Entity, Erase Back, 3..View, Erase Back

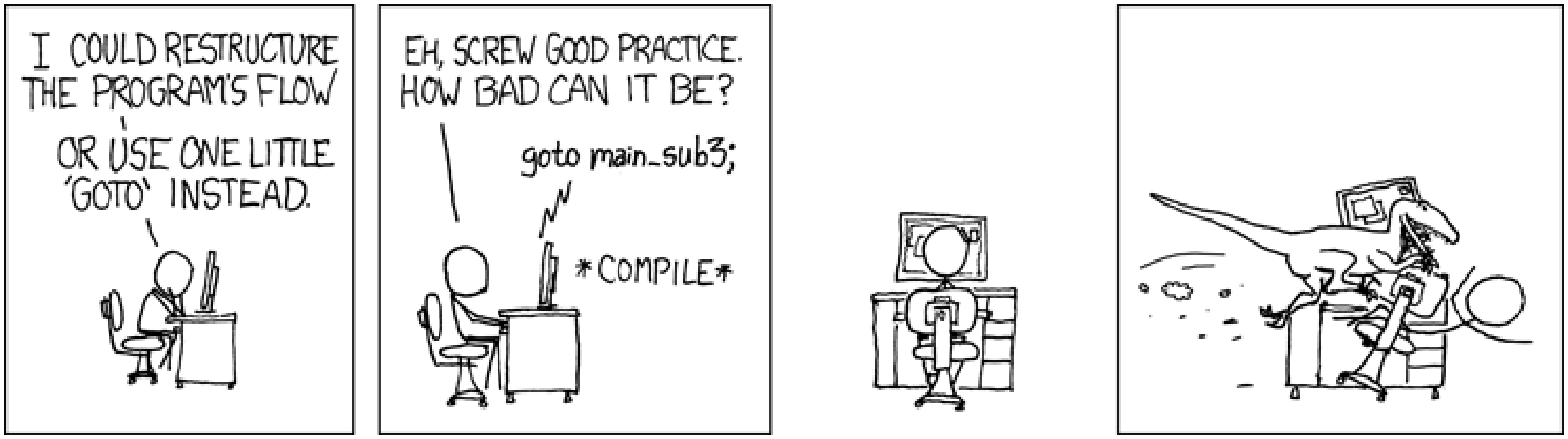
View Color: 0

Text Depth Offset: 1

Text in View Options	Index	Constant
Labels, Entities and Colors		
Label Parameters	0	FVI_LABEL
Coordinate System	1	FVI_CSYS
Point	2	FVI_POINT
Curve	3	FVI_CURVE
Combined Curve	95	FVI_COMPOSITE_CURVE
Curve - Mesh Size	24	FVI_CURVE_MESHSIZE
Curve/Surface Directions	39	FVI_CURVE_ACCURACY
Surface	4	FVI_SURFACE

```
Sub Main
    Dim App As femap.model
    Set App = feFemap()

    Dim viewobj As femap.View
    Set viewobj = App.feView

    Dim viewID As Long
    App.feAppGetActiveView(viewID)

    viewobj.Get(viewID)
    viewobj.Draw(0) = False
    viewobj.Label(0) = 73
    viewobj.ColorMode(0) = 1
    viewobj.color(0) = 1
    viewobj.RenderPushLabel = (1)
    viewobj.Put(viewID)
End Sub
```

5. CONTROLLING PROGRAM FLOW


I COULD RESTRUCTURE THE PROGRAM'S FLOW
OR USE ONE LITTLE 'GOTO' INSTEAD.
EH, SCREW GOOD PRACTICE. HOW BAD CAN IT BE?
goto main_sub3;
COMPILE

5.1 Loops and If-Then Statements

Loops are commonly used to work through entities selected by a set object. “For” loops continue for a specified number of iterations. “While” loops check a condition at the beginning of the loop and continue until the condition is false.

The standard *If-Then* statement is a great way to filter entities (*If* the element is a beam, *then* list ID number) or control the flow of the program (*If* rc is equal to 0, *Then* exit the program). Incorporating *Else* and *ElseIf* statements simply opens up more flow options, reducing the need for multiple nested *If-Then* statements. Let take a look at some examples

Check to see if “nodeID” is greater than zero. If so, perform the specified actions. Repeat until nodeID ≯ 0.

```
Sub Main
 Dim App As femap.model
 Set App = feFemap()

 Dim nodeOBJ As femap.Node
 Set nodeOBJ = App.feNode

 Dim nodeSET As femap.Set
 Set nodeSET = App.feSet
 Dim nodeID As Long

 nodeSET.Select(FT_NODE, True, "Select Nodes")

 nodeID = nodeSET.First
 While nodeID > 0
  nodeOBJ.Get(nodeID)
  App.feAppMessage(2,CStr(nodeOBJ.x))
  App.feAppMessage(2,CStr(nodeOBJ.y))
  App.feAppMessage(2,CStr(nodeOBJ.z))
  nodeID = nodeSET.Next
 Wend

End Sub
```

If the background is black (0) then switch it to white (124), otherwise switch it to black (0).

```
Sub Main
 Dim App As femap.model
 Set App=feFemap()

 Dim vobj As femap.View
 Set vobj=App.feView

 Dim viewID As Long

 App.feAppGetActiveView(viewID)
 vobj.Get(viewID)

 If vobj.WindowBackColor=0 Then
  vobj.WindowBackColor=124
 Else
  vobj.WindowBackColor=0
 End If

 vobj.Put(viewID)
 App.feViewRegenerate(0)

End Sub
```

6. CONNECTING TO FEMAP AND EXCEL

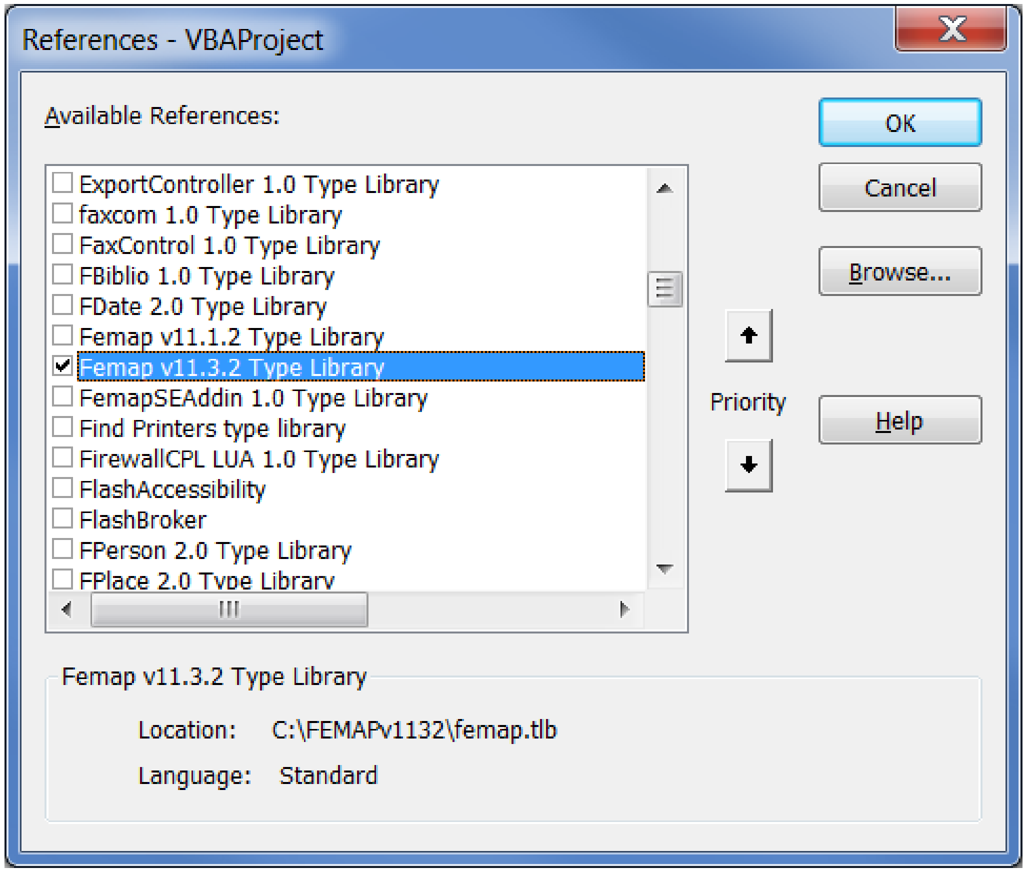
One of the abilities that makes the FEMAP API so powerful is the connection to MS Excel or other external programs. Whether you're dumping results to a spreadsheet or pushing a massive amount of load data into your model, connecting to external databases is a game changer.

Topics Covered:

- Connecting to FEMAP from Excel
- Printing information to an Excel Workbook
- Pulling information from an Excel Workbook

Workflow:

- Open a new Excel worksheet
- Activate the Developer Tab
- Open the Visual Basic Editor
- Import the program from Part 1
- Update the program to get it to cooperate with Excel


References - VBAProject

Available References:

- [ ] ExportController 1.0 Type Library
- [ ] faxcom 1.0 Type Library
- [ ] FaxControl 1.0 Type Library
- [ ] FBiblio 1.0 Type Library
- [ ] FDate 2.0 Type Library
- [ ] Femap v11.1.2 Type Library
- [✓] Femap v11.3.2 Type Library
- [ ] FemapSEAddin 1.0 Type Library
- [ ] Find Printers type library
- [ ] FirewallCPL LUA 1.0 Type Library
- [ ] FlashAccessibility
- [ ] FlashBroker
- [ ] FPerson 2.0 Type Library
- [ ] FPlace 2.0 Type Library

Femap v11.3.2 Type Library

Location: C:\FEMAPv1132\femap.tlb

Language: Standard

Remember this? Why is it important?

Method 1**
 `Dim App As femap.model`
 `Set App = feFemap()`
	• Does not work from VB in Excel.
	 • Provides autocomplete to assist with programming
	 • Ensures connection to current FEMAP session

Method 2*
 `Dim App As femap.model`
 `Set App = GetObject(,"femap.model")`
	• Requires the user to add the Femap Type Library to the Excel Workbook References.
	 • Provides autocomplete to assist with programming

Method 3
 `Dim App As Object`
 `Set App = GetObject(, "femap.model")`
	• Does not require special setup in Excel (References)
	 • There's no autocomplete

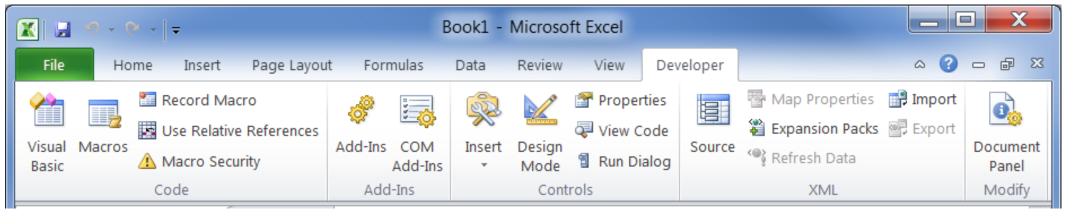
Important information from the FEMAP help file:

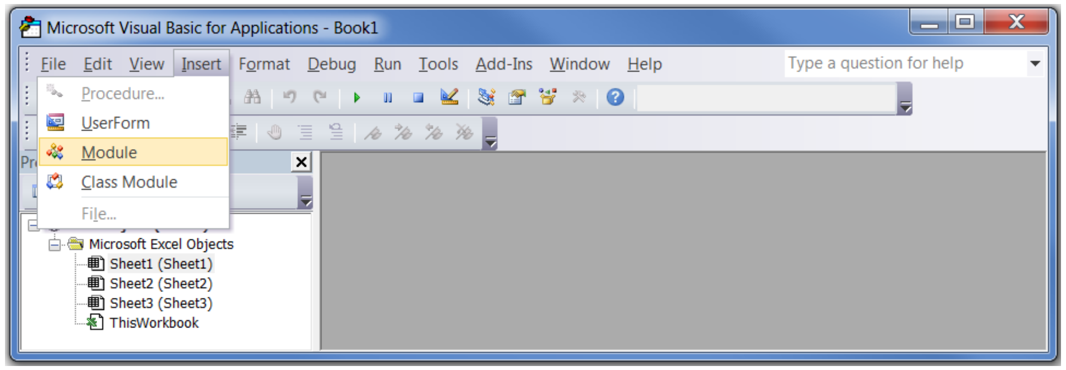
"When you are connecting to an existing session there are some special considerations. If you are programming in an environment outside of FEMAP (anything but the FEMAP API Programming Window), then you must use "GetObject" to connect. This uses standard OLE/COM mechanisms to find the FEMAP object, but imposes a limitation. If you have multiple copies of FEMAP running (multiple processes, not multiple models open in the same FEMAP), GetObject will always connect you to the process that was started first. This is simply a limitation of the OLE/COM interface and can not be avoided.

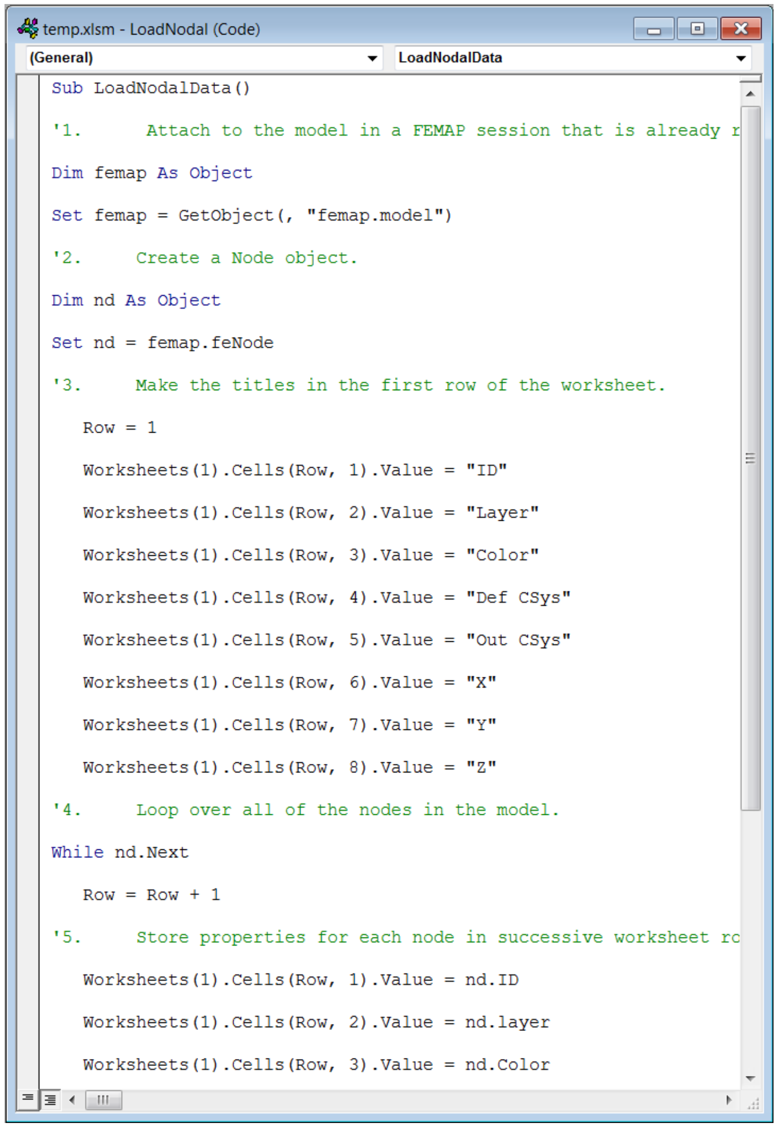
If you are using the integrated WinWrap environment (API Programming) however, you can overcome this limitation by never using "GetObject". Instead, you can connect to the current FEMAP session, no matter how many FEMAP processes are running by using"

Excel > Developer > Visual Basic > Insert > Module

From the Developer tab of Excel, click on the Visual Basic icon. Once the Visual Basic window opens, click Insert > Module. Paste the program into the Module. You can now close Visual Basic.



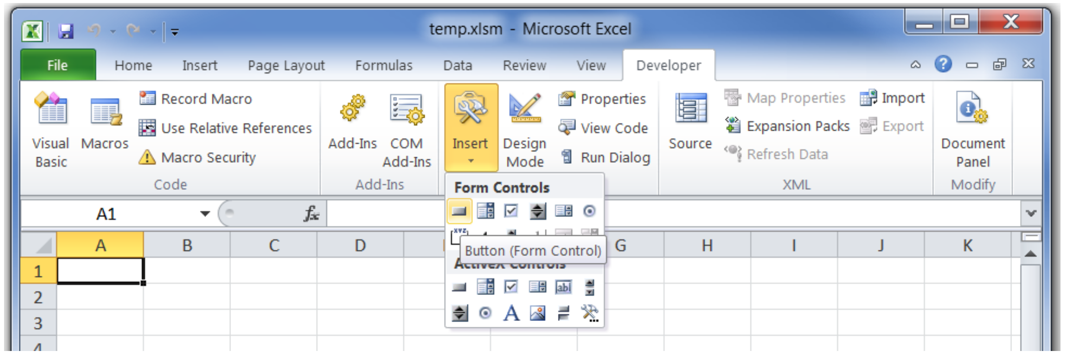


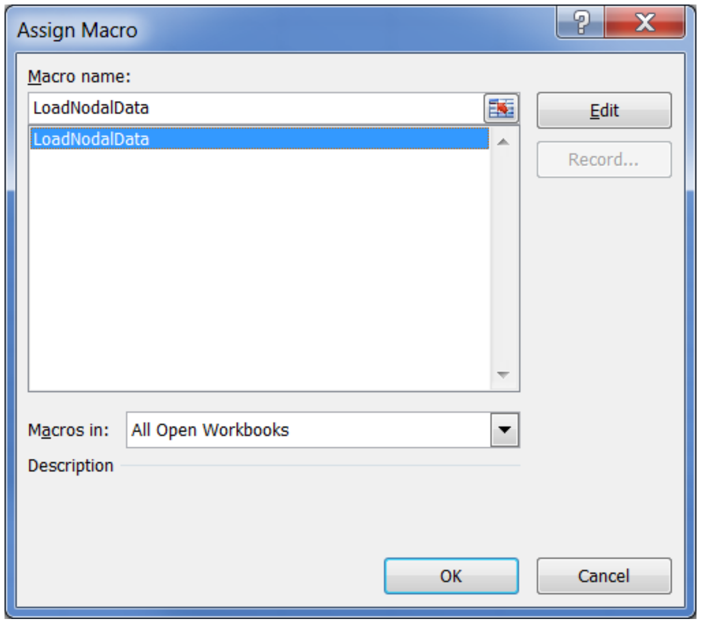


```
Sub LoadNodalData()
'1.      Attach to the model in a FEMAP session that is already r
Dim femap As Object
Set femap = GetObject(, "femap.model")
'2.     Create a Node object.
Dim nd As Object
Set nd = femap.feNode
'3.     Make the titles in the first row of the worksheet.
   Row = 1
   Worksheets(1).Cells(Row, 1).Value = "ID"
   Worksheets(1).Cells(Row, 2).Value = "Layer"
   Worksheets(1).Cells(Row, 3).Value = "Color"
   Worksheets(1).Cells(Row, 4).Value = "Def CSys"
   Worksheets(1).Cells(Row, 5).Value = "Out CSys"
   Worksheets(1).Cells(Row, 6).Value = "X"
   Worksheets(1).Cells(Row, 7).Value = "Y"
   Worksheets(1).Cells(Row, 8).Value = "Z"
'4.     Loop over all of the nodes in the model.
While nd.Next
   Row = Row + 1
'5.     Store properties for each node in successive worksheet r
   Worksheets(1).Cells(Row, 1).Value = nd.ID
   Worksheets(1).Cells(Row, 2).Value = nd.layer
   Worksheets(1).Cells(Row, 3).Value = nd.Color
```

Excel > Developer > Insert > Form Controls

Insert a button into the spreadsheet. Click and drag to set the button size. When you let go of the mouse button, an “Assign Macro” dialog box will appear. Select the LoadNodalData program that you just created in Visual Basic. The button is now linked to the program and is fully functional. Rename the button by right clicking on the button and modifying the text.

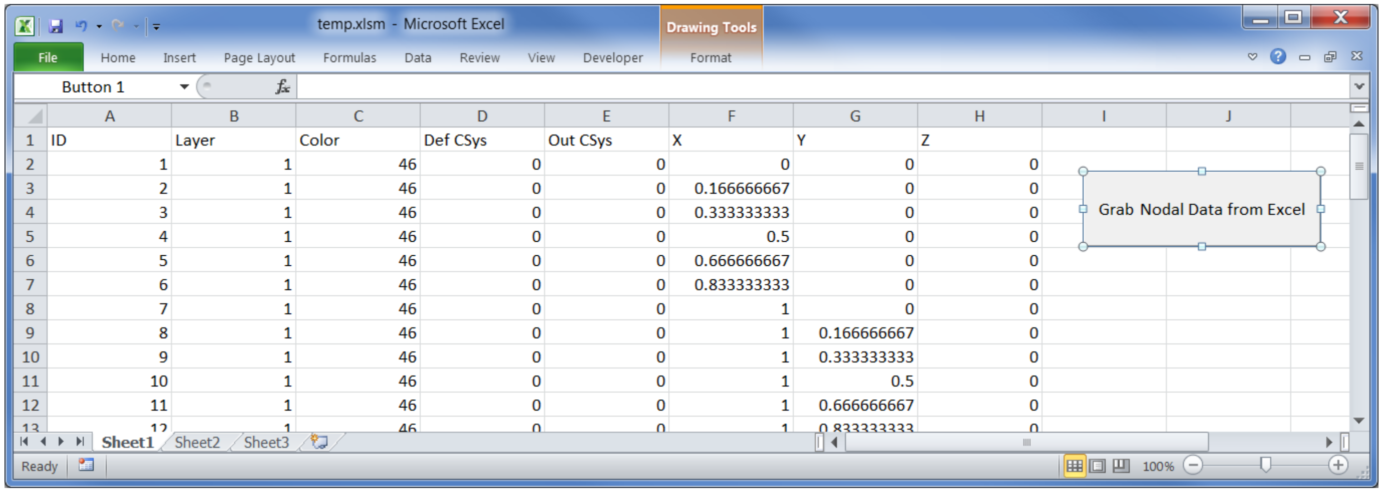




Macro name: LoadNodalData

LoadNodalData

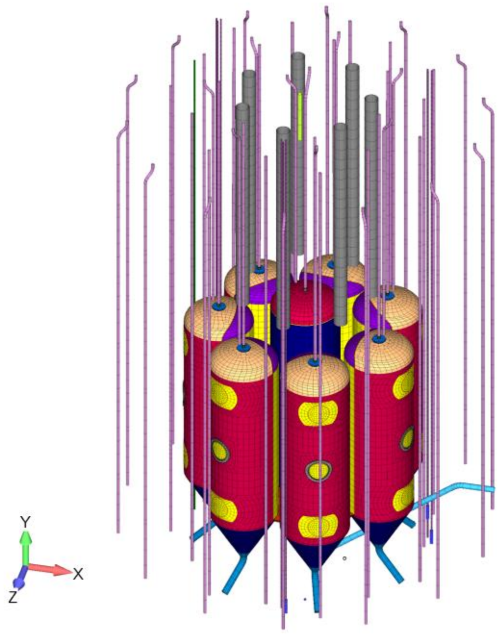
Macros in: All Open Workbooks

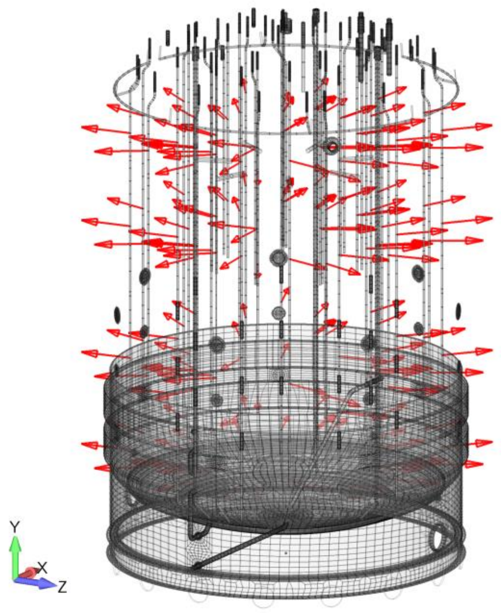


ID	Layer	Color	Def CSys	Out CSys	X	Y	Z
1	1	46	0	0	0	0	0
2	1	46	0	0	0.166666667	0	0
3	1	46	0	0	0.333333333	0	0
4	1	46	0	0	0.5	0	0
5	1	46	0	0	0.666666667	0	0
6	1	46	0	0	0.833333333	0	0
7	1	46	0	0	1	0	0
8	1	46	0	0	1	0.166666667	0
9	1	46	0	0	1	0.333333333	0
10	1	46	0	0	1	0.5	0
11	1	46	0	0	1	0.666666667	0
12	1	46	0	0	1	0.833333333	0

6.1 SHOW AND TELL: PRESSURE VESSEL PIPING LOAD APPLICATOR

Years ago, I had a pressure vessel project…THE pressure vessel project. From the geometry, to the loads, to the documentation requirements, the complexity was endless. One example of a mind-numbing task was applying the internal piping loads. There were eight internal vessels (pulse jet mixers, PJMs) within the parent vessel and it was possible for each one to put load on the internal piping of the vessel. For each pipe, the load acted on a vector from the centerline of the PJM to the centerline of the pipe. The magnitude of the load was a function of height and distance from the PJM. It was brutal and the FEMAP API was my saving grace.



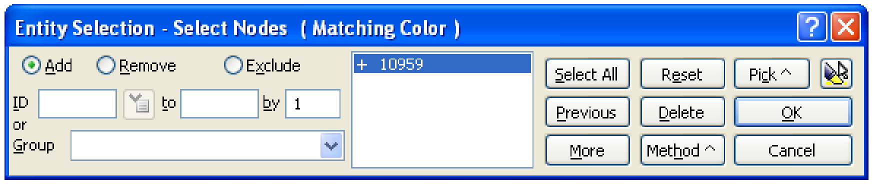


The first step was extracting nodes from the FEMAP model. We needed node ID number and coordinates. This information was used in combination to calculate the direction and magnitude of the load. The next step was to push the loads calculated in the Excel workbook, back into the FEMAP database. The process takes about 30 seconds with the API. Applying the loads manually takes about 8 hours.

Here's the workflow:

Click the "Gather Nodal Data" Button, Switch to the FEMAP Window

When you switch to the FEMAP window, you will see a nodal entity selection dialog box waiting for you. Select some of the pink nodes on the beam elements. The color of the nodes centered between supports has been modified to make them easier to find. To select all of the pink nodes, use the "Color" selection method.


Entity Selection - Select Nodes (Matching Color)

Add [✓] Remove Exclude

ID ___ to ___ by 1

or

Group

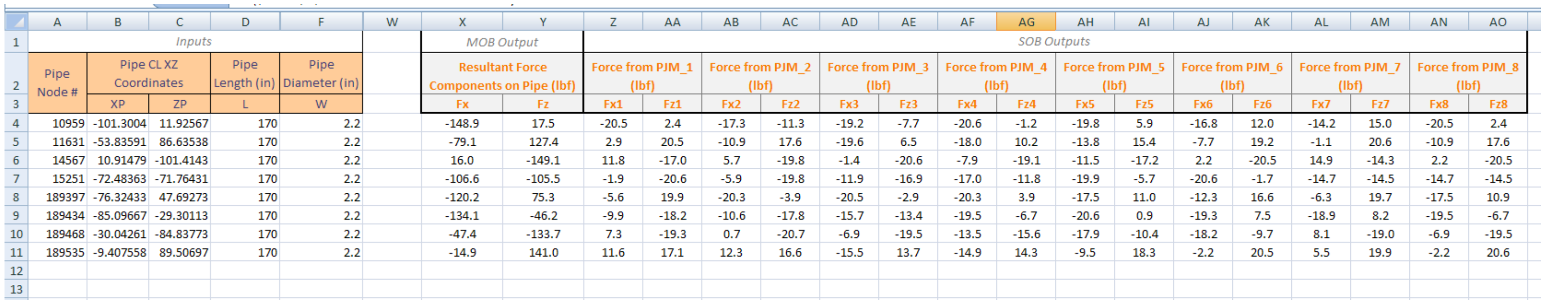
+ 10959

Switch to the Excel Spreadsheet

Data has been written into the spread sheet for the nodes you selected in FEMAP.

Enter Additional Data into Excel

The spread sheet requires a pipe length and pipe diameter to calculate loads. For this particular model, the pipe is 170" by 2.2". Now that the spreadsheet has created loads we will use another program to apply them to the FEA model.



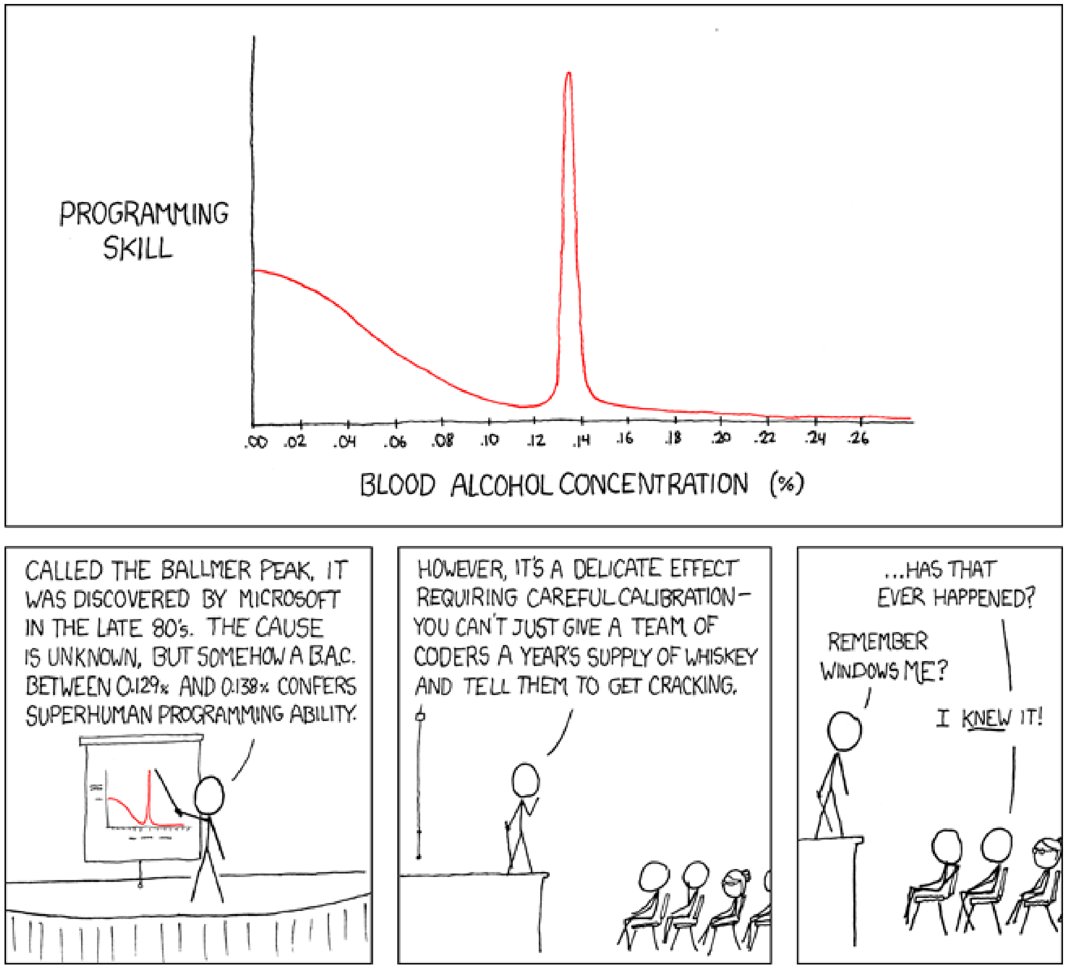
Inputs					MOB Output	
Pipe Node #	Pipe CL XZ Coordinates		Pipe Length (in)	Pipe Diameter (in)	Resultant Force Components on Pipe (lbf)	
	XP	ZP	L	W	Fx	Fz
10959	-101.3004	11.92567	170	2.2	-148.9	17.5
11631	-53.83591	86.63538	170	2.2	-79.1	127.4
14567	10.91479	-101.4143	170	2.2	16.0	-149.1
15251	-72.48363	-71.76431	170	2.2	-106.6	-105.5
189397	-76.32433	47.69273	170	2.2	-120.2	75.3
189434	-85.09667	-29.30113	170	2.2	-134.1	-46.2
189468	-30.04261	-84.83773	170	2.2	-47.4	-133.7
189535	-9.407558	89.50697	170	2.2	-14.9	141.0

SOB Outputs															
Force from PJM_1 (lbf)		Force from PJM_2 (lbf)		Force from PJM_3 (lbf)		Force from PJM_4 (lbf)		Force from PJM_5 (lbf)		Force from PJM_6 (lbf)		Force from PJM_7 (lbf)		Force from PJM_8 (lbf)	
Fx1	Fz1	Fx2	Fz2	Fx3	Fz3	Fx4	Fz4	Fx5	Fz5	Fx6	Fz6	Fx7	Fz7	Fx8	Fz8
-20.5	2.4	-17.3	-11.3	-19.2	-7.7	-20.6	-1.2	-19.8	5.9	-16.8	12.0	-14.2	15.0	-20.5	2.4
2.9	20.5	-10.9	17.6	-19.6	6.5	-18.0	10.2	-13.8	15.4	-7.7	19.2	-1.1	20.6	-10.9	17.6
11.8	-17.0	5.7	-19.8	-1.4	-20.6	-7.9	-19.1	-11.5	-17.2	2.2	-20.5	14.9	-14.3	2.2	-20.5
-1.9	-20.6	-5.9	-19.8	-11.9	-16.9	-17.0	-11.8	-19.9	-5.7	-20.6	-1.7	-14.7	-14.5	-14.7	-14.5
-5.6	19.9	-20.3	-3.9	-20.5	-2.9	-20.3	3.9	-17.5	11.0	-12.3	16.6	-6.3	19.7	-17.5	10.9
-9.9	-18.2	-10.6	-17.8	-15.7	-13.4	-19.5	-6.7	-20.6	0.9	-19.3	7.5	-18.9	8.2	-19.5	-6.7
7.3	-19.3	0.7	-20.7	-6.9	-19.5	-13.5	-15.6	-17.9	-10.4	-18.2	-9.7	8.1	-19.0	-6.9	-19.5
11.6	17.1	12.3	16.6	-15.5	13.7	-14.9	14.3	-9.5	18.3	-2.2	20.5	5.5	19.9	-2.2	20.6

Click the Button, Switch to the FEMAP Window

The API has updated the FEMAP database with the loads from Excel but they might not show up in the Model Info Tree. Regenerating the image (Window > Regenerate, Ctrl G) will update the Model Info Tree with the new loads.

7. POST PROCESSING


PROGRAMMING SKILL
.00 .02 .04 .06 .08 .10 .12 .14 .16 .18 .20 .22 .24 .26
BLOOD ALCOHOL CONCENTRATION (%)
CALLED THE BALLMER PEAK, IT WAS DISCOVERED BY MICROSOFT IN THE LATE 80'S. THE CAUSE IS UNKNOWN, BUT SOMEHOW A B.A.C. BETWEEN 0.129% AND 0.138% CONFERS SUPERHUMAN PROGRAMMING ABILITY.
HOWEVER, IT'S A DELICATE EFFECT REQUIRING CAREFUL CALIBRATION – YOU CAN'T JUST GIVE A TEAM OF CODERS A YEAR'S SUPPLY OF WHISKEY AND TELL THEM TO GET CRACKING.
...HAS THAT EVER HAPPENED?
REMEMBER WINDOWS ME?
I KNEW IT!

7.1 Free Body Diagram Generator

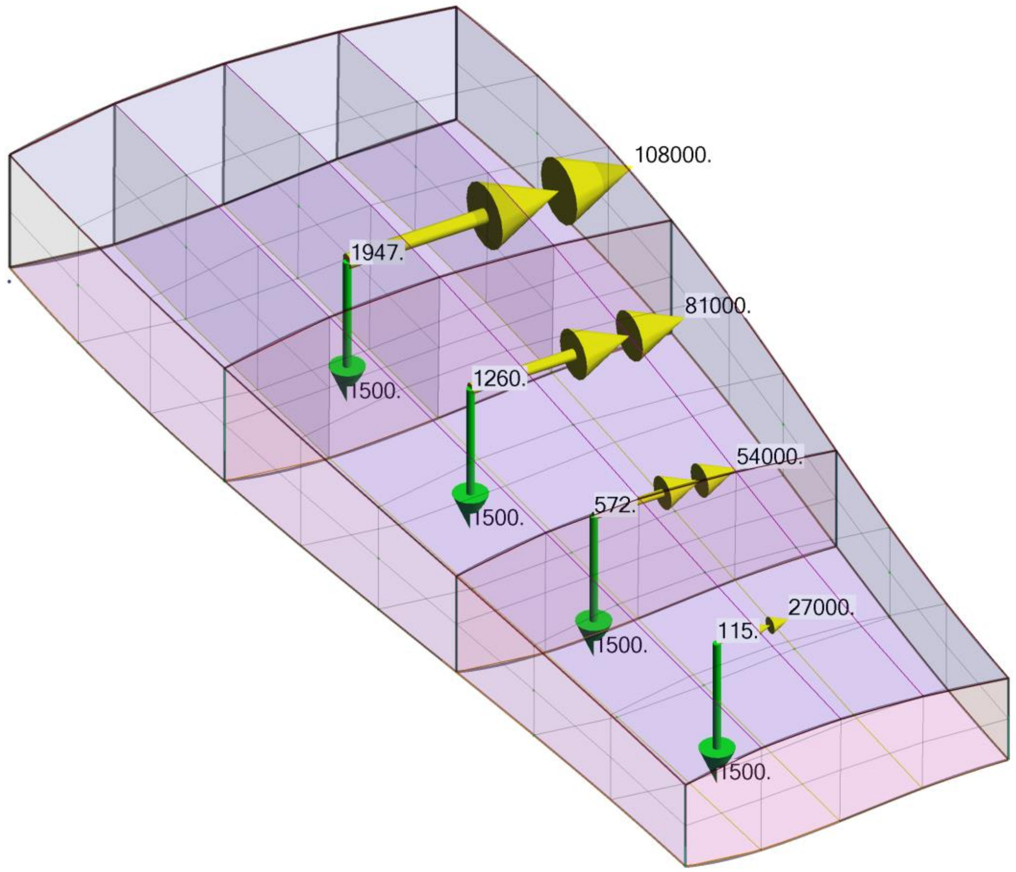
Freebody Diagrams (or FBDs) are an advanced post technique, yet they are beautifully simple at the same time. It's all about summing the forces and moments for a selection of elements and nodes. The trick is carefully selecting the entities and making sure that the "Freebody Contributions" are logical. "Section Cut" display mode operates in the same manner as the Interface Load, but rather than selecting elements and nodes, the user simply selects a cutting plane. FEMAP will automatically select nodes along the cutting plane and elements on one side of the plane.

Topics Covered:

- FBD Object Properties and Methods

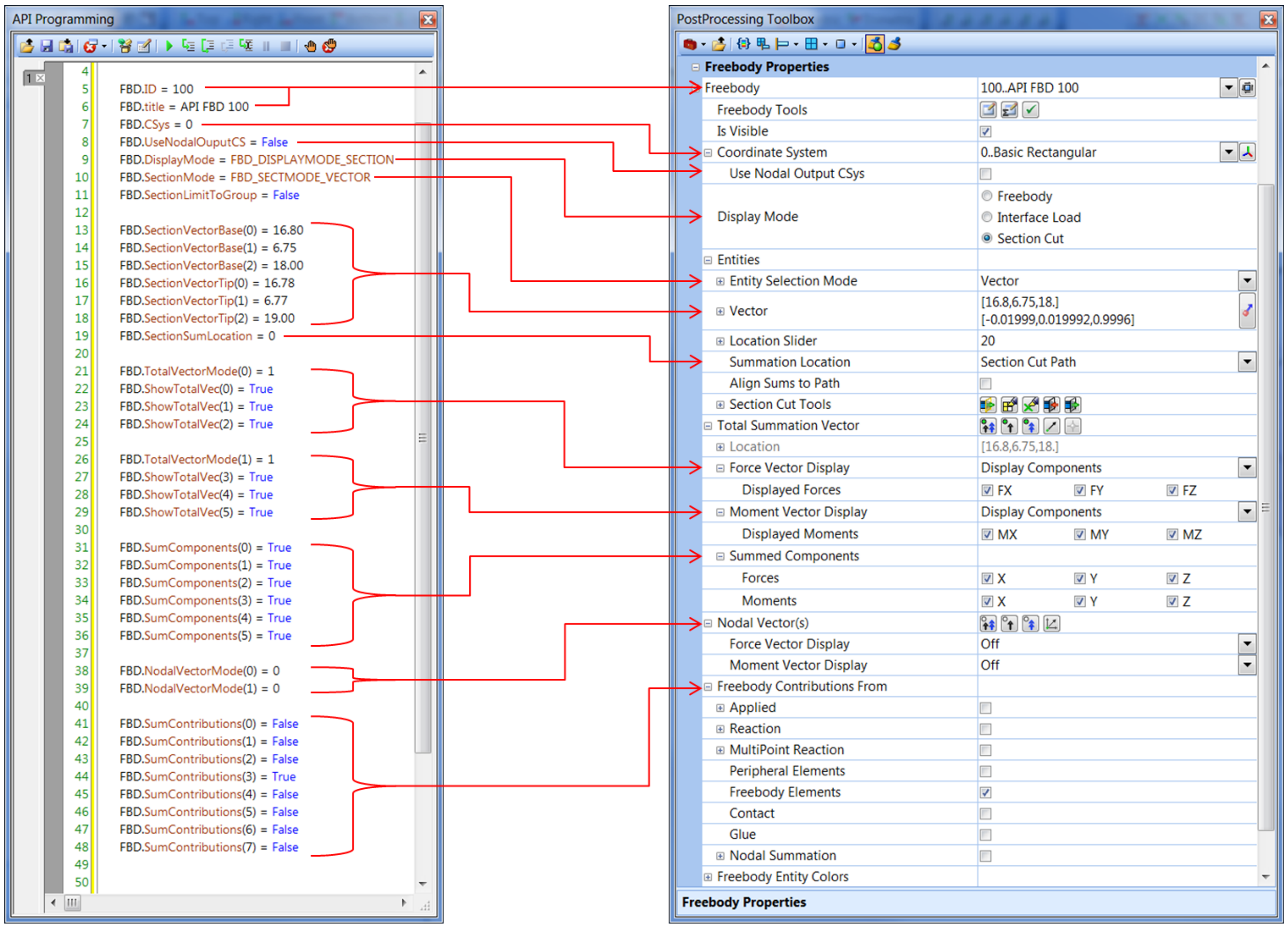
Workflow:

- Prompt the user to select a vector (this will be the normal vector of the section cut plane)
- Print the vector information to Excel
- Dimension a FBD Object
- Hardcode some of the FBD Object Properties
- Define other FBD Object Properties using information from Excel
- Put the FBDs into FEMAP database
- Extract FBD force data from the FEMAP data and print to Excel



7.2 Freebody Properties

The most common problems with freebody diagrams are selecting entities and choosing the correct contributions. Using the Section Cut display mode eliminates the former while hard coding FBD options into an API eliminates the later.



```
FBD.ID = 100
FBD.title = API FBD 100
FBD.CSys = 0
FBD.UseNodalOuputCS = False
FBD.DisplayMode = FBD_DISPLAYMODE_SECTION
FBD.SectionMode = FBD_SECTMODE_VECTOR
FBD.SectionLimitToGroup = False

FBD.SectionVectorBase(0) = 16.80
FBD.SectionVectorBase(1) = 6.75
FBD.SectionVectorBase(2) = 18.00
FBD.SectionVectorTip(0) = 16.78
FBD.SectionVectorTip(1) = 6.77
FBD.SectionVectorTip(2) = 19.00
FBD.SectionSumLocation = 0

FBD.TotalVectorMode(0) = 1
FBD.ShowTotalVec(0) = True
FBD.ShowTotalVec(1) = True
FBD.ShowTotalVec(2) = True

FBD.TotalVectorMode(1) = 1
FBD.ShowTotalVec(3) = True
FBD.ShowTotalVec(4) = True
FBD.ShowTotalVec(5) = True

FBD.SumComponents(0) = True
FBD.SumComponents(1) = True
FBD.SumComponents(2) = True
FBD.SumComponents(3) = True
FBD.SumComponents(4) = True
FBD.SumComponents(5) = True

FBD.NodalVectorMode(0) = 0
FBD.NodalVectorMode(1) = 0

FBD.SumContributions(0) = False
FBD.SumContributions(1) = False
FBD.SumContributions(2) = False
FBD.SumContributions(3) = True
FBD.SumContributions(4) = False
FBD.SumContributions(5) = False
FBD.SumContributions(6) = False
FBD.SumContributions(7) = False
```

Freebody Properties

Freebody	100..API FBD 100
Freebody Tools	
Is Visible	✓
Coordinate System	0..Basic Rectangular
Use Nodal Output CSys	☐
Display Mode	○ Freebody ○ Interface Load ● Section Cut
Entities	
Entity Selection Mode	Vector
Vector	[16.8,6.75,18.] [-0.01999,0.019992,0.9996]
Location Slider	20
Summation Location	Section Cut Path
Align Sums to Path	☐
Section Cut Tools	
Total Summation Vector	
Location	[16.8,6.75,18.]
Force Vector Display	Display Components
Displayed Forces	✓ FX ✓ FY ✓ FZ
Moment Vector Display	Display Components
Displayed Moments	✓ MX ✓ MY ✓ MZ
Summed Components	
Forces	✓ X ✓ Y ✓ Z
Moments	✓ X ✓ Y ✓ Z
Nodal Vector(s)	
Force Vector Display	Off
Moment Vector Display	Off
Freebody Contributions From	
Applied	☐
Reaction	☐
MultiPoint Reaction	☐
Peripheral Elements	☐
Freebody Elements	✓
Contact	☐
Glue	☐
Nodal Summation	☐
Freebody Entity Colors	

Freebody Properties

8. ANATOMY OF A SIMPLE API

```
Sub Main

Dim App As Object
Set App = GetObject(,"femap.model")

Dim entitySet As Object
Set entitySet = App.feSet

Dim entityType As Long
entityType = 7

Dim clearSet As Boolean
clearSet = True

Dim messageString As String
messageString = "Please Select the Nodes You Would Like To Move”

rc = entitySet.Select(entityType, clearSet, messageString)

Dim setID As Long
setID = entitySet.ID

Dim vecMove(3) As Double
vecMove(0) = 10
vecMove(1) = 0
vecMove(2) = 0

Dim vecLength As Double
rc = App.feVectorLength(vecMove, vecLength)

rc = App.feMoveBy(entityType, setID, False, vecLength, vecMove)

End Sub
```

Now we’ll walk through a simple API. All this API does is move a set of selected nodes 10 units in the x direction.

Yes, there is a function that will do this directly without an API, but we are starting simple. The entire script is shown on the left. We will walk through each step in this API.

8.1 DEFINING AN OBJECT

```
Sub Main

Dim App As Object
Set App = GetObject(,"femap.model")

Dim entitySet As Object
Set entitySet = App.feSet

Dim entityType As Long
entityType = 7

Dim clearSet As Boolean
clearSet = True

Dim messageString As String
messageString = "Please Select the Nodes You Would Like To Move”

rc = entitySet.Select(entityType, clearSet, messageString)

Dim setID As Long
setID = entitySet.ID

Dim vecMove(3) As Double
vecMove(0) = 10
vecMove(1) = 0
vecMove(2) = 0

Dim vecLength As Double
rc = App.feVectorLength(vecMove, vecLength)

rc = App.feMoveBy(entityType, setID, False, vecLength, vecMove)

End Sub
```

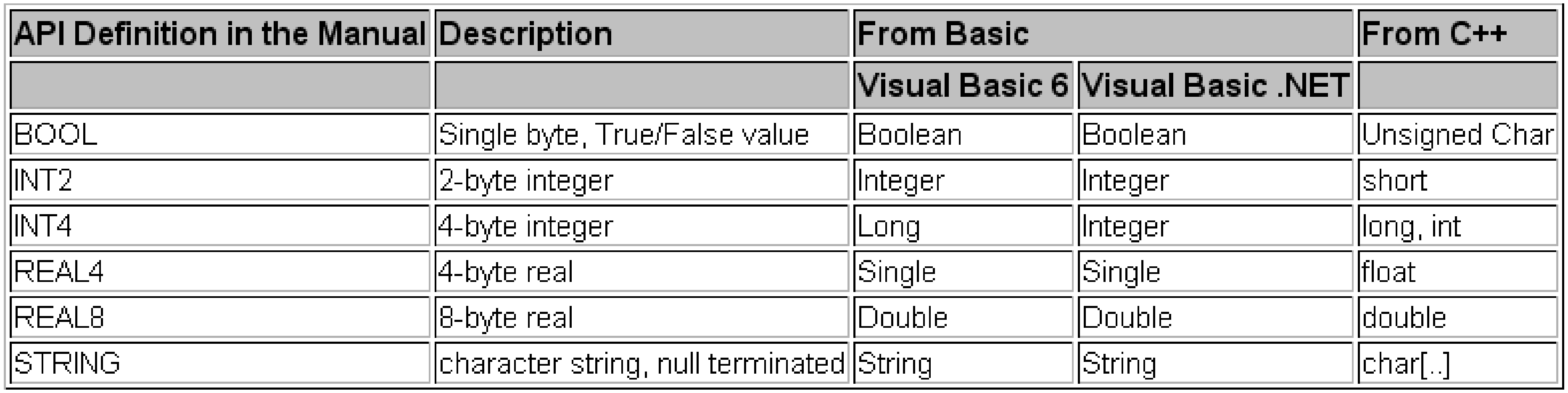
Sub Main at the top of the script signifies that what follows is the main program.

Next we create an object called “*App*”. We then set this object equal to the current FEMAP session. Essentially, this creates the ***FEMAP Application Object*** and calls it *App*. So the object *App* now has all the Properties of the ***FEMAP Application Object***.

What we want to do will also require the help of another object, the ***Set Object***. In this program, we are calling it *“entitySet”* and creating it using the ***feSet Method***. *entitySet* now has all the Properties and Methods inherent in the ***Set Object***.

8.2 DATA TYPES

Visual Basic requires the programmer to declare all variables before they are used as well as what type of data they will be. The six data types are shown below. WinWrap corresponds to the Visual Basic 6 data types. This table can be found in FEMAP API Help Section 1.2 “Data Types”.



API Definition in the Manual	Description	From Basic		From C++
		Visual Basic 6	Visual Basic .NET	
BOOL	Single byte, True/False value	Boolean	Boolean	Unsigned Char
INT2	2-byte integer	Integer	Integer	short
INT4	4-byte integer	Long	Integer	long, int
REAL4	4-byte real	Single	Single	float
REAL8	8-byte real	Double	Double	double
STRING	character string, null terminated	String	String	char[..]

8.3 DIMENSIONING VARIABLES

```
Sub Main

Dim App As Object
Set App = GetObject(,"femap.model")

Dim entitySet As Object
Set entitySet = App.feSet

Dim entityType As Long
entityType = 7

Dim clearSet As Boolean
clearSet = True

Dim messageString As String
messageString = "Please Select the Nodes You Would Like To Move”

rc = entitySet.Select(entityType, clearSet, messageString)

Dim setID As Long
setID = entitySet.ID

Dim vecMove(3) As Double
vecMove(0) = 10
vecMove(1) = 0
vecMove(2) = 0

Dim vecLength As Double
rc = App.feVectorLength(vecMove, vecLength)

rc = App.feMoveBy(entityType, setID, False, vecLength, vecMove)

End Sub
```

Next we declare all the variables needed ***Select Method*** of the ***Set Object*** (this will be described in more detail on subsequent pages).

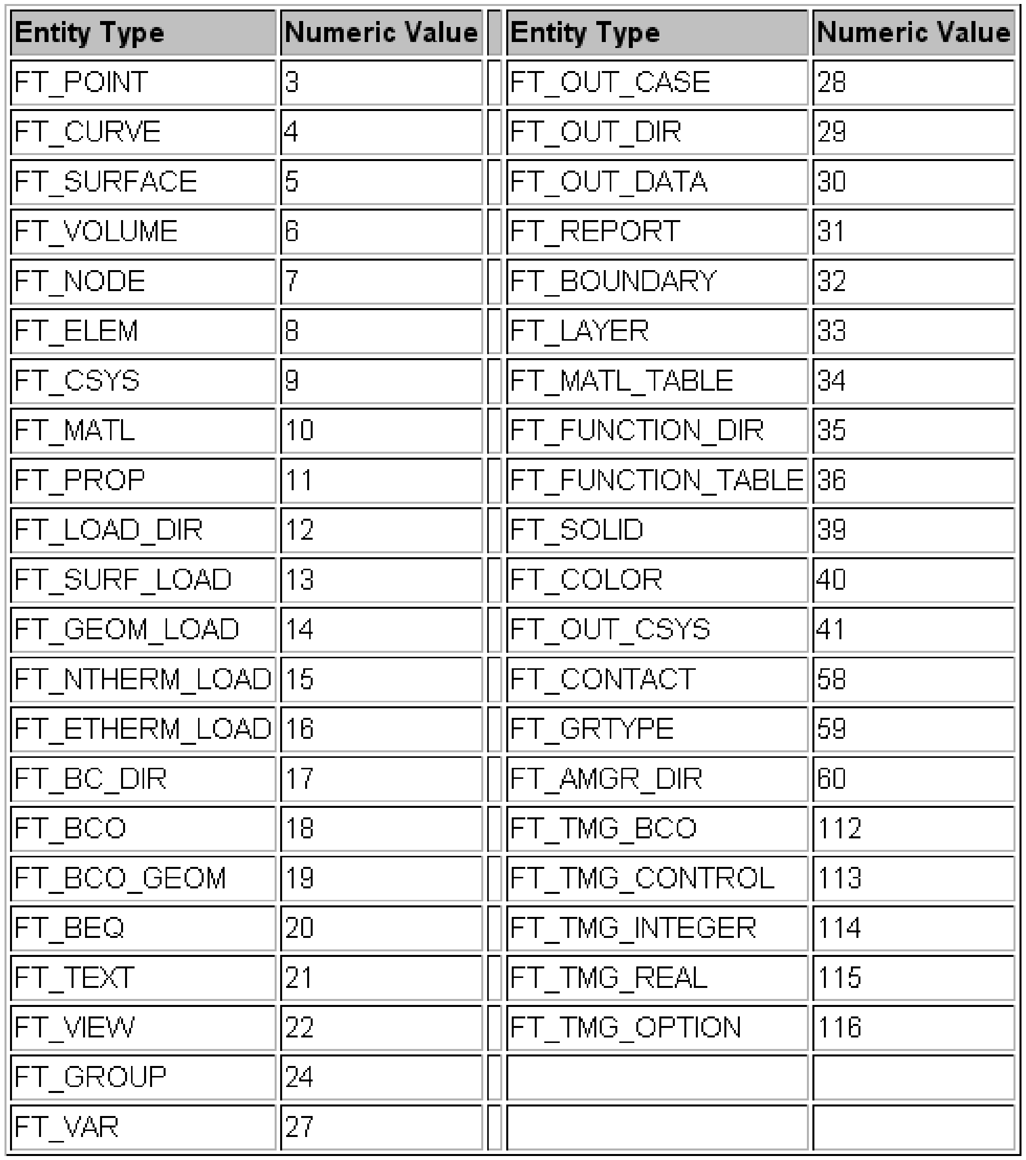
First, we declare a variable called “*entityType”* as a 4-byte integer data type (*Long*) and assign it a value. This will be used to specify what type of entity we will select with the ***Select Method***.

Then, we declare a variable called “*clearSet”* as a true/false value data type (*Boolean*) and assign it a value. This will be used to indicate that we want the ***Set Object*** cleared of any previous entities.

Finally, we declare a variable called *“messageString”* as a character string data type (*String*) and assign it a value. This will be the message the user sees when the ***Set Object*** uses the ***Select Method***.

8.4 ENTITY TYPES

Each entity in the FEMAP API is identified by a name and a number. The entity can be referred to by either. In the preceding piece of code where I refer to the node entity as the number 7, I could also have referred to it as FT_NODE. Either way the API will know to which entity type you are referring.



Entity Type	Numeric Value	Entity Type	Numeric Value
FT_POINT	3	FT_OUT_CASE	28
FT_CURVE	4	FT_OUT_DIR	29
FT_SURFACE	5	FT_OUT_DATA	30
FT_VOLUME	6	FT_REPORT	31
FT_NODE	7	FT_BOUNDARY	32
FT_ELEM	8	FT_LAYER	33
FT_CSYS	9	FT_MATL_TABLE	34
FT_MATL	10	FT_FUNCTION_DIR	35
FT_PROP	11	FT_FUNCTION_TABLE	36
FT_LOAD_DIR	12	FT_SOLID	39
FT_SURF_LOAD	13	FT_COLOR	40
FT_GEOM_LOAD	14	FT_OUT_CSYS	41
FT_NTHERM_LOAD	15	FT_CONTACT	58
FT_ETHERM_LOAD	16	FT_GRTYPE	59
FT_BC_DIR	17	FT_AMGR_DIR	60
FT_BCO	18	FT_TMG_BCO	112
FT_BCO_GEOM	19	FT_TMG_CONTROL	113
FT_BEQ	20	FT_TMG_INTEGER	114
FT_TEXT	21	FT_TMG_REAL	115
FT_VIEW	22	FT_TMG_OPTION	116
FT_GROUP	24		
FT_VAR	27		

8.5 Using the Capabilities of an Object

```
Sub Main

Dim App As Object
Set App = GetObject(,"femap.model")

Dim entitySet As Object
Set entitySet = App.feSet

Dim entityType As Long
entityType = 7

Dim clearSet As Boolean
clearSet = True

Dim messageString As String
messageString = "Please Select the Nodes You Would Like To Move”

rc = entitySet.Select(entityType, clearSet, messageString)

Dim setID As Long
setID = entitySet.ID

Dim vecMove(3) As Double
vecMove(0) = 10
vecMove(1) = 0
vecMove(2) = 0

Dim vecLength As Double
rc = App.feVectorLength(vecMove, vecLength)

rc = App.feMoveBy(entityType, setID, False, vecLength, vecMove)

End Sub
```

Objects have two types of capabilities:

- Methods (Functionality)
- Properties (Data)

Now that we have defined all of the variables that will be used as inputs, we want to do is collect from the user – what nodes they would like moved.

The ***Set Object*** has a handy Method that allows us to do this called ***Select***.

8.6 Object Method Syntax

rc = object.method(inputs, output)

The syntax in the above statement is standard. The object is what is being “asked” to act. The Method is what the object is being asked to do. The inputs are what the object needs in order to execute the Method. The output is what the object will “produce”, although often times, Methods will have no output.

The term rc is the return code and will generate a specific value depending on a number of object success states.

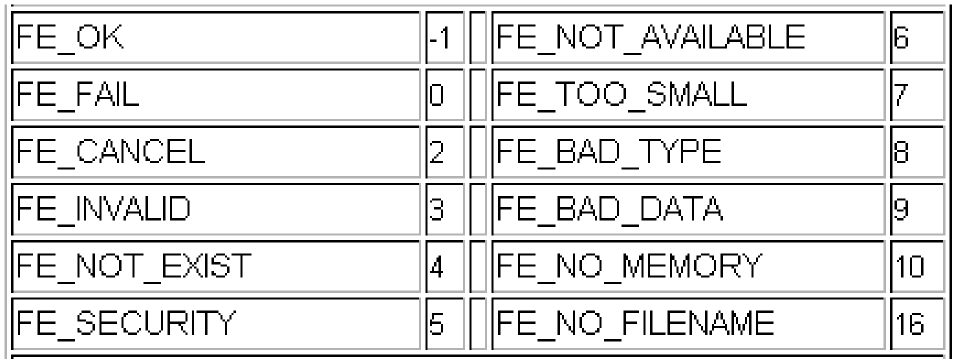
For more information, see FEMAP API Help Section 5.1.2, “Common Entity Methods”.

8.7 Return Codes

Often statements like the following are found in API’s:

rc = object.Method(inputs,output)

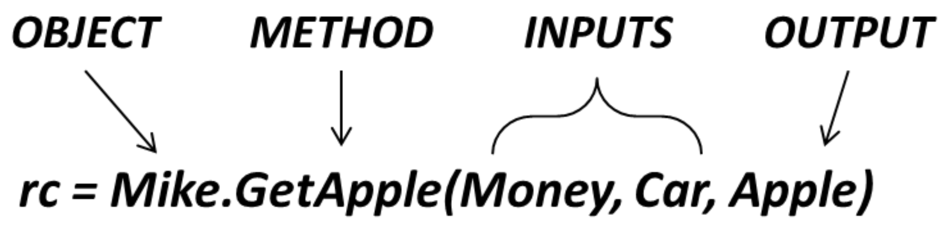
The rc stands for return code. After the object executes its Method, it returns a code that corresponds to its success in executing the Method. If the object is successful, a -1 is returned. If it is not successful, something else will be returned depending upon what went wrong. All the return codes are found in the table on the right.



FE_OK	-1		FE_NOT_AVAILABLE	6
FE_FAIL	0		FE_TOO_SMALL	7
FE_CANCEL	2		FE_BAD_TYPE	8
FE_INVALID	3		FE_BAD_DATA	9
FE_NOT_EXIST	4		FE_NO_MEMORY	10
FE_SECURITY	5		FE_NO_FILENAME	16

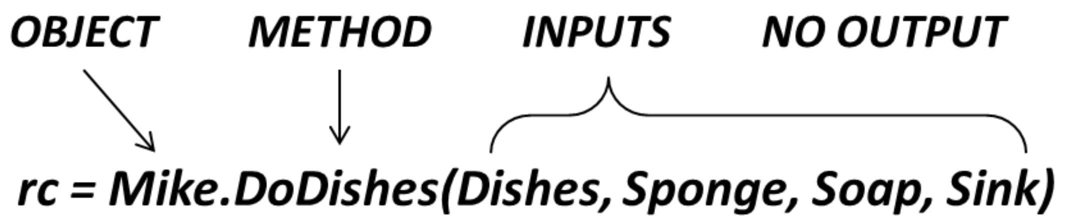
Methods that Produce Output

How this all works is best explained by a more concrete example. Think of an object as a person, a person who will do things that you ask. We will call this person “Mike”. Say we want “Mike” to go to the store and buy an apple. In order for “Mike” to do this, we need to provide him with a car and money. For this Method, “Mike” will produce an output: an apple. The statement would look like this:

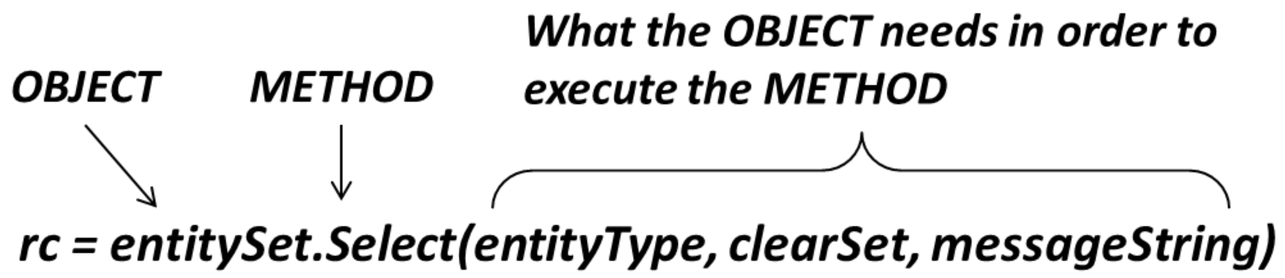


Methods that Produce No Output

Now suppose we want “Mike” to wash the dishes in the kitchen. We need to provide him with the dishes, soap, a sponge, and a sink. After he is done he will produce NO output for us because we haven’t asked him to bring us anything. All we have done is ask him to go off and do something. The statement looks much like the previous one.

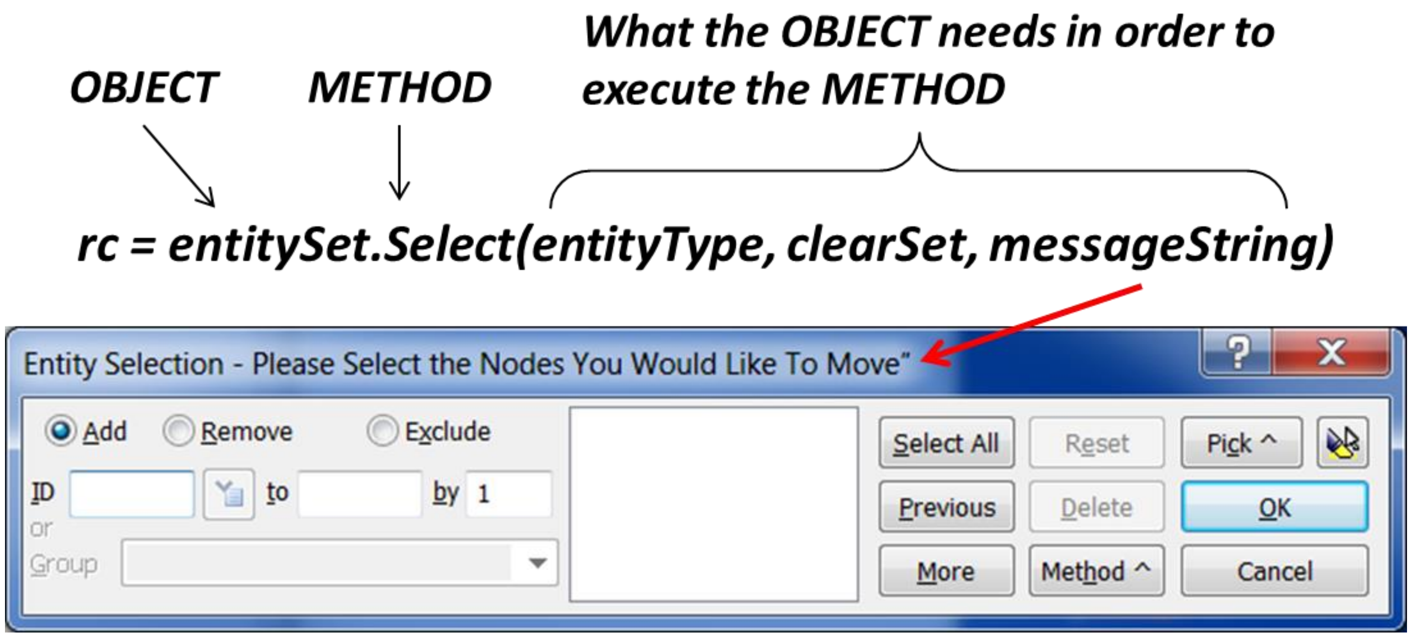


Sometimes we ask objects to organize things. Sometimes we will ask them to create or move things. The only time objects will have output is if we ask them to ***bring*** us something specific. This most likely seems fairly abstract, but once you see how it actually works you will see that it is very intuitive.


What the OBJECT needs in order to execute the METHOD
OBJECT
METHOD
rc = entitySet.Select(entityType, clearSet, messageString)

```
Dim entityType As Long
entityType = 7

Dim clearSet As Boolean
clearSet = True

Dim messageString As String
messageString = "Please Select the Nodes You Would Like To Move”

rc = entitySet.Select(entityType, clearSet, messageString)
```

The syntax of the *entitySet*. Select object Method follows the standard syntax. For this object there is no output, only inputs. This is because we are not “asking” the object for anything concrete (like a value); we are asking the object to place certain entities into a set. The effect is having our desired entities added to the *entitySet*. (Later, we will use an object that will produce an actual output, a required distance.)



```
rc = entitySet.Select(entityType, clearSet, messageString)
```

Entity Selection - Please Select the Nodes You Would Like To Move"

A ***Set Object*** is used to store a set of entities, i.e. a list of nodes. The ***Select Method*** displays the above shown dialog box so the user can select which nodes they are interested in. After the user selects these nodes, they are added to the set called *entitySet*.

In order to do this, the ***Set Object*** needs:

- to know what type of entity to ask for: *entityType*, which has already been set to 7 (corresponding to nodes),
- to know if the object should be cleared of any previous entities: *clearSet*, has already been set to True,
- to know what message to display at the top of the entity selection dialog: *messageString* has already been set to "Please Select the Nodes You Would Like To Move"

8.8 OBJECT PROPERTY SYNTAX

```
Sub Main

Dim App As Object
Set App = GetObject(,"femap.model")

Dim entitySet As Object
Set entitySet = App.feSet

Dim entityType As Long
entityType = 7

Dim clearSet As Boolean
clearSet = True

Dim messageString As String
messageString = "Please Select the Nodes You Would Like To Move"

rc = entitySet.Select(entityType, clearSet, messageString)

Dim setID As Long
setID = entitySet.ID

Dim vecMove(3) As Double
vecMove(0) = 10
vecMove(1) = 0
vecMove(2) = 0

Dim vecLength As Double
rc = App.feVectorLength(vecMove, vecLength)

rc = App.feMoveBy(entityType, setID, False, vecLength, vecMove)

End Sub
```

Certain object capabilities require no input and do not provide output in the convectional way. These are called ***Properties***.

Such is the case with the ***entitySet.ID*** statement.

Instead this syntax returns the desired value to the variable on the left hand side of the equal sign. In this case *setID* will take on the ID number of the object ***entitySet***. A single program can have multiple set objects defined, each containing their own data. Each of these sets would have a specific ID to differentiate them.

Note: It's recommended that you never hard code set IDs in your program as it may have unintended consequences. When a ***Set Object*** is created, FEMAP will auto-assign the next available ID automatically. To reference that set later, simply use the ID Property rather than a hard-coded ID.

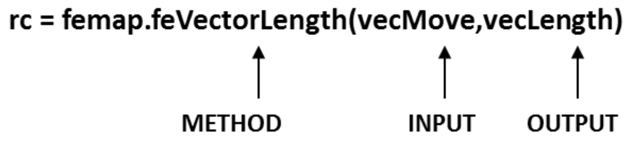
For more information, see FEMAP API Help Section 5.1.1, Common Entity Properties.

```
Sub Main

Dim App As Object
Set App = GetObject(,"femap.model")

Dim entitySet As Object
Set entitySet = App.feSet

Dim entityType As Long
entityType = 7

Dim clearSet As Boolean
clearSet = True

Dim messageString As String
messageString = "Please Select the Nodes You Would Like To Move”

rc = entitySet.Select(entityType, clearSet, messageString)

Dim setID As Long
setID = entitySet.ID

Dim vecMove(3) As Double
vecMove(0) = 10
vecMove(1) = 0
vecMove(2) = 0

Dim vecLength As Double
rc = App.feVectorLength(vecMove, vecLength)

rc = App.feMoveBy(entityType, setID, False, vecLength, vecMove)

End Sub
```

We will now use the ***feVectorLength Method*** of the ***FEMAP Application Object*** to find the magnitude of the nodal move we will be requesting.

First we declare a 3 dimensional array, composed of 8-byte real numbers called “*vecMove*”. This array will represent the vector along which the translation will take place. We then specify each value in the array. This will be the ***feVectorLength Method*** input.

Then we declare a variable called “*vecLength”* as an 8-byte real number (*Double*). No value is assigned because it will be used for the ***feVectorLength Method*** output.



What we are asking of the ***FEMAP Application Object*** is for it to take our vector, called *vecMove*, and tell us how long it is. What the object gives us is a new value for *vecLength*. If the operation is successful, *rc* will be given the value of –1.

```
Sub Main

Dim App As Object
Set App = GetObject(,"femap.model")

Dim entitySet As Object
Set entitySet = App.feSet

Dim entityType As Long
entityType = 7

Dim clearSet As Boolean
clearSet = True

Dim messageString As String
messageString = "Please Select the Nodes You Would Like To Move"

rc = entitySet.Select(entityType, clearSet, messageString)

Dim setID As Long
setID = entitySet.ID

Dim vecMove(3) As Double
vecMove(0) = 10
vecMove(1) = 0
vecMove(2) = 0

Dim vecLength As Double
rc = App.feVectorLength(vecMove, vecLength)

rc = App.feMoveBy(entityType, setID, False, vecLength, vecMove)

End Sub
```

And last, but certainly not least, we will request that the ***FEMAP Application Object*** moves our nodes.

This Method, called ***feMoveBy,*** has the following inputs:

- what type of entity it is moving,
- what set contains the ID's of the entities to move,
- whether or not this is a radial translation,
- the length of the translation,
- a vector specifying the direction of the translation.

8.9 Conclusion

What is most interesting about the script we just explored is that it only does one thing: it moves the nodes. Everything else found in script exists only to provide the last command with the information it needs to make the move. This is fairly common. Often much of the API script is devoted to retrieving things from the database, interpreting them, changing them, and then finally inserting them back in.

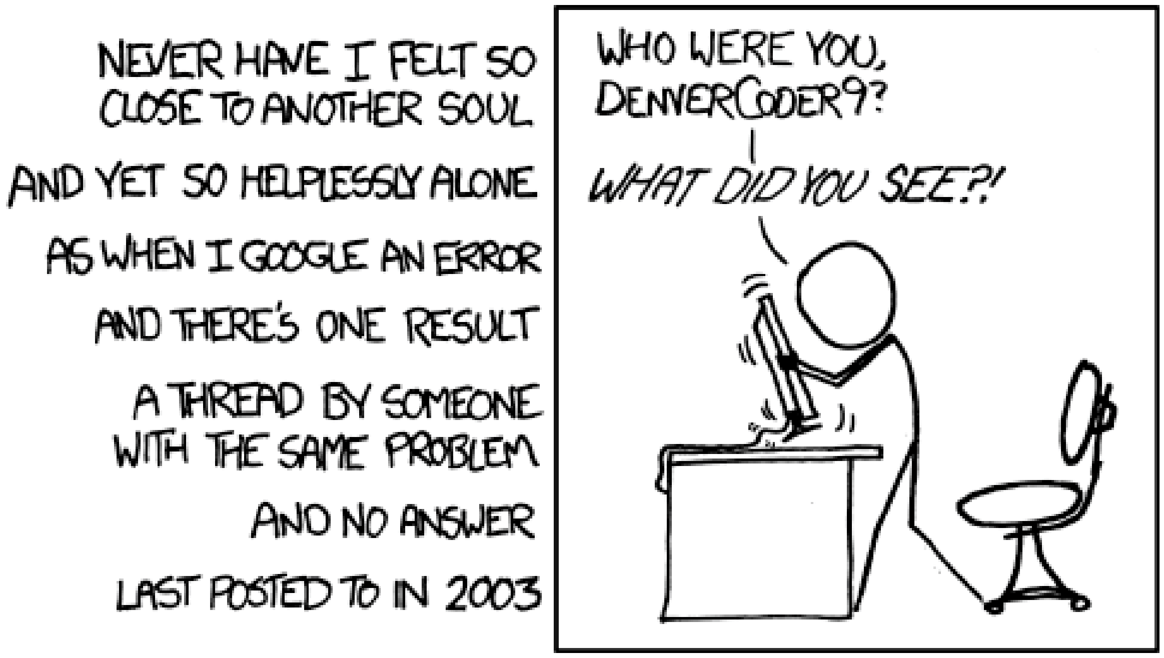
In our case, we retrieved the node numbers of that were to be moved, organized them into a set, and then requested that the ***FEMAP Application Object*** move them.

The previous example is a simple one that uses very little logic. There are no *for* or *while* loops and no *if then* statements, but all of the standard logic statements are available and are used frequently. Anyone with basic programming skills should be able to utilize them as they would in any other language.

You should now understand the basics needed to read and understand basic API's. The only way to become a pro at writing them is to sit down and do it. In no time you will find that the structure and capabilities are extremely powerful. You will also find that you will never again need to scratch your head and say, "I wish the FEMAP programmers would have included this feature."

9. REMEMBER, YOU'RE NOT ALONE!

Reach out to us at www.AppliedCAx.com for APIs, Technical Seminars and more information on FEMAP and NX Nastran. For more information on API programming with FEMAP, check out our online course.


NEVER HAVE I FELT SO CLOSE TO ANOTHER SOUL
AND YET SO HELPLESSLY ALONE
AS WHEN I GOOGLE AN ERROR
AND THERE'S ONE RESULT
A THREAD BY SOMEONE WITH THE SAME PROBLEM
AND NO ANSWER
LAST POSTED TO IN 2003
WHO WERE YOU, DENVERCODER9?
WHAT DID YOU SEE?!

And thanks https://xkcd.com/ for all the great comics!

AppliedCAx
We do this every day

SIEMENS
Ingenuity for life

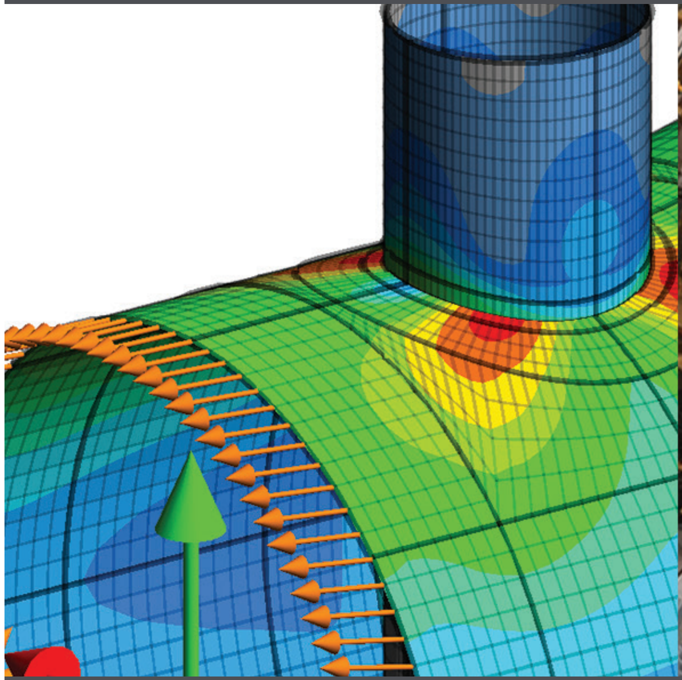
Siemens PLM Software for the USA

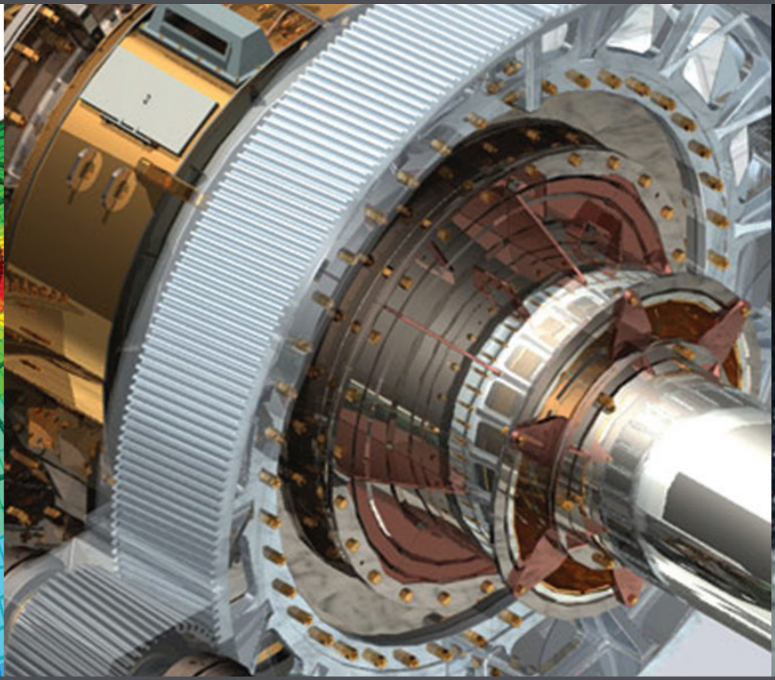
AppliedCAx.com

FEMAP & STAR-CCM+

LICENSES | TRAINING | SUPPORT

CAE Support • CAD Workflows • CAM Posts • PLM Architecture





FEMAP • NX CAD • NX CAM • Solid Edge • Simcenter 3D • STAR-CCM+ • Teamcenter