SIEMENS

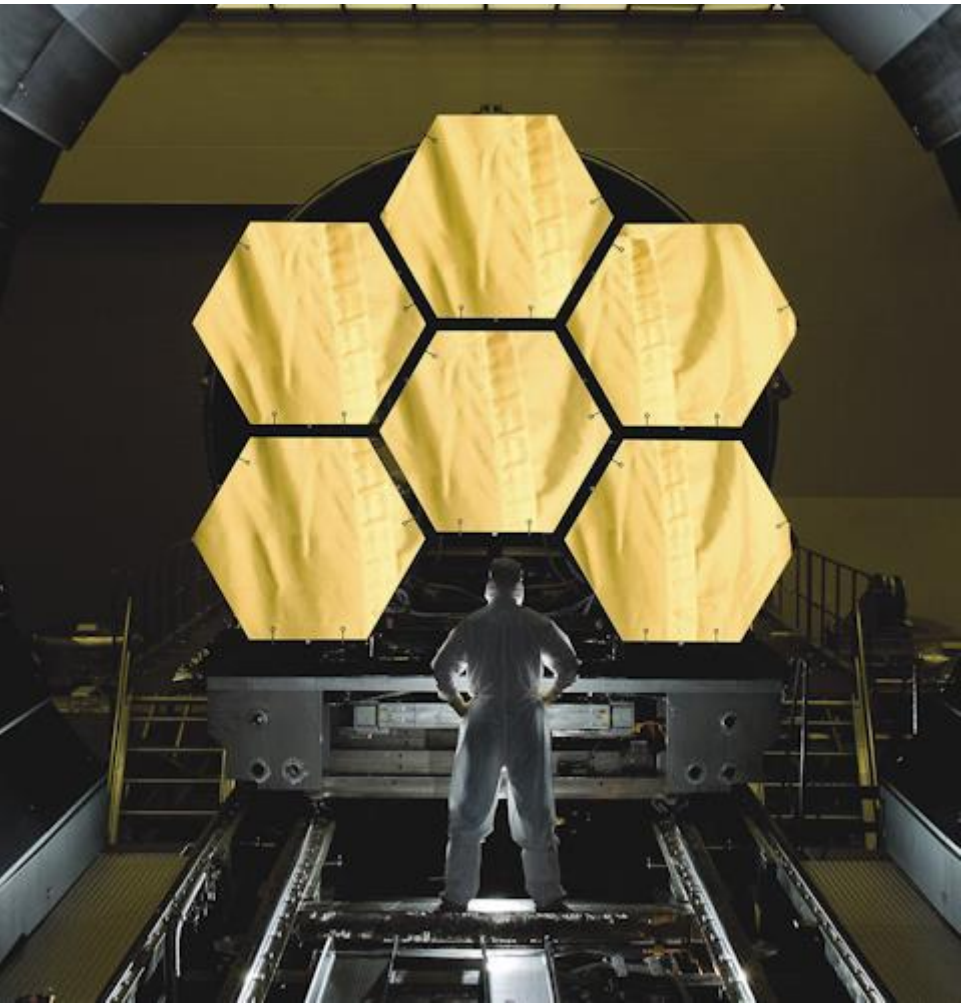
Patrick Kriengsiri

Introduction to the FEMAP API

FEMAP SYMPOSIUM 2014
Discover New Insights

Femap Symposium 2014
May 14-16, Atlanta, GA, USA

Agenda

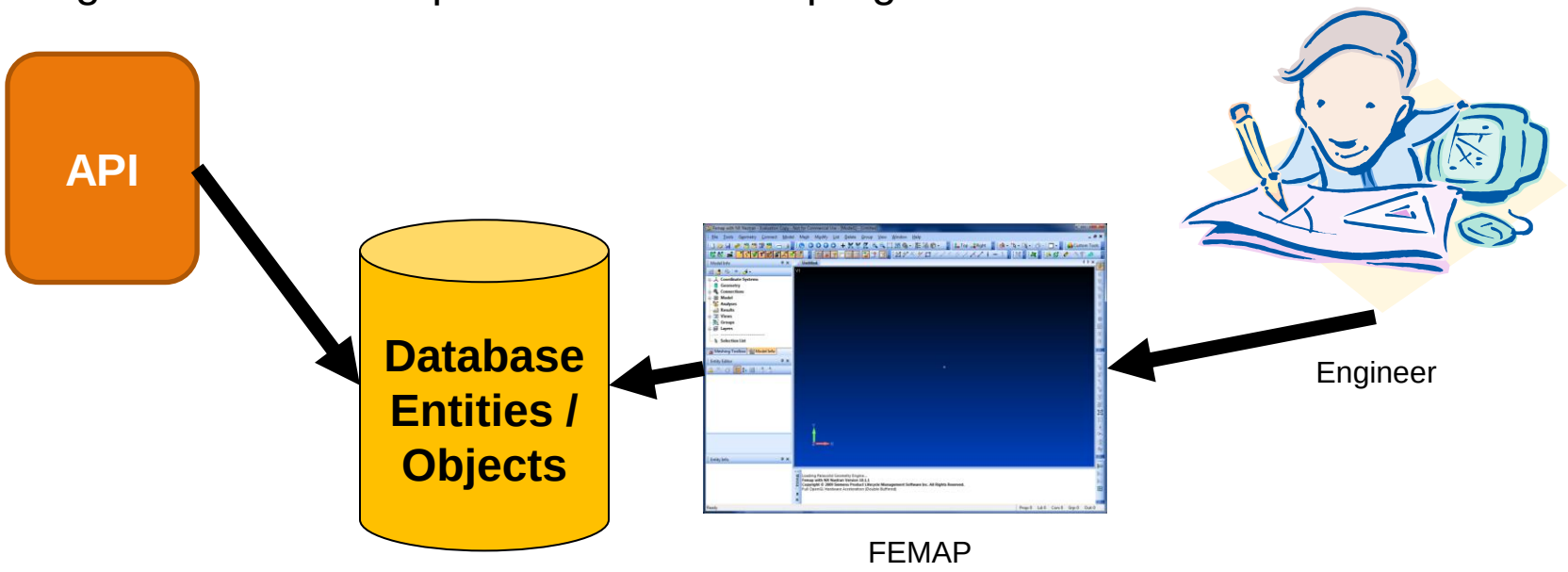


- What is the FEMAP API
- What's Possible with the FEMAP API
- FEMAP API Fundamentals
  - FEMAP API Objects
  - Using the API Programming Window
  - FEMAP Application Object
  - FEMAP Set Objects
  - FEMAP Entity Objects
- FEMAP API Examples
- Q&A

What is the FEMAP API?

What is the FEMAP API?

- The FEMAP *A*pplication *P*rogramming *I*nterface allows programmatic access to the FEMAP database as well as to FEMAP application functionality
- FEMAP includes development environment that allows for the creation of custom programs to extend and enhance FEMAP functionality
- Programs can be scripts or stand-alone programs



SIEMENS

What's Possible with the FEMAP API?

FEMAP can be viewed as a development platform

- Out-of-the-box functionality may fit the FEA needs of 95% of the users 95% of the way
- It's not possible to be a perfect Pre-Post for everyone
  - Different companies / people have different FEA needs
  - Each company has their own methods of doing analysis
  - Each individual within a company may have their own style

Programs can be created to do simple things such as automate routine tasks or more complicated things, such as automating specific analysis tasks

SIEMENS

What's Possible with the FEMAP API?

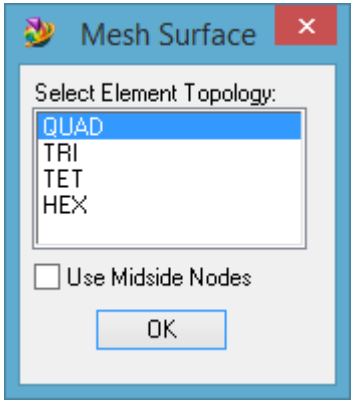
FEMAP development is limited by the mind and needs of the developer/user. Within the context of the application framework,

- The majority of FEMAP functions are accessible programmatically
- FEMAP selection system is available
- Custom text and graphics are available
- Programs can range in complexity from basic scripts to add-ins / standalone programs

What's Possible with the FEMAP API?

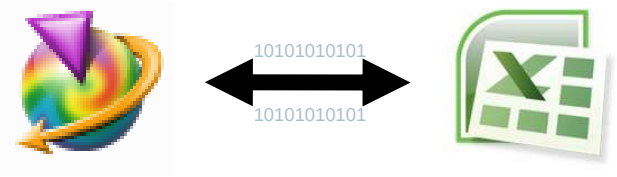
Custom user interfaces can be created and used with FEMAP programs

- Simple custom dialogs an be created in the API Programming Window
- More complicated dialogs or programs can be created as separate programs and used as FEMAP add-ins

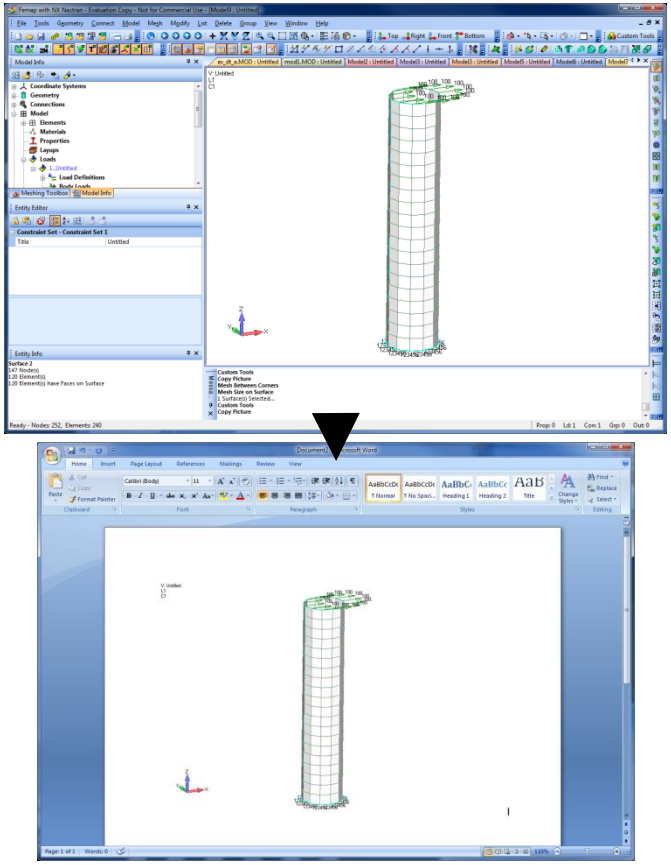


What's Possible with the FEMAP API?

Using the Windows COM interface, FEMAP can communicate with other programs



- Communication occurs on a two-way street
  - FEMAP can be used to drive other programs
  - Other programs can drive FEMAP

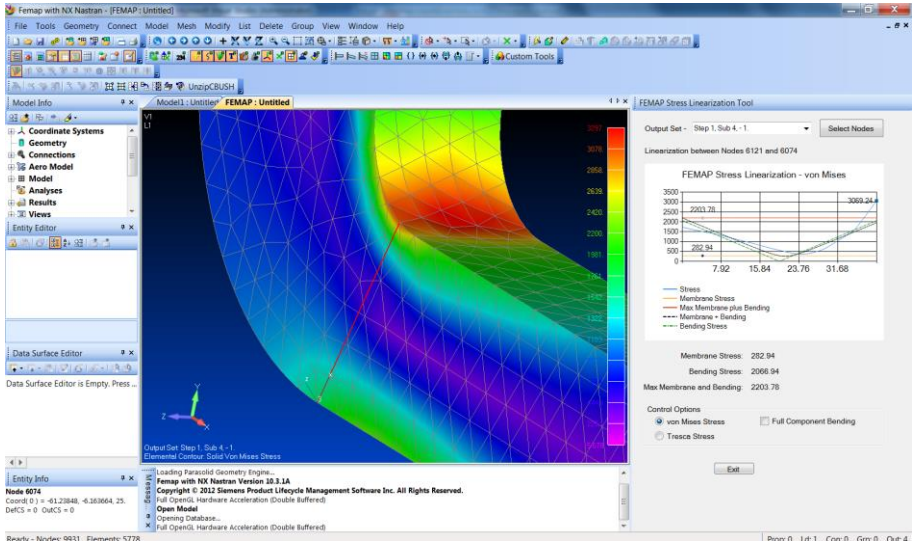


- Example: Use FEMAP to write a custom MS Word analysis report

What's Possible with the FEMAP API?

FEMAP Add-ins can also be created that behave like standard dockable panes

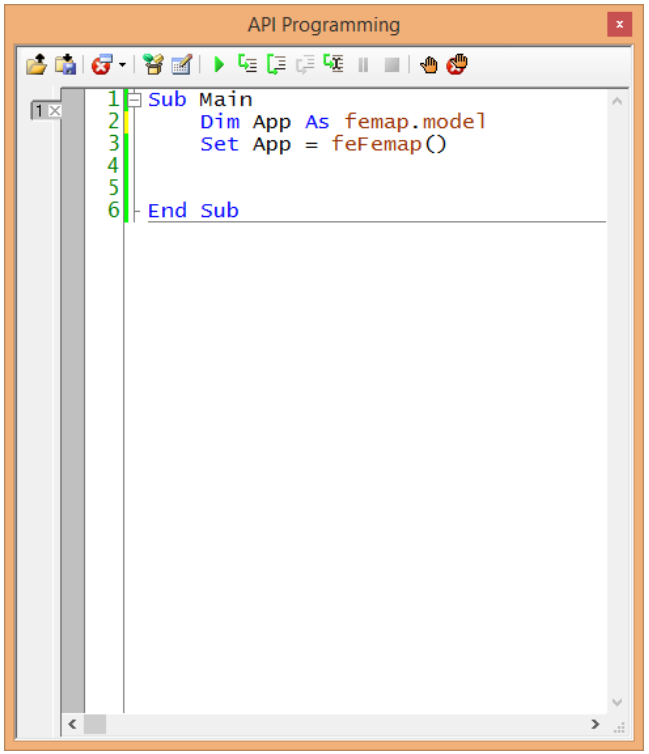
- Similar to TMG or SA Toolkit
- Standard dockable pane behaviors are available (docking, tabbed panes, pinned, etc)
- If using Visual Studio, full compliment of Windows Forms are available
- Programs can be headless – the UI is not required



FEMAP API Fundamentals

Programming for FEMAP is done primarily through the API Programming Window

- API Programs are written in Visual Basic for Applications, not a proprietary command language
- The FEMAP API is *object oriented*, meaning interactions with the API are performed by querying and manipulating a series of objects
  - Objects have properties and methods
    - Properties describe things about the object,
      - Car.color = red
      - Element.topology = CQUAD4
    - Methods provide a way of interacting with an object
      - Car.drive(50, “grocery store”)
      - Surface.mesh()

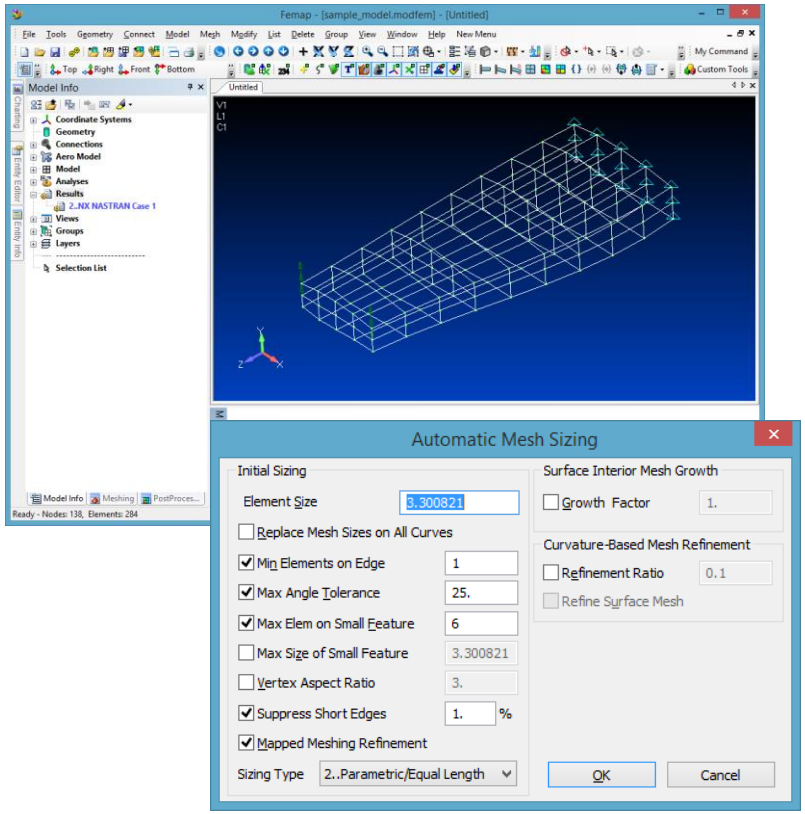


Types of FEMAP Objects

There are three main types of FEMAP objects – the FEMAP Application Object, FEMAP Entity Objects and FEMAP Tool Objects

Application Object

- This is the foundation for all FEMAP API programs
- Conceptually, the application object is the FEMAP program itself
- Properties include program settings and global variables (such as those found in the *Parameters* command)
- Methods include those functions available via menus, info tree, context menus, etc.



FEMAP Application Object
Creating the Application Object

SIEMENS

Access the FEMAP “application” is the first step in creating an API Program. The application object provides programmatic access to functions provided in the FEMAP GUI as well as direct access to all of the entity objects. Without an application object, no FEMAP API functionality is available.

Example:

```
Sub Main

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

End Sub
```

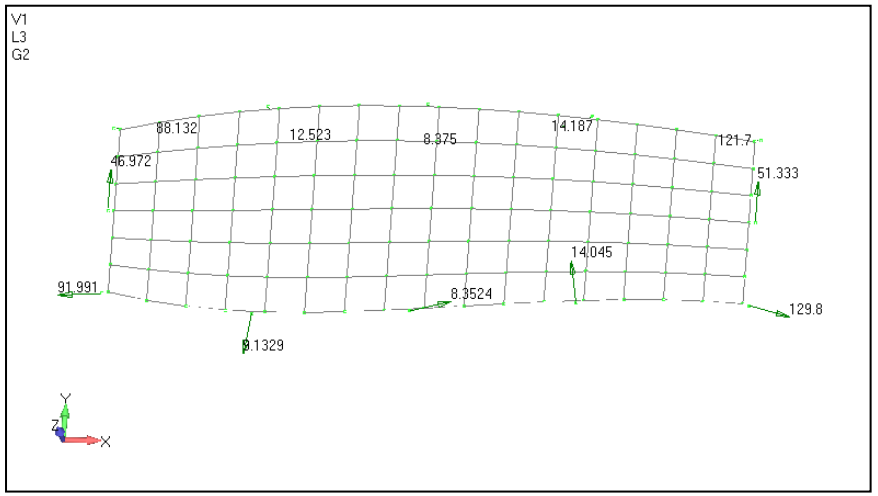
Info *	This is the default code stub in a new program created in the FEMAP API Programming Window.

Info *	The feFemap method that’s used to initialize the application object is only available within FEMAP and only in versions 10.0+

Types of FEMAP Objects

Entity Objects

- Entity objects are anything at are created and stored in the database – geometry, nodes, elements, analysis sets, etc.
- Entity objects can be created on their own or, often times, via application object methods



Tool Objects

- Tool objects are a special class of object that provide additional functionality and are generally not stored in the database
- Data Table, File Reader, Sets, Beam Calculator, Data Maps, etc.

Anatomy of a Program

When the FEMAP API Programming window is first opened, a program stub is automatically created that provides the foundation of an API program

```
Sub Main

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

End Sub
```

Beginning of program

Create a FEMAP variable

Beginning of program

End of program

This code stub is the basis of the majority of FEMAP programs. The API programming window can be used as a generic VBA interpreter without creating the FEMAP object

Creating and Initializing FEMAP Objects

The majority of FEMAP objects need to be declared as well as initialized before they can be used.

- The `Dim` keyword declares the variable of a certain data type
- In VBA, the `Set` instruction will initialize the variable
- This is similar in concept to a pointer in other languages, however there is no memory management required of the user, except in unmanaged code

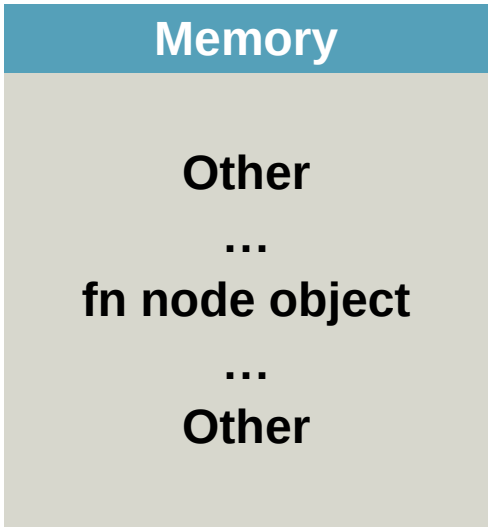
Example:

```
' FN is declared as a FEMAP node and memory is reserved
Dim fn As femap.Node
' FN is initialized as a node object from the active FEMAP session
Set fn = App.feNode
```

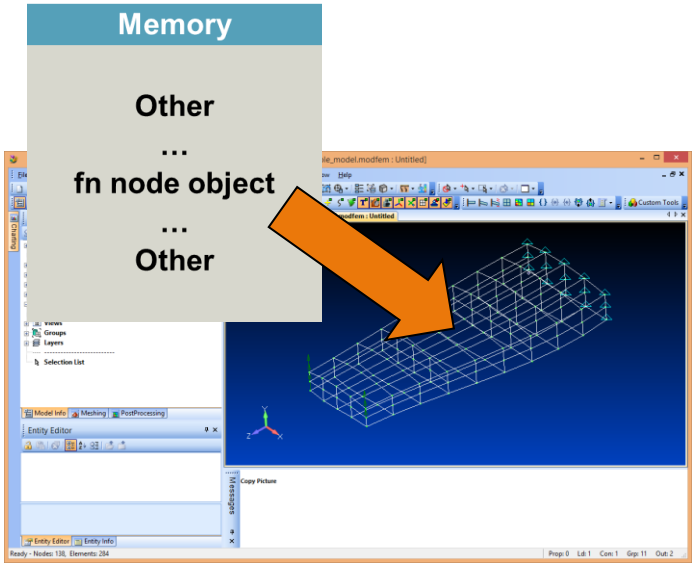
Info *	All objects, with the exception of the application object, are initialized using an application object method

Creating and Initializing FEMAP Objects

```
Dim fn As femap.Node
```




```
Set fn = App.feNode
```

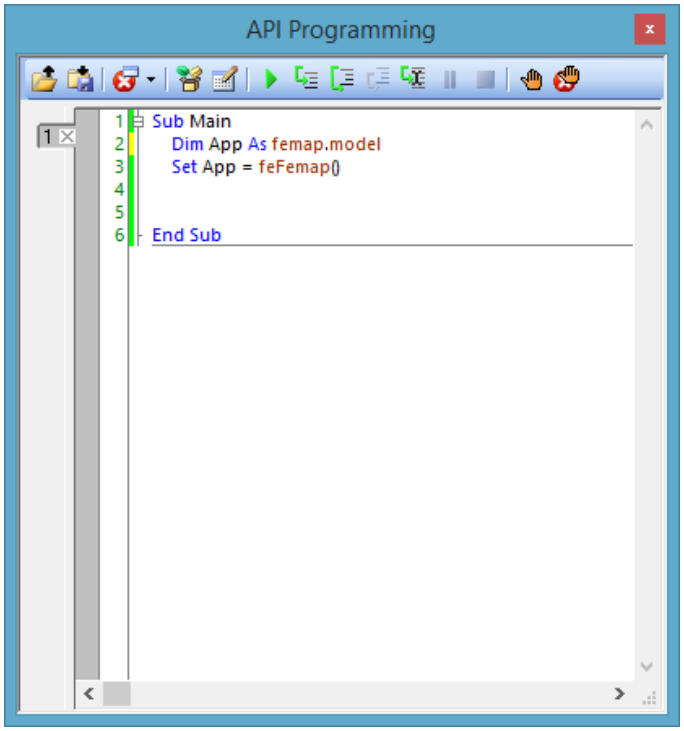


FEMAP API Programming Window

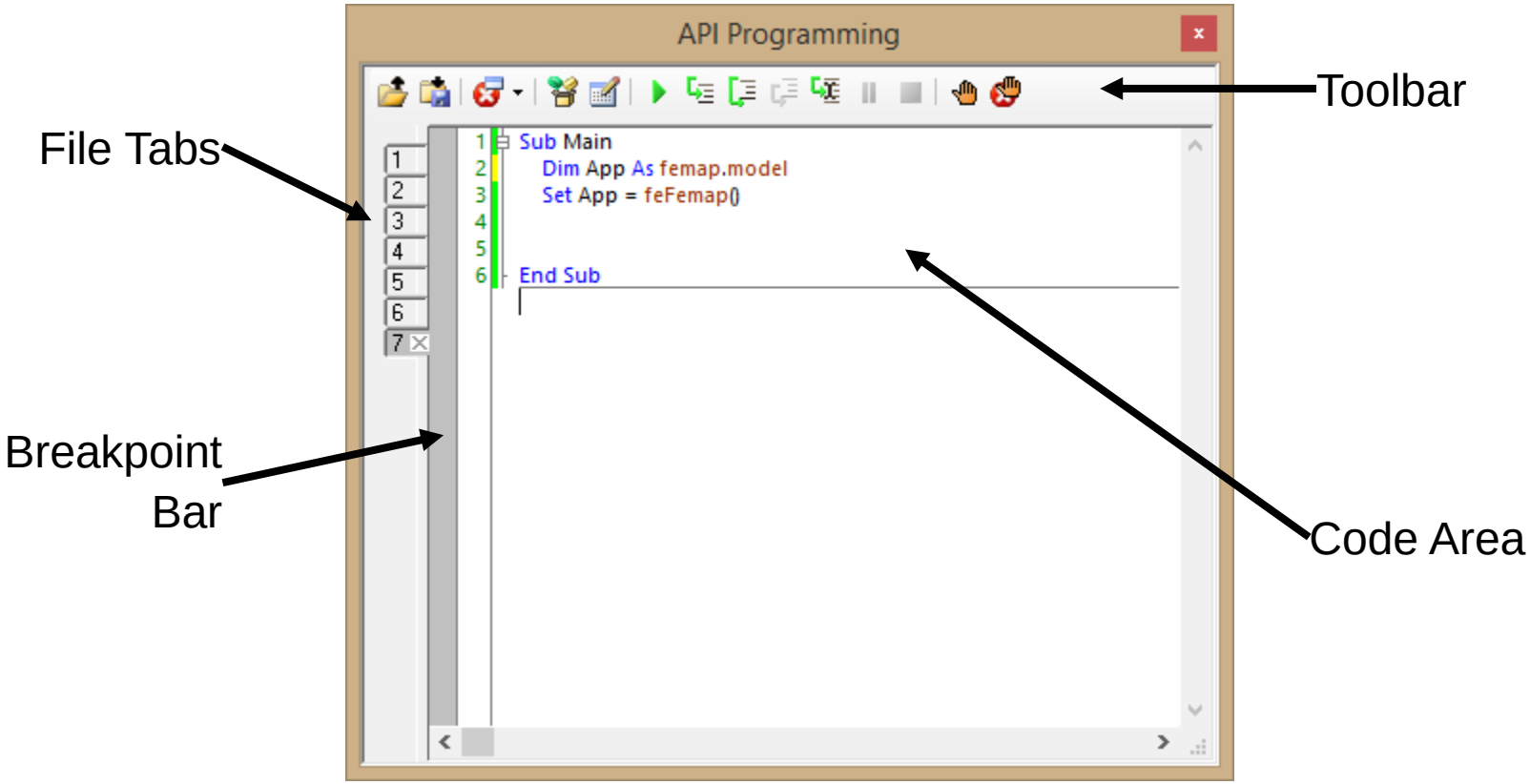
The FEMAP API Programming Window is an Integrated Development Environment available within the FEMAP UI.

- Access via Tools -> Programming -> API Programming or with the API Programming Button on the Panes Toolbar

- Features
  - Code completion
  - Syntax highlighting
  - GUI creation
  - Object browser
  - Debugging



API Programming Window Areas



FEMAP Set Objects

Sets are a critical concept within the FEMAP API

- Fundamentally, Sets are a list of entity IDs
  - Sets can only store IDs in the range of valid entity IDs (1 to 99,999,999)
  - IDs are stored within sets from lowest to highest, regardless of the order in which they were added
  - IDs are either in or out; they can not be stored multiple times
  - Sets only know what IDs are in contained within; they have no knowledge of what entity type it is related to
- Sets are often used to pass one ore more entity IDs within different API calls

Set ID = 1
1
3
5
9
14
17
22
…
10000

Creating FEMAP Sets

FEMAP Sets are created using the femap.Set object and are referenced using the feSet application object method.

Example:

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

    Dim fs As femap.Set
    Set fs = App.feSet

    fs.AddAll( FT_NODE )
    App.feViewShow( FT_NODE, fs.ID )
End Sub
```

Info *	Generally, when a method calls for a Set of IDs, it's looking for the ID property of the Set. Alternatively a negative ID may often be provided as an argument to specify a single ID without having to create a Set. See the API Reference Manual for specifics.

Adding IDs to Sets

Entity IDs can be added to sets using a variety of methods

Method	Description
`Add`	Adds a single ID to the Set
`AddRange`	Adds a range of IDs to the Set
`AddAll`	Adds all IDs of a given entity to the Set
`AddSet`	Adds all IDs from an existing Set to the current Set
`AddRule`	Adds IDs to a Set based on a grouping rule using a single ID
`AddSetRule`	Adds IDs to a Set based on a grouping rule using IDs from an existing Set
`AddArray`	Adds an array of Values
`AddGroup`	Adds IDs of an entity type from a Group to the Set

Info

*

See section 4.6.2.7 of the API Reference Manual for group selection rule enumerations.

Adding IDs to Sets

Example:

```
fs.AddAll( FT_NODE )                ' Adds all nodes
fs.Add( 1 )                         ' Adds ID 1
fs.AddRange( 1, 10, 2 )             ' Adds 1, 3, 5, 7, 9
fs.AddRule( 1, FGD_ELEM_BYMATL )    ' Adds all elements using property 1
fs.AddArray( 4, Array( 10, 11, 12, 13) )    ' Adds 10, 11, 12, 13
```

Caution !	Remember that Sets are simply a list of IDs – they are not aware of any entity types. In the example above, the Set will contain all IDs based on the methods called.

SIEMENS

Set Object IDs

In the majority of cases, Set objects containing lists of IDs are passed to FEMAP functions via the .ID property

- The Set ID is automatically assigned by FEMAP in the constructor
  - Example: `App.feViewShow( FT_NODE, fs.ID )`
- When the Set object loses scope, it's automatically cleaned up by FEMAP
  - Like other objects, they do no persist beyond variable scope
  - Set objects must be recreated and re-populated every time a program is run

Caution !	**It is not recommended to ever hard code a set ID – this may potentially cause problems with FEMAP as well as other APIs**

Set Object IDs

Example:

AVOID!!!

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

    Dim fs As femap.Set
    Set fs = App.feSet

    fs.ID = 1
    fs.AddAll( FT_NODE )
    App.feViewShow( FT_NODE, 1 )

End Sub
```

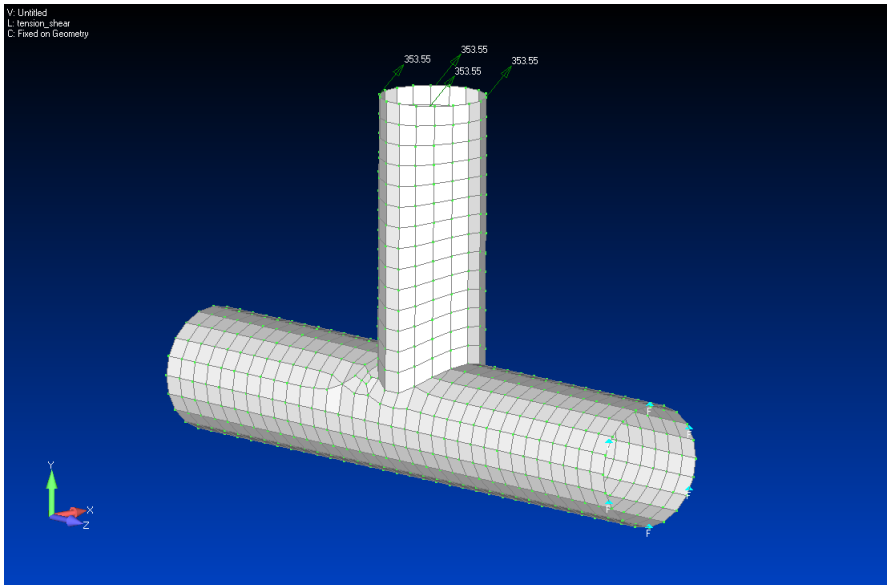
OK!

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

    Dim fs As femap.Set
    Set fs = App.feSet

    fs.AddAll( FT_NODE )
    App.feViewShow( FT_NODE, fs.ID )

End Sub
```

FEMAP Entity Objects

FEMAP Entity objects are anything that can be created and stored in a model

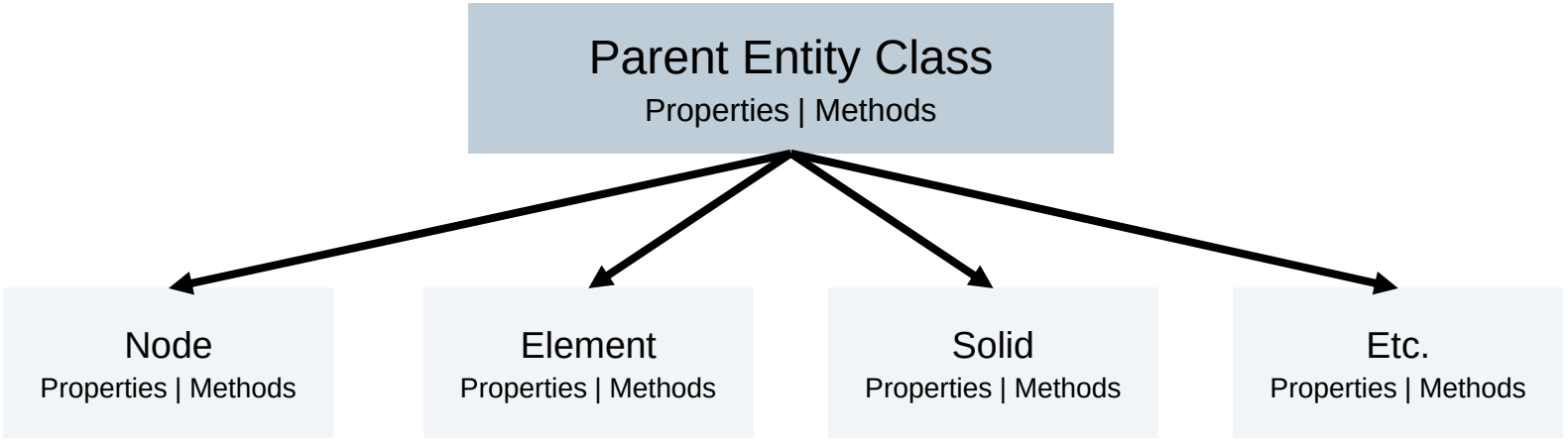
- Entity objects do not cover the tools and methods that may be used to create these entities
- Working with entity objects essentially provides direct access to the FEMAP database
  - All entities can be created/modified programmatically



Parent Entity Object

All Entity objects (nodes, elements, groups, layers, etc) are derived from a single parent class

- They share a common set of properties
- They share a common set of methods
- Individual entity objects have their own set of properties and methods that are specifically related to those object types



SIEMENS

Working with Entities

Like other FEMAP objects, entity objects need to be declared as well as initialized (using the Set instruction in VBA)

Example:

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

    ' Declare
    Dim fn As femap.Node
    ' Declare
    Set fn = App.feNode

End Sub
```

SIEMENS

Retrieving Entities from the Model

After an object is initialized, it can assume any of the properties of or use any of the methods associated with that entity type. At this point

- It's just an object that exists in memory
- There is no direct relationship to any entity that exists in the database
- The properties values can be unique to that entity or can assume those of any existing ones in the database

To load the properties from an existing entity object to the instance of an entity, use the Get method to retrieve an existing ID.

`Get()` – Loads an existing entity's properties into the current entity	
`INT4 id`	Existing entity ID

Retrieving Entities to the Model

To commit an object to the database, use the Put method

- If the specified ID doesn't currently exist in the model, a new entity will be created in the database with all the same properties as the "Put" entity
- If the specified ID does exist, the entity in the database is updated to reflect all the properties of the "Put" entity

`Put()` – Puts the existing entity's properties into the database	
`INT4 id`	Entity ID; new or existing

Info

*

Property validation is normally only performed on the Put action. Generally, properties are only validated on the Put action

Tip

When calling Put, it's rarely recommended to hard-code the ID. It's much safer to reference the entity's ID property when modifying an existing entity, or to use the NextEmptyID method to automatically determine a new ID when creating

FEMAP API Examples
Show All Nodes in a Model

SIEMENS

Example

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

    Dim fs As femap.Set
    Set fs = App.feSet
    fs.AddAll( FT_NODE )

    App.feViewShow( FT_NODE, fs.ID )

End Sub
```

FEMAP API Examples
Randomizing Property Colors

SIEMENS

Example:

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

    Dim fp As femap.Prop
    Dim randomColor As Integer
    Dim thisView As Long

    Set fp = App.feProp fp.Reset()

    While fp.Next()
        randomColor = Round(Rnd()*149, 0)
        fp.color = randomColor
        fp.Put ( fp.ID )
    Wend

    App.feAppGetActiveView(thisView)
    App.feViewRegenerate(thisView)
    MsgBox("Random colors assigned to properties")

End Sub
```

FEMAP API Examples
Set Group Automatic Add

SIEMENS

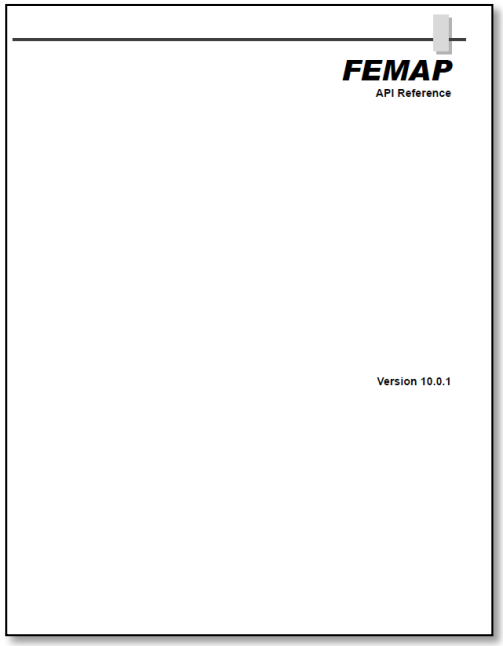
Example

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

    App.Info_GroupAutomaticAdd = -1

End Sub
```

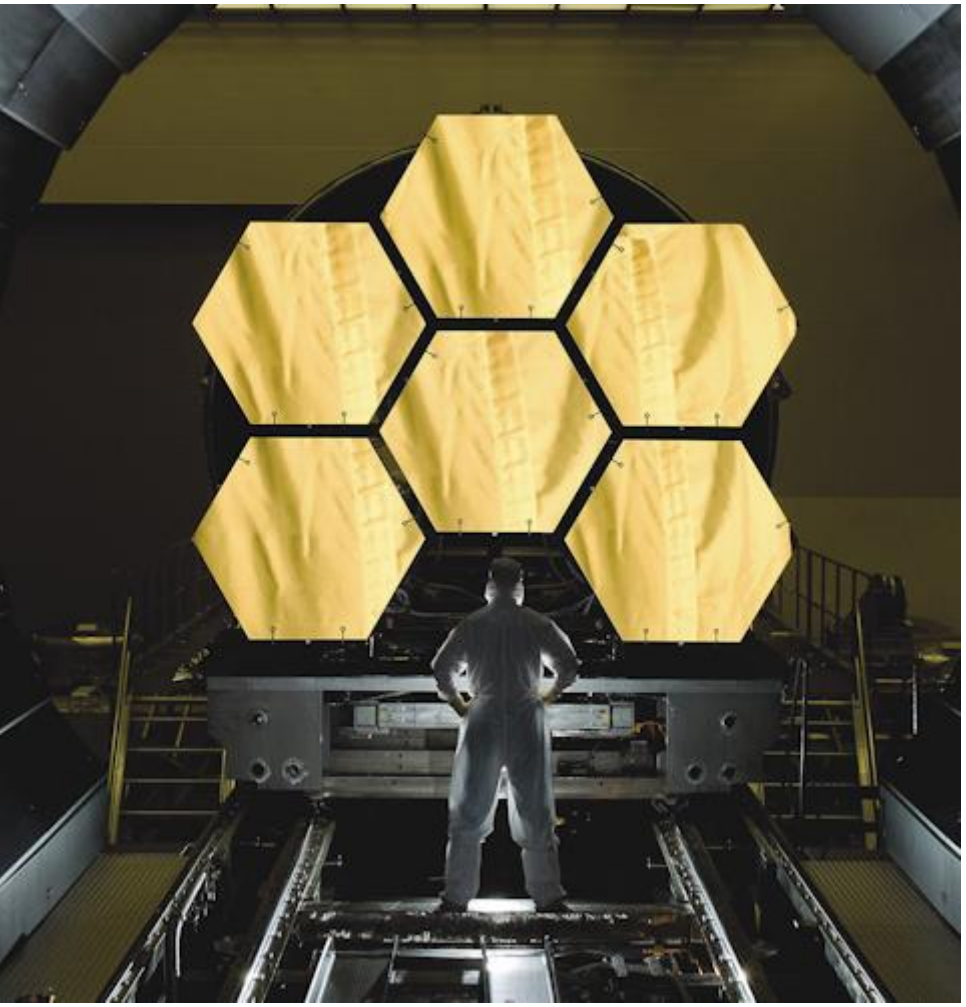
More Information / Resources

Beyond this class, additional examples may be found

- The FEMAP API Reference Manual
  - Found in <FEMAP ROOT>/pdf/api.pdf or Help->Programming
  - 2000 page reference manual for all FEMAP API functionality
  - Include basic examples of API programs
- In the Custom Tools menu
  - All utilities in the Custom Tools Menu are shipped as API programs
  - The source code is viewable in the API programming window or via a standard text editor
- On the FEMAP Online Community
  - http://community.plm.automation.siemens.com/



Introduction to the FEMAP API



Patrick Kriengsiri
FEMAP Development

411 Eagleview Blvd
Exton, PA, 19341

Phone: 404-353-6596

E-mail:
patrick.kriengsiri@siemens.com