NREL
NATIONAL RENEWABLE ENERGY LABORATORY

Spatial Meshes

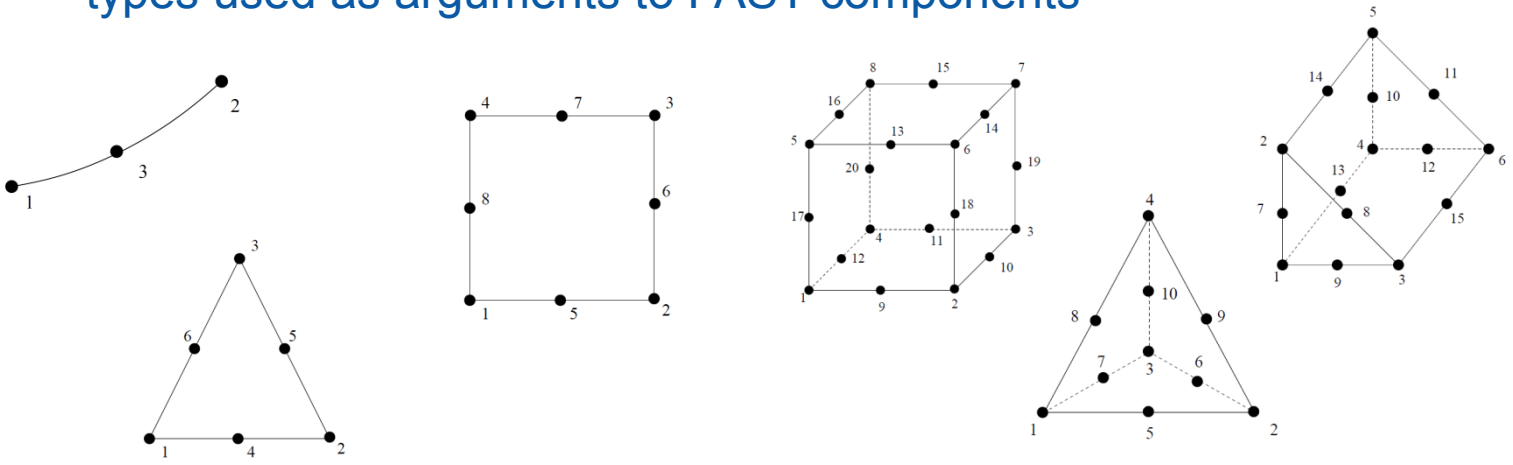
NREL/DOE Workshop on the New Modularization Framework for the FAST Wind Turbine CAE Tool

John Michalakes and Michael Sprague

October 8, 2012

Spatial Meshes

- FAST Mesh data type
  - Defines points, lines, surfaces, and volumes and operations thereon
    - Standard isoparametric mapping closely related to FEA
    - Elements and numbers define unique interpolation
    - Optional midside nodes allow curved line/surface with quadratic polynomials
  - Arrays associated with each mesh node
    - Position in coordinate system (x,y,z)
    - Fields: displacement, force, rotational velocity, translational velocity, moment, … , scalars
  - Required for representing meshed data within Input and Output data types used as arguments to FAST components



Independent Spatial Discretizations

- MeshType data type:
  - NNodes giving the finite number of logical nodes (points in space)
  - RemapFlag (discussed later)
  - Fields: vectors of length NNodes that store values associated at each node
    - Always defined: Position (NNodes 3-tuples: x,y,z coordinates of each node)
    - May also be defined:
      - Displacement (NNodes 3-tuples: x,y,z displacements at each node)
      - Orientation (NNodes 9-tuples: Direction Cosine Matrix)
      - Rotational velocity (NNodes 3-tuples)
      - Translational velocity (NNodes 3-tuples)
      - Arbitrary number of scalars (NScalars by NNodes array)
  - Connectivity information (stored as integer vectors) that organizes the nodes into elements

Lines	Tetrahedra
Triangles	Hexahedra
Quadrilaterals	Etc.

Operations on meshes

- Definition
  - Declaration of meshes
  - Creation of mesh instances and allocation of fields
  - Spatio-location of mesh nodes
  - Construction of nodes into elements
  - Committing the mesh
- Use
  - Setting and accessing fields
  - Copying meshes
  - Packing mesh data
  - Destroying meshes

Operations on meshes: Definition

- Declaration
  - Meshes are defined in Registry to generate declarations of component-contributed MODULE

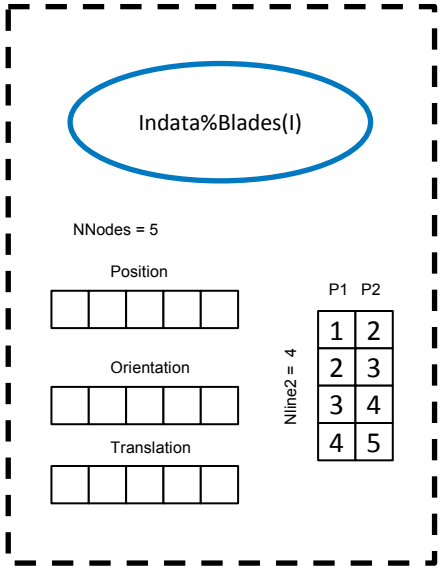
```
# Part of Registry file: Registry-ModuleNameX.txt
  ...
# ..... Input argument type ..........................................................
# Define inputs that are contained on the mesh here:
typedef  ModuleNameX/ModNmX InputType   MeshType Blades   {:}   - - "Allocatable array of blade meshes"
typedef   ^                  ^          ReKi     aScalar   -    - - "Scalar variable"   "units"
typedef   ^                  ^           ^       anArray  {:}   - - "Allocatable array" "units"

# ..... Output argument type .........................................................
typedef   ^                OutputType  MeshType Blades   {:}   - - "Allocatable array of blade meshes"
typedef   ^                  ^          ReKi     bScalar   -    - - "Scalar variable"   "units"
typedef   ^                  ^           ^       bArray  {2}{3} - - "2 by 3 2-D array"  "units"
  ...
```

Click here to see generated code

Operations on meshes: Definition

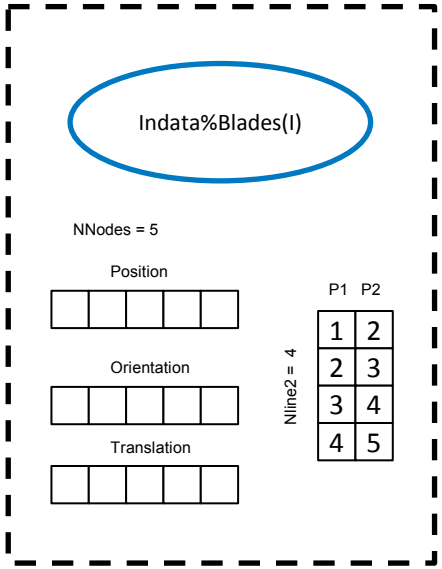
- Declaration
  - Meshes are defined in Registry to generate declarations of component-contributed MODULE
- Allocation
  - An instance of a mesh is created with a call to MeshCreate or MeshCopy in component-supplied Init routine



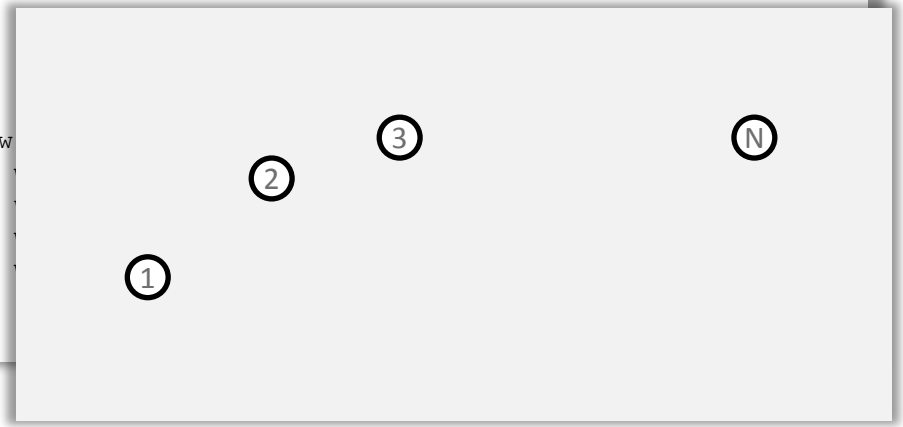

```
TYPE(ModName_InputType)  :: InData
TYPE(ModName_OutputType) :: OutData
    . . .
DO I = 1, SIZE( InData%Blades )
  CALL MeshCreate ( InData%Blades(I)          ! New mesh to be created                        &
                   ,IOS=COMPONENT_INPUT       ! It will be used as output from my component   &
                   ,NNodes=5                  ! It will contain five nodes, total             &
                   ,Orientation=.TRUE.        ! It will convey orientation                    &
                   ,Translation=.TRUE.        ! It will convey translation                    &
                  )
```

Operations on meshes: Definition

- Declaration
  - Meshes are defined in Registry to generate declarations of component-contributed MODULE
- Allocation
  - An instance of a mesh is created with a call to MeshCreate or MeshCopy in component-supplied Init routine



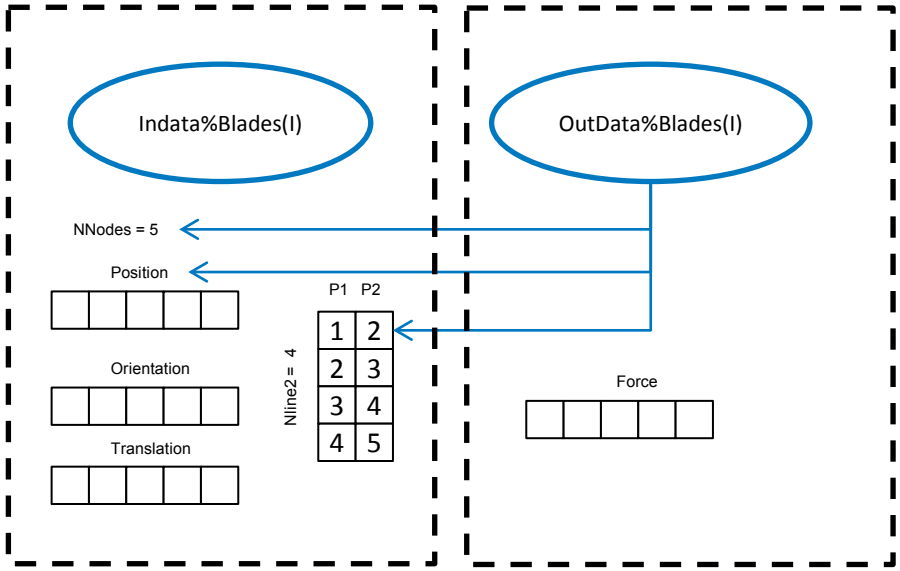

```
TYPE(ModName_InputType)  :: InData
TYPE(ModName_OutputType) :: OutData
    . . .
DO I = 1, SIZE( InData%Blades )
  CALL MeshCreate ( InData%Blades(I)          ! New
                   ,IOS=COMPONENT_INPUT       ! It
                   ,NNodes=5                  ! It
                   ,Orientation=.TRUE.        ! It
                   ,Translation=.TRUE.        ! It
                  )
```



Operations on meshes: Definition

Meshes may be unique or may have “siblings” to avoid duplicating position and connectivity data between input and output to a component.

- Declaration
  - Meshes are defined in Registry to generate declarations of component-contributed MODULE
- Allocation
  - An instance of a mesh is created with a call to MeshCreate or MeshCopy in component-supplied Init routine



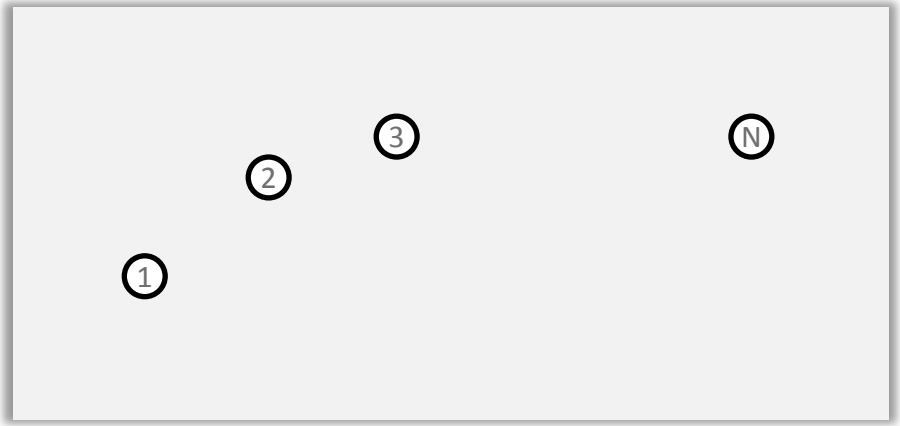

```
TYPE(ModName_InputType)  :: InData
TYPE(ModName_OutputType) :: OutData
    . . .
DO I = 1, SIZE( InData%Blades )
  CALL MeshCreate ( InData%Blades(I)          ! New mesh to be created                        &
                   ,IOS=COMPONENT_INPUT       ! It will be used as output from my component   &
                   ,NNodes=5                  ! It will contain five nodes, total             &
                   ,Orientation=.TRUE.        ! It will convey orientation                    &
                   ,Translation=.TRUE.        ! It will convey translation                    &
                  )
    ...

  CALL MeshCopy   (  InData%Blades(I)         ! Existing mesh                                 &
                   ,OutData%Blades(I)         ! New mesh to be created as a sibling           &
                   ,CtrlCode=MESH_SIBLING     ! It will be used as output from my component   &
                   ,Force=.TRUE.              ! It will convey forces                         &
                  )
```

Operations on meshes: Definition

- Spatio-location of nodes
  - Set the X, Y, Z coordinates of each node in the Mesh

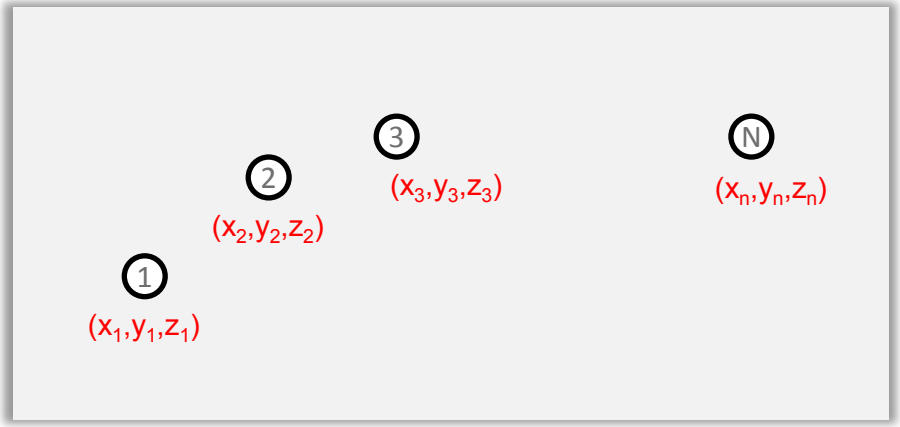
```
DO I = 1, SIZE( InputData%Blades(I) )
      . . .
   ! connect the nodes into 2-node line elements
   DO INode = 1, InputData%Blades(I)%NNodes
     READ(IU,*) Pos(1:3)
     CALL MeshPositionNode(              &
                InputData%Blades(I),     &
                INode=INode,             &
                Pos=Pos(1:3),            &
                ErrStat=ErrStat,         &
                ErrMess=ErrMess)
   END DO
```



Operations on meshes: Definition

- Spatio-location of nodes
  - Set the X, Y, Z coordinates of each node in the Mesh

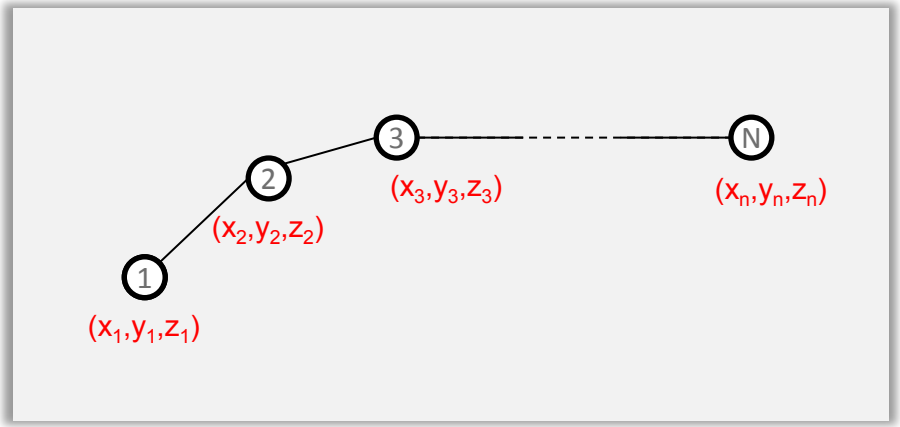
```
DO I = 1, SIZE( InputData%Blades(I) )
      . . .
   ! connect the nodes into 2-node line elements
   DO INode = 1, InputData%Blades(I)%NNodes
     READ(IU,*) Pos(1:3)
     CALL MeshPositionNode(              &
                InputData%Blades(I),     &
                INode=INode,             &
                Pos=Pos(1:3),            &
                ErrStat=ErrStat,         &
                ErrMess=ErrMess)
   END DO
```



Operations on meshes: Definition

- Spatio-location of nodes
  - Set the X, Y, Z coordinates of each node in the Mesh
- Construction
  - Connectivity established using calls to MeshConstructElement from the component-supplied ModName_Init routine
    - Join individual points into an element and elements as neighbors
    - Spaces must agree between elements

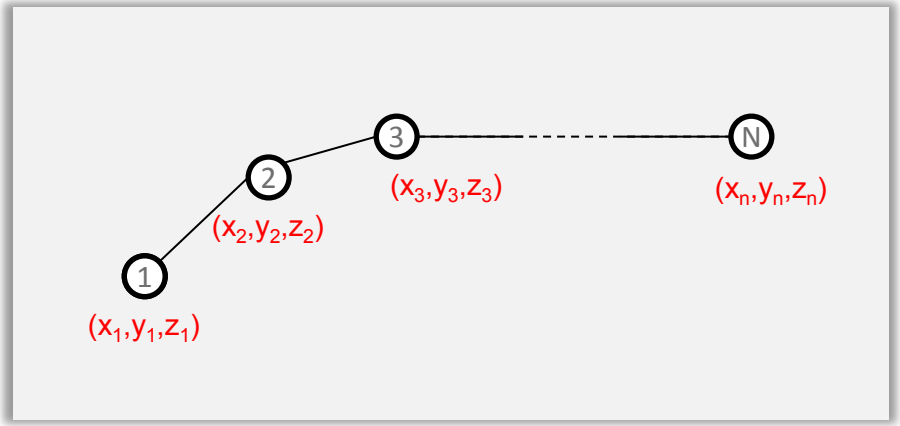
```
DO I = 1, SIZE( InputData%Blades(I) )
      . . .
 ! connect the nodes into 2-node line elements
   DO INode = 1, InputData%Blades(I)%NNodes-1
      CALL MeshConstructElement(             &
                     InputData%Blades(I),    &
                     ELEMENT_LINE2,          &
                     P1=INode, P2=INode+1,   &
                     ErrStat=ErrStat,        &
                     ErrMess=ErrMess)
   END DO
```



Operations on meshes: Definition

- Spatio-location of nodes
  - Set the X, Y, Z coordinates of each node in the Mesh
- Construction
  - Connectivity established using calls to MeshConstructElement from the component-supplied ModName_Init routine
    - Join individual points into an element and elements as neighbors
    - Spaces must agree between elements
- Committing the mesh
  - precompute traversal information, neighbor lists and other information

```
CALL CommitMesh(InputData%Blades(I),ErrStat,ErrMess)
```



Link to simple example

Operations on meshes: Usage

- Setting and accessing fields in a mesh
  - Fields are accessed directly from the mesh itself

    OutData%Mesh%force(inode) = …
  - Functions for iterating over elements in a mesh
    - MeshNextElement (start a traversal or provide next element)
    - MeshNextElemNeighbor (start a traversal over adjacent elements)
    - MeshElemNumNeighbors (returns number of neighboring elements)
  - These return information that allows code to iterate over nodes in a given element
    - The index and kind of element in the mesh
    - The number of nodes in that element

Operations on meshes: Usage

- Copying meshes
  - Create an all new copy of a mesh
  - Create a sibling of a mesh
  - Update position and fields of a mesh from another mesh
- Packing mesh data
  - Works the same as packing and unpacking of FAST derived data types
- Destroying meshes
  - Deallocate memory

Coupling considerations using meshes

- Static, Sliding, and Deforming Meshes
  - The mapping between module interface meshes is performed at any time step when the RemapFlag variable is set to TRUE in the ModMesh module
    - In cases where the interface meshes do not move relative to each other, the mapping should only be done once at initialization.
    - Either module might request a new mapping in the event of significant mesh distortion or significant relative motion between interface meshes.
    - In cases where an interface mesh in a module will follow an interface mesh of another module, both meshes should be initialized in the undeflected position, and RemapFlag should be set to FALSE after initialization.

Status

- Prototype ModMeshType and ModMesh modules being developed
- Prototype application for meshes and Registry
  - Aerodyn in new Framework
- Programmer's handbook and reference documentation