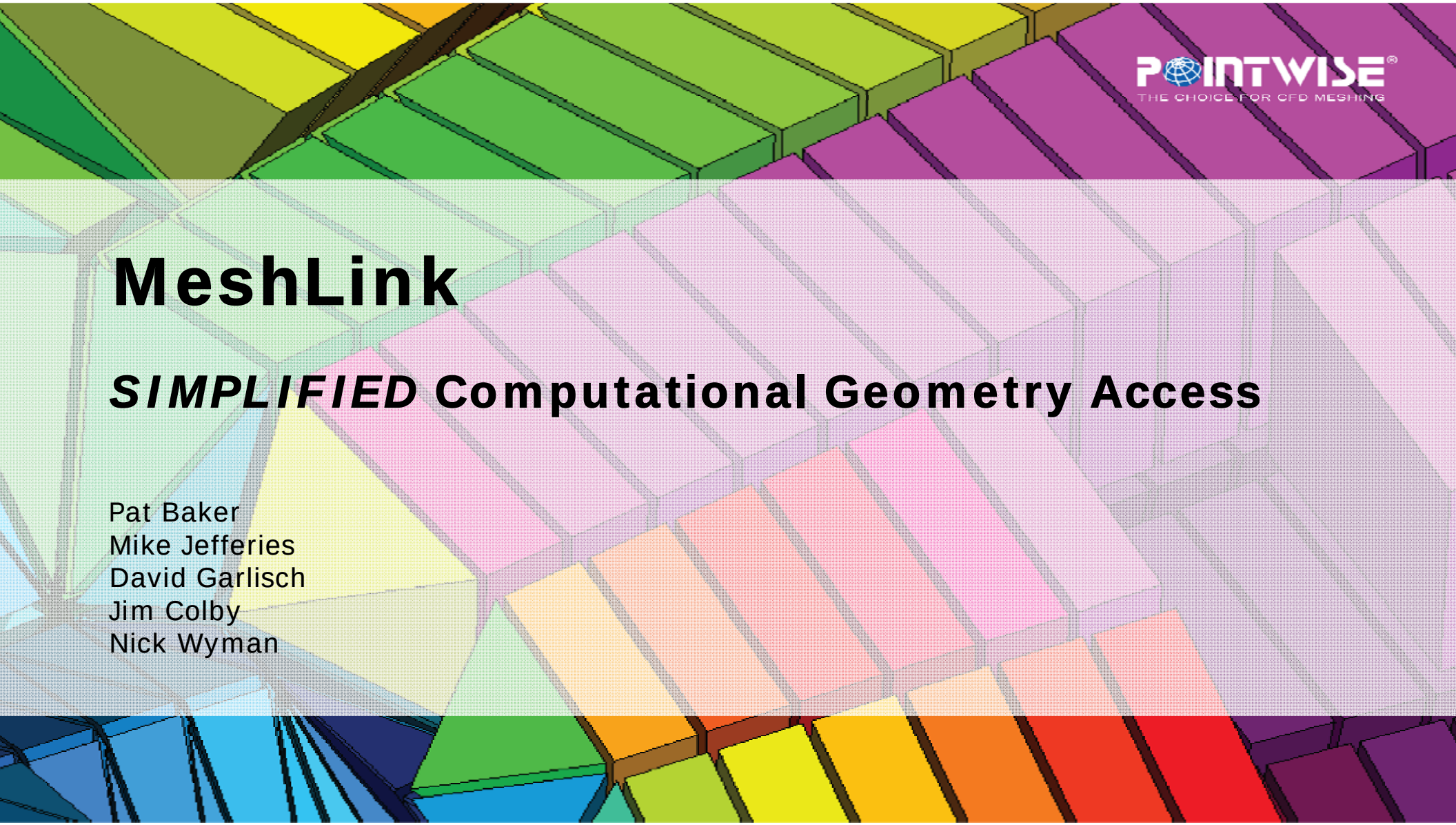


MeshLink

SIMPLIFIED Computational Geometry Access

Pat Baker
Mike Jefferies
David Garlisch
Jim Colby
Nick Wyman

Overview

The MeshLink library contains functions for import of geometry-mesh association data, import of computational geometry and querying, point projection, and evaluation of computational geometry.

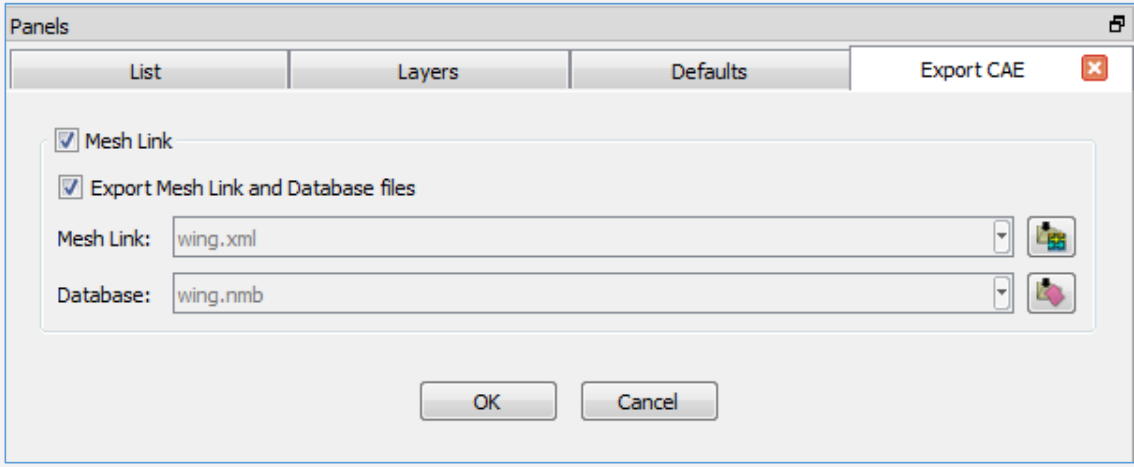
API functions are mesh oriented

Requires geometry-mesh associativity information from the mesh generator

Geometry-Mesh Associativity

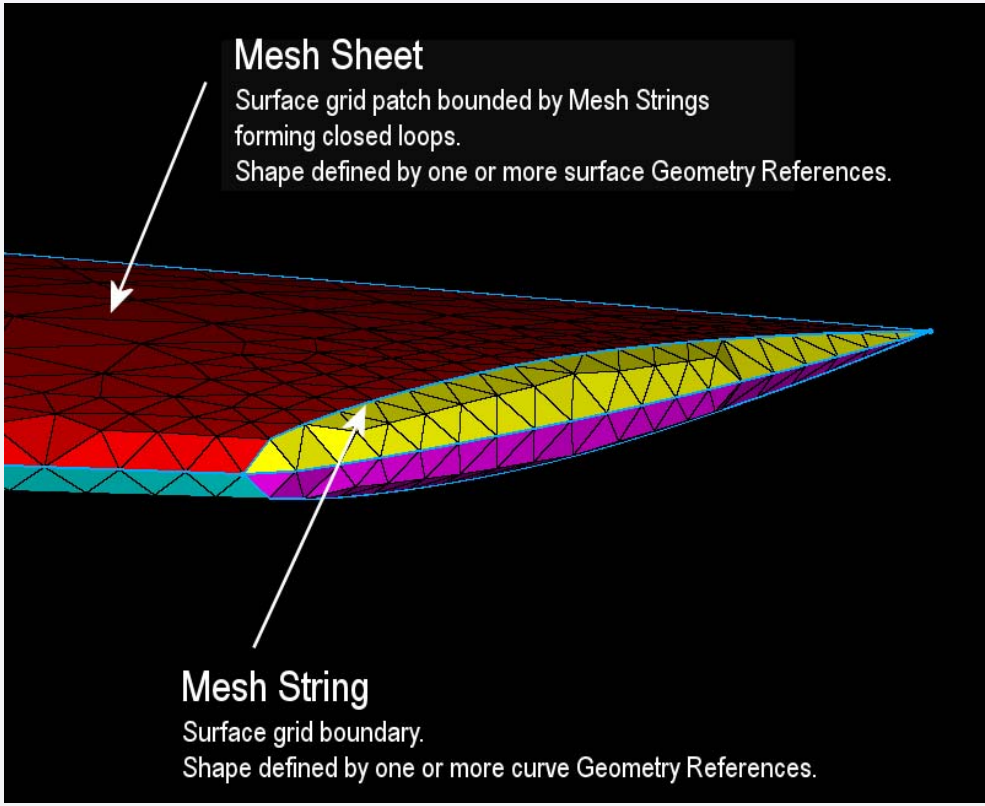
- MeshLink Schema
  - Defines the language for describing geometry-mesh associativity

- MeshLink File - XML representation of MeshLink schema data
  - Conforms to XML standards
  - Defines geometry-mesh associativity
  - Does not define the complete mesh
    - Geometry data file stores the geometry
    - Mesh data file stores the full mesh
  - Allows for validation against schema
  - Export with CAE mesh in Pointwise V18.3



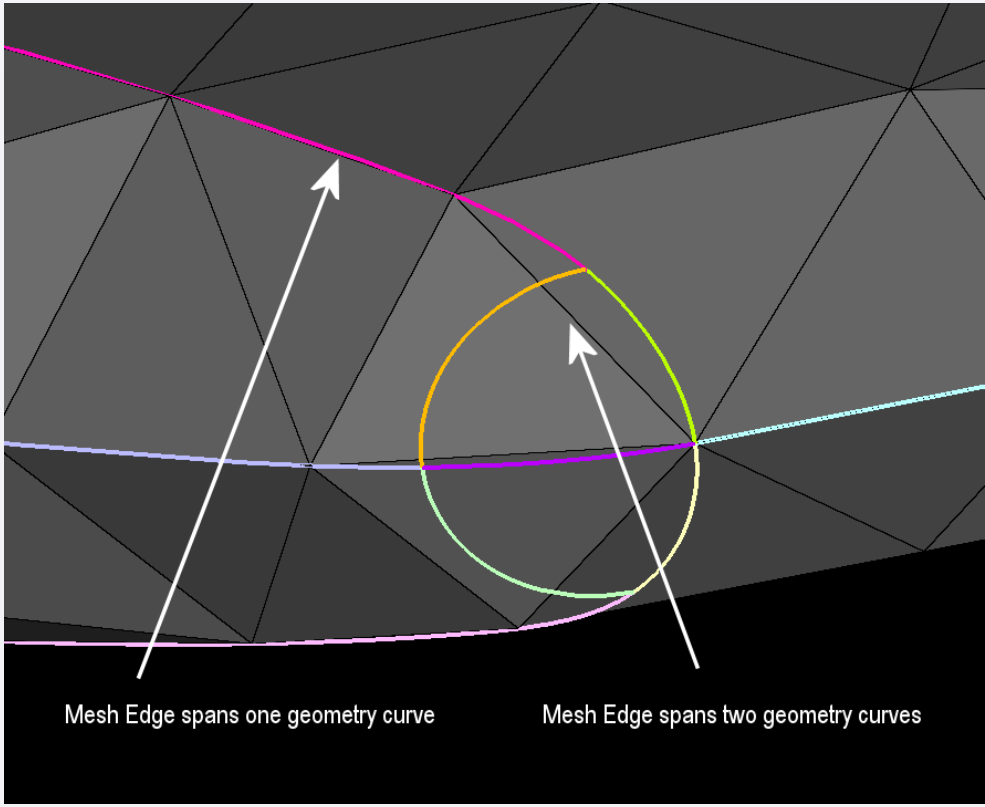
Mesh Associativity - Topology

- Mesh String – 1D collection of Mesh Edges
  - Pointwise Connector
- Mesh Sheet – 2D collection of Mesh Faces
  - Pointwise Domain
- Mesh Model – 3D collection of Mesh Strings and Mesh Sheets
  - Pointwise Block
- Optional geometry association
- Can be defined explicitly in MeshLink file or by reference to mesh data



Mesh Associativity - Elements

- Mesh Point – reference to a point in mesh data file
  - Defined by reference to mesh data only
- Mesh Edge – reference to an edge in mesh data file
  - Defined by reference to mesh data OR by point references
- Mesh Face – reference to a face in mesh data file
  - Defined by reference to mesh data OR by point references
- May have geometry association or utilize parent topology association



Computational Geometry Association

POINTWISE®
THE CHOICE FOR CFD MESHING

- Geometry reference
  - Defined by reference (name, ID, etc.) to entity in the geometry file
  - Globally unique GID integer ID required
  - Groups of geometry entities may be assigned a GID
- Mesh topology/elements
  - Optional GREF attribute matching defined GID
  - May be different at each topological level
    - Mesh String
      - May specify geometry curve OR surface GREF
      - Parent association for contained Mesh Edges
      - Optional Parametric Vertex info
    - Mesh Sheet
      - Surface GREF expected
      - Parent association for contained Mesh Faces
      - Optional Parametric Vertex info
    - Mesh Edge
      - Optional geometry curve OR surface GREF
    - Mesh Face
      - Optional surface GREF

```
<GeometryFile filename="sphere_ml.nmb" aref="2">
  <GeometryReference gid="1" ref="surface-6"/>
  <GeometryReference gid="2" ref="surface-3"/>
</GeometryFile>
<!-- geometry group containing GREf's 1 & 2 -->
<GeometryGroup gid="3">1 2</GeometryGroup>
```

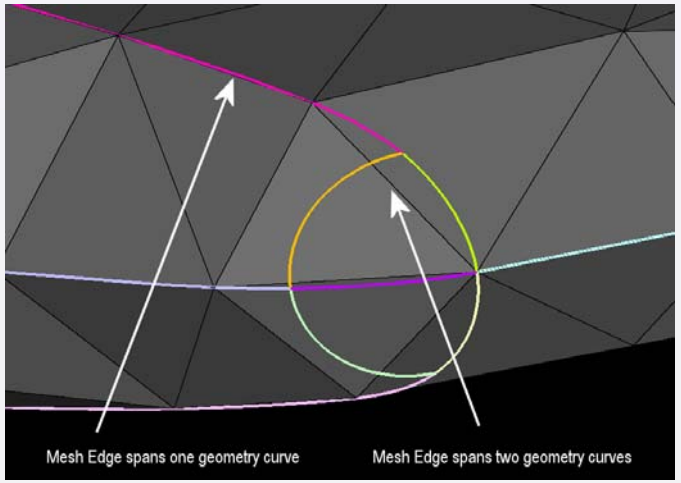
```
<MeshString name="root/con-8" gref="8">
  <MeshEdge etype="Edge2" count="2">9 6 6 3</MeshEdge>
</MeshString>
```

```
<MeshSheet name="root/dom-2" gref="3">
  <MeshFace etype="Quad4">
   1 2 5 4
  </MeshFace>
  <MeshFace etype="Quad4" gref="2">
   2 3 6 5
  </MeshFace>
  <MeshFace etype="Quad4" format="text" count="2">
   4 5 8 7
   5 6 9 8
  </MeshFace>
</MeshSheet>
```

Geometry Association for Mesh Points

POINTWISE®
THE CHOICE FOR CFD MESHING

- May have different association at each topological level
  - E.g. When splitting an edge, want the geometry curve association rather than face's surface association
- Parametric Vertex
  - Defined by reference (name, ID, etc.) to point in mesh file
  - GREF attribute matching defined non-group GID required
  - DIM attribute specifies dimension of data content
  - Content specifies coordinates in geometric entity




```
<MeshString name="root/con-8" gref="8">
  <ParamVertex vref="9" gref="8" dim="1">0.811</ParamVertex>
  <ParamVertex vref="6" gref="8" dim="1">0.924</ParamVertex>
  <ParamVertex vref="3" gref="2" dim="1">1.000</ParamVertex>
  <MeshEdge etype="Edge2" count="2">9 6 6 3</MeshEdge>
</MeshString>
```

Geometry Association for Mesh Points

POINTWISE®
THE CHOICE FOR CFD MESHING

- Mesh topology
  - Optional Parametric Vertex at Model/Sheet/String level
  - May be different at each topological level
- Mesh String
  - Parametric Vertex defines geometry coords for points in contained Mesh Edges
- Mesh Sheet
  - Parametric Vertex defines geometry coords for points in contained Mesh Faces
- Mesh Model
  - Parametric Vertex defines geometry coords for points in absence of Mesh String/Sheet
  - Non-unique!

```
<MeshString name="root/con-8" gref="8">
  <ParamVertex vref="9" gref="8" dim="1">0.411</ParamVertex>
  <ParamVertex vref="6" gref="8" dim="1">0.524</ParamVertex>
  <ParamVertex vref="3" gref="2" dim="1">0.000</ParamVertex>
  <MeshEdge etype="Edge2" count="2">9 6 6 3</MeshEdge>
</MeshString>
```

```
<!-- my special points defined at the MeshModel topology level -->
<MeshPointReference gref="15" count="2">18 17 </MeshPointReference>
<ParamVertex vref="17" gref="15" dim="1">0.1234</ParamVertex>
<ParamVertex vref="18" gref="15" dim="1">0.567</ParamVertex>
```

Working with Mesh Associativity

POINTWISE®
THE CHOICE FOR CFD MESHING

- MeshTopo
  - Generic class for common data
    - Unique ID
    - GREF – geometry reference
    - AREF – attribute reference
    - Name – unique string identifying the entity
    - REF – reference to entity in mesh data file
    - ParamVertices – parametric coordinate info
  - Provides get/set functions for typical workflow patterns

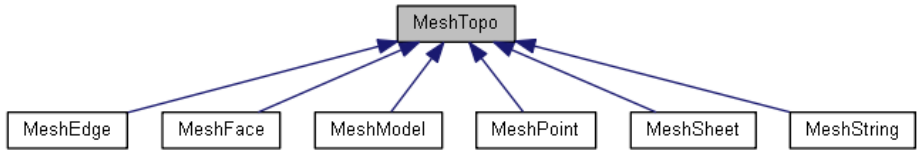
Public Member Functions | Protected Attributes | Static Private Attri
Lis

MeshTopo Class Reference

Base class for mesh topology entities. More...

`#include <MeshTopo.h>`

Inheritance diagram for MeshTopo:



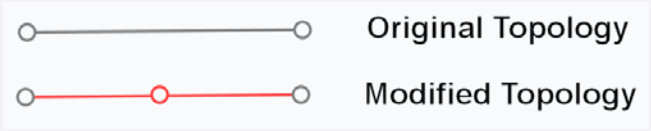
[legend]

Public Member Functions

	MeshTopo (**MLINT** mid, **MLINT** aref, **MLINT** gref, std::string &name) Constructor. More...
	MeshTopo (std::string &ref, **MLINT** mid, **MLINT** aref, **MLINT** gref, std::string &name) Construct with reference to entity in mesh data. More...
virtual **MLINT**	**getID** () const Return the ID of this **MeshTopo**. More...
virtual **MLINT**	**getGref** () const Return the GID of **GeometryGroup** associated with this **MeshTopo**. More...
virtual **MLINT**	**getAref** () const Return the AttID of **MeshLinkAttribute** associated with this **MeshTopo**. More...
virtual void	**getName** (const char **name) const Return the name of this **MeshTopo**. More...

Working with Mesh Strings

- MeshString Class
  - Derived from MeshTopo
  - 1D (curve) mesh topology
  - Container for Mesh Edges
  - Provides query functions based on edge indices & reference
- Usage Example: Splitting an edge by inserting the edge mid-point



1. Query MeshString for parent edge by indices
2. Project the new point onto the geometry associated with the parent edge
3. Add two new child edges to the MeshString
4. Delete the parent edge from the MeshString

Public Member Functions | Private Attributes | Static Private Attributes | Friends

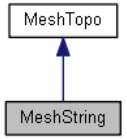
MeshString Class Reference

List of all member

1D (curve) mesh topology More...

`#include <MeshString.h>`

Inheritance diagram for MeshString:



[legend]

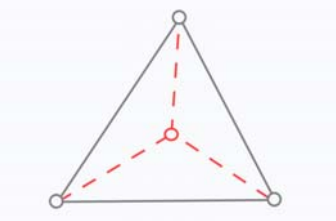
Public Member Functions

	MeshString (**MLINT** mid, **MLINT** aref, **MLINT** gref, std::string &name) Constructor. More...
	MeshString (std::string &ref, **MLINT** mid, **MLINT** aref, **MLINT** gref, std::string &name) Construct with reference to String entity in mesh data. More...
virtual bool	**addEdge** (**MLINT** i1, **MLINT** i2, **MLINT** mid, **MLINT** aref, **MLINT** gref, std::string &name, **ParamVertex** *pv1, **ParamVertex** *pv2, bool mapID) Add a **MeshEdge** to the **MeshString** using indices. More...
virtual bool	**addEdge** (std::string &ref, **MLINT** mid, **MLINT** aref, **MLINT** gref, std::string &name, **ParamVertex** *pv1, **ParamVertex** *pv2, bool mapID) Add a **MeshEdge** to the **MeshString** using reference. More...
virtual **MeshEdge** *	**findEdgeByInds** (**MLINT** i1, **MLINT** i2) const Find a **MeshEdge** in the **MeshString** associativity data. More...
virtual void	**deleteEdgeByInds** (**MLINT** i1, **MLINT** i2) Delete a **MeshEdge** from the **MeshString** associativity data. More...

Working with Mesh Sheets

- MeshSheet Class
  - Derived from MeshTopo
  - 2D (surface) mesh topology
  - Container for Mesh Faces
  - Provides query functions based on face indices & reference

- Usage Example: Splitting a triangular face by inserting the face mid-point



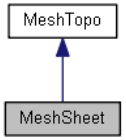
1. Query MeshSheet for parent face by indices
2. Project the new point onto the geometry associated with the parent face
3. Add three new child faces to the MeshSheet
4. Delete the parent face from the MeshSheet

MeshSheet Class Reference

2D (surface) mesh topology More...

`#include <MeshSheet.h>`

Inheritance diagram for MeshSheet:



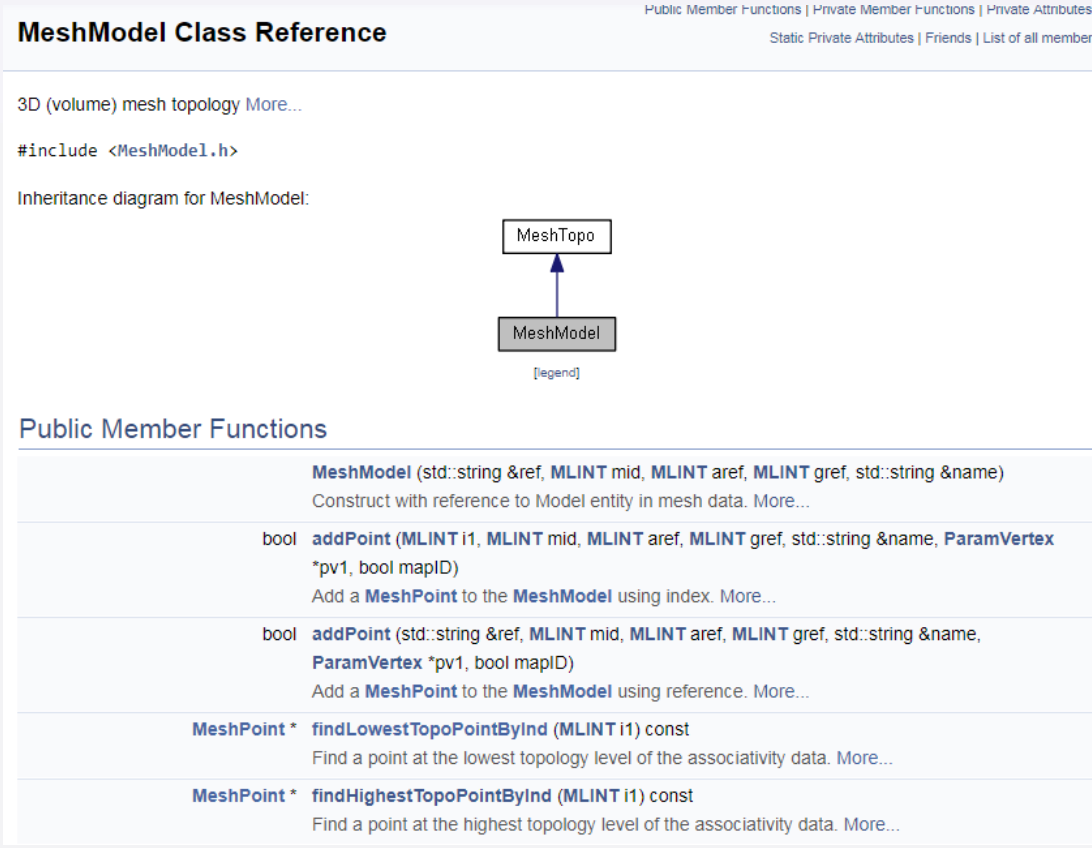
[legend]

Public Member Functions

	MeshSheet (**MLINT** mid, **MLINT** aref, **MLINT** gref, std::string &name) Constructor. More...
	MeshSheet (std::string &ref, **MLINT** mid, **MLINT** aref, **MLINT** gref, std::string &name) Construct with reference to Sheet entity in mesh data. More...
virtual bool	**addFace** (**MLINT** i1, **MLINT** i2, **MLINT** i3, **MLINT** mid, **MLINT** aref, **MLINT** gref, std::string &name, **ParamVertex** *pv1, **ParamVertex** *pv2, **ParamVertex** *pv3, bool mapID) Add a tri **MeshFace** to the **MeshSheet** using indices. More...
virtual **MeshFace** *	**findFaceByInds** (**MLINT** i1, **MLINT** i2, **MLINT** i3, **MLINT** i4=**MESH_TOPO_INDEX_UNUSED**) const Find a **MeshFace** in the **MeshSheet** associativity data. More...
virtual void	**deleteFaceByInds** (**MLINT** i1, **MLINT** i2, **MLINT** i3, **MLINT** i4=**MESH_TOPO_INDEX_UNUSED**) Delete a **MeshFace** from the **MeshSheet** associativity data. More...

Working with Mesh Models

- MeshModel Class
  - Derived from MeshTopo
  - 3D (surface) mesh topology
  - Container for Mesh Strings/Sheets
  - Provides query functions based on point/edge/face indices & reference
- Mainly a way to organize data associated with regions in the mesh data file
- Can be used to query Point/Edge/Face associativity
  - Point/Edge results are non-unique
  - E.g. Only one point returned matching index, but may be present in multiple Mesh Strings/Sheets with different association



MeshModel Class Reference

Public Member Functions | Private Member Functions | Private Attributes
Static Private Attributes | Friends | List of all member

3D (volume) mesh topology More...

`#include <MeshModel.h>`

Inheritance diagram for MeshModel:

Public Member Functions

	MeshModel (std::string &ref, **MLINT** mid, **MLINT** aref, **MLINT** gref, std::string &name) Construct with reference to Model entity in mesh data. More...
bool	**addPoint** (**MLINT** i1, **MLINT** mid, **MLINT** aref, **MLINT** gref, std::string &name, **ParamVertex** *pv1, bool mapID) Add a **MeshPoint** to the **MeshModel** using index. More...
bool	**addPoint** (std::string &ref, **MLINT** mid, **MLINT** aref, **MLINT** gref, std::string &name, **ParamVertex** *pv1, bool mapID) Add a **MeshPoint** to the **MeshModel** using reference. More...
MeshPoint *	**findLowestTopoPointByInd** (**MLINT** i1) const Find a point at the lowest topology level of the associativity data. More...
MeshPoint *	**findHighestTopoPointByInd** (**MLINT** i1) const Find a point at the highest topology level of the associativity data. More...

Geometry Kernel

- Generic GeometryKernel base class
- GeometryKernel operations
  - Geometry file import
  - Parametric evaluation
  - Point projection to computational geometry
- Multiple geometry kernels may be used
- Extended with Pointwise Project Geode kernel
  - https://www.pointwise.com/geode/

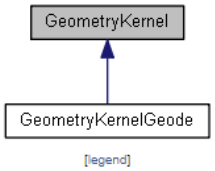
POINTWISE®
THE CHOICE FOR CFD MESHING

GeometryKernel Class Reference

Base class for geometry kernel interface. More...

`#include <GeometryKernel.h>`

Inheritance diagram for GeometryKernel:



Public Member Functions

virtual bool	**read** (const char *filename) Read the geometry data file. More...
virtual bool	**projectPoint** (const **GeometryGroup** *group, const **MLVector3D** point, **ProjectionData** &projectionData) Project a Cartesian point onto the Geometry group. More...
virtual bool	**getProjectionXYZ** (**ProjectionData** &projectionData, **MLVector3D** point) Return the projection hit Cartesian coordinates. More...
virtual bool	**getProjectionUV** (**ProjectionData** &projectionData, **MLVector2D** UV) Return the projection hit entity parametric coordinates. More...
virtual bool	**getProjectionEntityName** (**ProjectionData** &projectionData, std::string &name) Return the projection hit entity name. More...
virtual bool	**evalXYZ** (**MLVector2D** UV, const std::string &entityName, **MLVector3D** xyz) Evaluate the Cartesian coordinates at the entity parametric coordinates. More...
virtual bool	**evalRadiusOfCurvature** (**MLVector2D** UV, const std::string &entityName, **MLREAL** *minRadiusOfCurvature, **MLREAL** *maxRadiusOfCurvature)

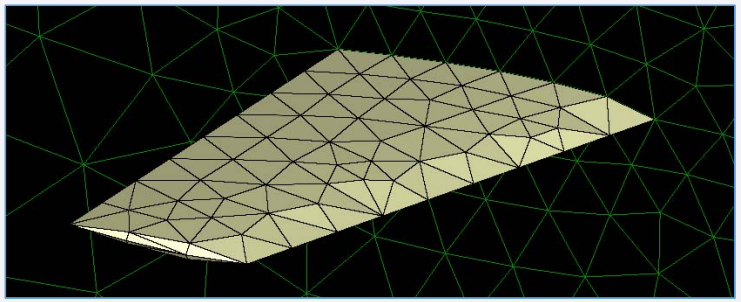
Code Example – Project Point Onto Edge Geometry

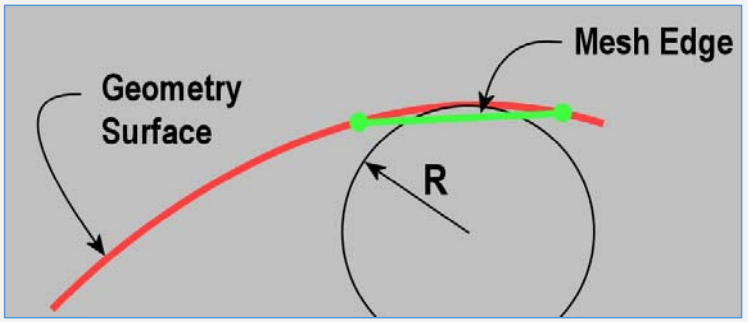
POINTWISE®
THE CHOICE FOR CFD MESHING

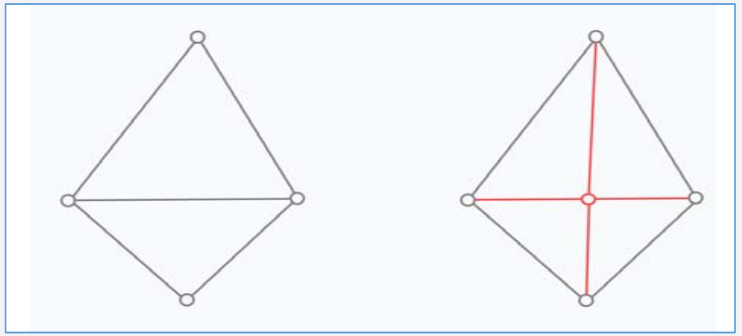
```
MLVector3D point = { 1.0, 2.0, 3.0 };
MeshTopo *meshObject = meshModel->findLowestTopoEdgeByInds( 101, 102 );
if ( !meshObject ) return 1; // missing from mesh assoc. database
MLINT gid = meshObject->getGref();
GeometryGroup *geomGroup = meshAssoc->getGeometryGroupByID( gid );
if ( geomGroup ) {
        // meshObject is associated with a geometry group
        // project point onto the geometry group
        ProjectionData projectionData( geomKernel );
        if ( geomKernel->projectPoint( geomGroup, point, projectionData ) ) {
                // projection successful, get the position
                MLVector3D projectedPt;
                geomKernel->getProjectionXYZ( projectionData, projectedPt );
        }
}
```

Demonstration

- Refine an unstructured mesh to meet geometry resolution goal
  - Coarse mesh created in Pointwise
  - Mesh exported to NASA FUN3D
  - Geometry exported to NMB
  - MeshLink file exported to XML
- Refinement Goals
  - Resolve geometry curvature such that mesh edge circular arc subtension < 20 deg
  - Minimum edge length - 0.5% chord
  - Maximum triangle aspect ratio – 20
  - Minimum triangle included angle – 5 deg
- Refinement Method
  - Only refine surface mesh
    - Clearer example of mechanics
  - Split edge where geometry is poorly resolved
    - 4 new edges, 4 new faces
  - Surface mesh exported to VRML

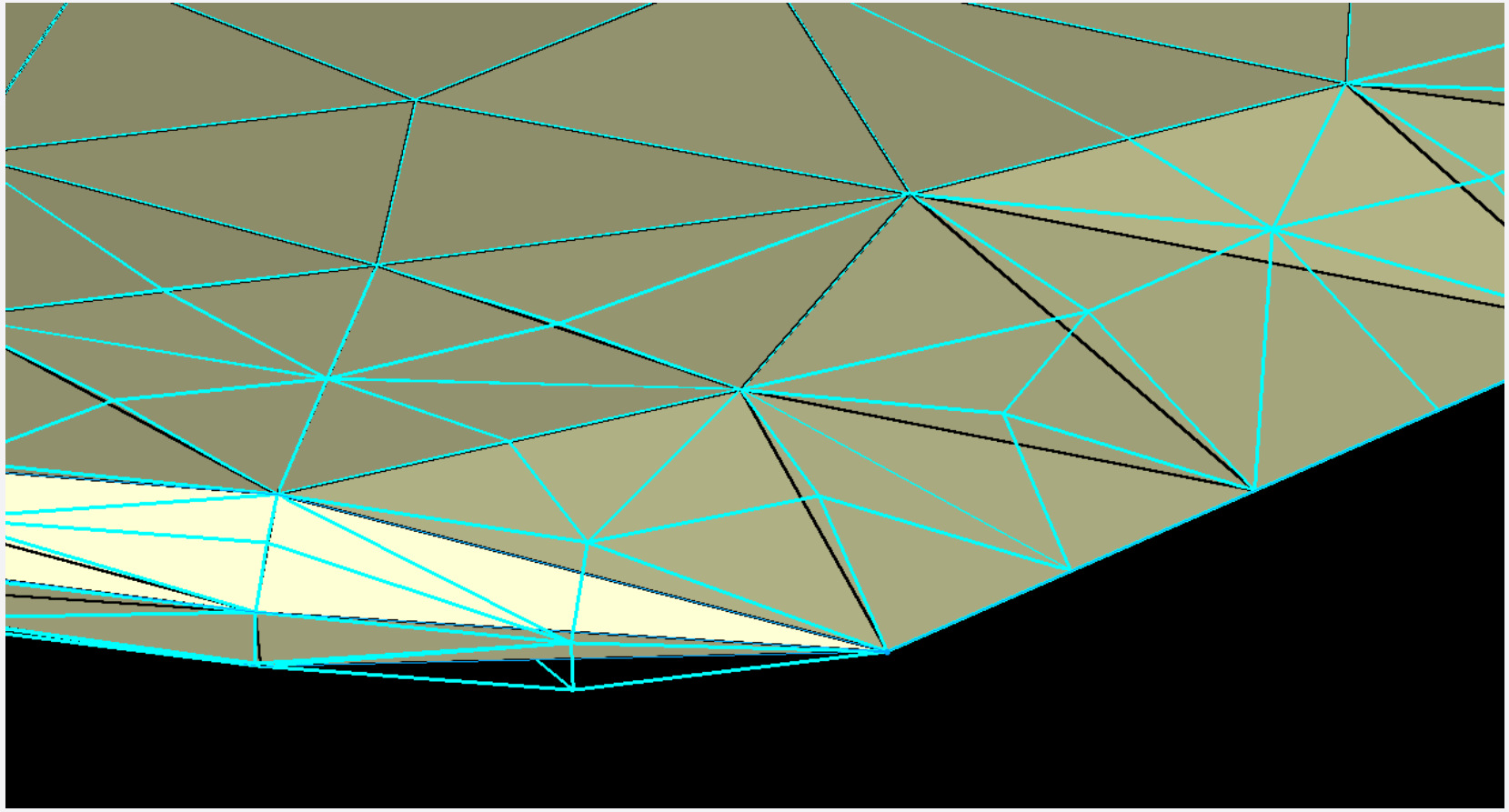






Demonstration Results

POINTWISE®
THE CHOICE FOR CFD MESHING



New edges (cyan) after one generation of splitting

Demonstration Results

POINTWISE®
THE CHOICE FOR CFD MESHING

- Initial Mesh
  - Number of Faces: 860
  - Number of Edges: 1290
  - Number of Constrained Edges: 307
  - Avg. Edge Arc Subtension: 68.8 deg
  - Max. Edge Arc Subtension: 1856.4 deg
    - Edge Length : 1.8968e-01
    - Rad. of Curvature : 5.8541e-03

- Refined Mesh
  - Performed 8921 edge splits in 6 generations
  - Number of Faces: 18702
  - Number of Edges: 28053
  - Number of Constrained Edges: 4272
  - Avg. Edge Arc Subtension: 7.2 deg
  - Max. Edge Arc Subtension: 20.0 deg
    - Edge Length : 1.1836e-02
    - Rad. of Curvature : 3.3909e-02