OpenSim::Smith2018ContactMesh Class Reference

This component is used to represent the articular surfaces in the **Smith2018ArticularContactForce** component. More...

Inheritance diagram for OpenSim::Smith2018ContactMesh:

OpenSim Properties, Sockets, Outputs, Inputs

Properties (single-value)

std::string	**mesh_file** "Path to triangle mesh geometry file representing the contact surface " "(supports .obj, .stl, .vtp)." More...
double	**elastic_modulus** "Uniform Elastic Modulus value for every triangle in mesh. " "The default value is 1000000.0 Pa." More...
double	**poissons_ratio** "Uniform Poissons Ratio value for every triangle in mesh. " "The default value is 0.5." More...
double	**thickness** "Uniform thickness of elastic layer for entire mesh. " "The default value is 0.005 meters" More...
bool	**use_variable_thickness** "Compute the local thickness for each triangle in mesh_file by " "calculating the distance along a normal ray cast from the center of " "each triangle in the mesh_file to intersection with the " "mesh_back_file. If use_variable_thickness is true, mesh_back_file " "must be defined and the 'thickness' property value is not used." "The Default value is false." More...
SimTK::Vec3	**scale_factors** "[x,y,z] scale factors applied to vertex locations of the mesh_file " "and mesh_back_file meshes." More...

Properties (optional)

std::string	**mesh_back_file** "Path to traingle mesh geometry file representing the backside of " "contact surface elastic layer (bone / backside of artifical " "component) mesh geometry file (supports .obj, .stl, .vtp). " More...
double	**min_thickness** "Minimum thickness threshold for elastic layer [m] when calculating " "cartilage thickness for each triangle." More...
double	**max_thickness** "Maximum thickness threshold for elastic layer [m] when calculating " "cartilage thickness for each triangle." More...

Sockets

PhysicalFrame	**scale_frame** "When using the ScaleTool, the scale factors from this frame will be " "used to scale the mesh." More...

OpenSim Properties, Sockets, Outputs, Inputs inherited from OpenSim::ContactGeometry

OpenSim Properties, Sockets, Outputs, Inputs inherited from OpenSim::Component

Public Member Functions

	Smith2018ContactMesh ()
	Smith2018ContactMesh (const std::string &name, const std::string &**mesh_file**, const **PhysicalFrame** &**frame**)
	Smith2018ContactMesh (const std::string &name, const std::string &**mesh_file**, const **PhysicalFrame** &**frame**, const SimTK::Vec3 &**location**, const SimTK::Vec3 &**orientation**)
	Smith2018ContactMesh (const std::string &name, const std::string &**mesh_file**, const **PhysicalFrame** &**frame**, const SimTK::Vec3 &**location**, const SimTK::Vec3 &**orientation**, bool **use_variable_thickness**, const std::string &**mesh_back_file**, double **min_thickness**, double **max_thickness**)
SimTK::ContactGeometry	**createSimTKContactGeometry** () const override Create a new SimTK::ContactGeometry based on this object. More...
const SimTK::PolygonalMesh &	**getPolygonalMesh** () const
int	**getNumFaces** () const
int	**getNumVertices** () const
const std::set< int > &	**getNeighborTris** (int tri) const
const std::vector< std::vector< int > > &	**getRegionalTriangleIndices** () const
const double &	**getTriangleThickness** (int i) const
const double &	**getTriangleElasticModulus** (int i) const
const double &	**getTrianglePoissonsRatio** (int i) const
const SimTK::Vector &	**getTriangleAreas** () const
const SimTK::Vector_< SimTK::Vec3 > &	**getTriangleCenters** () const
const SimTK::Vector_< SimTK::UnitVec3 > &	**getTriangleNormals** () const
const SimTK::Matrix_< SimTK::Vec3 > &	**getFaceVertexLocations** () const
const SimTK::Vector_< SimTK::Vec3 > &	**getVertexLocations** () const
const **OBBTreeNode** &	**getOBBTreeNode** () const
int	**getOBBNumTriangles** () const
bool	**rayIntersectMesh** (const SimTK::Vec3 &origin, const SimTK::UnitVec3 &direction, const double &min_proximity, const double &max_proximity, int &tri, SimTK::Vec3 intersection_point, SimTK::Real &distance) const
void	**generateDecorations** (bool fixed, const **ModelDisplayHints** &hints, const SimTK::State &s, SimTK::Array_< SimTK::DecorativeGeometry > &geometry) const override

	Optional method for generating arbitrary display geometry that reflects this Component at the specified *state*. More...
const **PhysicalFrame** &	**getMeshFrame** () const

Property-related functions

const std::string &	**get_mesh_file** () const Get the value of the **mesh_file** property. More...
std::string &	**upd_mesh_file** () Get a writable reference to the **mesh_file** property. More...
void	**set_mesh_file** (const std::string &value) Set the value of the **mesh_file** property. More...
const double &	**get_elastic_modulus** () const Get the value of the **elastic_modulus** property. More...
double &	**upd_elastic_modulus** () Get a writable reference to the **elastic_modulus** property. More...
void	**set_elastic_modulus** (const double &value) Set the value of the **elastic_modulus** property. More...
const double &	**get_poissons_ratio** () const Get the value of the **poissons_ratio** property. More...
double &	**upd_poissons_ratio** () Get a writable reference to the **poissons_ratio** property. More...
void	**set_poissons_ratio** (const double &value) Set the value of the **poissons_ratio** property. More...
const double &	**get_thickness** () const Get the value of the **thickness** property. More...
double &	**upd_thickness** () Get a writable reference to the **thickness** property. More...
void	**set_thickness** (const double &value) Set the value of the **thickness** property. More...
const bool &	**get_use_variable_thickness** () const Get the value of the **use_variable_thickness** property. More...
bool &	**upd_use_variable_thickness** () Get a writable reference to the **use_variable_thickness** property. More...
void	**set_use_variable_thickness** (const bool &value) Set the value of the **use_variable_thickness** property. More...
const std::string &	**get_mesh_back_file** () const Get the value of the **mesh_back_file** property. More...
std::string &	**upd_mesh_back_file** () Get a writable reference to the **mesh_back_file** property. More...
void	**set_mesh_back_file** (const std::string &value) Set the value of the **mesh_back_file** property. More...
const double &	**get_min_thickness** () const Get the value of the **min_thickness** property. More...
double &	**upd_min_thickness** () Get a writable reference to the **min_thickness** property. More...
void	**set_min_thickness** (const double &value) Set the value of the **min_thickness** property. More...
const double &	**get_max_thickness** () const Get the value of the **max_thickness** property. More...
double &	**upd_max_thickness** () Get a writable reference to the **max_thickness** property. More...
void	**set_max_thickness** (const double &value) Set the value of the **max_thickness** property. More...
const SimTK::Vec3 &	**get_scale_factors** () const Get the value of the **scale_factors** property. More...
SimTK::Vec3 &	**upd_scale_factors** () Get a writable reference to the **scale_factors** property. More...
void	**set_scale_factors** (const SimTK::Vec3 &value) Set the value of the **scale_factors** property. More...

Socket-related functions

void	**connectSocket_scale_frame** (const **Object** &object) Connect the 'scale_frame' **Socket** to an object of type **PhysicalFrame**. More...

- **Public Member Functions inherited from OpenSim::ContactGeometry**
- **Public Member Functions inherited from OpenSim::ModelComponent**
- **Public Member Functions inherited from OpenSim::Component**
- **Public Member Functions inherited from OpenSim::Object**

Auto-generated functions

static **Smith2018ContactMesh** *	**safeDownCast** (**OpenSim::Object** *obj) For use in MATLAB and Python to access the concrete class. More...
static const std::string &	**getClassName** () This returns "Smith2018ContactMesh". More...
Smith2018ContactMesh *	**clone** () const override Create a new heap-allocated copy of the concrete object to which this Object refers. More...
const std::string &	**getConcreteClassName** () const override Returns the class name of the concrete Object-derived class of the actual object referenced by this Object, as a string. More...

Additional Inherited Members

- **Static Public Member Functions inherited from OpenSim::ContactGeometry**
- **Static Public Member Functions inherited from OpenSim::ModelComponent**
- **Static Public Member Functions inherited from OpenSim::Component**
- **Static Public Member Functions inherited from OpenSim::Object**

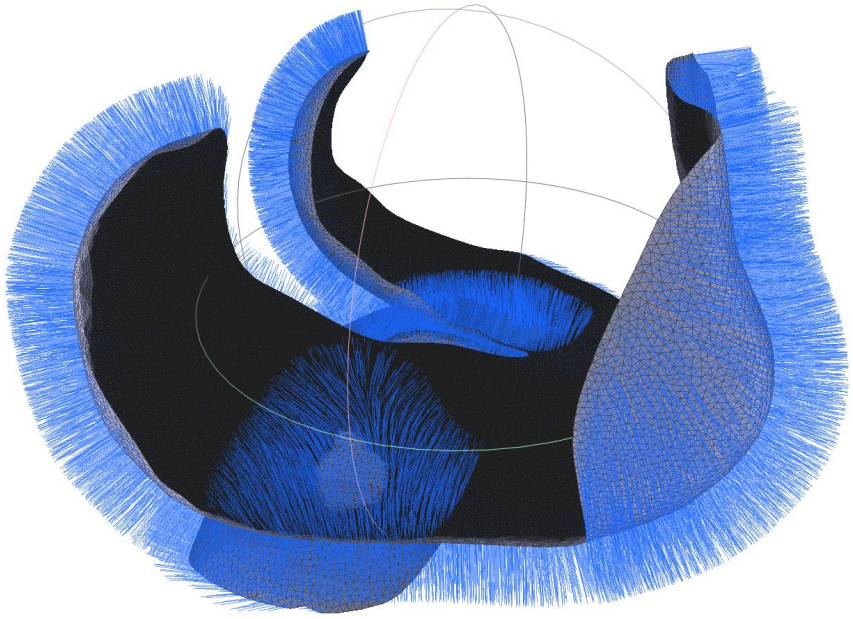
Detailed Description

This component is used to represent the articular surfaces in the **Smith2018ArticularContactForce** component.

The **Smith2018ContactMesh** reads in a triangle mesh from the mesh_file property (.stl, .vtp, .obj) that represents only the contact surface. This component can only be used with the **Smith2018ArticularContactForce** component to be in contact with another **Smith2018ContactMesh**.

Mesh Requirements

The mesh does not need to be closed (ie water tight) and a smaller number of triangles in the mesh will lead to faster collision detection performance. The normal vectors of the mesh should be pointing outwards from the articular surface towards the opposing contact mesh. Misdirected triangle normals is a common issue when constructing new meshes.



Variable Thickness

The **Smith2018ContactMesh** can calculate the local thickness at each triangle to generate spatially varying thickness maps. Here the optional mesh_back_file property must be defined using a mesh to represent the subchondral bone or back side of an implant component. The thickness is calculated by casting a normal ray from the center of each triangle in the mesh_file mesh towards the mesh_back_file mesh. The thickness is clipped so that it remains within the range defined by min_thickness and max_thickness.

Scaling

The mesh can be linearly scaled in the x,y,z directions using the scale_factors property. When using the **ScaleTool**, these scale factors are set based on the frame set in the scale_frame socket. It is generally advisable to scale both **Smith2018ContactMesh** meshes used as a contacting pair by the same scale factors to ensure the congruency of the articulating surfaces is not substaintially altered. For example, in a knee joint the scale_frame socket for both the femur and tibia Smith2018ContactMeshes would be connected to the femur frame to ensure both meshes are scaled by the femur scale factors.

Choosing the coarseness of the mesh

Because the contact force and potential energy are calculated based on the triangle areas and normals, and the derivatives of these outputs with respect to joint coordinates are commonly calculated in integrators and optimizers, the best performance will be achieved with a smooth mesh whose triangle areas are similar sized. In some extreme cases where the mesh becomes excessively coarse, the simulation slow down caused by jumps in the computed forces outweighs the speed up in collison detection gained by reducing the number of triangles in the mesh. For details on a convergence study based on triangle area see [1]. Note that the GPU implementation described in the paper is not implemented here.

Collison Detection

The collision detection algorithm is described in the Smith2018ArticularContact class description. The **Smith2018ContactMesh** stores all geometric mesh data and also performs ray intersection tests with a individual mesh triangles or an Oriented Bounding Box (OBB) hierarchy. Here, a SimTK::OrientedBoundingBox is constructed for the mesh_file geometry using code adapted from SimTK::ContactGeometry::TriangularMesh::OBBTreeNodeImpl.

Constructor & Destructor Documentation

◆ Smith2018ContactMesh() [1/4]

```
OpenSim::Smith2018ContactMesh::Smith2018ContactMesh ( )
```

◆ Smith2018ContactMesh() [2/4]

```
OpenSim::Smith2018ContactMesh::Smith2018ContactMesh ( const std::string &      name,
                                                      const std::string &      mesh_file,
                                                      const PhysicalFrame &    frame
                                                    )
```

◆ Smith2018ContactMesh() [3/4]

```
OpenSim::Smith2018ContactMesh::Smith2018ContactMesh ( const std::string &      name,
                                                      const std::string &      mesh_file,
                                                      const PhysicalFrame &    frame,
                                                      const SimTK::Vec3 &      location,
                                                      const SimTK::Vec3 &      orientation
                                                    )
```

◆ Smith2018ContactMesh() [4/4]

```
OpenSim::Smith2018ContactMesh::Smith2018ContactMesh ( const std::string &      name,
                                                      const std::string &      mesh_file,
                                                      const PhysicalFrame &    frame,
                                                      const SimTK::Vec3 &      location,
                                                      const SimTK::Vec3 &      orientation,
                                                      bool                     use_variable_thickness,
                                                      const std::string &      mesh_back_file,
                                                      double                   min_thickness,
                                                      double                   max_thickness
                                                    )
```

Member Function Documentation

◆ clone()

Smith2018ContactMesh* OpenSim::Smith2018ContactMesh::clone ( ) const inline override virtual

Create a new heap-allocated copy of the concrete object to which this Object refers.

It is up to the caller to delete the returned object when no longer needed. Every concrete object deriving from Object implements this pure virtual method automatically, via the declaration macro it invokes (e.g., **OpenSim_DECLARE_CONCRETE_OBJECT()**). Note that the concrete class overrides modify the return type to be a pointer to the *concrete* object; that still overrides the base class method because the return type is covariant with (that is, derives from) Object.

Implements **OpenSim::ContactGeometry**.

◆ connectSocket_scale_frame()

void OpenSim::Smith2018ContactMesh::connectSocket_scale_frame (const Object & object) inline

Connect the 'scale_frame' **Socket** to an object of type **PhysicalFrame**.

Call **finalizeConnections()** afterwards to update the socket's connectee path property. The reference to the connectee set here takes precedence over the connectee path property.

◆ createSimTKContactGeometry()

SimTK::ContactGeometry OpenSim::Smith2018ContactMesh::createSimTKContactGeometry ( ) const inline override virtual

Create a new SimTK::ContactGeometry based on this object.

Implements **OpenSim::ContactGeometry**.

◆ generateDecorations()

void
OpenSim::Smith2018ContactMesh::generateDecorations (bool **fixed,**
const ModelDisplayHints & **hints,**
const SimTK::State & **state,**
SimTK::Array_< SimTK::DecorativeGeometry > & **appendToThis**
) **const** override virtual

Optional method for generating arbitrary display geometry that reflects this Component at the specified *state*.

This will be called once to obtain ground- and body-fixed geometry (with *fixed=true)*, and then once per frame (with *fixed=false)* to generate on-the-fly geometry such as rubber band lines, force arrows, labels, or debugging aids.

Please note that there is a precondition that the state passed in to generateDecorations be realized to Stage::Position. If your component can visualize quantities realized at Velocity, Dynamics or Acceleration stages, then you must check that the stage has been realized before using/requesting stage dependent values. It is forbidden to realize the model to a higher stage within generateDecorations, because this can trigger costly side- effects such as evaluating all model forces even when performing a purely kinematic study.

If you override this method, be sure to invoke the base class method first, using code like this:

```
void MyComponent::generateDecorations
   (bool                                    fixed,
    const ModelDisplayHints&                hints,
    const SimTK::State&                     state,
    SimTK::Array_<SimTK::DecorativeGeometry>&   appendToThis) const
{
    // invoke parent class method
    Super::generateDecorations(fixed,hints,state,appendToThis);
    // ... your code goes here
    // can render velocity dependent quanities if stage is Velocity or higher
    if(state.getSystemStage() >= Stage::Velocity) {
        // draw velocity vector for model COM
    }
    // can render computed forces if stage is Dynamics or higher
    if(state.getSystemStage() >= Stage::Dynamics) {
        // change the length of a force arrow based on the force in N
    }
}
```

Parameters

[in]	**fixed**	If `true`, generate only geometry that is fixed to a **PhysicalFrame**, configuration, and velocity. Otherwise generate only such dependent geometry.
[in]	**hints**	See documentation for **ModelDisplayHints**; you may want to alter the geometry you generate depending on what you find there. For example, you can determine whether the user wants to see debug geometry.
[in]	**state**	The State for which geometry should be produced. See below for more information.
[in,out]	**appendToThis**	Array to which generated geometry should be *appended* via the `push_back()` method.

When called with *fixed=true only* modeling options and parameters (Instance variables) should affect geometry; time, position, and velocity should not. In that case **OpenSim** will already have realized the *state* through Instance stage. When called with *fixed=false,* you may consult any relevant value in *state*. However, to avoid unnecessary computation, **OpenSim** guarantees only that *state* will have been realized through Position stage; if you need anything higher than that (reaction forces, for example) you should make sure the *state* is realized through Acceleration stage.

Reimplemented from **OpenSim::Component**.

◆ get_elastic_modulus()

const double& OpenSim::Smith2018ContactMesh::get_elastic_modulus ( ) const inline

Get the value of the **elastic_modulus** property.

◆ get_max_thickness()

const double& OpenSim::Smith2018ContactMesh::get_max_thickness ( ) const inline

Get the value of the **max_thickness** property.

◆ get_mesh_back_file()

const std::string& OpenSim::Smith2018ContactMesh::get_mesh_back_file ( ) const inline

Get the value of the **mesh_back_file** property.

◆ get_mesh_file()

const std::string& OpenSim::Smith2018ContactMesh::get_mesh_file ( ) const inline

Get the value of the **mesh_file** property.

◆ get_min_thickness()

const double& OpenSim::Smith2018ContactMesh::get_min_thickness ( ) const inline

Get the value of the **min_thickness** property.

◆ get_poissons_ratio()

const double& OpenSim::Smith2018ContactMesh::get_poissons_ratio ( ) const inline

Get the value of the **poissons_ratio** property.

◆ get_scale_factors()

const SimTK::Vec3& OpenSim::Smith2018ContactMesh::get_scale_factors ( ) const inline

Get the value of the **scale_factors** property.

◆ get_thickness()

const double& OpenSim::Smith2018ContactMesh::get_thickness ( ) const inline

Get the value of the **thickness** property.

◆ get_use_variable_thickness()

const bool& OpenSim::Smith2018ContactMesh::get_use_variable_thickness ( ) const inline

Get the value of the **use_variable_thickness** property.

◆ getClassName()

static const std::string& OpenSim::Smith2018ContactMesh::getClassName ( ) inline static

This returns "Smith2018ContactMesh".

See **getConcreteClassName()** if you want the class name of the underlying concrete object instead.

◆ getConcreteClassName()

const std::string& OpenSim::Smith2018ContactMesh::getConcreteClassName ( ) const inline override virtual

Returns the class name of the concrete Object-derived class of the actual object referenced by this Object, as a string.

This is the string that is used as the tag for this concrete object in an XML file. Every concrete class derived from Object automatically overrides this method via the declaration macro it uses. See **getClassName()** to get the class name of the referencing (possibly abstract) class rather than the concrete object.

See also
getClassName()

Implements **OpenSim::ContactGeometry**.

◆ getFaceVertexLocations()

const SimTK::Matrix_<SimTK::Vec3>& OpenSim::Smith2018ContactMesh::getFaceVertexLocations ( ) const inline

◆ getMeshFrame()

const PhysicalFrame& OpenSim::Smith2018ContactMesh::getMeshFrame ( ) const inline

◆ getNeighborTris()

const std::set<int>& OpenSim::Smith2018ContactMesh::getNeighborTris (int tri) const inline

◆ getNumFaces()

int OpenSim::Smith2018ContactMesh::getNumFaces ( ) const inline

◆ getNumVertices()

int OpenSim::Smith2018ContactMesh::getNumVertices ( ) const inline

◆ getOBBNumTriangles()

int OpenSim::Smith2018ContactMesh::getOBBNumTriangles ( ) const inline

◆ getOBBTreeNode()

const OBBTreeNode& OpenSim::Smith2018ContactMesh::getOBBTreeNode ( ) const inline

◆ getPolygonalMesh()

const SimTK::PolygonalMesh& OpenSim::Smith2018ContactMesh::getPolygonalMesh ( ) const inline

◆ getRegionalTriangleIndices()

const std::vector<std::vector<int> >& OpenSim::Smith2018ContactMesh::getRegionalTriangleIndices ( ) const inline

◆ getTriangleAreas()

```
const SimTK::Vector& OpenSim::Smith2018ContactMesh::getTriangleAreas ( ) const
```
inline

◆ getTriangleCenters()

```
const SimTK::Vector_<SimTK::Vec3>& OpenSim::Smith2018ContactMesh::getTriangleCenters ( ) const
```
inline

◆ getTriangleElasticModulus()

```
const double& OpenSim::Smith2018ContactMesh::getTriangleElasticModulus ( int i ) const
```
inline

◆ getTriangleNormals()

```
const SimTK::Vector_<SimTK::UnitVec3>& OpenSim::Smith2018ContactMesh::getTriangleNormals ( ) const
```
inline

◆ getTrianglePoissonsRatio()

```
const double& OpenSim::Smith2018ContactMesh::getTrianglePoissonsRatio ( int i ) const
```
inline

◆ getTriangleThickness()

```
const double& OpenSim::Smith2018ContactMesh::getTriangleThickness ( int i ) const
```
inline

◆ getVertexLocations()

```
const SimTK::Vector_<SimTK::Vec3>& OpenSim::Smith2018ContactMesh::getVertexLocations ( ) const
```
inline

◆ rayIntersectMesh()

```
bool OpenSim::Smith2018ContactMesh::rayIntersectMesh ( const SimTK::Vec3 &      origin,
                                                       const SimTK::UnitVec3 &  direction,
                                                       const double &           min_proximity,
                                                       const double &           max_proximity,
                                                       int &                    tri,
                                                       SimTK::Vec3              intersection_point,
                                                       SimTK::Real &            distance
                                                     )                          const
```

◆ safeDownCast()

```
static Smith2018ContactMesh* OpenSim::Smith2018ContactMesh::safeDownCast ( OpenSim::Object * obj )
```
inline static

For use in MATLAB and Python to access the concrete class.

Example: `cObj = Smith2018ContactMesh.safeDownCast(obj)`. This is equivalent to `dynamic_cast<Smith2018ContactMesh*>(obj)` in C++.

◆ set_elastic_modulus()

void OpenSim::Smith2018ContactMesh::set_elastic_modulus (const double & value) inline

Set the value of the **elastic_modulus** property.

◆ set_max_thickness()

void OpenSim::Smith2018ContactMesh::set_max_thickness (const double & value) inline

Set the value of the **max_thickness** property.

◆ set_mesh_back_file()

void OpenSim::Smith2018ContactMesh::set_mesh_back_file (const std::string & value) inline

Set the value of the **mesh_back_file** property.

◆ set_mesh_file()

void OpenSim::Smith2018ContactMesh::set_mesh_file (const std::string & value) inline

Set the value of the **mesh_file** property.

◆ set_min_thickness()

void OpenSim::Smith2018ContactMesh::set_min_thickness (const double & value) inline

Set the value of the **min_thickness** property.

◆ set_poissons_ratio()

void OpenSim::Smith2018ContactMesh::set_poissons_ratio (const double & value) inline

Set the value of the **poissons_ratio** property.

◆ set_scale_factors()

void OpenSim::Smith2018ContactMesh::set_scale_factors (const SimTK::Vec3 & value) inline

Set the value of the **scale_factors** property.

◆ set_thickness()

void OpenSim::Smith2018ContactMesh::set_thickness (const double & value) inline

Set the value of the **thickness** property.

◆ set_use_variable_thickness()

void OpenSim::Smith2018ContactMesh::set_use_variable_thickness (const bool & value) inline

Set the value of the **use_variable_thickness** property.

◆ upd_elastic_modulus()

double& OpenSim::Smith2018ContactMesh::upd_elastic_modulus ( ) inline

Get a writable reference to the **elastic_modulus** property.

◆ upd_max_thickness()

double& OpenSim::Smith2018ContactMesh::upd_max_thickness ( ) inline

Get a writable reference to the **max_thickness** property.

◆ upd_mesh_back_file()

std::string& OpenSim::Smith2018ContactMesh::upd_mesh_back_file ( ) inline

Get a writable reference to the **mesh_back_file** property.

◆ upd_mesh_file()

std::string& OpenSim::Smith2018ContactMesh::upd_mesh_file ( ) inline

Get a writable reference to the **mesh_file** property.

◆ upd_min_thickness()

double& OpenSim::Smith2018ContactMesh::upd_min_thickness ( ) inline

Get a writable reference to the **min_thickness** property.

◆ upd_poissons_ratio()

double& OpenSim::Smith2018ContactMesh::upd_poissons_ratio ( ) inline

Get a writable reference to the **poissons_ratio** property.

◆ upd_scale_factors()

SimTK::Vec3& OpenSim::Smith2018ContactMesh::upd_scale_factors ( ) inline

Get a writable reference to the **scale_factors** property.

◆ upd_thickness()

double& OpenSim::Smith2018ContactMesh::upd_thickness ( ) inline

Get a writable reference to the **thickness** property.

◆ upd_use_variable_thickness()

bool& OpenSim::Smith2018ContactMesh::upd_use_variable_thickness ( ) inline

Get a writable reference to the **use_variable_thickness** property.

OpenSim Property, Socket, Output, Input Documentation

◆ elastic_modulus

double OpenSim::Smith2018ContactMesh::elastic_modulus

"Uniform Elastic Modulus value for every triangle in mesh. " "The default value is 1000000.0 Pa."

This property appears in XML files under the tag **<elastic_modulus>**. This property was generated with the **OpenSim_DECLARE_PROPERTY** macro; see **Property** to learn about the property system.

See also
get_elastic_modulus(), **upd_elastic_modulus()**, **set_elastic_modulus()**

◆ max_thickness

double OpenSim::Smith2018ContactMesh::max_thickness

"Maximum thickness threshold for elastic layer [m] when calculating " "cartilage thickness for each triangle."

This property appears in XML files under the tag **<max_thickness>**. This property was generated with the **OpenSim_DECLARE_OPTIONAL_PROPERTY** macro; see **Property** to learn about the property system.

See also
get_max_thickness(), **upd_max_thickness()**, **set_max_thickness()**

◆ mesh_back_file

std::string OpenSim::Smith2018ContactMesh::mesh_back_file

"Path to traingle mesh geometry file representing the backside of " "contact surface elastic layer (bone / backside of artifical " "component) mesh geometry file (supports .obj, .stl, .vtp). "

This property appears in XML files under the tag **<mesh_back_file>**. This property was generated with the **OpenSim_DECLARE_OPTIONAL_PROPERTY** macro; see **Property** to learn about the property system.

See also
get_mesh_back_file(), **upd_mesh_back_file()**, **set_mesh_back_file()**

◆ mesh_file

std::string OpenSim::Smith2018ContactMesh::mesh_file

"Path to triangle mesh geometry file representing the contact surface " "(supports .obj, .stl, .vtp)."

This property appears in XML files under the tag **<mesh_file>**. This property was generated with the **OpenSim_DECLARE_PROPERTY** macro; see **Property** to learn about the property system.

See also
get_mesh_file(), **upd_mesh_file()**, **set_mesh_file()**

◆ min_thickness

double OpenSim::Smith2018ContactMesh::min_thickness

"Minimum thickness threshold for elastic layer [m] when calculating " "cartilage thickness for each triangle."

This property appears in XML files under the tag **<min_thickness>**. This property was generated with the **OpenSim_DECLARE_OPTIONAL_PROPERTY** macro; see **Property** to learn about the property system.

See also
get_min_thickness(), **upd_min_thickness()**, **set_min_thickness()**

◆ poissons_ratio

double OpenSim::Smith2018ContactMesh::poissons_ratio

"Uniform Poissons Ratio value for every triangle in mesh. " "The default value is 0.5."

This property appears in XML files under the tag **<poissons_ratio>**. This property was generated with the **OpenSim_DECLARE_PROPERTY** macro; see **Property** to learn about the property system.

See also
get_poissons_ratio(), **upd_poissons_ratio()**, **set_poissons_ratio()**

◆ scale_factors

SimTK::Vec3 OpenSim::Smith2018ContactMesh::scale_factors

"[x,y,z] scale factors applied to vertex locations of the mesh_file " "and mesh_back_file meshes."

This property appears in XML files under the tag **<scale_factors>**. This property was generated with the **OpenSim_DECLARE_PROPERTY** macro; see **Property** to learn about the property system.

See also
get_scale_factors(), **upd_scale_factors()**, **set_scale_factors()**

◆ scale_frame

PhysicalFrame OpenSim::Smith2018ContactMesh::scale_frame

"When using the ScaleTool, the scale factors from this frame will be " "used to scale the mesh."

In an XML file, you can set this **Socket**'s connectee path via the **<socket_scale_frame >** element. This socket was generated with the **OpenSim_DECLARE_SOCKET** macro; see **AbstractSocket** for more information.

See also
connectSocket_scale_frame()

◆ thickness

double OpenSim::Smith2018ContactMesh::thickness

"Uniform thickness of elastic layer for entire mesh. " "The default value is 0.005 meters"

This property appears in XML files under the tag **<thickness>**. This property was generated with the **OpenSim_DECLARE_PROPERTY** macro; see **Property** to learn about the property system.

See also
get_thickness(), **upd_thickness()**, **set_thickness()**

◆ use_variable_thickness

bool OpenSim::Smith2018ContactMesh::use_variable_thickness

"Compute the local thickness for each triangle in mesh_file by " "calculating the distance along a normal ray cast from the center of " "each triangle in the mesh_file to intersection with the " "mesh_back_file. If use_variable_thickness is true, mesh_back_file " "must be defined and the 'thickness' property value is not used." "The Default value is false."

This property appears in XML files under the tag **<use_variable_thickness>**. This property was generated with the **OpenSim_DECLARE_PROPERTY** macro; see **Property** to learn about the property system.

See also

get_use_variable_thickness(), **upd_use_variable_thickness()**, **set_use_variable_thickness()**

The documentation for this class was generated from the following file:

- OpenSim/Simulation/Model/Smith2018ContactMesh.h