Frontmesh

Reference Manual

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Collection< Item > Class Template Reference

`#include <frontmesh.h>`

Public Member Functions

- int **number** ()
- void **add** (Item *item)
- void **push** (Item *item)
- void **insert** (Item *item)
- void **append** (Item *item)
- void **remove** ()
- void **erase** ()
- void **remove** (Item *item)
- void **erase** (Item *item)
- void **goFirst** ()
- void **goLast** ()
- void **goNext** ()
- void **goPrev** ()
- bool **isFirst** ()
- bool **isLast** ()
- void **goTag** ()
- void **setTag** ()
- void **untag** ()
- bool **isTagged** ()
- void **removeTagged** ()
- void **tagFirst** ()
- void **tagNext** ()
- void **tagPrev** ()
- bool **tagIsFirst** ()
- bool **tagIsCurrent** ()
- Item * **getItem** ()
- void **replaceItem** (Item *newitem)
- Item * **getTaggedItem** ()
- void **clean** ()
- void **wipe** ()
- void **show** (string name)

3.1.1 Detailed Description

template<class Item>
class Collection< Item >

Collection class implements linked list operations on other data strutures.

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

- frontmesh.h

3.2 Node Struct Reference

`#include <frontmesh.h>`

Public Member Functions

- **Node** (Point *Q)
- **Node** (Point *Q, Segment *A, Segment *B)
- REAL **coordinate** (int i)
- REAL * **coordinates** ()
- REAL **distance** (REAL y[ ])
- REAL **distance** (Node *node)
- REAL **distance** (Node *node, REAL dist[ ])
- void **addSegs** (Segment *A, Segment *B)
- void **add** (Segment *seg)
- int **numNeibs** ()
- void **replace** (Segment *oldseg, Segment *newseg)
- bool **isNeib** (Node *node)
- void **show** (string name)
- void **setSegs** (Segment *s0, Segment *s1)
- Collection< Segment > * **getSegs** ()
- Segment * **getSeg** (Node *node)
- Segment * **getOtherSeg** (Segment *seg)
- void **remove** (Segment *seg)

Public Attributes

- int **index**
- Collection< Segment > **segs**
- Point * **P** = NULL

3.2.1 Detailed Description

Node contains a point and a list of segments which are connected to it

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

- frontmesh.h

3.3 Point Struct Reference

`#include <frontmesh.h>`

Public Member Functions

- **Point** (Vector &newX, Vector &newF)
- **Point** (int ind, REAL *newX, REAL *newF)
- void **set** (Vector &newX, Vector &newF)
- void **setRange** (REAL r)
- REAL **bond** (Point *Q)
- void **show** (string name)

Public Attributes

- REAL **range** = 0.0
- int **i** = 0
- Vector **X**
- Vector **F**

3.3.1 Detailed Description

Point data strucure combines position vector with the surface normal vector.

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

- frontmesh.h

3.4 Segment Class Reference

`#include <frontmesh.h>`

Public Member Functions

- **Segment** (Node *n0, Node *n1, Node *n2)
- void **setTriangle** (Triangle *tri, int vert)
- Triangle * **getTriangle** ()
- void **setEndNodes** (Node *n0, Node *n1)
- Vector * **getNorm** ()
- void **setNode** (int i, Node *n)
- Node * **getNode** (int i)
- Node ** **getNodes** ()
- Node * **getOtherNode** (Node *n)
- Node * **getApex** ()
- void **setApex** (Node *nd)
- REAL **length** ()
- void **show** (string name)
- void **setNorm** ()
- Point * **getVertex** ()

3.4.1 Detailed Description

Segment cosists of two nodes and a vector in the direction perpendicular to the line connecting the nodes. The direction points inside the already generated mesh and is used to avoid points already in the mesh when selecting a new point for a surface triangle.

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

- frontmesh.h

3.5 Triangle Class Reference

`#include <frontmesh.h>`

Public Member Functions

- **Triangle** (Point ∗A, Point ∗B, Point ∗C)
- void **setVert** (int i, Point ∗P)
- Point ∗ **getVert** (int i)
- bool **replaceVert** (Point ∗P, Point ∗Q)
- bool **has** (Point ∗point)
- REAL **perimeter** ()
- REAL **area** ()
- void **show** (string name)

3.5.1 Detailed Description

Triangle holds pointers to three points which form its vertices

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

- frontmesh.h

3.6 Vector Class Reference

`#include <frontmesh.h>`

Public Member Functions

- **Vector** (REAL x, REAL y, REAL z)
- **Vector** (REAL X[ ])
- **Vector** (Vector ∗X)
- **Vector** (Vector &X)
- REAL ∗ **vec** ()
- void **set** (int i, REAL x)
- void **set** (REAL ∗X)
- REAL **get** (int i)
- void **set** (Vector ∗X)
- void **set** (Vector &X)
- void **add** (Vector &X)
- void **add** (Vector ∗X)
- void **sub** (Vector ∗X)
- void **sub** (Vector &X)
- void **sub** (Vector &X, Vector &Y)
- void **sub** (Vector &X, Vector ∗Y)
- void **sub** (Vector ∗X, Vector ∗Y)
- void **mid** (Vector &A, Vector &B)
- void **mid** (Vector ∗A, Vector ∗B)
- void **mul** (REAL c)
- void **rot** (REAL angle, Vector &axis)
- REAL **len** ()
- void **one** ()
- void **cross** (Vector ∗A, Vector ∗B)
- void **show** (string name)

3.6.1 Detailed Description

Vector class. Implements vector operations.

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

- frontmesh.h

Chapter 4

File Documentation

4.1 frontmesh.cc File Reference

Implementation of the propagating front routines.

```
#include <cstdlib>
#include <unistd.h>
#include <string.h>
#include <math.h>
#include <iostream>
#include <fstream>
#include "frontmesh.h"
```

Functions

- void **doc** ()
- void **usage** (char *prog)
- int main (int argc, char *argv[])
- bool isIntersecting (Point *A, Point *B, Segment *seg0, Segment *seg, Node *node, int level)
- REAL maxBond (Point *P, Collection< Point > &points, Segment *seg, Point *&Q)
- REAL maxBond (Point *P, Collection< Node > &nodes, Segment *seg, Point *&Q)
- void moveNode (Node *node, Collection< Node > &setA, Collection< Node > &setB)
- int frontmesh (int nPoints, REAL *Coords, REAL *Snorms, int *Indices, int *&Faces)

Variables

- bool **verbose** = false
- bool **silent** = false

4.1.1 Detailed Description

Implementation of the propagating front routines.

Author: Andrei Smirnov andrei.v.smirnov@gmail.com

4.1.2 Function Documentation

4.1.2.1 frontmesh()

```
int frontmesh (
            int nPoints,
            REAL * Coords,
            REAL * Snorms,
            int * Indices,
            int *& Faces )
```

Generates Mesh Using Propagating Front Method.

4.1.2.2 isIntersecting()

```
bool isIntersecting (
            Point * A,
            Point * B,
            Segment * seg0,
            Segment * seg,
            Node * node,
            int level )
```

Checks if a segment connecting points A and B intersects other segments in the front.

4.1.2.3 main()

```
int main (
            int argc,
            char * argv[] )
```

Main routine perfoming IO and calling the mesher.

4.1.2.4 maxBond() [1/2]

```
REAL maxBond (
            Point * P,
            Collection< Point > & points,
            Segment * seg,
            Point *& Q )
```

Finds point with maximum bonding to a given point

4.1.2.5 maxBond() [2/2]

```
REAL maxBond (
            Point * P,
            Collection< Node > & nodes,
            Segment * seg,
            Point *& Q )
```

Finds a front node with maximum bonding to a given point.

4.1.2.6 moveNode()

```
void moveNode (
            Node * node,
            Collection< Node > & setA,
            Collection< Node > & setB )
```

Moves node from one collection to another.

4.2 frontmesh.h File Reference

Header file with definitions of basic data structures.

Classes

- class Vector
- struct Point
- class Collection< Item >
- class Segment
- struct Node
- class Triangle

Macros

- #define **INT** int
- #define **REAL** double
- #define **BIG** 1.e30
- #define **SMALL** 1.e-30
- #define **DIM** 3
- #define **NV** 3
- #define **MIN_ASPECT_RATIO** 0.005
- #define **ZIP_PROXIMITY** 0.2
- #define **RANGE** 2.5

Functions

- REAL **dist** (Vector &A, Vector &B)
- REAL **dist** (Vector *A, Vector *B)
- REAL **dot** (Vector &A, Vector &B)
- REAL **dot** (Vector &A, Vector *B)
- REAL **dot** (Vector *A, Vector *B)
- void **add** (Vector &A, Vector &B, Vector &C)
- void **mid** (Vector &A, Vector &B, Vector &C)
- void **sub** (Vector &A, Vector &B, Vector &C)
- void **mul** (REAL c, Vector &A)
- void **cross** (Vector &A, Vector &B, Vector &C)
- bool **clockwise** (Vector *A, Vector *B, Vector *C, Vector *D)
- bool **intersecting** (Vector *A, Vector *B, Vector *C, Vector *D, Vector *E)
- REAL **perimeter** (Point *A, Point *B, Point *C)
- REAL **area** (Point *A, Point *B, Point *C)

Variables

- bool **silent**
- bool **verbose**

4.2.1 Detailed Description

Header file with definitions of basic data structures.

Index