Fade3D

v0.98

Generated by Doxygen 1.8.13

Contents

1 Main Page

1.1 3D Delaunay Triangulation for C++: Fade3D

Fade3D is a 3D Delaunay triangulation (tetrahedralization) library for C++. It generalizes the algorithms of the established Fade2D library from 2D to 3D. Fade3D is very fast.

1.1.1 Background and Development State

Development of Fade2D and Fade3D has started in 2009. The triangulation library Fade2D has gone public in 2010, it was successful and consequently most development time ran into Fade2D and its extensions while the tetrahedralization library Fade3D has been kept for internal use. This year Fade3D has received new attention and development work. It has been turned into a C++ library, more documentation has been written and test routines with random geometric objects have been run for weeks to ensure to the greatest possible extent that Fade3D is stable and robust. And this is still the primary goal: Making Fade3D absolutely stable and robust. All bugs and inconveniences from earlier beta versions have been fixed and there are no known problems in Fade3D v0.99. The software is now on the threshold of industrial usability, commercial testers are welcome.

Nevertheless we call the present Fade3D tetrahedralization v0.99 a beta version. Please don't hesitate to report anything you discover.

1.1.2 Download and Getting Started

Download Fade3D 0.99beta

Fade works without installation. Just unzip and compile the contained example source codes.

1.1.2.1 Linux and Apple Users:

- Enter the **examples** directory, type **make** and start the executable **./allExamples3D**

1.1.2.2 Windows Users:

- Enter the **examples**/**vs20XX_exampleProject**/ directory, open the contained solution (∗.sln) file and compile. Find the executable in the Win32 or x64 folder. It is best to run the example from a command line window.

1.1.2.3 Directory Contents

- **examples**
  Source code of the examples.
- **include_fade3d**
  Header files
- **Win32** and **x64**
  The DLL's for 32- and 64-bit Windows. Visual Studio 2008, 2010, 2012, 2013, 2015 and 2017 is supported
- **lib_${DISTRO}_${ARCHITECTURE}**
  The shared libraries (∗.so) for Linux and Mac.
- **doc**
  PDF Documentation

1.1.3 License

A student license for non-commercial research software is availablei free of charge. Soon Fade3D will exit the beta testing stage and then commercial licenses will also be available. Commercial testers are already welcome. If you use a free-of-charge version please put a link to this software on your website.

1.1.4 Release notes / History

Version 0.99beta, October 29th., 2017
In previous versions insert(vector<Point3> vPoints,vector<Point3*>& vHandles) returned the pointers in vHandles in arbitrary order. Altough this was not really a bug it was unexpected and is fixed now: The order of the pointers returned in vHandles corresponds to the order in vPoints now. Small internal improvements have been made to detect duplicate vertices early. Support for Raspberry PI has been added.

Version 0.98beta, September 26th. , 2017
Bugfix: Now the locate() method works
Version 0.97beta, September 19th. , 2017
First public release of the Fade3D library. Please report any problems you may find so that they can be fixed quickly.

2 Class Index

2.1 Class List

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

3 Class Documentation

3.1 FADE3D::Ball3 Class Reference

Sphere.

`#include <Ball3.h>`

Public Member Functions

- Ball3 (double x, double y, double z, double sqRadius_)
- double getRadius () const
- double getSqRadius () const
- Point3 getCenter () const

Protected Attributes

- Point3 **center**
- double **sqRadius**

Friends

- std::ostream & **operator**<< (std::ostream &stream, const Ball3 &b)

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Ball3()

```
FADE3D::Ball3::Ball3 (
            double x,
            double y,
            double z,
            double sqRadius_ )
```

Parameters

x,y,z	coordinates of the center
*sq↩ Radius_*	is the squared radius

3.1.2 Member Function Documentation

3.1.2.1 getCenter()

```
Point3 FADE3D::Ball3::getCenter ( ) const
```

Get the center

3.1.2.2 getRadius()

```
double FADE3D::Ball3::getRadius ( ) const
```

Get the radius

3.1.2.3 getSqRadius()

```
double FADE3D::Ball3::getSqRadius ( ) const
```

Get the squared radius

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

- /home/geom/repo/dev/geomDev/dt3/dt3Library/Ball3.h

3.2 FADE3D::Bbox3 Class Reference

Axis-aligned minimal 3D bounding box.

```
#include <Bbox3.h>
```

Public Member Functions

- Bbox3 ()
  Constructor.
- Bbox3 (const std::vector< Point3 > &vPoints)
  Constructor.
- bool isValid ()
  Check if the bounding box is valid.
- bool add (const std::vector< Point3 > &vPoints)
  Add points to the bounding box.
- bool add (const Point3 &p)
  Add a point to the bounding box.
- Bbox3 operator+ (Bbox3 &b)
  Add another bounding box.
- Point3 getMinPoint ()
  Get the minimum corner.
- Point3 getMaxPoint ()
  Get the maximum corner.
- double getMinCoord ()
  Get the smallest coordinate.
- double getMaxCoord ()
  Get the largest coordinate.

- double getRangeX () const

  Get the x-range.
- double getRangeY () const

  Get the y-range.
- double getRangeZ () const

  Get the z-range.
- double getMaxRange () const

  Get the maximum range.
- double getMinX () const

  Get the minimal x coordinate.
- double getMinY () const

  Get the minimal y coordinate.
- double getMinZ () const

  Get the minimal z coordinate.
- double getMaxX () const

  Get the maximal x coordinate.
- double getMaxY () const

  Get the maximal y coordinate.
- double getMaxZ () const

  Get the maximal z coordinate.

Protected Attributes

- double **minX**
- double **minY**
- double **minZ**
- double **maxX**
- double **maxY**
- double **maxZ**

Friends

- class **HC3**
- std::ostream & **operator**<< (std::ostream &stream, Bbox3 &pC)

3.2.1 Constructor & Destructor Documentation

3.2.1.1 Bbox3() [1/2]

```
FADE3D::Bbox3::Bbox3 ( )  [inline]
```

Constructor. The bounds of the bounding box are initialized to -DBL_MAX and +DBL_MAX values. The bounding box becomes valid as soon as points are added.

3.2.1.2 Bbox3() [2/2]

```
FADE3D::Bbox3::Bbox3 (
            const std::vector< Point3 > & vPoints )  [inline], [explicit]
```

This constructor computes the axis aligned minimal bounding box of the points in vPoints

3.2.2 Member Function Documentation

3.2.2.1 add() [1/2]

```
bool FADE3D::Bbox3::add (
            const std::vector< Point3 > & vPoints )  [inline]
```

Add `vPoints` to the bounding box.

Returns

true if the bounds of the present Bbox3 have changed
false otherwise

3.2.2.2 add() [2/2]

```
bool FADE3D::Bbox3::add (
            const Point3 & p )  [inline]
```

Add `p` to the bounding box

Returns

true if the bounds of the present Bbox3 have changed
false otherwise

3.2.2.3 getMaxCoord()

```
double FADE3D::Bbox3::getMaxCoord ( )  [inline]
```

Returns

the maximum of (maxX,maxY,maxZ)

3.2.2.4 getMaxPoint()

```
Point3 FADE3D::Bbox3::getMaxPoint ( )  [inline]
```

Returns

the point with the largest coordinates of the bounding box

3.2.2.5 getMaxRange()

```
double FADE3D::Bbox3::getMaxRange ( ) const  [inline]
```

Returns

the maximum of rangeX,rangeY and rangeZ

3.2.2.6 getMaxX()

```
double FADE3D::Bbox3::getMaxX ( ) const  [inline]
```

3.2.2.7 getMaxY()

```
double FADE3D::Bbox3::getMaxY ( ) const  [inline]
```

3.2.2.8 getMaxZ()

```
double FADE3D::Bbox3::getMaxZ ( ) const  [inline]
```

3.2.2.9 getMinCoord()

```
double FADE3D::Bbox3::getMinCoord ( )  [inline]
```

Returns

the minimum of (minX,minY,minZ)

3.2.2.10 getMinPoint()

```
Point3 FADE3D::Bbox3::getMinPoint ( )  [inline]
```

Returns

the point with the smallest coordinates of the bounding box

3.2.2.11 getMinX()

```
double FADE3D::Bbox3::getMinX ( ) const  [inline]
```

3.2.2.12 getMinY()

```
double FADE3D::Bbox3::getMinY ( ) const  [inline]
```

3.2.2.13 getMinZ()

```
double FADE3D::Bbox3::getMinZ ( ) const  [inline]
```

3.2.2.14 getRangeX()

```
double FADE3D::Bbox3::getRangeX ( ) const  [inline]
```

Returns

the x-range maxX-minX

3.2.2.15 getRangeY()

```
double FADE3D::Bbox3::getRangeY ( ) const  [inline]
```

Returns

the y-range maxY-minY

3.2.2.16 getRangeZ()

```
double FADE3D::Bbox3::getRangeZ ( ) const  [inline]
```

Returns

the z-range maxZ-minZ

3.2.2.17 isValid()

```
bool FADE3D::Bbox3::isValid ( )  [inline]
```

Check if the bounding box has valid bounds. After construction the bounds are initialized to DBL_MAX and DBL↩ _MIN. As soon as the first point is added Bbox3 becomes valid.

3.2.2.18 operator+()

```
Bbox3 FADE3D::Bbox3::operator+ (
            Bbox3 & b )  [inline]
```

Add another Bbox3 to the present one.

Returns

the axis aligned minimal bounding box of the union of the two boxes.

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

- /home/geom/repo/dev/geomDev/dt3/dt3Library/Bbox3.h

3.3 FADE3D::Edge3 Class Reference

Edge of a tetrahedron.

`#include <Edge3.h>`

Public Member Functions

- Edge3 (Tet3 *pTet_, const int opp3_, const int opp2_)
  Constructor of Edge3.
- Tet3 * getTet () const
  Get the tetrahedron.
- int getOpp3Index () const
  Get the opp3ITI index.
- int getOpp2Index () const
  Get the opp2ITI index.
- int getSourceIndex () const
  Get the source index.
- int getTargetIndex () const
  Get the target index.
- Point3 * getSourceVtx () const
  Get the source vertex of the edge.
- Point3 * getTargetVtx () const
  Get the target vertex of the edge.
- bool operator== (const Edge3 &rhs) const
  Check if two undirected edges coincide.
- bool operator!= (const Edge3 &rhs) const
  Check if two undirected edges are different.

Friends

- std::ostream & **operator**<< (std::ostream &stream, Edge3 &e)

3.3.1 Detailed Description

An Edge3 is represented by a Tet3 pointer and two two IntraTetIndices *opp3* and *opp2*. The Edge of the tetrahedron is selected as follows: At first, opp3 selects the facet (triangle) of the tetrahedron opposite to the corner addressed by opp3. Then opp2 selects the edge in this triangle opposite to the vertex at position opp2. See the image where opp3=3 and opp2=0. The edge proceeds from corner 1 to corner 2.

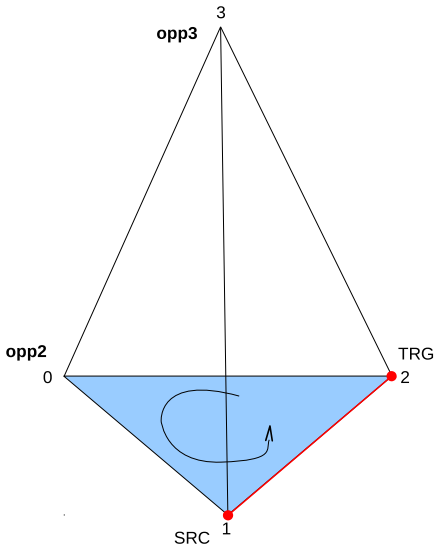

Figure 1 Edge (0,1) of a tetrahedron, selected by opp3=3 and opp2=0

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Edge3()

```
FADE3D::Edge3::Edge3 (
            Tet3 * pTet_,
            const int opp3_,
            const int opp2_ )
```

Parameters

p↩ Tet↩ _	Tetrahedron
opp3↩ _	selects one of the four triangles of the tetrahedron
opp2↩ _	selects one of the edges of the triangle

3.3.3 Member Function Documentation

3.3.3.1 getSourceIndex()

```
int FADE3D::Edge3::getSourceIndex ( ) const
```

Returns

the index of the tetrahedron which selects the source vertex of the edge

3.3.3.2 getTargetIndex()

```
int FADE3D::Edge3::getTargetIndex ( ) const
```

Returns

the index of the tetrahedron which selects the target vertex of the edge

3.3.3.3 operator"!=()

```
bool FADE3D::Edge3::operator!= (
            const Edge3 & rhs ) const
```

Two Edge3 objects are different when they do not refer to the same undirected edge.

3.3.3.4 operator==()

```
bool FADE3D::Edge3::operator== (
            const Edge3 & rhs ) const
```

Two edges are equal when their vertices coincide (undirected edge) even if the two objects use different tetrahedra that meet on this edge.

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

- /home/geom/repo/dev/geomDev/dt3/dt3Library/Edge3.h

3.4 FADE3D::Facet3 Class Reference

Side of a tetrahedron.

```
#include <Facet3.h>
```

Public Member Functions

- Facet3 (Tet3 *pTet_, const int opp3_)
- Tet3 * getTet ()
- int getOpp3Index () const
- int getIntraTetIndex (int ith) const
- Edge3 getEdge (int ith) const
- bool operator== (const Facet3 &other) const
- bool operator!= (const Facet3 &other) const

3.4.1 Detailed Description

A Facet3 is one of the four sides (triangles) of a tetrahedron and it is represented by a Tet3 pointer and the opposite IntraTetIndex.

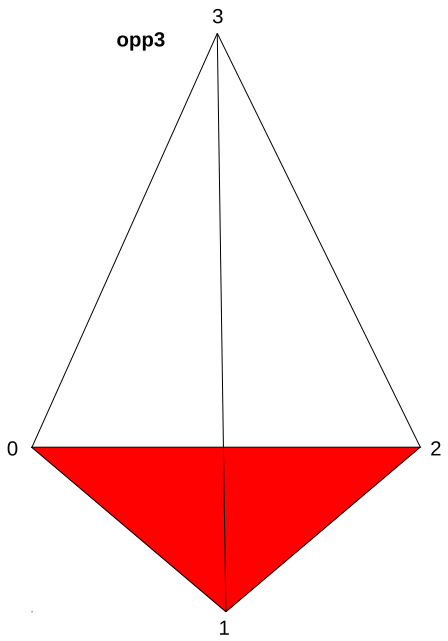

Figure 2 Edge (0,1,2) of a tetrahedron, selected by opp3=3

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Facet3()

```
FADE3D::Facet3::Facet3 (
            Tet3 * pTet_,
            const int opp3_ )
```

Parameters

p↩ Tet↩ _	is a pointer to a Tet3
opp3↩	selects the side of `pTet`

3.4.3 Member Function Documentation

3.4.3.1 getEdge()

```
Edge3 FADE3D::Facet3::getEdge (
            int ith ) const
```

Get the `ith` edge of the facet

Parameters

ith	{0,1,2} selects the edge to be returned

3.4.3.2 getIntraTetIndex()

```
int FADE3D::Facet3::getIntraTetIndex (
            int ith ) const
```

Get the index of the `ith` vertex of the facet

3.4.3.3 getOpp3Index()

```
int FADE3D::Facet3::getOpp3Index ( ) const
```

Get the IntraTetIndex

Returns

the IntraTetIndex of the tetrahedron which selects the side

3.4.3.4 getTet()

```
Tet3* FADE3D::Facet3::getTet ( )
```

Get the Tet3

3.4.3.5 operator"!=()

```
bool FADE3D::Facet3::operator!= (
            const Facet3 & other ) const
```

Check if two Facet3 objects refer to a different triangle

3.4.3.6 operator==()

```
bool FADE3D::Facet3::operator== (
            const Facet3 & other ) const
```

Check if two Facet3 objects refer to the same triangle

Returns

true if the present Facet3 and `other` refer to the same facet and false otherwise

Note

Inner facets of a triangulation are shared by two tetrahedra, thus the same facet can be expressed with two different Tet3 objects and opposite indices.

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

- /home/geom/repo/dev/geomDev/dt3/dt3Library/Facet3.h

3.5 FADE3D::Fade_3D Class Reference

3D Delaunay triangulation - the main class

```
#include <Fade_3D.h>
```

Public Member Functions

- void getTetsAroundVertex (Point3 *pVtx, std::vector< Tet3 *> &vTetOut)
  Get all tetrahedra around a vertex.
- bool checkValidity (const std::string &msg, bool bCheckSphereInc)
  Check validity of the tetrahedral mesh.
- void show (const std::string &filename, Point3 *pVtx=NULL)
  Draw a 3D scene.
- Point3 * insert (Point3 &p)
  Insert a single 3D point.
- void insert (std::vector< Point3 > &vInputPoints)
  Insert a vector of 3D points.
- void insert (std::vector< Point3 > &vInputPoints, std::vector< Point3 *> &vHandlesOut)
  Insert 3D points from vInputPoints and store pointers in vHandles.
- Tet3 * locate (const Point3 &p)
  Locate a tetrahedron which contains p The Fade_3D class can be used as a data structure for point location. This method returns a pointer to a tetrahedron which contains p.
- void getTetrahedra (std::vector< Tet3 *> &vTetrahedra) const
  Get all Tet3 (tetrahedra)
- void getVertices (std::vector< Point3 *> &vVertices) const
  Get all vertices.
- bool is3D () const
  Check if the triangulation is 3D.

Static Public Member Functions

- static void printLicense ()

  Print your license type.

3.5.1 Detailed Description

Fade_3D represents a 3D Delaunay triangulation (tetrahedralization)

3.5.2 Member Function Documentation

3.5.2.1 checkValidity()

```
bool FADE3D::Fade_3D::checkValidity (
            const std::string & msg,
            bool bCheckSphereInc )
```

This is a debug method, primary ment for internal use to check if the internal data strucutre is valid. It may be time-consuming. Don't use this method unless you assume that something is wrong.

Parameters

msg	is a debug string that will be shown when the check fails
bCheckSphereInc	specifies if the empty sphere property shall be checked also

3.5.2.2 getTetrahedra()

```
void FADE3D::Fade_3D::getTetrahedra (
            std::vector< Tet3 *> & vTetrahedra ) const
```

Parameters

out	*vTetrahedra*	is used to return Tet3 pointers.

3.5.2.3 getTetsAroundVertex()

```
void FADE3D::Fade_3D::getTetsAroundVertex (
            Point3 * pVtx,
            std::vector< Tet3 *> & vTetOut )
```

Parameters

in	*pVtx*	is the query vertex
out	*vTetOut*	is used to return the incident tetrahedra

3.5.2.4 getVertices()

```
void FADE3D::Fade_3D::getVertices (
            std::vector< Point3 *> & vVertices ) const
```

Parameters

out	*vVertices*	is used to return Point3 pointers. The order does not necessrily coincide with the insertion order.

Note

When duplicate points are inserted then only one copy is made and consequently only one vertex pointer is returned for them. Thus the number of points returned by the present method can be smaller than the number of inserted points.

3.5.2.5 insert() [1/3]

```
Point3* FADE3D::Fade_3D::insert (
            Point3 & p )
```

Parameters

p	is the point to be inserted.

The triangulation keeps a copy of *p* and returns a pointer to this copy. If duplicate points are inserted the returned pointer is always the same (the one of the very first insertion).

Returns

a pointer to the point in the triangulation

Note

This method is fast but it is even faster to pass all points at once if possible. See void insert(const std↩ ::vector<Point3>& vInputPoints)

3.5.2.6 insert() [2/3]

```
void FADE3D::Fade_3D::insert (
            std::vector< Point3 > & vInputPoints )
```

Parameters

vInputPoints	contains the points to be inserted.

3.5.2.7 insert() [3/3]

```
void FADE3D::Fade_3D::insert (
            std::vector< Point3 > & vInputPoints,
            std::vector< Point3 *> & vHandlesOut )
```

Parameters

in	*vInputPoints*	contains the points to be inserted.
out	*vHandlesOut*	(empty) is used to return Point3 pointers

Internally, the triangulation keeps copies of the inserted points which are returned in *vHandles* (in the same order). If duplicate points are contained in vInputPoints then only one copy will be made and a pointer to this unique copy will be stored in vHandles for every occurance.

3.5.2.8 is3D()

```
bool FADE3D::Fade_3D::is3D ( ) const
```

Check if the triangulation containts tetrahedra. This is the case if at least 4 non-coplanar vertices exist.

3.5.2.9 locate()

```
Tet3* FADE3D::Fade_3D::locate (
            const Point3 & p )
```

Parameters

p	is the query point

Returns

a pointer to a Tet3 object (or NULL if is3D()==false or if *p* is outside the triangulation)

3.5.2.10 show()

```
void FADE3D::Fade_3D::show (
            const std::string & filename,
            Point3 * pVtx = NULL )
```

This method draws all tetrahedra. The output is a *.list file for Geomview

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

- /home/geom/repo/dev/geomDev/dt3/dt3Library/Fade_3D.h

3.6 FADE3D::Point3 Class Reference

Vertex.

`#include <Point3.h>`

Public Member Functions

- Point3 ()
  Constructor.
- Point3 (const double x, const double y, const double z)
- Point3 (const Point3 &p_)
- double x () const
  Get the x coordinate.
- double y () const
  Get the y coordinate.
- double z () const
  Get the u coordinate.
- void xyz (double &x_, double &y_, double &z_) const
  Access all coordinates at once.
- Tet3 * getOneTet () const
  Get one incident tetrahedron.
- void **exchange** (double x, double y, double z)
- void **debug** ()
- int **getCustomIndex** ()
- bool operator< (const Point3 &p) const
  Less than operator.
- bool operator== (const Point3 &p) const
  Equality operator.
- Vector3 **operator-** (const Point3 &other) const
- Point3 **operator+** (const Vector3 &vec) const

Protected Attributes

- double coordX
- double **coordY**
- double **coordZ**
- Tet3 * **pAssociatedTet**
- int **customIndex**

Friends

- class **Tet3**
- class **HC3**
- struct **Validator**
- std::ostream & **operator**<< (std::ostream &stream, const Point3 &pnt)
- std::istream & **operator**>> (std::istream &stream, Point3 &pnt)

3.6.1 Constructor & Destructor Documentation

3.6.1.1 Point3() [1/3]

```
FADE3D::Point3::Point3 ( )
```

Coordinates are initialized to -DBL_MAX, the custom index is initialized to -1, the associated incident tetrahedron pointer is initialized to NULL.

3.6.1.2 Point3() [2/3]

```
FADE3D::Point3::Point3 (
            const double x,
            const double y,
            const double z )
```

This constructor initializes the custom index to -1 and the associated incident tetrahedron to NULL.

3.6.1.3 Point3() [3/3]

```
FADE3D::Point3::Point3 (
            const Point3 & p_ )
```

The copy constructor copies the coordinates and the custom index but not the associated incident tetrahedron

3.6.2 Member Function Documentation

3.6.2.1 getOneTet()

```
Tet3* FADE3D::Point3::getOneTet ( ) const
```

Returns

an incident tetrahedron if one exists
NULL otherwise

3.6.2.2 operator<()

```
bool FADE3D::Point3::operator< (
            const Point3 & p ) const
```

Compares the coordinates of the points

3.6.2.3 operator==()

```
bool FADE3D::Point3::operator== (
            const Point3 & p ) const
```

Compares the coordinates of the points

3.6.3 Member Data Documentation

3.6.3.1 coordX

```
double FADE3D::Point3::coordX  [protected]
```

Deprecated, will be removedDeprecated, will be removed

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

- /home/geom/repo/dev/geomDev/dt3/dt3Library/Point3.h

3.7 FADE3D::Segment3 Class Reference

Line segment.

```
#include <Segment3.h>
```

Public Member Functions

- **Segment3** (const Point3 &src, const Point3 &trg)
- Point3 getSrc () const

  Get the source point.
- Point3 getTrg () const

  Get the target point.

Friends

- std::ostream & **operator**<< (std::ostream &stream, Segment3 seg)

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

- /home/geom/repo/dev/geomDev/dt3/dt3Library/Segment3.h

3.8 FADE3D::Tet3 Class Reference

Tetrahedron.

```
#include <Tet3.h>
```

Public Member Functions

- Point3 getCircumcenter ()

  Get Circumcenter.
- void getCorners (Point3 *&p0, Point3 *&p1, Point3 *&p2, Point3 *&p3) const

  Get Corners.
- Point3 * getCorner (const int ith) const

  Get Corner.
- bool hasVertex (const Point3 *p) const

  Has Vertex.
- bool hasVertex (const Point3 &p) const

  Has Vertex.
- int getIntraTetIndex (const Point3 *p) const

  Get IntraTetIndex.
- int getIntraTetIndex (const Tet3 *pNeigTet) const

  Get IntraTetIndex.
- Tet3 * getOppTet (const int ith) const

  Get Opposite Tetrahedron.
- Tet3 * getOppTet (const Point3 *pOppVtx) const

  Get Opposite Tetrahedron.
- Point3 * getOppVtxInOppTet (const int ith, bool bNullAllowed) const

  Get Opposite Tetrahedron.
- Edge3 getEdge (const int opp3, const int opp2)

  Get Edge.
- Facet3 getFacet (const int opp3)

  Get Facet.

Static Public Member Functions

- static std::pair< int, int > getEdgeIndices (int opp3, int opp2)

  Get Edge Indices.

Friends

- std::ostream & **operator**<< (std::ostream &stream, const Tet3 &pC)

3.8.1 Detailed Description

The 4 corners of a Tetrahedron (Tet3) are addressed by the Intra-Tetrahedron-Indices 0,1,2 and 3. For short we refer to them as the IntraTetIndices. A Tet3 is oriented and thus its IntraTetIndices appear in a specific order. Here is a memory hook (see the image): When a triangle with counterclockwise indices 0,1,2 lies on the floor then the remaining vertex 3 of the tetrahedron lies above this triangle.

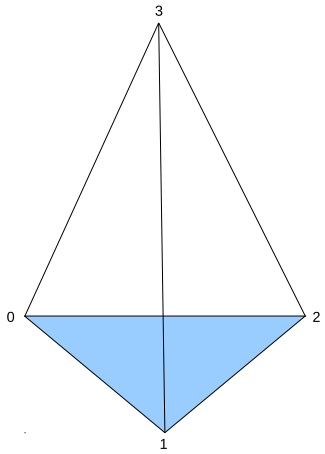

Figure 3 IntraTetIndices: A counterclockwise triangle 0,1,2 on the floor and corner 3 above

3.8.2 Member Function Documentation

3.8.2.1 getCircumcenter()

```
Point3 FADE3D::Tet3::getCircumcenter ( )
```

Returns

the center of a sphere that passes through the 4 corners of the present Tet3.

3.8.2.2 getCorner()

```
Point3* FADE3D::Tet3::getCorner (
            const int ith ) const
```

Returns

the corner with the `ith` IntraTetIndex

3.8.2.3 getCorners()

```
void FADE3D::Tet3::getCorners (
            Point3 *& p0,
            Point3 *& p1,
            Point3 *& p2,
            Point3 *& p3 ) const
```

Used to access all corners of the present Tet3 at once

3.8.2.4 getEdge()

```
Edge3 FADE3D::Tet3::getEdge (
            const int opp3,
            const int opp2 )
```

return an Edge3 addressed by the two IntraTetIndices `opp3` (selects the opposite triangle) and `opp2` (selects the edge of this triangle)

3.8.2.5 getEdgeIndices()

```
static std::pair<int,int> FADE3D::Tet3::getEdgeIndices (
            int opp3,
            int opp2 )  [static]
```

return a pair of IntraTetIndices that specify an edge of the tetrahedron.

The edge is selected through two IntraTetIndices: At first `opp3` selects the opposite triangle. Then `opp2` selects the opposite edge in this triangle. The triangle has a counterclockwise orientation (see the image) and thus the two returned indices are exactly defined.

3.8.2.6 getFacet()

```
Facet3 FADE3D::Tet3::getFacet (
            const int opp3 )
```

return the Facet3 opposite from the vertex addressed by `opp3`.

3.8.2.7 getIntraTetIndex() [1/2]

```
int FADE3D::Tet3::getIntraTetIndex (
            const Point3 * p ) const
```

Returns

the IntraTetIndex of the vertex `p`.

3.8.2.8 getIntraTetIndex() [2/2]

```
int FADE3D::Tet3::getIntraTetIndex (
            const Tet3 * pNeigTet ) const
```

The present Tet3 and `pNeigTet` must be neighbors. The present method returns the IntraTetIndex of the corner of the present Tet3 opposite to the shared triangle.

3.8.2.9 getOppTet() [1/2]

```
Tet3* FADE3D::Tet3::getOppTet (
            const int ith ) const
```

return the opposite tetrahedron of the `ith` corner, i.e., the tetrahedron adjacent at the facet opposite to the `ith` vertex.

3.8.2.10 getOppTet() [2/2]

```
Tet3* FADE3D::Tet3::getOppTet (
            const Point3 * pOppVtx ) const
```

return the opposite tetrahedron of `pOppVtx`, i.e., the tetrahedron adjacent at the facet opposite to `pOppVtx`.

3.8.2.11 getOppVtxInOppTet()

```
Point3* FADE3D::Tet3::getOppVtxInOppTet (
            const int ith,
            bool bNullAllowed ) const
```

return the opposite vertex in the `ith` opposite tetrahedron.

3.8.2.12 hasVertex() [1/2]

```
bool FADE3D::Tet3::hasVertex (
            const Point3 * p ) const
```

Returns

true if any of the four vertex pointers corresponds to `p`

3.8.2.13 hasVertex() [2/2]

```
bool FADE3D::Tet3::hasVertex (
            const Point3 & p ) const
```

Returns

true if the coordinates of any of the four corner vertices correspond to the coordinates of `p`.

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

- /home/geom/repo/dev/geomDev/dt3/dt3Library/Tet3.h

3.9 FADE3D::Vector3 Class Reference

3D Vector

`#include <Vector3.h>`

Public Member Functions

- Vector3 (const double x_, const double y_, const double z_)
  Constructor.
- Vector3 ()
  Constructor The vector is initialized to (0,0,0)
- double x () const
  Get the x-value.
- double y () const
  Get the y-value.
- double z () const
  Get the z-value.
- void set (const double x_, const double y_, const double z_)
  Set the values.
- double length () const
  Get the length of the vector.
- double operator∗ (const Vector3 &other) const
  Scalar product.
- Vector3 operator∗ (double val) const
  Multiply by a scalar value.
- Vector3 operator/ (double val) const
  Divide by a scalar value.

Protected Attributes

- double **valX**
- double **valY**
- double **valZ**

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

- /home/geom/repo/dev/geomDev/dt3/dt3Library/Vector3.h

3.10 FADE3D::Visualizer3 Class Reference

Geomview visualizations.

`#include <Visualizer3.h>`

Public Member Functions

- **Visualizer3** (const std::string &filename)
- void **closeFile** ()
- void **openFile** (std::string filename)
- void **writeBall** (Ball3 ∗ball)
- void **writeBall** (const Point3 &center, double weight, bool bTransparent=false)
- void **writeBalls** (const std::vector< Ball3 > &vBalls, bool bTransparent=false)
- void **writeSegment** (const Point3 &src, const Point3 &trg, const std::string &c)
- void **writePolygon** (std::vector< Point3 > &vPoints, const std::string &c)
- void **writeTetrahedron** (Tet3 ∗pTet, const std::string &c)
- void **writeTriangle** (Triangle3 ∗pT, const std::string &c)
- void **writeTriangles** (std::vector< Triangle3 ∗> vTriangles, const std::string &c)
- void **writeBbox** (const Bbox3 &bbx, const std::string &c)
- void **writePoint** (const Point3 &p, unsigned lineWidth, const std::string &color)
- void **writePoints** (std::vector< Point3 > &vPoints, unsigned lineWidth, const std::string &color)
- void **writePoints** (std::vector< Point3 ∗> &vPoints, unsigned lineWidth, const std::string &color)

Static Public Member Functions

- static std::string **getNextColor** ()
- static std::string **getColor** (unsigned ith)
- static std::string **getColorName** (unsigned ith)

Static Public Attributes

- static std::string **CLIGHTBLUE**
- static std::string **CDARKBLUE**
- static std::string **CYELLOW**
- static std::string **CPINK**
- static std::string **CBLACK**
- static std::string **CLIGHTBROWN**
- static std::string **CDARKBROWN**
- static std::string **CORANGE**
- static std::string **CPURPLE**
- static std::string **CGRAY**
- static std::string **CLIGHTGRAY**
- static std::string **CRED**
- static std::string **CGREEN**
- static std::string **CWHITE**
- static std::string **CRIMSON**
- static std::string **CDARKORANGE**
- static std::string **CGOLDENROD**
- static std::string **COLIVE**
- static std::string **CLAWNGREEN**
- static std::string **CGREENYELLOW**
- static std::string **CPALEGREEN**
- static std::string **CMEDSPRINGGREEN**
- static std::string **CLIGHTSEAGREAN**
- static std::string **CCYAN**
- static std::string **CSTEELBLUE**
- static std::string **MIDNIGHTBLUE**
- static std::string **CWHEAT**

Protected Member Functions

- void **startList** (const size_t numPoints, const size_t numTriangles, bool bWithEdges=true)
- void **endList** ()

Protected Attributes

- std::ofstream **outFile**

Static Protected Attributes

- static const std::string **colorNames** [27]
- static const std::string **colorArray** [27]
- static int **nextColorIdx**

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

- /home/geom/repo/dev/geomDev/dt3/dt3Library/Visualizer3.h

Index