

THUNDER

Generated by Doxygen 1.8.14

Contents

1	File Index	1
1.1	File List	1
2	File Documentation	3
2.1	include/Geometry/Euler.h File Reference	3
2.1.1	Detailed Description	4
2.1.2	Function Documentation	4
2.1.2.1	alignZ()	4
2.1.2.2	angle() [1/3]	4
2.1.2.3	angle() [2/3]	5
2.1.2.4	angle() [3/3]	5
2.1.2.5	direction()	5
2.1.2.6	quaternion() [1/2]	6
2.1.2.7	quaternion() [2/2]	6
2.1.2.8	quaternion_conj()	7
2.1.2.9	quaternion_mul()	7
2.1.2.10	randRotate2D()	7
2.1.2.11	randRotate3D()	7
2.1.2.12	reflect3D()	8
2.1.2.13	rotate2D() [1/2]	8
2.1.2.14	rotate2D() [2/2]	8
2.1.2.15	rotate3D() [1/4]	8
2.1.2.16	rotate3D() [2/4]	9
2.1.2.17	rotate3D() [3/4]	9
2.1.2.18	rotate3D() [4/4]	9
2.1.2.19	rotate3DX()	10
2.1.2.20	rotate3DY()	10
2.1.2.21	rotate3DZ()	10
2.1.2.22	scale3D()	12
2.1.2.23	translate3D()	12
	Index	13

Chapter 1

File Index

1.1 File List

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

include/Geometry/ Euler.h	
Some description about Euler.h	3

Chapter 2

File Documentation

2.1 include/Geometry/Euler.h File Reference

some description about [Euler.h](#)

```
#include <cmath>
#include <gsl/gsl_math.h>
#include "Macro.h"
#include "Typedef.h"
#include "Precision.h"
#include "Random.h"
#include "Functions.h"
```

Functions

- void [quaternion_mul](#) (dvec4 &dst, const dvec4 &a, const dvec4 &b)
Calculate the product of two quaternions.
- dvec4 [quaternion_conj](#) (const dvec4 &quat)
Calculate the conjugate quaternion of a quaternion.
- void [angle](#) (double &phi, double &theta, const dvec3 &src)
Calculate ϕ and θ given a certain direction \mathbf{v} .
- void [angle](#) (double &phi, double &theta, double &psi, const dmat33 &src)
Calculate ϕ , θ and ψ of the rotation represented by the rotation matrix \mathbf{R} .
- void [angle](#) (double &phi, double &theta, double &psi, const dvec4 &src)
Calculate ϕ , θ and ψ of the rotation represented by the quaternion \mathbf{q} .
- void [quaternion](#) (dvec4 &dst, const double phi, const double theta, const double psi)
Calculate the quaternion \mathbf{q} for representation the rotation, given 3 Euler angles ϕ , θ and ψ .
- void [quaternion](#) (dvec4 &dst, const double phi, const dvec3 &axis)
- void **quaternion** (dvec4 &dst, const dmat33 &src)
- void [rotate2D](#) (dmat22 &dst, const dvec2 &vec)
- void [rotate2D](#) (dmat22 &dst, const double phi)
- void [direction](#) (dvec3 &dst, const double phi, const double theta)
- void [rotate3D](#) (dmat33 &dst, const double phi, const double theta, const double psi)
- void [rotate3D](#) (dmat33 &dst, const dvec4 &src)
- void [rotate3DX](#) (dmat33 &dst, const double phi)
- void [rotate3DY](#) (dmat33 &dst, const double phi)

- void [rotate3DZ](#) (dmat33 &dst, const double phi)
- void [alignZ](#) (dmat33 &dst, const dvec3 &vec)
- void [rotate3D](#) (dmat33 &dst, const double phi, const char axis)
- void [rotate3D](#) (dmat33 &dst, const double phi, const dvec3 &axis)
- void [reflect3D](#) (dmat33 &dst, const dvec3 &plane)
- void [translate3D](#) (mat44 &dst, const dvec3 &vec)
- void [scale3D](#) (dmat33 &dst, const dvec3 &vec)
- void **swingTwist** (dvec4 &swing, dvec4 &twist, const dvec4 &src, const dvec3 &vec)
- void **randDirection** (dvec2 &dir)
- void [randRotate2D](#) (dmat22 &rot)
- void **randQuaternion** (dvec4 &quat)
- void [randRotate3D](#) (dmat33 &rot)

2.1.1 Detailed Description

some description about [Euler.h](#)

Details about [Euler.h](#)

2.1.2 Function Documentation

2.1.2.1 [alignZ\(\)](#)

```
void alignZ (
    dmat33 & dst,
    const dvec3 & vec )
```

This function calculates the rotation matrix for aligning a direction vector to Z-axis.

Parameters

<i>dst</i>	the rotation matrix
<i>vec</i>	the direction vector

2.1.2.2 [angle\(\)](#) [1/3]

```
void angle (
    double & phi,
    double & theta,
    const dvec3 & src )
```

Calculate ϕ and θ given a certain direction \mathbf{v} .

Parameters

out	<i>phi</i>	ϕ
out	<i>theta</i>	θ
in	<i>src</i>	v

2.1.2.3 angle() [2/3]

```
void angle (
    double & phi,
    double & theta,
    double & psi,
    const dmat33 & src )
```

Calculate ϕ , θ and ψ of the rotation represented by the rotation matrix **R**.

Parameters

out	<i>phi</i>	ϕ
out	<i>theta</i>	θ
out	<i>psi</i>	ψ
in	<i>src</i>	R

2.1.2.4 angle() [3/3]

```
void angle (
    double & phi,
    double & theta,
    double & psi,
    const dvec4 & src )
```

Calculate ϕ , θ and ψ of the rotation represented by the quaternion **q**.

Parameters

out	<i>phi</i>	ϕ
out	<i>theta</i>	θ
out	<i>psi</i>	ψ
in	<i>src</i>	q

2.1.2.5 direction()

```
void direction (
```

```
dvec3 & dst,
const double phi,
const double theta )
```

This function calculates the direction vector given phi and theta. The 2-norm of this direction vector is 1.

Parameters

<i>dst</i>	the direction vector
<i>phi</i>	phi
<i>theta</i>	theta

2.1.2.6 quaternion() [1/2]

```
void quaternion (
    dvec4 & dst,
    const double phi,
    const double theta,
    const double psi )
```

Calculate the quaternion **q** for representation the rotation, given 3 Euler angles ϕ , θ and ψ .

Parameters

<i>dst</i>	the quaternion to be calculated
<i>phi</i>	phi
<i>theta</i>	theta
<i>psi</i>	psi

2.1.2.7 quaternion() [2/2]

```
void quaternion (
    dvec4 & dst,
    const double phi,
    const dvec3 & axis )
```

This function calculates the quaternion given rotation angle and rotation axis.

Parameters

<i>dst</i>	the quaternion to be calculated
<i>phi</i>	the rotation angle
<i>axis</i>	the rotation axis (unit vector)

2.1.2.8 quaternion_conj()

```
dvec4 quaternion_conj (
    const dvec4 & quat )
```

Calculate the conjugate quaternion of a quaternion.

Returns

the conjugate quaternion

Parameters

in	<i>quat</i>	a quaternion
----	-------------	--------------

2.1.2.9 quaternion_mul()

```
void quaternion_mul (
    dvec4 & dst,
    const dvec4 & a,
    const dvec4 & b )
```

Calculate the product of two quaternions.

Parameters

out	<i>dst</i>	product, a quaternion
in	<i>a</i>	left multiplier, quaternion
in	<i>b</i>	right multiplier, quaternion

2.1.2.10 randRotate2D()

```
void randRotate2D (
    dmat22 & rot )
```

This function generates a random unit quaternion.

2.1.2.11 randRotate3D()

```
void randRotate3D (
    dmat33 & rot )
```

This function generates a random 3D rotation matrix.

2.1.2.12 reflect3D()

```
void reflect3D (
    dmat33 & dst,
    const dvec3 & plane )
```

This function calculates the transformation matrix of reflection against a certain plane given by its normal vector.

Parameters

<i>dst</i>	the rotation matrix
<i>plane</i>	the normal vector the reflection plane

2.1.2.13 rotate2D() [1/2]

```
void rotate2D (
    dmat22 & dst,
    const dvec2 & vec )
```

This function calculates the rotation matrix given the a unit vector.

Parameters

<i>dst</i>	the rotation matrix
<i>vec</i>	the unit vector

2.1.2.14 rotate2D() [2/2]

```
void rotate2D (
    dmat22 & dst,
    const double phi )
```

This function calculates the rotation matrix given phi in 2D.

Parameters

<i>dst</i>	the rotation matrix
<i>phi</i>	phi

2.1.2.15 rotate3D() [1/4]

```
void rotate3D (
    dmat33 & dst,
```

```
const double phi,
const double theta,
const double psi )
```

This function calculates the rotation matrix given phi, theta and psi.

Parameters

<i>dst</i>	the rotation matrix
<i>phi</i>	phi
<i>theta</i>	theta
<i>psi</i>	psi

2.1.2.16 rotate3D() [2/4]

```
void rotate3D (
    dmat33 & dst,
    const dvec4 & src )
```

This function calculates the rotation matrix given a quaternion.

Parameters

<i>dst</i>	the rotation matrix
<i>src</i>	the quaternion

2.1.2.17 rotate3D() [3/4]

```
void rotate3D (
    dmat33 & dst,
    const double phi,
    const char axis )
```

This function calculates the rotation matrix of rotation along a certain axis (X, Y or Z) of phi.

Parameters

<i>dst</i>	the rotation matrix
<i>axis</i>	a character indicating which axis the rotation is along

2.1.2.18 rotate3D() [4/4]

```
void rotate3D (
    dmat33 & dst,
```

```
const double phi,
const dvec3 & axis )
```

This function calculates the rotation matrix of rotation along a certain axis given by a direction vector of *phi*.

Parameters

<i>dst</i>	the rotation matrix
<i>phi</i>	<i>phi</i>
<i>axis</i>	the direction vector indicating the axis

2.1.2.19 rotate3DX()

```
void rotate3DX (
    dmat33 & dst,
    const double phi )
```

This function calculates the rotation matrix of rotation along X-axis of *phi*.

Parameters

<i>dst</i>	the rotation matrix
<i>phi</i>	<i>phi</i>

2.1.2.20 rotate3DY()

```
void rotate3DY (
    dmat33 & dst,
    const double phi )
```

This function calculates the rotation matrix of rotation along Y-axis of *phi*.

Parameters

<i>dst</i>	the rotation matrix
<i>phi</i>	<i>phi</i>

2.1.2.21 rotate3DZ()

```
void rotate3DZ (
    dmat33 & dst,
    const double phi )
```

This function calculates the rotation matrix of rotation along Z-axis of phi.

Parameters

<i>dst</i>	the rotation matrix
<i>phi</i>	phi

2.1.2.22 scale3D()

```
void scale3D (
    dmat33 & dst,
    const dvec3 & vec )
```

This function calculates the transformation matrix of scaling.

Parameters

<i>dst</i>	the transformation matrix
<i>vec</i>	a 3-vector of which vec[0] indicates the scale factor along X axis, vec[1] indicates the scale factor along Y axis and vec[2] indicates the scale factor along Z axis

2.1.2.23 translate3D()

```
void translate3D (
    mat44 & dst,
    const dvec3 & vec )
```

This function calculates the singular matrix of translation of a certain vector.

Parameters

<i>dst</i>	the singular matrix
<i>vec</i>	the translation vector

Index

alignZ
 Euler.h, [4](#)
angle
 Euler.h, [4](#), [5](#)

direction
 Euler.h, [5](#)

Euler.h
 alignZ, [4](#)
 angle, [4](#), [5](#)
 direction, [5](#)
 quaternion, [6](#)
 quaternion_conj, [6](#)
 quaternion_mul, [7](#)
 randRotate2D, [7](#)
 randRotate3D, [7](#)
 reflect3D, [7](#)
 rotate2D, [8](#)
 rotate3DX, [10](#)
 rotate3DY, [10](#)
 rotate3DZ, [10](#)
 rotate3D, [8](#), [9](#)
 scale3D, [12](#)
 translate3D, [12](#)

include/Geometry/Euler.h, [3](#)

quaternion
 Euler.h, [6](#)
quaternion_conj
 Euler.h, [6](#)
quaternion_mul
 Euler.h, [7](#)

randRotate2D
 Euler.h, [7](#)
randRotate3D
 Euler.h, [7](#)
reflect3D
 Euler.h, [7](#)
rotate2D
 Euler.h, [8](#)
rotate3DX
 Euler.h, [10](#)
rotate3DY
 Euler.h, [10](#)
rotate3DZ
 Euler.h, [10](#)
rotate3D
 Euler.h, [8](#), [9](#)

scale3D
 Euler.h, [12](#)

translate3D
 Euler.h, [12](#)