# THUNDER

Generated by Doxygen 1.8.14

# **Contents**

Index

1	File	Index			1
	1.1	File Lis	t		1
2	File	Docume	entation		3
	2.1	include	/Geometry	y/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	<b>angle()</b> [1/3]	4
			2.1.2.3	<b>angle()</b> [2/3]	5
			2.1.2.4	<b>angle()</b> [3/3]	5
			2.1.2.5	direction()	5
			2.1.2.6	<b>quaternion()</b> [1/2]	6
			2.1.2.7	<b>quaternion()</b> [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

13

# **Chapter 1**

# File Index

4	4	 	i		- 4
٦.	п.	-11	Δ	ΙI	CT.

Here is a list of all documented files with brief description
---

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

2 File Index

# **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  $\phi$  and  $\theta$  given a certain direction  $\mathbf{v}$ .

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

Calculate  $\phi$ ,  $\theta$  and  $\psi$  of the rotation represented by the rotation matrix  ${\bf R}$ .

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

Calculate  $\phi$ ,  $\theta$  and  $\psi$  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  ${\bf q}$  for representing the rotation, given 3 Euler angles  $\phi$ ,  $\theta$  and  $\psi$ .

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

Calculate the quaternion  $\mathbf{q}$  for representing the rotation, given the rotation axis  $\mathbf{r}$  and the rotation angle around this axis  $\phi$ 

- 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 ( \label{eq:dmat33 & dst,}  const dvec3 & vec )
```

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

#### **Parameters**

dst	the rotation matrix
vec	the direction vector

```
2.1.2.2 angle() [1/3]
```

Calculate  $\phi$  and  $\theta$  given a certain direction  $\mathbf{v}$ .

#### **Parameters**

out	phi	$\phi$
out	theta	$\theta$
in	src	$\mathbf{v}$

# **2.1.2.3 angle()** [2/3]

Calculate  $\phi$ ,  $\theta$  and  $\psi$  of the rotation represented by the rotation matrix  ${\bf R}.$ 

#### **Parameters**

out	phi	$\phi$
out	theta	$\theta$
out	psi	$\psi$
in	src	R

# **2.1.2.4** angle() [3/3]

Calculate  $\phi,\,\theta$  and  $\psi$  of the rotation represented by the quaternion  ${\bf q}.$ 

# **Parameters**

out	phi	$\phi$
out	theta	$\theta$
out	psi	$\psi$
in	src	$\mathbf{q}$

#### 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**

dst	the direction vector
phi	phi
theta	theta

# **2.1.2.6 quaternion()** [1/2]

Calculate the quaternion  ${\bf q}$  for representing the rotation, given 3 Euler angles  $\phi$ ,  $\theta$  and  $\psi$ .

#### **Parameters**

out	dst	q
in	phi	$\phi$
in	theta	$\theta$
in	psi	$\psi$

# **2.1.2.7 quaternion()** [2/2]

Calculate the quaternion  $\bf q$  for representing the rotation, given the rotation axis  $\bf r$  and the rotation angle around this axis  $\phi$ .

#### **Parameters**

dst	the quaternion to be calculated
phi	the rotation angle
axis	the rotation axis (unit vector)

#### 2.1.2.8 quaternion\_conj()

Calculate the conjugate quaternion of a quaternion.

#### Returns

the conjugate quaternion

#### **Parameters**

in	quat	a quaternion
----	------	--------------

#### 2.1.2.9 quaternion\_mul()

Calculate the product of two quaternions.

#### **Parameters**

out	dst	product, a quaternion
in	а	left multiplier, quaternion
in	b	right multiplier, quaternion

#### 2.1.2.10 randRotate2D()

This function generates a random unit quaternion.

# 2.1.2.11 randRotate3D()

```
void randRotate3D ( {\tt dmat33~\&~rot~)}
```

This function generates a random 3D rotation matrix.

#### 2.1.2.12 reflect3D()

```
void reflect3D ( \mbox{dmat33 \& } dst, \mbox{const dvec3 \& } plane \mbox{)}
```

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

#### **Parameters**

dst	the rotation matrix
plane	the normal vector the reflection plane

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

const dvec2 & vec )

#### **Parameters**

dst	the rotation matrix
vec	the unit vector

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

#### **Parameters**

dst	the rotation matrix
phi	phi

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

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

#### **Parameters**

dst	the rotation matrix
phi	phi
theta	theta
psi	psi

# 2.1.2.16 rotate3D() [2/4]

```
void rotate3D ( \label{eq:dmat33 & dst,} $$ const dvec4 & src )
```

This function calculates the rotation matrix given a quaternion.

#### **Parameters**

dst	the rotation matrix
src	the quaternion

# 2.1.2.17 rotate3D() [3/4]

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

#### **Parameters**

dst	the rotation matrix
axis	a character indicating which axis the rotation is along

#### **2.1.2.18** rotate3D() [4/4]

```
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**

dst	the rotation matrix
phi	phi
axis	the direction vector indicating the axis

#### 2.1.2.19 rotate3DX()

```
void rotate3DX ( \label{eq:dmat33 \& dst,}  const double phi )
```

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

#### **Parameters**

dst	the rotation matrix
phi	phi

#### 2.1.2.20 rotate3DY()

```
void rotate3DY ( \label{eq:dmat33 \& dst,}  const double phi )
```

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

# **Parameters**

dst	the rotation matrix
phi	phi

# 2.1.2.21 rotate3DZ()

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

# **Parameters**

dst	the rotation matrix
phi	phi

# 2.1.2.22 scale3D()

```
void scale3D ( \label{eq:dmat33 & dst,}  const dvec3 & vec )
```

This function calculates the transformation matrix of scaling.

#### **Parameters**

dst	the transformation matrix
vec	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()

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

#### **Parameters**

dst	the singular matrix
vec	the translation vector

# Index

Euler.h, 8, 9

alignZ Euler.h, 4 angle	scale3D Euler.h, translate3D
Euler.h, 4, 5 direction Euler.h, 5	Euler.h,
Euler.h alignZ, 4 angle, 4, 5 direction, 5 quaternion, 6 quaternion_conj, 7 quaternion_mul, 7 randRotate2D, 7 randRotate3D, 7 reflect3D, 7 rotate2D, 8 rotate3DX, 10 rotate3DY, 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, 7 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	

12

12