Orbit Test

Generated by Doxygen 1.8.8

Sun Nov 27 2016 14:27:04

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 Angle Class Reference

An angle in radians.

```
#include <Angle.h>
```

Public Member Functions

- Angle (double newr)

    Constructor with radians set.
- Angle ()

    Default constructor, angle set to 0.
- double radians () const

    Get angle as radians.
- double degrees () const

    Get angle as degrees.
- void radians (double newr)

    Set angle as radians.
- void degrees (double newd)

    Set angle as degrees.
- double operator() () const

    Get angle as radians.
- void operator() (double newr)

    Set angle as radians.
- Angle operator+ (const Angle b) const

    Allows addition of angles.
- Angle operator- (const Angle b) const

    Allows subtraction of angles.
- Angle operator+ (const double b) const

    Allows addition of angles.
- Angle operator- (const double b) const

    Allows subtraction of angles.
- operator double () const

    Allows casting to double.

Protected Attributes

- double rad

  Internal storage of angle in radians.

3.1.1 Detailed Description

An angle in radians.

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

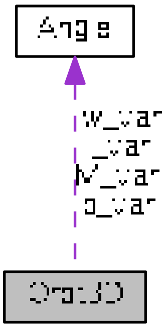
- include/Angle.h
- src/Angle.cpp

3.2 Orbit3D Class Reference

Represents an orbit around an unspecified central body.

`#include <Orbit3D.h>`

Collaboration diagram for Orbit3D:



Public Member Functions

- double e () const

  Eccentricity.
- void e (double newe)

  Set Eccentricity.
- double eccentricity () const

  Alias of e()
- void eccentricity (double newe)

  Alias of e(double)
- double a () const

  Semi-major axis.
- void a (double newa)

  Set Semi-major axis.

- double semiMajorAxis () const

  Alias of a()
- void semiMajorAxis (double newa)

  Alias of a(double)
- Angle i () const

  Inclination.
- void i (Angle newi)

  Set Inclination.
- Angle inclination () const

  Alias of i()
- void inclination (Angle newi)

  Alias of i(Angle)
- Angle o () const

  Longitude of Ascending Node.
- void o (Angle newo)

  Set Longitude of Ascending Node.
- Angle w () const

  Argument of periapsis.
- void w (Angle neww)

  Set Argument of periapsis.
- Angle argumentOfPeriapsis () const

  Alias of w()
- void argumentOfPeriapsis (Angle neww)

  Alias of w(Angle)
- Angle M () const

  Mean Anomaly.
- void M (Angle newM)

  Set Mean anomaly.
- Angle meanAnomaly () const

  Alias of M()
- void meanAnomaly (Angle newM)

  Alias of M(Angle)
- double PeriAndApoDistance () const

  Returns r_ap + r_per.
- double r_per () const

  Distance to Periapsis.
- double r_ap () const

  Distance to Apoapsis.
- double MajorAxis () const

  Major axis.
- double b () const

  Semi Minor Axis.
- double semiMinorAxis () const

  Alias of b()
- double MinorAxis () const

  Minor Axis.
- double T (double mew) const

  Orbital period.
- double orbitalPeriod (double mew) const

  Alias of T()
- double calculateaForSpecificT (double T, double mew) const

Calculates semi-major axis required for selected orbital period.
- double calculateSyncOrbit (double srp, double mew) const

    Calculates semi-major axis for synchronous orbit.
- Angle u () const

    Argument of Latitude.
- Angle argumentOfLatitude () const

    Alias of u()
- Angle l () const

    True Longitude.
- Angle trueLongitude () const

    Alias of l()
- Angle f () const

    True anomaly.
- void f (Angle newf)

    Set True anomaly.
- Angle trueAnomaly () const

    Alias of f()
- void trueAnomaly (Angle newf)

    Alias of f(Angle)
- Angle E () const

    Eccentric Anomaly.
- Angle eccentricAnomaly () const

    Alias of E()
- double ell () const

    Semi-Latus Rectum.
- double semiLatusRectum () const

    Alias of ell()
- double latusRectum () const

    Latus Rectum.
- double p () const

    Focal Parameter.
- double focalParameter () const

    Alias of p()
- OrbitalShape shape () const

    Shape of Orbit.
- double c () const

    Linear Eccentricity.
- double linearEccentricity () const

    Alias of c()
- double radiusTrueAnomaly () const

    Radius from True Anomaly.
- double radiusEccentricAnomaly () const

    Radius from Eccentric Anomaly.
- double epsilon (double mew) const

    Specific Orbital Energy.
- double specificOrbitalEnergy (double mew) const

    Alias of epsilon()
- double v (double mew) const

    Mean Orbital Speed.
- double meanOrbitalSpeed (double mew) const

    Alias of v(double)

- Vector3D hBar () const

  Specific Relative Angular Momentum.
- Vector3D specificRelativeAngularMomentum () const

  Alias of hBar()
- Vector3D osvr () const

  Orbital state vector position.
- Vector3D osvv () const

  Orbital state vector velocity.
- Vector3D lineOfNodes () const

  Line of nodes vector.
- double n (double mew) const

  Mean Motion.
- double meanMotion (double mew) const

  Alias of n(double)
- double meanLongitude () const

  Mean longitude.
- double longitudeOfPeriapsis () const

  Longitude of Periapsis.

Protected Attributes

- double e_var

  Eccentricity.
- double a_var

  Semi-major axis.
- Angle i_var

  Inclination.
- Angle o_var

  Longitude of Ascending Node.
- Angle w_var

  Argument of periapsis.
- Angle M_var

  Mean anomaly.

3.2.1 Detailed Description

Represents an orbit around an unspecified central body.

This class defines an orbit around a central body using Keplerian elements. The elements used to define the orbit are the eccentricity, semi-major axis, inclination, longitude of ascending node, argument of periapsis, and mean anomaly.

All other return values are calculated in terms of these elements. The Standard Gravitational Parameter (mew) is required to calculate any elements that are based on the mass of the planet.

3.2.2 Member Function Documentation

3.2.2.1 double Orbit3D::c ( ) const

Linear Eccentricity.

This is undefined for parabolic orbits and this function will throw an exception

3.2.2.2 double Orbit3D::calculateaForSpecificT (double *T,* double *mew*) const

Calculates semi-major axis required for selected orbital period.

Parameters

T	The amount of time in seconds

3.2.2.3 double Orbit3D::calculateSyncOrbit (double *srp,* double *mew*) const [inline]

Calculates semi-major axis for synchronous orbit.

Parameters

srp	The time it takes for one rotation in seconds

3.2.2.4 Angle Orbit3D::f ( ) const

True anomaly.

Undefined in circular orbits, use argumentOfLatitude()

3.2.2.5 double Orbit3D::p ( ) const

Focal Parameter.

Focal parameter is infinite for circular orbits and this function will throw an exception.

3.2.2.6 double Orbit3D::PeriAndApoDistance ( ) const

Returns r_ap + r_per.

Divides the semi-major axis by 2 giving the sum of the distance between the apoapsis and periapsis.

3.2.2.7 Angle Orbit3D::u ( ) const

Argument of Latitude.

Undefined in circular orbits with zero inclination, use trueLongitude()

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

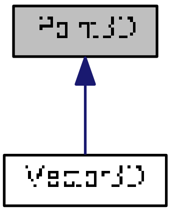
- include/Orbit3D.h
- src/Orbit3D.cpp

3.3 Point3D Class Reference

A point in 3-dimensional space.

```
#include <Point3D.h>
```

Inheritance diagram for Point3D:



Public Member Functions

- double x () const

  Get coord x.

- void x (double newx)

  Set coord x.

- double y () const

  Get coord y.

- void y (double newy)

  Set coord y.

- double z () const

  Get coord z.

- void z (double newz)

  Set coord z.

3.3.1 Detailed Description

A point in 3-dimensional space.

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

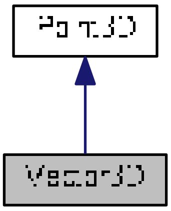
- include/Point3D.h
- src/Point3D.cpp

3.4 Vector3D Class Reference

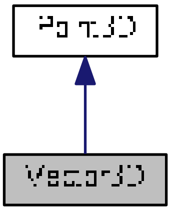
A vector in 3-dimensional space.

```
#include <Vector3D.h>
```

Inheritance diagram for Vector3D:



Collaboration diagram for Vector3D:



Public Member Functions

- Vector3D (double newx, double newy, double newz)

  Constructor with coordinates.
- Vector3D ()

  Default constructor, makes zero vector.
- Vector3D (Point3D p1, Point3D p2)

  Create a vector representing 2 sets of coordinates.
- Angle alpha () const

  Angle from x axis to vector.
- void alpha (Angle newa)

  Modify alpha angle.
- Angle beta () const

  Angle from y axis to vector.
- void beta (Angle newa)

  Modify beta angle.
- Angle gamma () const

  Angle from z axis to vector.
- void gamma (Angle newa)

  Modify gamma angle.

- Angle theta () const

  Angle from z axis to vector.
- Angle phi () const

  Angle from x axis to 2D vector.
- double magnitude () const

  Returns magnitude of vector.
- bool isZero () const

  Checks if vector has no magnitude or direction.
- bool isUnit () const

  Check if vector magnitude is 1.
- bool operator== (const Vector3D b) const

  Allows vectors to be compared.
- bool isOpposite (const Vector3D b) const

  Checks if vector is opposite.
- bool isParallel (const Vector3D b) const

  Checks if vector is parallel.
- bool isAntiparallel (const Vector3D b) const

  Checks if vector is anitparallel.
- Vector3D operator+ (const Vector3D &b) const

  Allows vectors to be added.
- Vector3D operator- (const Vector3D &b) const

  Allows vectors to be subtracted.
- Vector3D operator∗ (const double &b) const

  Allows multiply vector by scalar.
- Vector3D operator/ (const double &b) const

  Allows division vector by scalar.
- double dotProduct (const Vector3D b) const

  Generate dot product of 2 vectors.
- Vector3D crossProduct (const Vector3D b) const

  Generate cross product of 2 vectors.
- Angle findAngle (const Vector3D b) const

  Calculate angle between 2 vectors.

3.4.1 Detailed Description

A vector in 3-dimensional space.

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

- include/Vector3D.h
- src/Vector3D.cpp

Index