Documentation for NetVecCad’s

Foundations Namespace

Paul Schrum

5 Jan 2015

Classes in the Foundations namespace are low level building blocks upon which all or most of NetVecCad depends. Many other packages may use these classes. In reality they are as essential, basic, and axiomatic as the Double class. In fact, many of these classes are wrappers around Double values and what makes them “different” are the methods and constraints applied to them.

# Foundations

Class Hierarchy for Angle

|  |  |  |  |
| --- | --- | --- | --- |
| Level 0 | Angle | | |
| Level 1 Inheritance | Azimuth | Deflection | Slope |

## Class Angle

is a wrapper around a Double. The sole value is kept as radians (to optimize trig function speed). There are no constraints on the value. Value 0.0 is to the right. Positive values increase counter-clockwise. Values may be written or read as degrees, radians, or turns.

### Tests

#### Angle\_cos\_minus120\_equalsMinus0\_5

#### Angle\_sin\_minus150\_equalsMinus0\_5

#### Angle\_tan\_minus45\_equalsMinus1

#### Angle\_fromDxDy\_quadrant1\_1010\_equals45Degrees

#### Angle\_fromDxDy\_quadrant1\_1010\_equals135Degrees

#### Angle\_fromDxDy\_quadrant1\_1010\_equalsMinus135Degrees

#### Angle\_fromDxDy\_quadrant1\_1010\_equalsMinus45Degrees

#### Angle\_settingTo1\_shouldResultIn\_equals57\_2957795Degrees

#### Angle\_ComputeRemainder\_ScaledByDenominator

## Foundations.Angles

### Class Azimuth

represents a direction in a format common in surveying in North America and other places. Azimuth is and Angle, but the conventions are transformed. Value of 0.0 represents due north. Positive values increase clockwise. Values are constrained to be between 0.0 and 2π - δ radians (0.0 and 359.9999999999 degrees). The value of an Azimuth may be set to (written) as some value outside this range, but the value is immediately normalized to and persisted as a value in this range.

#### Tests

##### Azimuth\_1\_30\_addDeflection\_Pos2\_15\_shouldYieldNewAzimuth\_3\_45

##### Azimuth\_setFromXY

##### Azimuth\_Arithmatic\_subtraction

##### Azimuth\_Arithmatic\_addition

### Class Deflection

represents a change of direction. ~~There are no constraints on the value.~~ I’m not sure if it is constrained to +/ 360 degrees or +/- 180 or not. Common operations are Deflection defl = Azimuth2 – Azimuth1; and Azimuth3 = Azimuth2 + defl;

#### Tests

##### Deflection\_setTo\_Pos1Rad\_shouldBe\_Pos1Rad

##### Deflection\_setTo\_Neg1Rad\_shouldBe\_Neg1Rad

##### Deflection\_setTo\_Pos6Rad\_shouldBe\_Pos6Rad

##### Deflection\_setTo\_Pos2\_shouldBe\_Pos2Degrees

##### Deflection\_setTo\_neg5\_\_18\_\_29\_5

##### Deflection\_Right\_divideBy2\_isCorrect

##### Deflection\_Left\_divideBy2\_isCorrect

### Class Slope

represents a line traversing increasing z values along its positive length in x (uphill) or decreasing along its positive length in x (downhill). Somehow vertical slopes are allowed. I have to investigate more to be sure I understand this.

#### Tests

##### Slope\_from100PercentByConstructor\_is45Degrees

##### Slope\_IsVerticalUpward

## Class Degree

represents the “Degree” unit which divides circles into 360 degrees. ToString() returns a string ending with °.

### Tests

#### Degree\_sin90\_returns1p0

#### Degree\_Atan2Of10And0\_returns90degrees

#### Degree\_AsinOf1overSqrt2\_shouldEqual45degrees

## Class extendDoubleForPtsDegree

Methods:

### Degree AsPtsDegree(this Double val)

### Dictionary<String, Double> AsParts(this Double val) (OBSOLETE)

### FP

### Double ToRadians

### Double ToDegrees

# Other Classes (Documentation Pending)

## Foundations.Coordinates

### BoundingBox

### IboundingBox

### Point

### Ray

### StationOffsetElevation

### Vector

## Foundations.WorkingUnits

### Length

## Foundations.Symbology

### Foundations.Symbology.Color

#### Color

#### ColorAsBrush

### Feature

### FeatureList

### Style