MiscCoord

Inherits From: Object

Declared In: misckit/miscgiskit/MiscCoord.h

Class Description

MiscCoord is an abstract superclass, but it can be used on it's own if necessary. MiscCoord objects are containers for an arbitrary number of ***points***. A ***point*** is a vector of double precision floating point numbers which represent a ªlocationº, usually spatial, in an arbitrary ***coordinate system***. Coordinate systems come in many flavors, and the MiscCoord class does not specify any particular one: that is the purpose of subclasses. There are currently subclasses for mathematical coordinate systems such as cartesian, cylindrical, and spherical and geographic coordinate systems such as world and Universal Transverse Mercator (in numerous guises). More generally (and technically), a coordinate system for a MiscCoord subclass may be any one to one point mapping relation between two domains.

Most operations on MiscCoords are independant of the coordinate system in use. The primary need for specific information about the system in use is for conversion to another coordinate system. For this purpose, each

MiscCoord has the id of a ***conversion agent*** and a ***constants object***. Constants objects contain unchanging information that defines the relationship of the coordinate system of the particular MiscCoord object to some ***base reference frame*** in that coordinate system. The conversion agent is an object that handles requests for conversion of points to a different coordinate system. A conversion agent is an object that satisfies the **MiscCoordConversionServer** protocol. The particular one used should be one that can handle most common conversion requests directly, ie the agent for a cartesian MiscCoord handles conversions to cylindrical and spherical coordinates itself. Conversions to other systems may also exist, and the conversion agent is of class **MiscCoordConverter** the conversion will be ***subcontracted*** to another agent if necessary and possible. This allows the coord class into a profusion of subclasses while avoiding a complexity explosion inside the conversion agent.

A MiscCoord is accessed using the concept of ***current point*** and ***current block***. Block operations use the current point as the start of the current block and a block size to specify the end of the block. Single point read and write operation use only the current point.

There are no special methods for controlling the storage allocation. Storage is expanded or contracted automatically to fullfil requests that set the current block.

As mentioned earlier, MiscCoord will usually be subclassed. However an example of general usage will illuminate the actual simplicity of the object, something which may not have been made clear by the admittedly technical discussion above.

The following example creates two MiscCoords. Note that this is very similar to what would happen inside a subclass initialization method; in typical usage there would only be a single message required for the creation of each coord:

myConverter = **[**MyConverterClass **new];**
myConstants1 = **[[**MyConstantsClass1 **alloc] *initWithSomeArgs:*** ..... **];**
myConstants2 = **[[**MyConstantsClass2 **alloc] *initWithSomeArgs:*** ..... **];**

aCoord = **[[**MyFirstCoordClass alloc**]**
initDescription: "Upper Stranmillis Road" **converter:** *myConverter* **constants:** *myConstants1***];**

anotherCoord = **[[**MySecondCoordClass alloc**]**
initDescription: "Lower Stranmillis Road" **converter:** *myConverter* **constants:** *myConstants2***];**

The initialization creates storage for only one point, so if we need more we need to specify the storage requirements:

[*aCoord* **selectAndSetNumPoints:** 0 **blockSize:** 10**];**
[*anortherCoord* **selectAndSetNumPoints:** 0 **blockSize:** 5**];**

Now select a point, write values to it and read them back:

double *first*, *second*, *third***;**

if (! [*aCoord* **selectExistingPoints:** 3 **blockSize:** 1**]) exit (-1);**

[*aCoord* **setCoord:** 1.23456789 **:** 2.345678901 **:** 3.145196**];**
[*aCoord* **coord:** **&***first* **: &***second* **: &***third* **];**

Convert and append the contents of the first object to the second object:

[*aCoord* **selectExistingPoints:** 0
blockSize:[*aCoord* **numPoints]];**
[*anortherCoord* **selectAndSetMinPoints: [***anotherCoord* **numPoints] blockSize:[***aCoord* **numPoints]];**

if (! [*aCoord* **convert:** *anotherCoord***]) printf (**"Could not convert between coordinates systems");

Convert the first object and overwrite the contents second object. Second object will have the same size and descriptive text as the first if the operations succeeds:

if (! [*aCoord* **convertCoord:** *anotherCoord***]) printf (**"Could not convert between coordinates systems");

Instance Variables

char ***description**;
unsigned int **curIndex**;
unsigned int **curBlockSize**;
id **constants**;
id <MiscCoordConverterServer> **converter**;
Storage **pntStorage**;

description	A text description of the MiscCoord object. ie "Hillside Crescent"
curIndex	Index of the first point in the currently selected block.
curBlockSize	Size of the currently selected block.
constants	Id of the constants object .
converter	Id of a primary conversion agent.
pntStorage	Storage object containing the points. Storage is organized as a set of vectors of double precision floating point numbers.

Adopted Protocols

CoordConverterClient + radiansToDegrees:
+ degreesToRadians:
+ toDegreesOnlyDegrees:minutes:seconds:
+ fromDegreesOnly:degrees:minutes:seconds:
+ dimensions
- constants;
- curPtr
- curBlockSize

Method Types

Initialization
- initDescription:converter:constants:
- init
- free

Selecting blocks of points
- selectExistingPoints:blockSize:
- selectAndSetNumPoints:blockSize:
- selectAndSetMinPoints:blockSize:
- numPoints
- curIndex

Coordinate System Conversion - convertCoord:
- convert:
- converter

Accessing current point - setCoord: : :

	- coord: : : - coord1 - coord2 - coord3
Text description	- description - setDescription:
Archiving	- write: - read:

Instance Methods

convert:

- (BOOL) **convert:**<CoordConverterClient>*aCoord*

Convert the selected block of points in **self** into the coordinate system of *aCoord* and store the result into the selected block of points in *aCoord*. Returns YES if a conversion occured, NO if it did not. Conversions can be refused for a number of reasons:

There is no method available that can go directly from the coordinate system of **self** to that of *aCoord*.
The two objects have different numbers of dimensions.
aCoord does not conform to the **MiscCoordConverterClient** protocol.
The selected block sizes of **self** and *aCoord* are different.

See also: - **converter**, - **convertCoord:**

convertCoord:
- (BOOL) **convertCoord:**<CoordConverterClient>*aCoord*

Convert the entire contents of **self** into the coordinate system of *aCoord*. Result is stored in *aCoord*. The description text is copied as well. *aCoord* is resized to hold the same number of points as **self**. Returns YES if a conversion occured, NO if it did not. Conversions can be refused for a number of reasons:

There is no method available that can go directly from the coordinate system of **self** to that of *aCoord*.
The two objects have different numbers of dimensions.
aCoord does not conform to the **MiscCoordConverterClient** protocol.

See also: - **converter**, - **convert:**

converter
- <MiscCoordConverterServer>**converter**

Return the converter id.

See also: - **convert:**, - **convertCoord**

coord:::
- **coord:**(double*)*coord1* **:**(double*)*coord2* **:**(double*)*coord3*

Get the x,y and z values of the point at the current index. Returns **self**.

See also: - **setCoord:::**

coord1
- (double)**coord1**

Returns the first coordinate value of the point at the current index.

See also: - **coord2**, - **coord3**

coord2
- (double)**coord2**

Returns the second coordinate value of the point at the current index.

See also: - **coord1**, - **coord3**

coord3
- (double)**coord3**

Returns the third coordinate value of the point at the current index.

See also: - **coord1**, - **coord2**

curIndex
- (unsigned int)**curIndex**

Return the index of the beginning of the selected block of points.

See also: - **selectExistingPoints:blockSize:**, - **selectAndSetNumPoints:blockSize:**, - **selectAndSetMinPoints:blockSize:**, - **numPoints**

free
- **free**

Free the coord, its storage and description text. Also attempts to free the constants and converter objects. If they are
shared by other MiscCoords, they should be of classes that protect against freeing.

See also: - **init**, **initDescription:converter:constants:**

init
- **init**

Create a coord object with no converter and no constants.

See also: - **free**, **initDescription:converter:constants:**

initDescription:converter:constants:
- **initDescription:**(char *)*text* **converter:**<MiscCoordConverterServer>*converter* **constants:***anObject*

Initialize a coord. It makes a private copy of *text*. Storage is initialized to hold one object. To expand the storage do the following:

[myCoord **selectAndSetNumPoints:**0 **blockSize:** size**];**

See also: - **init**, - **free**

numPoints
- (unsigned int)**numPoints**

Return the number of points.

See also: - **selectExistingPoints:blockSize:**, - **selectAndSetNumPoints:blockSize:**, - **selectAndSetMinPoints:blockSize:**, - **curIndex**

read:

- **read:**(NXTypedStream *)*stream*

Reads the object from the typed stream *stream*. Returns **self**.

See also: - **write:**

selectExistingPoints:blockSize:

- (BOOL) **selectExistingPoints:**(unsigned int)*startIndex* **blockSize:**(unsigned int)*blockSize*

Select a block of points starting at *startIndex* of length *blockSize*. I f the block does not fit inside the existing capacity of the coord, returns NO. Often used to mark a block of points for conversion. Otherwise the block is selected and it returns YES.

See also: - **selectAndSetNumPoints:blockSize:**, - **selectAndSetMinPoints:blockSize:**, - **numPoints**, - **curIndex**

selectAndSetNumPoints:blockSize:

- (BOOL) **selectAndSetNumPoints:**(unsigned int)*startIndex* **blockSize:**(unsigned int)*blockSize*

Select a block of points starting at *startIndex* of length *blockSize*. The number of points is set to exactly fit the request. Storage is truncated or expanded as necessary to make the requested block end and the end of the coord be the same. It always returns YES.

See also: - **selectExistingPoints:blockSize:**, - **selectAndSetMinPoints:blockSize:**, - **numPoints**

selectAndSetMinPoints:blockSize:

- (BOOL) **selectAndSetNumPoints:**(unsigned int)*startIndex* **blockSize:**(unsigned int)*blockSize*

Select a block of points starting at *startIndex* of length *blockSize*. The number of points is set to exactly fit the request. Storage is expanded if necessary to make the requested block end fit within the coord. It always returns YES.

See also: - **selectExistingPoints:blockSize:**, - **selectAndSetNumPoints:blockSize:**, - **numPoints**, - **curIndex**

setCoord:::

- **setCoord:**(double)*coord1* **:**(double)*coord2* **:**(double)*coord3*

Sets the x,y and z values of the point at the current index. Returns **self**.

See also: - **coord:::**

setDescription:

- **setDescription:** (char *) text

Load the description field with a private copy of *text*. Frees the old text, if any. Returns **self**.

See also: - **setDescription:**

write:

- **write:**(NXTypedStream *)*stream*

Writes the object to the typed stream *stream*. Returns **self**.

See also: - **read:**