MiscCoordConverter

Inherits From: Object

Declared In: misckit/miscgiskit/MiscCoordConverter.h

Class Description

MiscCoordConverter is an abstract superclass. It supplies the job setup and subcontractor selection framework on which all subclasses rely. In most cases the use of MiscCoordConverters will be hidden inside a MiscCoord class so that the typical user of this kit need never worry about sending messages directly to a MiscCoordConverter. They need only deal directly with Coord objects: all else happens magically. If you are truly interested in magic, then read on.

MiscCoordConverters can be written for any subclasses that conform to the **MiscCoordConverterClient** protocol. The MiscCoordConverter itself conforms to the **MiscCoordConverterServer** protocol, so that conversion services are not completely tied to this class.

A customer object, usually a subclass of MiscCoord, sends a **convert:to:** message to a MiscCoordConverter

object. In the following example, a request is sent to convert *m* points starting at *n1* in *aCoord* into *anotherCoord* starting at *n2*. If *aCoord* is too small, nothing is done. If *anotherCoord* is too small, it is expanded as required:

```
myConverter = [MyConverterClass [allocFromZone: [self zone]] init];
if  ([aCoord  selectExistingPoints:  n1 blockSize: m])
     {[anotherCoord selectAndSetMinPoints: n2 blockSize:  m];
    flag = [myConverterr convert: aCoord to: anotherCoord];
     }
```

When *myConverter* receives this message, it searches its list of services to see if it has a method that can convert points from the coordinate system of the class of *aCoord* to that of the class of *anotherCoord*. If there is none, the converter tries to subcontract the job to another registered converter. The MiscCoordConverter class keeps a list of every coord converter that is created, and each of them is polled to see if it provides a service that can satisfy the request.

If no subcontractor can be found, a boolean value of NO is returned. It is expected that through smart programming this will rarely be the case, and in fact the original target will usually be the one to fullfil the job. In any case, once a job is "accepted", *aCoord* and *anotherCoord* are polled using the **MiscCoordConverterClient** protocol and a "job description" is filled in by the **subContractor** (which is usually *myConverter*). **theTransform** method found earlier is executed by the **subContractor** for each of the **npoints** to be converted or copied from within *aCoord* to *anotherCoord*. When the conversion is completed, a boolean value of YES is returned.

Subclasses are fairly simple to write. The only required method is an **init** method (which may be hidden by a **new** method and not exported). Often a **free** method will be included to block deletion of what is usually the single instance of the class.

The init method calls the superclass **init** and then executes a sequence of **addService:convertsFrom:to:** methods, one for each service to be provided. These services are not unexported other than through the services list. Conversion service methods have no arguments: each gets all its information from the instance variables **srcConstants**, **dstConstants**, **dimensions**, **src**, **dst** and **npoints** that are set up by the CoordConverter prior to

executing that method. Do not modify them! **src** and **dst** should be used as double arrays of size dimension. Data should be taken from **src**[i], converted and written into **dst**[i]. The conversion method must have a void return value. Use **fastCopySelector** method to get a selector id for use in cases in which no processing of points is required, ie both objects are of the same class and have the same constants.

Converters are not archivable because the data in them is easily reproduced. This should be done in the awake methods of MiscCoordConverterClients. Besides which, most MiscCoordConverters have only once instance and using an old one instead of the current one would not be nice.

Instance Variables

id **services**;
SEL **theTransform**;
id **subContractor**;
id **srcConstants**;
id **dstConstants**;
unsigned int **dimensions**;
double ***src**;
double ***dst**;
unsigned int **npoints**;
BOOL **sameConstants;**

services	A Storage object containing a list of the services provided by the converter object.
theTransform	Selector of the transform selected for the current job.
subContractor	Id of the MiscCoordConverter object that supplies theTransform.
srcConstants	Id of the constants object associated with the source MiscCoord object.
dstConstants	Id of the constants object associated with the destination MiscCoord object.
dimensions	Dimensions of a point, ie the number of double precision floating point numbers

	per point.
src	Pointer to next point within the source MiscCoord object.
dst	Pointer to next point within the destination MiscCoord object.
npoints	The number of points to be converted or moved from source to destination.
sameConstants	Set to value returned by **isEqual:** on srcConstants and dstConstants.

Adopted Protocols

CoordConverterServer	- convert:to:

Method Types

Initialization	- init - free
Registration	- addService:convertsFrom:to: - fastCopySelector

Instance Methods

addService:convertsFrom:to:
- **addService:**(SEL)*serviceMethod* **convertsFrom:**(Class)*srcClass* **to:**(Class)*dstClass*

Add the selector for *serviceMethod* to our list of services. It is defined to be applicable for converting points from

the coordinate system of *srcClass* to that of *dstClass*. Convertor service methods must be methods with no arguments that return void and will usually not be exported for use outside of the defining MiscCoordConverter except by this method. They convert or copy a point consisting of **dimensions** number of double precision floating points from the **src** address to the destination address **dst**.

[self **addService:** *@selector(internalMethodReturningVoid)* **convertsFrom:** *aClass* **to:** *anotherClass***]**;

Returns **self**.

See also: - **fastCopySelector**

awake
- **awake**

Make the unarchived object a registered converter. You should never invoke this method directly. Returns **self**.

See also: - **write:**, - **read:**

fastCopySelector
- (SEL)**fastCopySelector**

Returns the selector for an internal method that will do a verbatim copy of **npoints** of size **dimensions** from **src** to **dst**. It will typically be used in **addService:convertsFrom:to:** calls:

[self **addService: [**self **fastCopySelector] convertsFrom:** *aClass* **to:** *theSameClass***]**;

Note that the fast copy service will fail if the **sameConstants** isn't true, such as if the source and destination coord objects have different reference frames. For the moment, implementation of a method that can copy items of the same class must either accept this; do the copy in two stages; or implement a special service method that

only uses the fast copy if the aforementioned condition is true, and otherwise carries out whatever mathematical transform is necessary.

See also: - **addService:convertsFrom:to:**

free
- free

Remove the object from the list of registered MiscCoordConverter subcontractors and frees it's list of services provided. Many subclasses will override this method and block free'ing entirely since the pointer will often be distributed widely.

See also: - **init**

init
- init

Registers the object with the list of registered MiscCoordConverter subcontractors and creates an initially empty Storage object to which descriptions of services provided by this object may be added. Subclasses will override this method but send it as part of their own init, prior to adding their description of services to the services list.

See also: - **free**