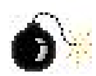
Simtools Library

Documentation and Implementation Status



Compatibility

- `1.0.4 -> 1.0.5: GUISwarm now inherits from GUIComposite. Because GUIComposite handles the passthru of archiving keys to tkobjc primitives, the method setControlPanelGeometryRecordName is no longer needed; instead, the macro SET_WINDOW_GEOMETRY_RECORD_NAME is provided.`

- `1.0.3 -> 1.0.4: All functions maintain backward compatibility. There are additional features, however, and features previously undocumented.`

Overview

Simtools is the stdlib of Swarm... In other words, it is the library where we have parked many miscellaneous classes which, while very useful, are not specific enough to be placed in any other library. So, for example, simtools contains InFile, ObjectLoader and other I/O classes, the ControlPanel and GUISwarm which support GUI interactions, but also UName which simply generates unique names for objects.

Documentation

The *`objectbase`* library is defined and documented following the basic structure of the `Library Interface Conventions`

`(defobj/library.html)`
. The following standard sections of documentation are available:

- `Interface Reference`
`(simtools/Interface.html)`

Simtools Library: Documentation and Implementation Status

The simtools library has undergone an upgrade to the status of a library as of 1.0.4. This means the interface now conforms to the library interfaces specifications, the format of the documentation now reflects this. Other than new features, there should be no affect on the user.

Marcus G. Daniels <mgd@santafe.edu>
(mailto:mgd@santafe.edu)

Last modified: 1998-01-08

globalTkInterp (from libtkobjc) is set to be a normal Tcl interpreter instead of a Tcl/Tk/BLT interpreter.

Object Types (with links to reference pages)

As simtools is a collection of tools that are only loosely related to each other - the class hierarchy is virtually flat. Most of the classes are standalone.

Contents:

- GUISwarm
(GUISwarm.html)
- ControlPanel
(ControlPanel.html)
- ActionCache
(ActionCache.html)
- ProbeDisplay
(ProbeDisplay.html)
- CompleteProbeDisplay
(CompleteProbeDisplay.html)
- ProbeDisplayManager
(ProbeDisplayManager.html)
- InFile
(InFile.html)
- OutFile
(OutFile.html)
- AppendFile
(AppendFile.html)
- ObjectLoader
(ObjectLoader.html)
- ObjectSaver
(ObjectSaver.html)
- QSort
(QSort.html)
- NSelect

Name

GUISwarm

Synopsis

```
//
// GUISwarm -- a version of the Swarm class which is graphics aware
//
@protocol GUISwarm <SwarmProcess>
@end
```

Description

GUISwarm is a subclass of Swarm that is used as a toplevel Swarm for simulations running with a graphical user interface. The GUISwarm creates a ControlPanel automatically for you and defines a "go" method that interprets the state of the ControlPanel to keep things running in response to user input. Users subclass GUISwarm much like they subclass a normal Swarm, implementing the same kind of buildObjects, buildActions, and activateIn methods. When you are done building your Observer Swarm, start it as a toplevel via `[myGUISwarm go]`.

The control panel places a few responsibilities on the GUISwarm subclass author. In particular, a message to `[controlPanel doTkEvents]` should be scheduled fairly frequently - only when that method is executed will the user interface update (and the stop button be checked). Also, it is often useful to use `[controlPanel setStateStopped]` to wait for the user to indicate they're ready for execution to proceed.

- go

  Start the activity running, and also handle user requests via the control panel. Returns either `Completed` (the model ran until requested to terminate) or `ControlStateQuit` (the user pressed the quit button).

Marcus G. Daniels <mgd@santafe.edu>
(mailto:mgd@santafe.edu)

Last modified: 1998-01-08

Get the current button state of the controlpanel. Is one of ControlStateRunning, ControlStateStopped, ControlStateStepping, ControlStateNextTime, or ControlStateQuit.

- setStateRunning, - setStateStopped, - setStateStepping, - setStateQuit, - setStateNextTime

  Sets the state to the indicated value, and also sends the appropriate stop or terminate message to the running activity tree. The - setStateStopped message is particularly useful since it will cause the simulation to stop until the user interactively sets it back in motion (in other words, this method is useful in generating a software-triggered pause).

Marcus G. Daniels <mgd@santafe.edu>
(mailto:mgd@santafe.edu)

Last modified: 1998-01-08

- setWindowGeometryRecordName: (const char *)name

  Set the name to be used for archiving the geometry of the control panel widget. There is no default; the position of the control panel will not be preserved unless you use this method.

Marcus G. Daniels <mgd@santafe.edu>
(mailto:mgd@santafe.edu)

Last modified: 1998-01-08

In order to record the window position of a ProbeDisplay, it is necessary to give it a name. This method is used to give an instance ProbeDisplay a name, which will used by the Archiver when recording its geometry information.

Regular Protocol

- getProbedObject
- getProbeMap
- update

  This method maintains consistency between the values of the probedObject's variables and the values which are displayed in the ProbeDisplay. Ideally, this method should be called every time the object is modified by the simulation. In practice, the user schedules an update on the probeDisplayManager which in turn communicates to all the active ProbeDisplays in the system.

Marcus G. Daniels <mgd@santafe.edu>
(mailto:mgd@santafe.edu)

Last modified: 1998-01-08

Name

ProbeDisplayManager

Synopsis

```
//
// ProbeDisplayManager --
// A singleton class whose instance is used to manage all the
// ProbeDisplays created by the user during a GUI run of the
// simulation.
//

@protocol ProbeDisplayManager <SwarmObject>
USING
- createProbeDisplayFor: anObject;
- createProbeDisplayFor: anObject setWindowGeometryRecord-
Name: (const char *)name;

- createCompleteProbeDisplayFor: anObject;
- createCompleteProbeDisplayFor: anObject setWindowGeometryRecord-
Name: (const char *)name;

- addProbeDisplay: probeDisplay;
- removeProbeDisplayFor: anObject;
- removeProbeDisplay: probeDisplay;
- setDropImmediatelyFlag: (BOOL)dropImmediateFlag;
- update;
@end
```

Description

- addProbeDisplay: pd

- removeProbeDisplay: pd

    These methods are used to specify which Probe Displays should be managed by the ProbeDisplayManager.

- update

    This method will recursively send an update message to all the Probe Displays managed by the ProbeDisplayManager.

Name

InFile - class to perform file input

Synopsis

```
//
// InFile --
// A class which was intended to support file input. There have been
// justified requests from our userbase to re-design this interface.
//

@protocol InFile <SwarmObject>
CREATING
+ create: aZone withName: (const char *)theName;
USING
- (int)getWord: (char *)aWord;
- (int)getLine: (char *)aLine;
- (int)getInt: (int *)anInt;
- (int)getUnsigned: (unsigned *)anUnsigned;
- (int)getLong: (long *)aLong;
- (int)getUnsignedLong: (unsigned long *)anUnsLong;
- (int)getDouble: (double *)aDouble;
- (int)getFloat: (float *)aFloat;
- (int)getChar: (char *)aChar;
- (int)unGetChar: (char)aChar;
- (int)skipLine;
@end
```

Description

This class is intended to simplify the input file-I/O in Swarm. It essentially deals with the detailed file opening and closing routines thus alleviating the need for C file I/O procedure calls.

`+ create: aZone withName: (const char *)theName`

This is the create method for InFiles, where theName is, of course the name of the file to open.

`- (int)getWord: (char *)aWord`

Name

OutFile - class to perform file output

Synopsis

```
//
// OutFile --
// A class which was intended to support file output.
//

@protocol OutFile <SwarmObject>
CREATING
+ create: aZone withName: (const char *)theName;
USING
- putString: (const char *)aString;
- putInt: (int) anInt ;
- putUnsigned: (unsigned) anUnsigned;
- putLong: (long) aLong ;
- putUnsignedLong: (unsigned long) anUnsLong ;
- putDouble: (double) aDouble ;
- putFloat: (float) aFloat ;
- putChar: (char) aChar ;
- putTab ;
- putNewLine ;
@end
```

Description

This class is intended to simplify output file-I/O in Swarm. It essentially deals with the detailed file opening and closing routines thus alleviating the need for C file I/O procedure calls.

+ create: aZone withName: (const char *)theName
  This is the create method for OutFiles, where theName is, of course the name of the file to open.

- putString: (const char *)aString

- putInt: (int) anInt

Name

AppendFile - class for appended file output

Synopsis

```
//
// AppendFile --
// A class to support appended file output.
//
@protocol AppendFile <SwarmObject>CREATING
+ create: aZone withName: (const char *)theName;
USING
- putString: (const char *)aString;
- putInt: (int)anInt;
- putUnsigned: (unsigned)anUnsigned;
- putLong: (long)aLong;
- putUnsignedLong: (unsigned long)anUnsLong;
- putDouble: (double)aDouble;
- putFloat: (float)aFloat;
- putChar: (char)aChar;
- putTab;
- putNewLine;
@end
```

Description

This class subclasses from OutFile, the only difference in functionality, being that it opens a given file in *Append*

mode, rather than *Overwrite*mode, which OutFile does.

+ create: aZone withName: (const char *)theName
  This is the create method for AppendFiles, where theName is, of course the name of the file to open.

- putString: (const char *)aString

- putInt: (int)anInt

- putUnsigned: (unsigned)anUnsigned;

Name

ObjectLoader - class to load objects IVars from a file

Synopsis

```
//
// ObjectLoader --
// An attempt to design some form of standard for object loading.
//

@protocol ObjectLoader <SwarmObject>
USING
+ load: anObject from: aFileObject;
+ load: anObject fromFileNamed: (const char *)aFileName;

- setFileObject: aFileObject;
- loadObject: anObject;
// In case the same class is being loaded multiple times...
- updateCache: exampleTarget;
@end
```

Description

This class is used to initialize the variables of a target object from a data file. The data file is required to have a

very simple format:

```
# A Comment

@begin

aVarName aValue
aVarName aValue # A comment
aVarName aValue
#
# A Comment
#
aVarName aValue
```

Name

ObjectSaver - class to save object's IVars to a file

Synopsis

```
//
// ObjectSaver --
// An attempt to design some form of standard for object saving.
// The ProbeMap argument is used to specify what variables get saved.
//

@protocol ObjectSaver <SwarmObject>
USING
+ save: anObject to: aFileObject;
+ save: anObject to: aFileObject withTemplate: aProbeMap;
+ save: anObject toFileNamed: (const char *)aFileName;
+ save: anObject toFileNamed: (const char *)aFileName withTemplate: (Probe-
Map *)aProbeMap;

- setFileObject: aFileObject;
- setTemplateProbeMap: aProbeMap;
- saveObject: anObject;
@end
```

Description

```
This class is used to write an object's variables to a specified file.  If only a subset of
the variables should be

written out, the set is specified by a template ProbeMap (where the ProbeMap will contain Probes
for those
variables which should be saved).
```

```
+ save:  anObject to:  aFileObject
+ save:  anObject to:  aFileObject withTemplate:  aProbeMap
+ save:  anObject toFileNamed:  (const char *)aFileName
+ save:  anObject toFileNamed:  (const char *)aFileName withTemplate:  aProbeMap
```

Name

QSort - class to sort a collection

Synopsis

```
//
// QSort --
// A class (not to be instantiated) wrapper around the C sort routine.
//

@protocol QSort <SwarmObject>
USING
+ sortObjectsIn: aCollection;
+ (void)sortObjectsIn: aCollection using: (SEL)aSelector;
+ (void)sortNumbersIn: aCollection;
+ (void)sortNumbersIn: aCollection us-
ing: (int(*)(const void*,const void*)) comp_fun;
+ (void)reverseOrderOf: aCollection;
@end
```

Description

QSort is simply a "wrapper" for the C native "qsort" function, as applied to a Swarm collection. The values will appear in ascending order by default. Reversing the order of a collection can be made by calling reverseOrderOf. All these methods modify the underlying collection, so any indexes should always be regenerated.

+(void) sortObjectsIn: aCollection
: Given a collection, this method will sort the objects with the "compare" function for the object. If the objects don't provide a compare function, it uses the default from the defobj library.

+(void)sortObjectsIn: aCollection using: (SEL)aSelector
: This method allows the user to specify a given message selector to use to "compare" the objects.

+(void) sortNumbersIn: aCollection;
: This method is intended specifically for integer comparisons, again using a default "compare" function. The default assumes numbers should be monotonically increasing.

Name

NSelect - class to select exactly N elements at random from a collection

---

Synopsis

```
//
// NSelect --
// A class (not to be instantiated) wrapper around a Knuth algorithm
// for the selection of exactly N elements form a collection without
// repetition.
//

@protocol NSelect <SwarmObject>
USING
+ (void)select: (int)n from: aCollection into: bCollection;
@end
```

Description

NSelect selects exactly N elements from a collection without repetition. A target collection must be provided.

Marcus G. Daniels <mgd@santafe.edu>
(mailto:mgd@santafe.edu)

Last modified: 1998-01-08

- + create: aZone setBaseNameObject: aStringObject
- - setBaseName: (const char *)aString
- - setBaseNameObject: aStringObject

  These messages should be used to set or reset the base name.
  Note: these methods will automatically reset the counter.
- - (const char *)getNewName
- - getNewNameObject

  Generates a new name in the requested format.
- - resetCounter

  Resets the counter used as a suffix in the unique names generated.

Marcus G. Daniels <mgd@santafe.edu>
(mailto:mgd@santafe.edu)

Last modified: 1997-12-19

Marcus G. Daniels <mgd@santafe.edu>
(mailto:mgd@santafe.edu)

Last modified: 1998-01-08

(mailto:mgd@santafe.edu)

Last modified: 1998-01-08