
Mousetrap
CreateDrop
createEnd
SwarmObject
getCompleteProbeMap
getInstanceName
getProbeForMessage:
getProbeForVariable:
getProbeMap
Mousetrap
displayWidget
ZoomRaster
modelSwarm
MousetrapModelSwarm
triggered
0
xCoord
20
yCoord
20
reload
setDisplayWidget:
trigger

```
[probeDisplayManager createProbeDisplayFor: anObject] ;
```

And that the contents of the returned ProbeDisplay will be based on a ProbeMap obtained from the probed object itself.

Probe support resident in the objectbase *(objectbase.html)* library is documented with that library. Extra description, above and beyond the bare interface description for precision probing and SwarmObject is below:

Precision setting for floating-point numbers with probes

Global setting of precision

There are two types of global precision setting via ProbeLibrary (objectbase/ProbeLibrary.html) :

-setDisplayPrecision: (int) nSigFigsSaved
: Sets the number of significant figures for floating point (and double-floating) numbers diplayed on a GUI widget. Currently this is only implemented for VarProbes. The display uses the %*g sprintf-style formatting, which can vary slightly from implementation to implementation. If you set the number of significant figures to 3, then a float of value of 0.6344346 is displayed as 0.634 on the GUI widget. Note that this in no way affects the underlying stored value of the floating point number.

-setSavedPrecision: (int) nSigFigsSaved
: Sets the global default for the saving of floats through ObjectSaver. All objects with floats and doubles as instance variables are saved with the precision specified by this method. This is independent of the displayed precision of the same instance variable on a GUI widget.

To actually initialise these defaults: in the top level swarm, you should add the calls to the global ProbeLibrary instance (which is actually created by the initSwarm call in main) during the - createBegin method (this sets the precision in the global instance, `before` any probes are checked out of the instance. If neither method is called on probeLibrary, then the precision defaults to six significant figures in both cases.

Example: Here's how it's done in `HeatbugObserverSwarm.m`:

```
+ createBegin: aZone
{
  HeatbugObserverSwarm * obj;
  id <ProbeMap> probeMap;
```

```
  id floatProbe;

  [...]
  // the -setFloatFormat is applied to the probe which is
  // "returned" from the call to probeLibrary
  floatProbe = [[probeLibrary getProbeForVariable: "randomMoveProbability"
    inClass: [self class]]
                                   setFloatFormat: ".3f"];

  // now we have the probe - put it back into the customMap
  [probeMap addProbe: floatProbe];

  [...]
  return obj;
}
```

Or, more compactly:

```
+ createBegin: aZone
{
  HeatbugModelSwarm *obj;
  id <ProbeMap> probeMap;

  [...]

  [probeMap addProbe: [[probeLibrary getProbeForVari-
able: "randomMoveProbability"
    inClass: [self class]]
                              setFloatFormat: "%.3f"]];
  [...]
  return obj;
}
```

SwarmObject Support for Probing

As mentioned earlier, all SwarmObjects can generate Probes/ProbeMaps. By default, they achieve this by ”checking out” the appropriate references from the probeLibrary.

```
Interface definition
(objectbase/SwarmObject.html)
from objectbase
(objectbase.html)
library.

  - (const char*)getInstanceName
```

On the one hand if an object to be probed is specified *without* any particular ProbeMap being specified, then the ProbeDisplay generated will provide a window of class DefaultProbeDisplay, which displays all the variables resident in that class structure.

On the other hand, if a ProbeMap *is* specified then the ProbeDisplay follows exactly the specification as represented by the contents of the ProbeMap (above). When used in this manner, ProbeDisplays can generate tailored interfaces to objects (so for example, we have purposefully hidden certain instance variables in the HeatbugModelSwarm class, and have shown only one of the methods which the class understands).

Graphical Interface Usage

Common to both the standard ProbeDisplay and the CompleteProbeDisplay:

- The different fields in the ProbeDisplay can be updated by typing in new values and pressing Return. However, certain fields (containing pointers or ids, for example) cannot be modified and will generate a beep if such a modification is attempted.
- If an instance variable/argument slot is defined to hold an object, then that object can be drag&dropped into another variable/argument slot by clicking on it with the first mouse button (a small rectangle with the name of the object will appear - simply drag it to another object-typed variable/argument

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

Last modified: 1998-01-08