1

ARI Research Note 87-49

CONCEPTS AND NOMENCLATURE IN THE WORLD MODELING SYSTEM
(A Lesson in Applied Metaphysics)

Greg Hood
University of California at Irvine

for

Contracting Officer's Representative
Judith Orasanu

BASIC RESEARCH LABORATORY
Michael Kaplan, Director

ari

U. S. Army
Research Institute for the Behavioral and Social Sciences

October 1987

Approved for public release; distribution unlimited.

U. S. ARMY RESEARCH INSTITUTE

FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency under the Jurisdiction of the Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON Technical Director	WM. DARRYL HENDERSON COL, IN Commanding

Research accomplished under contract for the Department of the Army

University of California at Irvine

Technical review by

Michael Kaplan

This report, as submitted by the contractor, has been cleared for release to Defense Technical Information Center (DTIC) to comply with regulatory requirements. It has been given no primary distribution other than to DTIC and will be available only through DTIC or other reference services such as the National Technical Information Service (NTIS). The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other official documentation.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER ARI Research Note 87-49	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) CONCEPTS AND NOMENCLATURE IN THE WORLD MODELING SYSTEM (A Lesson in Applied Metaphysics)		5. TYPE OF REPORT & PERIOD COVERED Interim Report January 85 - January 86 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Greg Hood		8. CONTRACT OR GRANT NUMBER(s) MDA903-85-C-0324
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of California at Irvine Department of Information and Computer Science Irvine, CA 92717		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2Q161102B74F
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Research Institute for the Behavioral and Social Sciences, 5001 Eisenhower Avenue, Alexandria, VA 22333-5600		12. REPORT DATE October 1987 13. NUMBER OF PAGES 9
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) - -		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE n/a

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

- -

18. SUPPLEMENTARY NOTES

Judith Orasanu, contracting officer's representative

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

World Modeler, Computer Models,
World Master, Artificial Intelligence
World Coordinator

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This research note provides the nomenclature information necessary to create a World Modeling organism or simulation. Nomenclature for worlds, objects, organisms, views, images, times, and states is provided. Information on checkpointing, tracing, creating log and image files, implementation guidelines, and file formats is also included. (Keywords:

Concepts and Nomenclature in the World Modeling System

or

A Lesson in Applied Metaphysics

Greg Hood
29 January 1986

Status: Proposed

1. Worlds

Each simulation that is to be run must have a unique *world name*. This name must be registered with the World Coordinator by placing it in the file "/usr/worldm/worlds.start", along with the filename of the desired World Master. At this point the authorized users of that world may be listed, along with the filenames of their desired User Interfaces.

2. Objects

All objects (both complex and primitive) within a world must have a name associated with them. The name of an object cannot be more than 31 characters in length. The rule for uniqueness is that the name of an object must be unique among it and its siblings. The full name of an object is formed by concatenating the names of all of its superobjects with its own name (in a fashion similar to Unix filenames). For example, if we have two tables, then the names may be as follows:

```
table1
            top
            leg1
            leg2
            leg3

table2
            top
            leg1
            leg2
            leg3
```

To designate the first leg of the second table the full name "table2.leg1" would be used. The name of an object must be given at the time the object is created. The User Interface will check that the specified name satisfies the uniqueness requirement. It is also possible to change the name of an object after it has been created.

3. Organisms

Each organism in a simulation must have a unique name. This is given at the time when the StartOrganism or AttachOrganism command is issued. It is possible to change an organism's name later on if that is required.

4. Views

Each view of the world which the Perq generates must have a name. Thus, a global view of the world may have the name "birdseye", while the view as seen from organism A's perspective may have the name "viewA". View names must be unique.

5. Images

A view when computed by the Perq may have more than one image request associated with it. An image request specifies some post-processing which is to be done to the view (thus turning it into an image), and also specifies what to do with the image (e g. display it on the Perq screen, save it in a file, or deliver it to an organism). Each series of images which is generated by the Perq must also be given a name. These must be unique. They may, however, coincide with the view name for that image.

6. Time

The time within a world is measured in seconds. When the world is first created, the time is set to 0. Thereafter, time increases in quantized steps, called timesteps. A timestep is typically a constant between 0.1 second and 1 second, although this can vary over the course of a simulation.

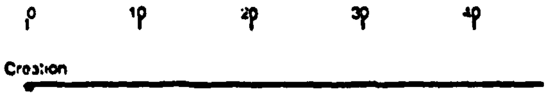

Figure 6-1:

A timeline is useful for visualizing the simulation (see Figure 6-1). At any time in the simulation, the world state may be saved by giving it a state name. For example, we may save the state at time 10 and at time 20. This is represented on the timeline as shown in figure 6-2. It is possible to restore the state

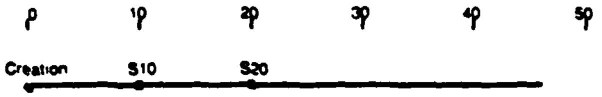

Figure 6-2:

of the world to that of a saved state. We may think of this going back to a node on the timeline. The simulation may be continued again with possibly different actions on the part of the user or the organisms within the world. Thus, it will not necessarily retrace the path of the old timeline, but will diverge to form a distinct branch. This process may happen repeatedly so that a complex tree may result as depicted in figure 6-3. The root of the tree is always the initial world state at time 0, termed the "creation" state.

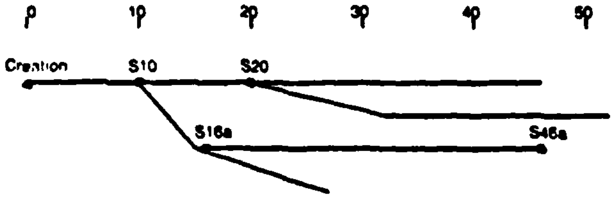

Figure 6-3:

7. States

Whenever an entire simulation is saved by the SaveState command, the state of the world is saved along with the internal mental states of all organisms within the world. The SaveState command will ask the user for the name of this saved state. The state of the world will be saved in a file of the form "<world name>.<state name>.WM.state" within the working directory of the world master. The state of each organism will be saved in a file of the form "<world name>.<state name>.<organism name>.state" within the working directory of the organism. The length of the world name plus the longest organism name plus the state name must not exceed 6 characters because of Unix 4.1 filesystem restrictions. Thus, it is good to try to choose short names until we get support for long filenames in Unix 4.2. In addition, a line describing the state will be appended to the file "<world name>.WM.stdir". There is one special state which is present in all worlds which is given the name "creation". This state is not listed in the state directory. Restoring the state to the "creation" will remove all objects and organisms from the world, since there is initially nothing in a world when it is first created.

8. Checkpointing

It is important that the system save the state of the world occasionally even in the absence of any explicit "Save State" command. This is so that in the event of a hardware or software failure, many hours of user and computer time will not be lost. Thus, after a certain number of timesteps, the entire simulations is checkpointed. A checkpoint operation is handled exactly as a SaveState command with state name "CKP<number>". The number is an integer which is the truncation of the world time multiplied by 1000. Thus, worlds will be saved in files with names of the form "<world name>.CKP<time>.WM.state" and organisms in files with names of the form "<world name>.CKP<time>.<organism name>.state". A line describing the checkpoint will also be added to the file "<world name>.WM.statedir". Checkpoint files are automatically deleted to conserve disk space just before the second following checkpoint file is written. This ensures that at least one good set of checkpoint files will be available at any time.

9. Traces

During the execution of a simulation, it is often necessary to record certain information such as successive views of the world, or a trace of an organism's internal state[1]. The log files produced during a portion of simulated time must be given a collective name, which will be termed a "trace". We may indicate this on the timeline graph by the labeling of a path on that graph with the trace name (see Figure 9-1). It is possible to return back to a state and then rerun the simulation from that point producing a new trace. The only restriction is that log files in the same trace cannot overlap in time. In other words, when a world state is restored, the trace name must be changed or else the previous log files generated for paths extending after the restored time will be lost. For example, in figure 9-1, if the state is restored to "State1" and the trace name is changed to "trace4" then the simulation can be replayed and both traces will be preserved (see figure 9-2). If, however, the trace name is not changed then the previous log files with the name "trace3" which extend after "State1" will be erased, and overwritten by the new log files (see figure 9-3).

10. Log files

Individual log files are created for each segment of the timeline graph. Thus, whenever a state is saved (including a checkpointed state), new log files will be opened for subsequent recording. The form of the log files for organisms is "<world name>.<trace name>.<organism name>.<time>.log". The "<time>" is the world time at the start of that segment multiplied by 1000 and truncated. In addition

[1] Using the *Viewer* program, it will be possible to rapidly display several logs on the same screen and sequence through them.

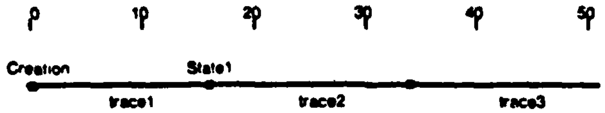

Figure 9-1:

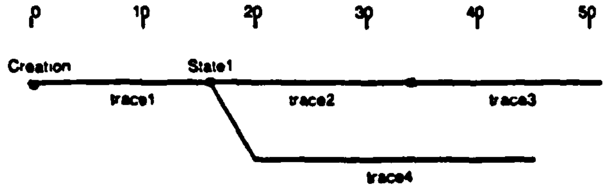

Figure 9-2:

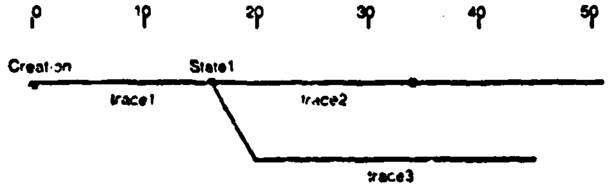

Figure 9-3:

the log file name is appended to the list of log files in that trace, which is stored in a file of the form "<world name>.<trace name>.<organism name>.logdir".

11. Image Files

A new file is created for each individual image generated regardless of the state saving which has taken place. The form of the filenames for images is "<world name>.<trace name>.<image name>.<time>.img". The "<time>" is the world time multiplied by 1000 and truncated. These filenames are also added to an image file list, which is kept in a file of the form "<world name>.<trace name>.<image name>.imgdir". Image files are stored in the directory "/sys/user/images/" on the Perq on which they were generated.

12. Implementation Guidelines

The new commands will require one new command to be added to the protocol:

Set Trace Name **c 68 <string trace name>**

[UI to WM, WM to UI, UI to PQ, WM to SEI, SEI to WM (unlikely)] This command sets the trace name for the simulation. The appropriate action on receipt of this message is:

1. Close all previous log files.

2. Set the internal trace name to its new value.

3. Open the file "<world name>.<trace name>.<organism name>.logdir", creating it if necessary.

4. Go through the file and find all previous log files which end after the current time, and delete them, including their entry in the .logdir file.

5. Whenever new log information is to be written, the program must then open the new log file of the form "<world name>.<trace name>.<organism name>.<time>.log".

The semantics of some existing commands will have to be extended to support the log file concept:

Save State **c 45 <string>**

[UI to WM, WM to UI, WM to SEI]
Upon receipt of this command an organism should make sure the log file has been closed, and the line in the log file directory updated to include the ending time for that log file. When subsequent information is to be written to the log file, a new log file will be opened.

Restore State **c 46 <string>**

[UI to WM, SEI to WM, WC to WM, WM to UI, WM to SEI]
The handling of log files upon receipt of this command is done as follows:

1. The currently open log file is closed.

2. The log file directory is updated.

3. The internal state is restored.

4. Find all log files listed in the log file directory which end after the restored time, and delete them, including their entry in the log file directory.

5. Whenever new log information is to be written, the program must then open the new log file of the form "<world name>.<trace name>.<organism name>.<time>.log".

13. File Formats

13.1. State file directories

```
<time1> <state name 1>
<time2> <state name 2>
...
```

The entries are in unsorted order.

13.2. State files

The format of these is specific to the program saving the state. The only constraint is that the program which saved it must be able to read it back to restore its state.

13.3. Log file directories

```
<time x 1000 and truncated> <starting time> <ending time>
...
```

The entries are in unsorted order.

13.4. Log files

Log files have the following format:

```
↑L
World time: <time>
<log file information>
...
...
↑L
World time: <time>
<log file information>
...
...
...
```

The chunks of information in these files is sorted according to world time.

13.5. Image file directories

```
<time x 1000 and truncated> <height in pixels> <width in pixels>
...
```

The entries are unsorted.

13.6. Image files

Image files on the Perq are in the special Accent file format for bitmaps so that LoadVPPicture can be used to rapidly read the image.