APPENDIX A: GLAIR STATUS REPORT

Progress Report of Work Performed from February to May, 1995
for
Amherst Systems Inc.
Subcontract No. 150-7176A

Design of GLAIR for the Foveal Robot

Stuart C. Shapiro and Henry Hexamoor
Department of Computer Science
and Center for Cognitive Science
State University of New York at Buffalo
226 Bell Hall
Buffalo, New York 14260

August 1, 1995

1. Overview

We received the Nomad200 simulation software for the FEVAHR robot. We have begun the following activities: (a) we are implementing the FEVAHR room using the simulation, including simulation of the vision system, and (b) we have developed the perceptuo-motor level component of FEVAHR to work with the Nomad200 simulation. The implemented PMA uses sonar data. The PMA does not yet use vision data and it is not yet coupled with the Knowledge Level (KL) component of FEVAHR.

We continued implementation of the KL component of FEVAHR with a SNePS network. We used the graphics tool Garnet to simulate the FEVAHR room. The KL component is made to work with the simulated room. In the remainder of this report we present the details of the KL component.

2. Knowledge Level Progress

We have partially implemented FEVAHR's Knowledge Level. This has involved partial implementation of:

- a grammar for understanding natural language input and for natural language generation,
- plans and primitive actions;
- alignment of KL representations of primitive actions and other entities with PML representations, and
- a simulated robot and environment.

2.1 The Garnet Simulation

Since this effort predated the arrival of the Nomad simulation software, we used the Garnet[1] graphical user interface package to create the simulated robot and environment.

According to the specifications, FEVAHR will be in a 17' x 17' room, containing

at least 1 named individual, e.g., ``John,"
at least 1 individual unique by description,
at least 1 indistinguishable class,

all of which will have a minimum dimension of 1'.

Figure 1 shows the objects in the simulated room. In the upper left is FEVAHR, represented by a cyan circle. In the lower-right is a blue square with a "name tag" representing John. In the upper right is a green circle representing "the green robot." In the lower left are three red circles representing "red robots."

The room itself is simulated by a graphics window whose size is a scaled 17' x 17'. In the same scale, John is $1\,1/2$' on a side, and each of the 5 robots is 1' in diameter.

John and the green and red robots can be moved by a user dragging them with the mouse, simulating their movement themselves, or their being moved by an agent external to FEVAHR. FEVAHR, however, moves only under program control.

All objects are implemented as Garnet CLOS objects. Their locations within the simulated room are maintained by Garnet as values of certain slots.

[1] Brad A. Myers & Andrew Mickish. "Overview of the Garnet System," Carnegie Mellon University, Oct., 1993.

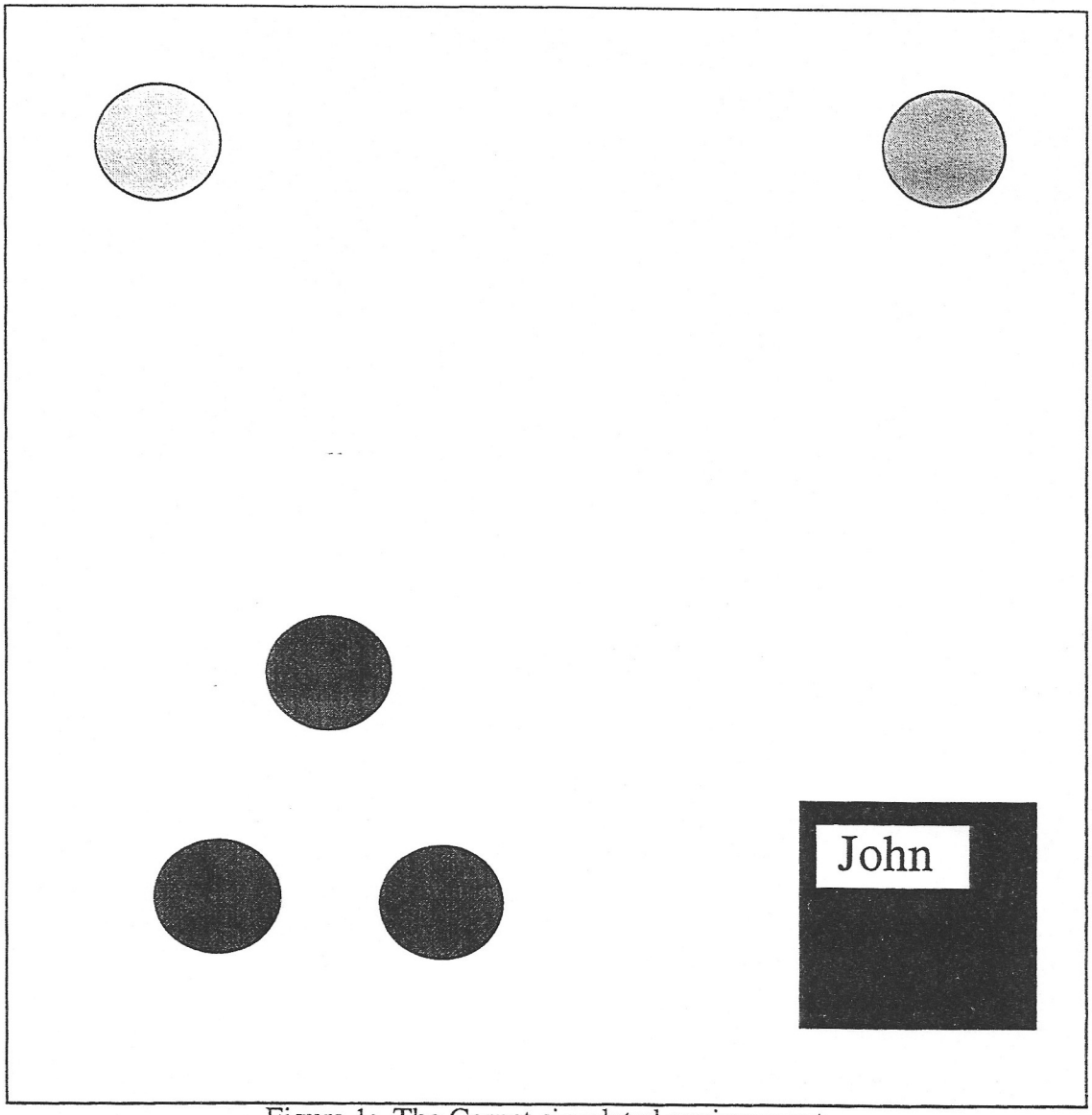

Figure 1: The Garnet simulated environment.

2.2 Alignment of Entities

At the KL, an entity is represented by a SNePS node representing the entity as FEVAHR thinks of it. At the PML and SAL, however, FEVAHR can only have a sensory impression of the entity. Therefore, SNePS nodes are aligned with descriptions, where a description is implemented as a list of the color of the entity and its shape. This description of John is `(#k<OPAL:BLUE-FILL> FEVAHR-WORLD:SQUARE)`. The green robot and the red robots don't have descriptions themselves. Instead the SNePS node representing the category of robots is aligned with the description `(NIL FEVAHR-WORLD:CIRCLE)`, the node representing the color green is aligned with the description `(#k<OPAL:GREEN-FILL> NIL)`, and the node representing the color red is aligned with the description `(#k<OPAL:RED-FILL> NIL)`.

The alignments, themselves, are implemented by a global assoc. list, ***alignments*** which is of the form $(\cdots(\ node\ .\ description)\ \cdots)$.

2.3 Natural Language Commands

According to the FEVAHR specifications, the minimal command language is
<command> ::= Stop | <action> <np>
<action> ::= Go to | Follow
<np> ::= <npr>
| (that | a) [<adj>] <n>

In fact, the currently implemented grammar includes
<command> ::= Stop | <action> <np>
<action> ::= Find | Go to | Follow
<np> ::= <npr>
| (that | this | the | a) <cat>
<cat> ::= [<adj>] <n>

The relevant lexicon is
<adj> ::= green | red
<np> ::= robot
<npr> ::= John

giving 1 + 3 x (1+ 4 x 3 x 1) = 40 different commands:

1. Stop.
2. Find John.
3. Find a robot.
4. Find the robot.
5. Find that robot.
6. Find this robot.
7. Find a green robot.
8. Find the green robot.
9. Find that green robot.
10. Find this green robot.
11. Find a red robot.
12. Find the red robot.
13. Find that red robot.
14. Find this red robot.
15. Go to John.
16. Go to a robot.
17. Go to the robot.
18. Go to that robot.
19. Go to this robot.
20. Go to a green robot.
21. Go to the green robot.
22. Go to that green robot.
23. Go to this green robot.
24. Go to a red robot.
25. Go to the red robot.
26. Go to that red robot.
27. Go to this red robot.
28. Follow John.
29. Follow a robot.
30. Follow the robot.
31. Follow that robot.
32. Follow this robot.
33. Follow a green robot.
34. Follow the green robot.
35. Follow that green robot.
36. Follow this green robot.
37. Follow a red robot.
38. Follow the red robot.
39. Follow that red robot.
40. Follow this red robot.

The 18 commands using "this" or "that" must be combined with a deictic gesture pointing to an object. In this simulation, we use clicking with the middle mouse button on an object in the

simulated environment. The six commands marked with a "*" are semantic anomalies, referring to "the *np*" when there is more than one. In fact, if a user uses one of these commands, FEVAHR responds "`Which one do you mean?`," and the user may then enter one of the deictic *np*s and point to the intended object.

Focusing on an object is simulated by storing the Garnet representation of the object in the global variable ***STM***, simulating iconic short term memory. The effects of the *Find* commands are to place an appropriate object in ***STM***. The *Go to* commands combine a *Find* with moving the simulated FEVAHR near the simulated object in the simulated environment. After a *Follow* command, the simulated FEVAHR goes to the appropriate object, and then stays near it if the user uses the mouse to move the simulated object. The *Stop* command cancels the *Follow* command, if necessary, and replaces whatever is in ***STM*** with **NIL**.

2.4 Primitive Actions

The primitive actions are *find*, *finddeictic*, *gotofocussed*, *staywithfocussed*, and *stop*. SNeRE, the SNePS rational engine, maintains the association between SNePS nodes representing primitive actions and actual Lisp functions that effect them. These Lisp functions represent the actions at the PM Level. The effects of the primitive actions are as follow:

(**find** ***object-node***) Finds the description associated with the SNePS *object-node*, then finds the Garnet object satisfying that description, and stores that object into ***STM***.

(**finddeictic** ***category-node***) Retrieves the Garnet object that the user points to or has just pointed to with the middle mouse button, and stores it into ***STM***. The actual *category-node* is ignored at this time, so if, for example, the user says "that red robot" and points to John or to the green robot, the object pointed to will be accepted with no complaint. This may be corrected at a later time.

(**gotofocussed**) Uses Garnet routines to move the simulated FEVAHR to a point near the Garnet object that is stored in ***STM***.

(**staywithfocussed**) Uses Garnet routines to move the simulated FEVAHR to a point near the Garnet object that is stored in ***STM***, and then uses the Garnet constraint mechanism to assure that the simulated FEVAHR stays near that other object even if the user moves it with the mouse.

(**stop**) Cancels the constraint on the location of the simulated FEVAHR, if necessary, and replaces the value of ***STM*** with **NIL**.

Both **gotofocussed** and **staywithfocussed** use the function (**near** ***figure ground***), which returns a point that is "near" the ground object, based on the sizes of both objects, and is between the ground object and the center of the room. For example, Figure 2 shows FEVAHR near John when they are in the lower right quarter of the room. shows FEVAHR near John when they are in the lower right quarter of the room.

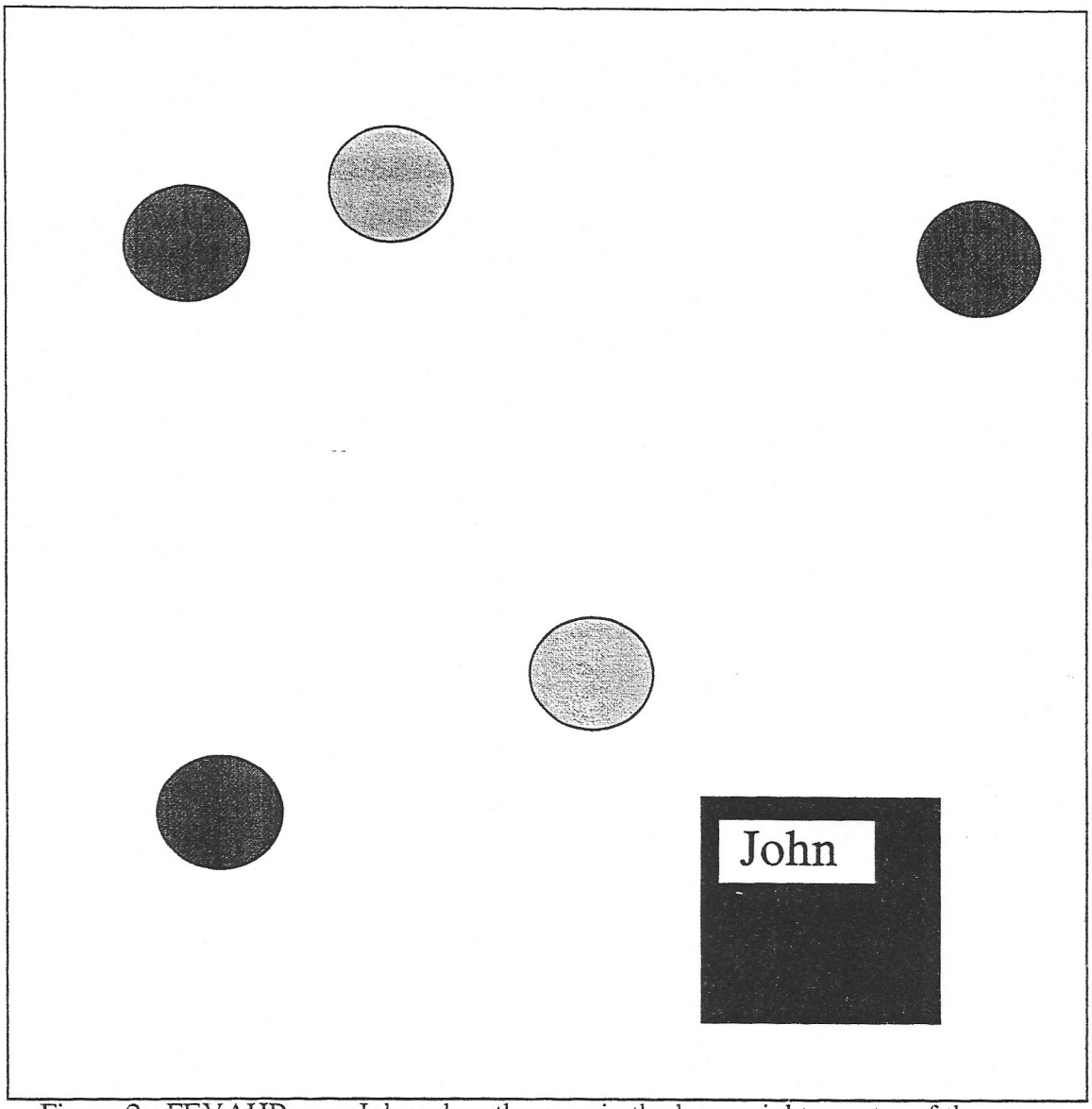

Figure 2: FEVAHR near John when they are in the lower right quarter of the room.

2.5 Complex Acts

The natural language commands *Find* and *Stop* are implemented by the primitive actions *find* and *stop*, respectively.

The natural language command *Go to np* is implemented by the complex act (**go *obj***), where ***obj*** is the node representing the entity denoted by *np*. The complex act (**go *obj***) is performed by first doing (**find *obj***), and then doing (**gotofocussed**).

The natural language command *Follow np* is implemented by the complex act (**follow *obj***), where ***obj*** is the node representing the entity denoted by *np*. The complex act (**follow *obj***) is performed by first doing (**go *obj***), and then doing (**staywithfocussed**).

If an *np* is *this* or *that* followed by some *cat*, the parser calls (**finddeictic *category-node***), where ***category-node*** is the node that represents the category denoted by *cat*. The parser then returns (**recognize *STM***) as the node that represents the denotion of the entire *np*.

The function (**recognize** ***object***) simulates vision by returning the SNePS node that represents the entity simulated by the Garnet object *object*. If that is the object simulating John or the green robot, the same node is returned that would have been if the *np* had been *John* or *the green robot* in the first place. However, if the Garnet object is one of those simulating a red robot, a new node is created and returned, and a belief is entered into the KL that the entity represented by this node is a red robot. This is the same action that would have been performed by the parser if the original *np* had been *a red robot*.