EXPLANATION OF MERCURY, THE VEOS HUMAN INTERFACE

Andy MacDonald

April 1993

Here's the general scoop on VEOS, FERN, and Mercury:

VR is basically a database visualization method. You make a database full of information about different entities in the world, and then display them in one form or another. Database fields could include things like color, picture, size, sound, loudness, etc. Or less physical properties like data values that you could interpret in any way you like. If you can buy into that, then let's move ahead:

VEOS provides the basic database storage and retrieval, and the communications methods between different computers (so you can distribute and share your database).

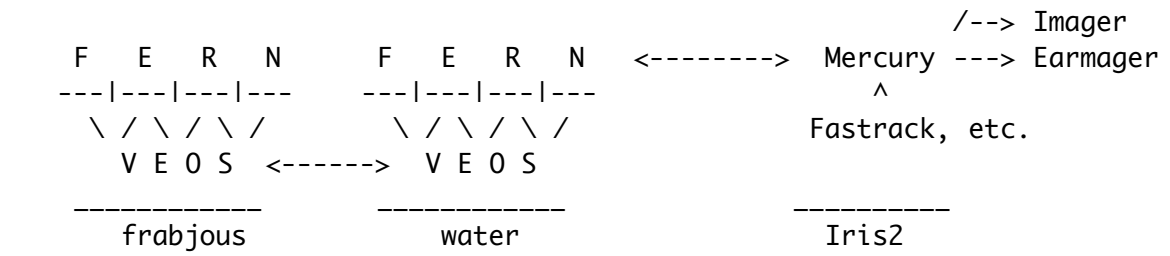
FERN provides the database management and distribution. It creates a common database structure and makes sure that the proper portions are updated with regards to the other entities in the system. Each entity has a part of the database that refers to its own data and a part that refers to data about other entities in the same world (space). To find out about other entities, just look in the proper database portion. To tell others about yourself, just write to the part of the database that automatically gets distributed around and don't worry about it. FERN takes care of all communication and distribution of that data. All you care about is where to read from and where to write. One other function that FERN provides is direct message sending between entities. That way, one entity could tell another entity what to do, provided that the receiving entity has created a method for that to happen. Unlike my example in the FERN paper, now when you want to fly, your wand sends a message to your virtual body to move. So the general database is distributed, but specific info is sent. Still with me?

Now, Mercury is what provides the human interface. In order to display the database to the human, you must take both the data and readings from sensors (polhemus, etc) and display the data to the senses (sound and graphics renderers), right? There have been lisp (i.e. 100% FERN/VEOS) entities that provide this interface to the human in the past, but Mercury takes all the functions necessary for the interface (sensing and displaying) and puts them into one package, written in C, and provides an interface to the database. So there are FERN entities in the virtual world that look to the rest of the world like body parts (head, hand, etc.), but are actually only communicating information to and from Mercury. So Mercury sits and reads the Fastrack about 30 times a second and updates head position and hand position as fast as possible (around 20 fps for a 2500 polygon model using triangle meshes), and gets updates from the rest of the virtual world at about 10 fps. So if you put, let's say, a spinning space needle entity in the world, you'd see the needle updated about 10 fps, but turning your head would happen much

faster, up to 30 fps (for now). And, btw, you could include as many Mercuries in the same world as you like. They're all just dipping into the distributed FERN database. Furthermore, you could build an interface for Mercury into any other world you wanted, as long as it made the proper calls to Mercury, so it is not solely based on VEOS and FERN, though that is the model that lets it be multi-user, etc.

So to hook up UNC and HITL, you'd need to write a FERN entity that would interface with whatever UNC world you're running. If you could just get the proper data sent in the proper format, we'd be in the same world at the same time. And the data that needs to get sent is somewhat minimal if your world is not heavily populated. There'd of course be some lag, but that wouldn't make _that_ much of a difference depending on the world, I'd think. The only other thing would be translating whatever models there are into both yours and our formats. Not that bad, right?

OK, so let me try to do it graphically, and then let you ask me whatever questions you like. Here it is:



So, FERN is talking through VEOS to other "nodes" in the system, in this example, frabjous and water, but it could be any of our machines. The Iris does _not_ participate in this FERN "pool" (as it's now called). A set of entities, in this case on water, know how to run and talk to Mercury and do. Mercury runs on the Iris and talks back and forth directly to these FERN entities. Similarly, FERN entities could be made to interface to other types of processes, like Mathematica, etc.