VEOS/FERN/MERC ONE PAGER

William Bricken

November 1993

The Virtual Environment Operating Shell (**VEOS**) has been iteratively developed at HITL over the last four years, providing high-visibility demonstrations, thesis support and design tools in dozens of projects. VEOS provides a comprehensive and unified management facility and distributed rapid prototyping environment for generation of, interaction with, and maintenance of shared virtual environments. It is UNIX-based, platform independent, and has been extensively tested on DEC, Sun, and SGI platforms. Low-level mechanisms within VEOS are hidden from the user. VEOS 3.0 code will be available for non-commercial purposes as shareware in early 1994.

Within VEOS, the **Kernel** manages processes, memory, and communication on a single hardware processor, using LISP as an interface language. **FERN** manages task decomposition on each node and distributed computing across nodes, using a Linda-like communication model that includes asynchronous, synchronous, and stream interactivity. **SensorLib** provides a library of spatial and body-oriented device drivers. The **Imager** provides hardware-independent stereo rendering for a variety of display techniques. **SpatialSound** is the auditory counterpart of the Imager.

Other systems built at HITL enhance the performance and functionality of the VEOS core. **Mercury** implements a participant system, using distributed processing to decouple the throughput performance of behavior transducers from the complexity of the virtual environment, delivering consistently high display rates independent of number of participants or computational load. **UM** is a generalized relational mapper which provides a simple graph-based interface for constructing arbitrary relations between input signals, state information, and output. The **Wand** is a hand-held interactivity device which allows the participant to identify, move, and change the attributes of virtual objects. **Voice** provides vocal command recognition.

VEOS extends programming metaphors to include first-class environments, biological models, and systems-oriented programming. An **entity** is a coupled collection of data, functionality and resources, which is programmed using a biological/environmental metaphor. Each entity within the virtual world is modular and self-contained, each entity is computationally independent and autonomous. Entities provide functions that define perception, action and motivation within a dynamic environment. **Perceive** functions determine which environmental transactions an entity has access to. **React** functions determine how an entity responds to environmental changes. **Persist** functions determine an entity's repetitive or goal-directed behavior.