VEOS: PRELIMINARY FUNCTIONAL ARCHITECTURE

William Bricken
April 1990

Virtual interface hardware sensors, such as the head-coupled unit, hand sensors, joysticks, yolks, and other devices, provide a digital signal which is generated by the physical actions of a participant (user) in a virtual environment. VEOS is the software substrate which mediates the interface between the symbolic computation generating the virtual environment and natural behavior of the participant within the virtual environment.

FUNCTIONALITY

The primary subsystems of the Virtual Environment Operating System (VEOS) include:

Interpretation
: couples the input activities of the participant to computational processes.

Modeling
: manages the computational behavior and model of virtual environment elements.

Display Integration
: integrates output signals from the model and from other sources to drive the display devices which create the virtual environment.

These three subsystems will perform the following functions:

The Interpreter

receives sensor signals,
screens errors,
initiates error correction and negotiation with participant,
integrates input into a common model representation, and
bypasses the model subsystem for signals that require rapid display,
do not interact with the model, or have specialized hardware.

The Virtual Environment Model

maintains and coordinates the representation, processing activity, and interaction between model elements,
manages the long term memory, library access, history storage, and database access associated with models,
manages the process allocation and load balancing between models and available hardware processing resources,
coordinates the connectivity and model consistency between multiple participants.

The Display Integrator

receives standardized output from the modeling subsystem,
receives specialized output from model-independent hardware,
receives error messages from the Interpreter,
converts signals into a participant-centered viewpoint,
controls viewpoint, perspective, and other hardware generated display functions, and
integrates multiple sources of environmental images (such as video, videodisc, digital libraries, and physical reality).

Our design philosophy is to assign a virtual processor to every distinct object in the virtual environment. Objects are organized hierarchically, with empty space itself as the root node. The architecture of the VEOS itself is the same as the architecture of every object within the virtual environment. Therefore, the properties of the operating system can be inherited by each object. This morphism of processing capabilities provides a unified basis to support arbitrary distribution of available processing power.

Internal Architecture

The primary organization within each modeled object is an input-process-output loop. Input is identified by the object's sensors, which themselves are subsystems. Output is defined by effectors. The structure of each object/system consists of:

Input buffer	(fed by sensors)
Priorities	(the internal value system on input)
Disposition	(rules triggered by selected input)
Knowledge	(state collected by rules)
Output buffer	(actions generated by rules)

Sensors store input in a buffer. A set of rules for attention select a single input item to compare to the trigger clauses in the set of disposition rules. When a particular rule is matched, the action it specifies is carried out. An action may be to store the input as knowledge or to cause an effector to change the state of the environment. Some rules may be contingent on stored

knowledge to be triggered. Some rules may be independent of input, they form the internal processing disposition of the system.

This architecture allows situated responses. Objects can react to environmental changes (as perceived by input sensors and filtered by priorities) dynamically and opportunistically. Objects can also learn from experience and internally abstract experiences to form idiosyncratic knowledge bases. When rules include inference over the knowledge base, an object can behave as an expert system.

Objects can be programmed by inserting rules into their disposition. The process of programming can be linked to voice commands. A voice command (such as "Add mass to this book") can trigger in the target object a metarule to insert a rule into the disposition or the knowledge base. The template for the ADD metacommand constructs a knowledge slot, queries for associated values and for authorization to change, and constructs appropriate externally defined sensors and effectors. The idea is to permit interactive, intuitive programming of systems within a virtual environment.

In summary, the software architecture of the VEOS and of virtual objects incorporates rule based logic programming locally in object-oriented autonomous systems. Programs are reactive and situational (data-driven) when internal priorities are satisfied by existing input, and autonomous and learning (goal-driven) when there is not prioritized input. Disposition rulebases are programmable by other systems (in particular by the participant).

Software Tools

The VEOS will provide a wide range of software tools for construction of and interaction with models. Our intent is to provide primitives which permit the design of environments with arbitrary characteristics. We hope to be able accommodate arbitrary predicate calculus theories and non-standard experimental logics (temporal, modal, situational) by changing inference engines within the VEOS. We are not addressing the development of the theories themselves, rather we wish to provide an empirical environment for theory exploration. The range of model programmability includes:

- object construction and editing
  - spatial enumeration
  - assembly of primitives
  - constructive solid geometry
  - boundary models
  - sweeps
  - properties
  - constraints
  - rules
  - relations

space construction and editing
	global properties (gravity, laws of motion)
	topological functions and deformities
	coordinate systems
	spatial construction capabilities
	global origins and world viewpoints
	edges, backdrops and scenery
	cleave objects out of background

abstraction construction and editing
	hierarchical composition
	process model and computational semantics
	abstraction rules, functions and relations
	abstract definition (generation, uniqueness, structural)

viewpoint control
	glide through
	scaling
	jack-in (teleport)
	multiple concurrent viewpoints (stereo)
	projection

boundary integrity
	collision detection
	contact maintenance
	joint articulation
	surface travel
	sensory ports
	personal space
	boundary self-maintenance

multiple concurrent participants
	initialization of common worlds
	uniqueness enforcement
	consistency maintenance and contradiction partitioning
	negotiation of mutual relationships

inhabitation
	sensor and effector mapping
	communication ports and input/output maps
	functional constraints

control of time
	parameterized animation speeds
	variable rate parallel clocks
	parameterized display rate and lag
	history
	time-tagged communication
	instant replay and backtracking

display and resource control
    successive display abstraction (wire-frame, hidden line
        removal, shading, texture, photorealism)
    variable polygons within a region
    foveal refinement
    distance object filters
    allocation of processors
    simulation interweaving

internal processes
    priorities
    computational and inferential mechanisms
    pattern matching
    constraint management
    short and long term memory
    behavior and experience storage
    hypothesis testing and information seeking

history and statistics
    store experience (categorical, ordinal, time-stamped)
    generalization
    classification
    distribution statistics
    correlational statistics
    trend analysis

HIGH-LEVEL TOOLS

Three examples of tools constructed from the VEOS functionality follow:

The Wand

The Wand is an interface tool which uses a simple physical device for a wide range of functions. The physical device is a rod with a 6 degree-of-freedom sensor on one end which supplies position and orientation information to the model. The sensor information inhabits a virtual rod held by a virtual hand. We assign functionality to the Wand by attaching a voice sensor to it and inserting rules into its set of dispositions. The general form of the response rule is "if recognize-input then generate-associated-output". Some functions of the Wand include:

Ray on/off:
    A ray emanates from the end of the virtual rod, collinear with it.

Identify:
    The first object which the ray penetrates returns its name.

Distance:
The length of the ray vector, expressed in the metric of the intervening space, is returned.

Connect:
Construct a communications port between the rod and the identified object.

Jack:
Teleport the viewpoint of the rod (along the ray vector) to the identified point on the object.

Grasp:
Attach the end of the ray to the identified object. When the Wand is moved, the object stays attached. When the Wand is rotated, the object rotates.

Normal:
Rotate the identified object so that the intersecting ray is normal to the object's surface.

Sight:
Jack into the Wand; the viewpoint of the participant issuing the command is linked to the ray vector.

Move faster/slower:
Move the viewpoint of the participant along the ray vector.

The Virtual Body

Since the participant is included within the virtual environment, the representation of self is fundamental to virtual interface design. The Virtual Body is the primary reference point, the interface between the user and the virtual environment. It provides direct access to computational graphic objects; it is the channel of direct action and control. Monitoring the Virtual Body provides the computational system with a complete record of actions taken by the participant.

The Virtual Body is a software toolkit for

-- attaching arbitrary hardware input devices to arbitrary representations of components of our body. Usually the linkage will emphasize naturalness.

-- making psychometric measurements of behavior in a virtual environment, and

-- maintaining coherence between the participant's model of physical activity and the virtual representation of that activity.

The unique aspects of the Virtual Body are:

1. The sensor measuring participant activity is designed to be transparent. Natural physical movement directly affects the computation; there is no apparent interface. The Virtual Body software maintains the illusion of direct interaction.

2. Mapping between physical action and computational effect is flexible and dynamic. A spoken word, for example can change the computational effect of shifting one's gaze from "Identify that object" to "Transport me to that object."

3. Physical actions in a virtual environment furnish psychometric data on performance, resource expenditure (load), and cognitive model. Since action can be taken literally (there is no symbolic transcription filtering the meaning of behavior), performance in a virtual environment mimics performance in reality. As long as the representation of the task is valid, the user's behavior directly indicates the user's ability to perform.

Components of the Virtual Body

Sensors are physical devices which sense natural movement. Sensors furnish a data stream to the computational environment. Sensors that are under development include:

Head tracking,
Eye tracking,
Voice recognition,
Hand movement (right and left),
Body movement(torso, arms, legs),
Touch locales (fingertips).

The Virtual Body includes software for:

-- mapping this data stream to representations of body components in a virtual environment display,

-- interpreting the data stream as instructions to change the virtual environment, and

-- collecting and analyzing the data stream as psychometric information.

SIMSTAT, an empirical programming environment

SIMSTAT is a virtual environment which provides statistical design and analysis fully integrated into a general simulation capacity which incorporates the scientist as a participant. During a physical experiment, the participant can enter the simulation of the experiment. The virtual display is driven by data streams which may originate from mathematical models, from other processes within the simulation, or from the physical experiment itself. The virtual environment is fully under the control of the designer, so that experimental factors can be varied with precision. The history mechanism within each object generates a stream of behavioral data which is linked to a general statistical analysis package. Correlational analysis is achieved by linking two objects to a common clock which time-stamps their respective behaviors.

SCHEDULE

Year 1: Development of VEOS for independent objects, prototype of interpretation and integration modules, simple models, viewpoint control, Wand and Virtual Body tools.

Year 2: Hierarchical objects, object, space, and abstraction editors, prototype for multiple concurrent participants, inhabitation.

Year 3: Complex models, boundary integrity, integration of several input devices, prototype parallelism.

Year 4: Display, resource, and time control, history and statistics, large world models.