EMBEDDING MATHEMATICS WITHIN A VIRTUAL WORLD

William Winn and William Bricken
January 1993

ABSTRACT

This project will provide for the first time an assessment of the effectiveness of direct interaction with abstractions in virtual environments for teaching mathematical problem solving. The goal is to embed the symbolic processes of algebra into the behavior of virtual objects. The project will develop virtual worlds which support abstract understanding without requiring symbol manipulation, thus providing an opportunity for problem solving through natural interaction. Two aspects of virtual environment design will be emphasized: embedding the interactive mathematical functionality into the behavior of virtual objects, and embedding pedagogical strategies. *Embedded mathematics* means that virtual objects know their correct computational behavior and can communicate it. *Embedded pedagogy* means that interactive virtual objects embody a teaching strategy in their behavior. The substitution of observable behavior for symbol manipulation encourages constructive interaction as a basis for student learning. The project will actively develop software design tools and virtual world experiences with teachers, and evaluate the utility of this approach for standard curricula. In general, all software embedding tools can be made available to the student within a virtual environment, providing constructive educational interaction with both definitional and transformational aspects of embedded concepts.

PROJECT SUMMARY

The goal of this project is to explore the effectiveness of virtual environments for teaching mathematical thinking, in particular, algebraic problem solving. The project will develop virtual worlds which support abstract understanding *without requiring symbol manipulation*, thus providing the opportunity for problem solving through natural interaction within a virtual environment. Two aspects of virtual environment design will be emphasized: embedding mathematical responsivity into the behavior of the virtual world, and embedding pedagogical strategies. *Embedded mathematics* means that virtual objects know their correct computational behavior and can communicate it. Further, the appearance, attributes, and behavior of the virtual object support its intended semantics. A virtual shopping cart, for example, might know how to add the costs of its contents. If the accounting is intended to keep a balanced budget, then the cart might present the consequences of purchasing its contents as the current balance in a (virtual) checkbook. *Embedded pedagogy* means that the interactive virtual objects embody a teaching strategy in their behavior. The cart might voice a warning to the user when it contains more than the checking account supports. The substitution of observable behavior for symbol manipulation encourages constructive interaction as a basis for learning. A student determining how

many 15-cent avocados can be purchased for a dollar can use the virtual cart as a tool (an incremental adder, for example), rather than backing into the problem abstractly by solving the equation 15x=100. In general, all software tools for constructing embedded behavior can be made available to the student within a virtual environment, allowing students to call upon existing definitions, axioms, and transformations (functions and rules) to constructively modify situations into simpler solutions.

Worlds will be constructed using the innovative VEOS software for virtual environments, which includes situated environmental/biological modeling and programmable entities. VEOS provides a substantive infrastructure (such as process coordination, communication, database management, sensor integration, graphics output management, platform compatibility, virtual body performance optimization, and the application programmers interface) which removes several layers of software complexity from the development of embedding tools.

Software design tools and virtual world experiences will be developed actively with teachers, and evaluated in the context of standard curricula. Data will be gathered on student performance in the virtual world, the extent of transfer to similar tasks in the physical world, and improvement on standardized tests. The project therefore will provide for the first time an assessment of the effectiveness of direct interaction in virtual environments for conveying understanding of mathematical problem solving.

PROJECT DESCRIPTION

We plan to explore how the software tools of virtual reality might help students who have trouble learning symbolic mathematics, by constructing both responsive environments and empowering tools. Our strength is in the project team's deep experience with creating educational virtual environments and the supporting software tools. We intend to assess the transfer effects of embedded mathematics for problem solving in classroom environments. Our contribution to Mathematics Education will be a generic set of software tools (independent of specific display technologies) which empower teachers and students to construct responsive pedagogical environments.

Embedded Symbolic Manipulation

Traditionally the domain knowledge of science and mathematics is represented *symbolically.*, and taught through manipulation of algebraic equations. The symbol systems taught in school exhibit many difficulties: they are difficult for novices to understand; they are abstract rather than contextually anchored; the operational meaning of many symbols is often unclear, ambiguous and context dependent; and mastery of a symbol system is usually confused with mastery of the system's referential content.

Virtual environments permit symbolic processes to be hidden from a participant, while the behavior of objects in the virtual environment provides direct evidence of the consequences of those symbolic processes. A virtual ball, for example, programmed to obey the laws of motion, will travel along the expected trajectory, mimicking physical behavior. The computation of the trajectory is hidden but evident, it is *embedded within the virtual object*. In the virtual world, however, the ball's simplest behavior is not in accord with the laws of motion. The non-programmed ball, having no substance, sticks to the exact spot of its release. Inertia, gravity, and force must be programmed in as symbolic computations. This apparent weakness, that virtual environments are heavily dependent on symbolic computation, is also a strength, for it permits complete computational control over observed behavior. In particular, we can program virtual worlds so that the behavior of objects in the environment embeds both symbolic and pedagogical processes. Students could learn literally though direct experience of behavioral consequences (in simulated environments with embedded symbolic processes).

By embedded computation, we mean more than interaction with a process. Embedding implies natural interaction with realistic (or at least naturalistic) objects in an environment, objects which react and respond to environmental perturbations (such as a participant's query), objects which met the expectations of interacting agents. Embedding is semantic rather than syntactic; interaction is well-intended rather than well-formed.

A handheld calculator embeds numerical computation, it's addition button performs as expected. The keypad enforces well-formed input by limiting transactions to keyed operations on digits. But people have difficulty sequencing complex numerical calculations on a calculator because it does not embed computational strategies, teaching techniques, or naturalistic language domains. Thus, the simple calculator provides a very weak embedded functionality. Stretching an environmental concept, we might say that embedded capabilities are the affordances of an object, they are provided by the object as part of its interactional character.

A computational interface has *natural semantics* when the display has a meaning that does not require explanation. Consider a house. A textual description requires reading skills, a procedural database (lists of coordinates) requires decoding, a picture can be recognized immediately but is not interactive. A house in a virtual environment is most like a physical house, you can walk in the front door and explore each room while looking around. Transformations can be expressed as natural behavior. Opening a window, for example, can be achieved by grabbing it and pulling it open, rather than by editing the symbolic attributes of the window to change the values that represent the location of the window. A virtual house has natural semantics, no one needs to explain it. Natural semantics is what a child learns before symbolic schooling. Most sciences have natural semantics, most symbolic studies (the three Rs) do not.

Embedded mathematics presents symbolic computational processes as behavior with natural semantics. Virtual objects are constructed so that they perform both the physical and the computational behaviors expected of them. Virtual objects which embedded mathematics can include word processors, checkbooks (the spreadsheet application), lumber (which could generate its own bill of lading when assembled into a house), most scientific visualizations (such as pollution clouds, deep-sea rifts, four-dimensional cubes, and galactic collisions), and intelligent software agents.

The thrust of this proposal is to embed more abstract concepts, such as the operation of mathematical division, into naturalistic object interactions. Students could compute an average, for example, by placing a set of objects in the same Averager basket. The basket itself would enforce identification of the metric being averaged, by either deduction, by action such as rejecting non-conforming objects, or by interaction with the student. Interactional style is also embedded within objects, with the same configurability (using deduction, action, and interaction).

More generally still, software tools which permit programmers to define embedded symbolic and pedagogical behaviors, *world-building tools*, can be provided to students as they learn. With the ease of interface provided by virtual environments, students can construct their own problem solving tools dynamically while interacting with the problem as manifest within the virtual environment. Thus, virtual environments provide a context for the development of meta-skills, generalized tools and strategies for solving classes of problems.

To assist the student in construction of problem-solving skills, we propose to *embed pedagogy* into the behavior of virtual objects.

PROJECT GOAL

We intend to demonstrate that students can learn abstract mathematical content in virtual environments without first having to master a symbol system. We also propose to show that after the concepts and rules of an embedded domain have been understood, the symbol system is learnable, meaningful, and can be used to represent the content of a domain and to communicate it to others.

The project's specific objectives follow:

1. To develop a virtual world that embodies algebraic rules in the behavior of objects, and embodies pedagogical strategies for making those rules accessible to a participant. This requires that we:

    a. Develop software tools which permit embedding algebraic and rule-based behavior in virtual objects.

b. Develop modeling techniques for incorporating designed educational problem solving experiences in virtual environments.

c. Develop pedagogical strategies that will lead to the understanding of mathematical concepts through experience in virtual environments.

d. Identify appropriate methods (which may not rely on symbolic abstraction) for assessing student comprehension of mathematical concepts and their transfer from the virtual to the real world.

2. To provide "proof of concept" that direct experience in designed virtual environments can lead to conceptual learning, without mediating symbolic representations. And to validate that mathematical content can be learned by non-symbolic techniques.

3. To conduct the project in close collaboration with teachers and students in classrooms.

4. To develop the first tools for empowering virtual world participants for the construction of conducive digital learning environments.

ANTICIPATED OUTCOME

The project will have two types of outcome: data and products.

Data

The data gathered in the project will either provide evidence that embedded mathematics and pedagogy is effective in improving students' understanding of abstract content or will furnish reasons why it is not. Because we expect this approach to be more effective with students who have difficulty with symbolic computation, we anticipate gathering correlational evidence for differential effects on students with different types of mathematical ability (spatial, symbolic, logical, kinesthetic, ...). We also anticipate gathering information on the ways in which learning from virtual environments impacts the classroom, and whether teachers can integrate these techniques into classroom activities.

Products

The project will allow us to create functioning virtual worlds in which students construct knowledge of mathematics. Effective pedagogical strategies for use with virtual environments will also be developed, as well as techniques for assessing their effectiveness. We will extend the rich base of virtual world programming tools developed over the last four years at HITL, customizing them for use in educational applications. These tools will serve as an infrastructure for the development of a non-symbolic curriculum in mathematics.

DIFFICULTIES OF LEARNING SYMBOLIC ALGEBRA

The difficulties children have when they begin to learn algebra are well documented (Bricken, 1987; O'Shea, 1986; Zehavi & Bruckheimer, 1984; Gerace & Mestre, 1982; Sleeman, 1984). Thwaites (1982) found that students are often baffled by algebra's non-visual nature, its apparent arbitrariness, its complexity, and how problems are expressed through its symbols. Students often do not understand what variables are, how letters are used to represent them, or how equality can be used (Rosnick, 1981; Kaput 1978; Bernard & Bright, 1982).

If students fail to understand algebraic representation, then the only way they can solve algebra problems is by the rote application of procedures they have memorized. This memorization is brittle, often both over- and under-generalized, and elaborated by motivations independent of the content of algebra (Bricken, 1987). Unfortunately, the rule-based approach to algebra is the one that is often taught, even though this does not promote the development of good conceptual models of algebra (Thwaites, 1982; Bright, 1981; Bernard & Bright, 1982; Greeno,1985).

Many attempts to address symbolic abstraction make it concrete through the use of manipulables, a positive approach supported by many experimental studies (Sowell, 1989). The symbols of algebra are typically reified either in physical objects or in computer representations or simulations of problems and problem-solving. Thus, showing numbers as sets of objects that students can actually count makes it easier for students to solve arithmetic word problems (Lindvall, Tamburino, & Robinson, 1982). Arranging numbers to multiply into two-dimensional matrices allows students to compute the answer by counting the cells in the matrix (Carrier, Post, & Heck, 1985). In the area of equation solving, we find studies of the effectiveness of using pan balances (Austin & Vollrath, 1989), spreadsheets (Watkins & Taylor, 1989), computer-based graphing (Waits & Demana, 1989), and a variety of other strategies (Shumway, 1989).

We propose to develop virtual worlds with objects which both obey the rules of mathematics and can be used as tools for problem solving. The flexibility of virtual objects allows us to transfer our semantic intentions more easily, to make them look and act like we expect. Virtual objects are situated in virtual environments, permitting us to design contextual cues, object interactions, and circumstantial affordances to assist meaning making.

As an example, imagine a student trying to solve the standard two unknown problem of the ages of two people, when the sum and product are known (say, $x+y=22$, $xy=120$). The computation can be approached by algebraic symbol manipulation, by geometric construction, by typing the equations into a symbolic processing program such as Mathematica, or by reasoning. Embedding mathematics in a virtual world provides automated symbolic, geometric, and computational solutions, but each of these presumes skill. Non-symbolic learners may not use abstract constructions, they may, for example, ask to see

all the pairs of integers that add to 22. They may then discover duplicate pairs due to the commutativity of addition. So they request to remove the duplicates. The next request is for each of these pairs to be multiplied. The pair (there might have been more than one) that creates the expected product is the desired solution.

Here, the learner prescribed a computational algorithm, but one that is not generally available in physical reality or in software without embedded mathematics. The virtual environment assists the search for a solution by removing all symbolic burden. Not only is computation and simplification immediate, generation, filtering and search of large sets of instances is also facilitated.

Embedded mathematical worlds allow students to interact with the objects and concepts of mathematics in natural, intuitive ways and perhaps therefore to construct understanding of abstract concepts more easily. Rather than trying to predict or to design or even to evaluate the best solution to a problem in a virtual world, we propose to provide software tools which empower students and teachers to design their own best solution tools to whatever problems they encounter, and to provide at least prototypically good examples of designed worlds for facilitating problem solving techniques.

RATIONALE FOR LEARNING IN VIRTUAL ENVIRONMENTS

Our expectations for the success of VR to help students learn abstract material, and the guidelines that direct our design of virtual environments, are based on the theories of constructivism, multiple intelligences, and representation. Constructivism leads us to expect that first-person experience will create deeper understanding of abstract concepts. Multiple intelligences suggest that learning has many pathways and many modalities. The theory of representation provides us with display and interaction techniques which remove the disassociation of abstract symbols while maintaining formal semantics for computation.

Non-symbolic representation

Virtual environments finesse the semantics/syntax barrier by providing the representational flexibility to construct symbols which reflect meaning as well as abstract patterns (W. Bricken, 1991). Interaction is direct and not mediated by an interface (M. Bricken, 1991). Objects in algebraic relationships can quite literally be picked up and moved around algebraically, while the student observes the consequences (Winn & Bricken, 1992).

Multi-modal learning

The comprehension of abstract rules and concepts can be significantly improved when represented by several sensory modalities. Animated computer graphics and graphs make it easier to learn concepts and rules in Chemistry (Kozma ,

Russell, Jones, Marx & Davis, 1993). Adding sound effects to graphic representations improves comprehension of complex data sets (Scaletti & Craig, 1991). Spatial display and interaction are particularly important (Larkin and Simon,1987; Paivio,1971, 1983; Winn, Li, and Schill, 1991). Inclusive environments have been shown to be emotionally involving and extremely easy to use (M. Bricken, 1991a; M. Bricken,1991b).

Knowledge construction

Students construct their own meaning by interacting with material rather than being taught something explicitly (Bransford, Sherwood, Hasselbring, Kinzer, & Williams, 1990; Cognition and Technology Group, 1991; Scardamalia, 1991; Spiro & Jehng, 1990; Spiro, Fel tovich, Jacobson, & Coulson, 1991) To specify a particular content organization or instructional strategy is counterproductive. These techniques are exemplified by the hypermedia system developed by Spiro and his colleagues (Spiro, Feltovich, Jacobson, & Coulson, 1991; Spiro & Jehng, 1990) that lets students learn problem-solving through the exploration of ill-structured domains such as literary criticism, military strategy, and cardio-vascular medicine; and by the interactive videodisk materials developed by Bransford et al. (1990) and the Cognition and Technology Group at Vanderbilt University (1991) that facilitate the solving of complex mathematics problems by allowing children to interact with dramatically presented adventures.

Some key theoretical elements behind constructivism are contained in Spiro's Cognitive Complexity Theory (Spiro, Coulson, Feltovitch, & Anderson, 1988; Spiro & Jehng, 1990). Cognitive Complexity Theory proposes a number of strategies to promote the acquisition of flexible knowledge. For the purposes of this project, the most important is that students "revisit the same material, at different times, in re-arranged contexts, for different purposes, and from different conceptual perspectives" (Spiro, Feltovitch, Jacobson, & Coulson, 1991).

SOFTWARE TECHNIQUES

The motivation to build a virtual environment interaction toolkit for mathematics education includes

- control of software complexity,
- an interface based in natural behavior and multiple intelligences,
- enhancement of native human abilities of spatial understanding and experiential learning,
- support of concurrent multiple participants and cooperative work, and
- direct, non-symbolic communication.

We currently use computers as symbol processors, interacting with them through a layer of symbolic mediation. The computer user, just like the reader of books, must provide cognitive effort to convert the screen’s representations

into the user's meanings. Virtual environment software systems, in contrast, provide interface tools which support natural behavior as input and direct perceptual recognition of output. The idea is to access digital data in the form most easy for our comprehension; this generally implies using representations that look and feel like the thing they represent.

Immersive environments redefine the relationship between experience and representation, in effect rendering the syntax-semantics barrier transparent. Reading, writing, and arithmetic are hidden from the computer interface, replaced by direct, non-symbolic environmental experience.

Virtual environment software attempts to restructure programming tools from the bottom up, in terms of *spatial, organic models.* The primary task of a virtual environment operating system is to make computation transparent, to empower the participant with *natural interaction.* The technical challenge is to create mediation languages which enforce rigorous mathematical computation while supporting intuitive actions such as talking, walking, pointing, grabbing, and abstraction (tool creation and problem solving). The design goal for natural interaction is simply *direct access to meaning*, interaction not filtered by a layer of textual representation. This implies both eliminating the keyboard as an input device, and minimizing the use of text as output.

Taxonomies of the component technologies and functionalities of virtual environment systems have only recently begun to develop (Naimark, 1991; Zeltzer, 1992; Robinett, 1992), maturing interest in virtual environments from a pre-taxonomic phenomenon to an incipient science. Ellis (1991) identifies the central importance of the environment itself, deconstructing it into content, geometry, and dynamics.

More comprehensive overviews have been published for VR research directions (Bishop *et al.* 1992), for software (Zyda *et al*, 1993), for system architectures (Appino *et al*, 1992), for operating systems (Coco, 1993), and for participant systems (Minkoff 1993).

Entities

The VEOS programming model is based on entities. 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 can function independently and autonomously. The biological/environmental metaphor introduced in VEOS is unique, originating from the artificial life community (Langton, 1988; Meyer & Wilson, 1991; Varela & Bourgine, 1992); it is a preliminary step toward providing a programming development environment for modeling and interacting with concurrent autonomous systems within an inclusive environment (Varela, 1979; Maturana & Varela, 1987).

The organization of each entity is based on a mathematical model of inclusion, permitting entities to serve as both objects and environments. Entities which *contain* other entities serve as their environment; the environmental component of each entity contains the global laws and knowledge of its contents. From a programming context, entities provide an integrated approach to variable scoping and to evaluation contexts. From a modeling point-of-view, entities provide modularity and uniformity within a convenient biological metaphor. From an educational point-of-view, entities permit modeling of natural and abstract phenomena with customized behaviors and "personalities" that can interact directly with each student in an individualized manner.

The functions normally included within (fern-entity...) define the following characteristics:

- **attributes:** (fern-put-boundary-attribute...) Properties which are associated with state values are constructed within the entity's boundary resource. Examples include position, color, sound, mass, and name.

- **workspace:** (fern-put-local...) Local memory and private workspace resources are reserved within a local partition of the database.

- **behavior:** (fern-define-method...) Methods which define an entity's response to the messages it receives are defined as functions which are evaluated within the local context.

- **processes:** (fern-persist...) Persistent processes within an entity are defined and initialized. An entity can engage in many processes which timeshare an entity's computational process resources.

- **perceptions:** (fern-perceive...) When specific changes occur in an entity's environment, the entity is immediately notified, modeling a perceptual capability. An entity can only access external data which it can perceive.

- **peripherals:** (sensor-init...) Connections to any physical sensors or input devices used by the entity are established and initialized.

- **functionality:** (define <function-name>...) Any particular functions required to achieve the above characteristics are defined within the entity's local context.

Complexity in Information Systems

A central design issue is the development of a computational infrastructure that supports the complexity of non-linear information systems while providing a seamless interface to the human attempting to understand the inherent

complexity of these systems. The rules of elementary algebra, for example, permit the construction of tremendously large useful systems, such as electricity distribution networks and VLSI microprocessors. Present day information environments call upon visualization, audio cueing, kinesthetic feedback, and the full dynamics of *participation within the complex environment.*

Complex natural phenomena and networked information environments can both be characterized as having large numbers of different varieties of "agents" (objects capable of storing, processing, and transmitting information) interacting with and modifying each other, with no single agent having complete knowledge of all of the rest of the agents. These agents are, in general, semi-autonomous and relatively simple, they may represent firms in an economy or antibodies in the immune system or quantum states in a lattice of atoms. The entity model implemented in the VEOS software was developed to place agent architectures at the heart of virtual environment control regimes.

APPENDIX
Software Infrastructure Tools for Constructing Embedded Educational Environments

We propose to build upon the VEOS tools (particularly the construction tools) to create toolkits for the construction of virtual environments with embedded behaviors that are easy-to-use by students and teachers.

Many of these tools are independent of display medium, sensory bandwidth, and input devices, insuring their utility in multimedia, windowing, and monitor-based contexts. The tools are classified into three general categories:

interface (physical to virtual)
construction (of virtual environments)
environmental (design and programming)

INTERFACE TOOLS

Mercury, The Virtual Body Mercury is a participant's interface to the virtual world (Minkoff, 1993). Mercury monitors the sensors that track the user's physical state, sends this information to the database, and displays the data in the database appropriately.

SensorLib SensorLib is a library of device pollers for VR sensors, such as 6 degree-of-freedom tracking devices (Polhemus, Logitec, Ascention), joysticks (GEO-Ball, SpaceBall), behavior sensors (BioMuse, MIDI instruments, Wand), and other VR input devices.

Imager The Imager is a fast graphical rendering system which provides consistent, hardware-independent imaging. The Imager handles the details of stereo rendering for both head-mounted, fully inclusive and liquid-crystal shutter displays, and can also render into an on-screen window.

SpatialSound The Sound Renderer is the auditory counterpart to the Imager. It provides spatially localized (3D) sound to virtual environments. The Voice system recognizes isolated speech, using Natural Language Understanding.

The Wand The Wand is an interface tool which uses 6 degree-of-freedom sensor on a rod to supply position and orientation information. The sensor information inhabits a virtual rod held by a virtual hand.

CONSTRUCTION TOOLS

UniversalMapper The UniversalMapper provides a graphical means of specifying entity behaviors. Using this system, we program entity attributes by drawing graphs that specify relationships between attributes.

EntityEditor This tool permits a designer to specify both form and behavior of entities. The editor will provide form-based templates (later extendible to behaviors in the virtual environment) for specifying entity attributes, workspace (local memory), behavior (methods), processes (persistent and reactive behavior), perceptions, peripherals (associated physical inputs), and functionality (local functions which support the maintenance of characteristics).

SpaceEditor The SpaceEditor provides first class modeling and dynamics tools for environments. Environments consist of a space and objects within that space. Characteristics of every object within a space can be abstracted to be characteristics of the space itself. Other properties of space include metric structure (grids, orderings, sets, reals), gradients (gravity, wind, electromagnetic forces), continuity, grain-size and coordinate systems.

DynamicsToolkit A software library of dynamical functions that enable the simulation of Newtonian motion, with the type of motion determined interactively by the user at run-time. The implementation of extremely efficient numerical algorithms will permit the simulation of complex systems (systems with large numbers of degrees of freedom) in real time. This toolkit also incorporates collision detection, surface contact maintenance, and joints.

DigitalStudio Humans use stories to structure their interaction with knowledge. DigitalStudio has the goal of providing story telling tools (characterization, dramatic tension, plot, voice) within a dynamic, interactive virtual environment. It includes a minimal behavioral vocabulary, timing and synchronization primitives, and a scripting language.

BoundaryLanguage The BoundaryLanguage project has as a goal to provide a functional experiential (visual, auditory, tactile and behavioral) programming language. We have been able to demonstrate that foundational mathematics (logic, integers, algebra and sets) can be expressed concretely, using 3D arrangements of physical things, such as blocks on a table, doors open or shut, rock walls that respond to gravity, the things of everyday life.

Boundary mathematics provides a formalism for the fundamental concepts of VR. Applied to particular virtual worlds, boundary mathematics provides spatial logic, a formal technique for embedding control structure in space rather than in tokens. Applied to programming, boundary mathematics provides a visual programming language. The formal foundations provide structure for computational implementations, identify tools for description and composition, and suggest techniques for interactivity (Bricken & Gullichsen, 1989; Bricken, 1993).

APPLICATIONS

VEOS was developed iteratively over three years, in the context of prototype development of demonstrations, theses and experiments. It was constantly under refinement, extension and performance improvement. For the purposes of embedding mathematics and pedagogy, the following applications suggest the range of design choices that has already been established.

Tours Tours allow a participant to navigate through an interesting environment. Examples of tours built in VEOS include a configurable (and flyable) aircraft, a topographically accurate replica of the Seattle area, and a metropolitan transit vehicle.

Physical Simulation Physical simulations are challenging since they require very precise control of the computation. Coco and Lion (1992) implemented a billiard ball simulation with eighteen dynamically interactive entities.

Multiparticipant Interactivity Block World allows four participants to independently navigate and manipulate moveable objects in a shared virtual space. Catch lets two participants play catch with a virtual ball while talking with spatially-localized voices. The Catch application emphasizes independent participant perceptions. Participants customized their personal view of a shared virtual environment in terms of color, shape, scale, and texture. Although the game of catch was experienced in a shared space, the structures in that space were substantively different for each participant.

Manufacturing Karen Jones (1992) developed a factory simulation application which provided interactive control over production resources allocation.

Spatial Perception Daniel Henry wrote a thesis on comparative human perception in virtual and actual spaces (Henry, 1992). He compared the subjective perception of size, form, and distance in both a real gallery and its virtual model.

Scientific Visualization Many applications have been built in VEOS for visualizing large or complex data sets, including satellite collected data of the Mars planet surface, changes in semiconductor junctions over varying voltages, and volcanically active deep-sea rifts.

Education Meredith Bricken and Chris Byrne (M. Bricken, 1991) led a program to give local youth the chance to build and experience virtual worlds. The program emphasized the cooperative design process of building virtual environments. These VEOS worlds employed the standard navigation techniques of the wand and many provided interesting interactive features. The implementations include an AIDS awareness game, a Chemistry World and a world which modeled events within an atomic nucleus.

BIBLIOGRAPHY

Agha, G. (1988) *Actors: a model of concurrent computation in distributed systems*. MIT Press.

Appino, P.A., Lewis, J.B., Koved, L., Ling, D.T., Rabenhorst, D. & Codella, C. (1992) An architecture for virtual worlds. *Presence*, 1(1), 1-17.

Arango, M., Berndt, D., Carriero, N., Gelertner, D. & Gilmore, D. (1990) Adventures with network linda, *Supercomputing Review*, October 1990, 42-46.

Betz, D. & Almy, T. (1992) XLISP 2.1 User's Manual.

Bishop, G., Bricken, W., Brooks, F., *et al*. (1992) Research directions in virtual environments: report of an NSF invitational workshop. *Computer Graphics* 26(3), 153-177.

Blanchard, C., Burgess, S., Harvill, Y., Lanier, J., Lasko, A., Oberman, M. & Teitel, M. (1990) Reality built for two: a virtual reality tool. *Proceedings 1990 Symposium on Interactive Graphics*, Snowbird Utah, 35-36.

Blau, B., Hughes, C.E., Moshell, J.M. & Lisle, C. (1992) Networked virtual environments. *Computer Graphics 1992 Symposium on Interactive 3D Graphics*, 157.

Bricken, M. (1991) Virtual worlds: no interface to design. in Benedikt, M. (ed) *Cyberspace first steps*. MIT Press, 363-382.

Bricken, W. (1990) Software architecture for virtual reality. *Human Interface Technology Lab Technical Report P-90-4*, University of Washington.

Bricken, W. (1991a) VEOS: preliminary functional architecture, *ACM Siggraph'91 Course Notes, Virtual Interface Technology*, 46-53. Also *Human Interface Technology Lab Technical Report M-90-2*, University of Washington.

Bricken, W. (1991b) A formal foundation for cyberspace. *Proceedings of Virtual Reality '91, The Second Annual Conference on Virtual Reality, Artificial Reality, and Cyberspace*, San Francisco, Meckler.

Bricken, W. (1992a) VEOS design goals. *Human Interface Technology Lab Technical Report M-92-1*, University of Washington.

Bricken, W. (1992b) Spatial representation of elementary algebra, *1992 IEEE Workshop on Visual Languages*, Seattle, IEEE Computer Society Press, 56-62.

Bricken, W., Pezely, D., Evenson, M. & Almquist, M. (1993) A second step towards virtual reality: the entity model and system design. *Human Interface Technology Lab Technical Report M-93-1*, University of Washington.

Bricken, W. & Gullichsen, E. (1989) An introduction to boundary logic with the LOSP deductive engine, *Future Computing Systems* 2(4)., also *Human Interface Technology Lab Technical Report P-89-1,* University of Washington.

Bricken, W., Pezely, D., Evenson, M. & Almquist, M. (1993) A second step towards virtual reality: the entity model and system design. *Human Interface Technology Lab Technical Report P-93-1,* University of Washington.

Coco, G. (1993) *The virtual environment operating system: derivation, function and form.* Masters Thesis, School of Engineering, University of Washington.

Coco, G. & Lion, D. (1992) Experiences with asychronous communication models in VEOS, a distributed programming facility for uniprocessor LANs. *Human Interface Technology Lab Technical Report R-93-2*, University of Washington.

Cogent Research, Inc. (1990) Kernel linda specification: version 4.0. Technical Note, Beaverton, Oregon.

Cruz-Neira, C., Sandin, D.J., DeFanti, T., Kenyon, R. & Hart, J. (1992) The cave: audio visual experience automatic virtual environment, *CACM* 35(6), 65-72.

Dershowitz, N. & Jouannaud, J. P. (1990) Chapter 6: rewite systems, *Handbook of Theoretical Computer Science,* Elsevier Science Publishers, 245-320.

Ellis, S.R. (1991) The nature and origin of virtual environments: a bibliographical essay. *Computer Systems in Engineering,* 2(4), 321-347.

Etzioni, O., Lesh, N. & Segal, R. (1992) Building softbots for UNIX. Department of Computer Science and Engineering, University of Washington.

Etzioni, O., Levy, H., Segal, R. & Thekkath, C. (1993) OS agents: using ai techniques in the operating system environment. Department of Computer Science and Engineering, University of Washington.

Emerson, T. (1993) Selected bibliography on virtual interface technology. *Human Interface Technology Lab Technical Report B-93-2*, University of Washington.

Feiner, S., MacIntyre, B. & Seligmann, D. (1992) Annotating the real world with knowledge-based graphics on a “see-through” head-mounted display. *Proceedings of Graphics Interface ‘92*, Vancouver Canada, 78-85.

Fisher, S., McGreevy, M., Humphries, J. & Robinett, W. (1986) Virtual environment display system, *ACM Workshop on Interactive 3D Graphics*, Chapel Hill, NC.

Fisher, S., Jacoby, R., Bryson, S., Stone, P., McDowell, I., Bolas, M., Dasaro, D., Wenzel, E. & Coler, C. (1991) The ames virtual environment

workstation: implementation issues and requirements. *Human-Machine Interfaces for Teleoperators and Virtual Environments*. NASA 20-24.

Gelertner, D. & Carriero, N. (1992) Coordination languages and their significance. *Communications of the ACM*, 35(2), 97-107.

Gelertner, D., & Philbin, J. (1990) Spending Your Free Time, *Byte*, May 1990.

Goldberg, A. (1984) *Smalltalk-80*, Xerox Corporation; Addison Wesley.

Green, M., Shaw, C., Liang, J. & Sun, Y. (1991) MR: a toolkit for virtual reality applications. Department of Computer Science, University of Alberta, Edmonton, Canada

Grimsdale, C. (1991) dVS: distributed virtual environment system. Product documentation, Division Ltd. Bristol, UK.

Grossweiler, R., Long, C., Koga, S. & Pausch, R. (1993) DIVER: a distributed virtual environment research platform, Computer Science Department, University of Virginia.

Henry, D. (1992) *Spatial perception in virtual environments: evaluating an architectural application.* Masters Thesis, School of Engineering, University of Washington.

Holloway, R., Fuchs, H. & Robinett, W. (1992) Virtual-worlds research at the University of north carolina at chapel hill, Course #9 Notes: Implementation of Immersive Virtual Environments, *SIGGRAPH'92* Chicago Ill.

Jones, K. (1992) *Manufacturing simulation using virtual reality*. Masters Thesis, School of Engineering, University of Washington.

Jul, E., Levy, H., Hutchinson, N. & Black, A. (1988) Fine-grained mobility in the emerald system. *ACM Transactions on Computer Systems*, 6(1), 109-133.

Kazman, R. (1993, to appear) HIDRA: an architecture for highly dynamic physically based multi-agent simulations. *International Journal of Computer Simulation*.

Langton, C. (1988) *Artificial life: proceedings of an interdisciplinary workshop on the synthesis and simulation of living systems.* Addison-Wesley

Maturana, H. & Varela, F. (1987) *The tree of knowledge*. New Science Library.

Meyer, J. & Wilson, S. (1991) *From animals to animats: proceedings of the first international conference on simulation of adaptive behavior.* MIT Press.

Minkoff, M. (1992) The FERN model: an explanation with examples. *Human Interface Technology Lab Technical Report R-92-3*, University of Washington.

Minkoff, M. (1993) *The participant system: providing the interface in virtual reality.* Masters Thesis, School of Engineering, University of Washington.

Naimark, M. (1991) Elements of realspace imaging: a proposed taxonomy. *Proceedings of the SPIE 1457, Stereoscopic Displays and Applications II.* SPIE 169-179

Oren, T., Salomon, G., Kreitman, K. & Don, A. (1990) Guides: characterising the interface. in Laurel, B. (ed) *The art of human-computer interface design.* Addison-Wesley.

Pezely, D.J., Almquist, M.D. & Bricken, W. (1992) Design and implementation of the meta operating system and entity shell. *Human Interface Technology Lab Technical Report R-91-5*, University of Washington.

Robinett, W. (1992) Synthetic experience: a proposed taxonomy. *Presence* 1(2), 229-247.

Robinett, W. & Holloway, R. (1992) Implementation of flying, scaling and grabbing in virtual worlds. *Computer Graphics 1992 Symposium on Interactive 3D graphics*. 189.

Spencer-Brown, G. (1969) *Laws of Form.* Bantam.

Tanimoto, S. (1993) The pixelspace cooperative-learning environment. Department of Computer Science and Engineering, University of Washington.

Torque Systems, Inc. (1992) Tuplex 2.0 software specification. Palo Alto, Calif.

Varela, F. (1979) *Principles of Biological Autonomy*. Elsevier North Holland.

Varela, F. & Bourgine, P. (1992) *Toward a practice of autonomous systems: proceedings of the first european conference on artificial life*. MIT Press.

von Eicken, T., Culler, D.E., Goldstein, S. C. & Schauser, K. E. (1992) Active messages: a mechanism for integrated communication and computation, *ACM*, 256-266.

VPL (1991) Virtual reality data-flow language and runtime system, body electric manual 3.0. VPL Research, Redwood City, CA.

Wenzel, E., Stone, P., Fisher, S. & Foster, S. (1990) A system for three-dimensional acoustic 'visualization' in a virtual environment workstation. *Proceedings of the First IEEE Conference on Visualization, Visualization '90*. IEEE 329-337.

West, A.J., Howard, T.L.J., Hubbold, R.J., Murta, A.D., Snowdon, D.N. & Butler, D.A. AVIARY - a generic virtual reality interface for real applications. Department of Computer Science, University of Manchester, UK.

Winn, W. A conceptual basis for educational applications of virtual reality. *Human Interface Technology Lab Technical Report R-93-9,* University of Washington.

Winn, W. & Bricken, W. Designing virtual worlds for use in mathematics education: the example of experiential algebra. *Educational Technology* 12/92, 12-18.

Wolfram, S. (1988) *Mathematica: a system for doing mathematics by computer*. Addison-Wesley.

Woodward, G. (1993) Interactive dynamics for virtual worlds: background and preliminary architecture. *Human Interface Technology Lab Technical Report R-93-8.* University of Washington.

Zeltzer, D., Pieper, S. & Sturman, D. (1989) An integrated graphical simulation platform. *Graphics Interface '89*, Canadian Information Processing Society, 266-274.

Zeltzer, D. (1992) Auonomy, interaction, and presence. *Presence*, 1(1), 127-132.

Zyda, M.J., Akeley, K., Badler, N., Bricken, W., Bryson, S., vanDam, A., Thomas, J. Winget, J., Witkin, A., Wong, E. & Zeltzer, D. (1993) Report on the state-of-the-art in computer technology for the generation of virtual environments, Computer Generation Technology Group, National Academy of Sciences, National Research Council Committee on Virtual Reality Research and Development.

Zyda, M.J., Pratt, D.R., Monahan, J.G. & Wilson, K.P. (1992) NPSNET: constructing a 3D virtual world. *Computer Graphics*, 3, 147.