OVERVIEW of INTERACTIVE SOFTWARE TOOLS

William Bricken
October 1994

This memo describes an integrated software tool suite for generation of, management of, and interaction with arbitrary virtual environments for multiple participants. It specifies the application programmer's interface tools required to evolve the desktop metaphor into the environmental metaphor. The suite is called **EXPERIENTIAL COMPUTING**.

Infrastructure tools:

- **Core**, the virtual environment operating system,
- **PS (ParticipantSystem),** the virtual body, including **Sensors**, **CIG (Computer Image Generator)**, and **Sound**
- **EntityManager**, for coordination and structuring of entities
- **Voice,** a natural language recognition tool
- **Wand**, a generalized interactivity tool

Modeling tools:

- **EntityEditor**, an interactive form-based editor for attributes and dispositions
- **SpaceEditor**, a configurable environmental entity
- **RelationEditor**, a graph-based relation editor
- **ActivityEditor**, a scripting tool for embedded narrative
- **DynamicsEditor,** for Newtonian dynamics with boundary integrity
- **EntityLibrary**, a cataloging tool for constructed entities
- **RecursiveFormer,** a modeler based on efficient rules

Environmental tools:

- **VirtualCommons**, a continuously active, multi-participant gateway
- **Express**, the non-programmers interface to the software suite, including editors, resource management, and process control.
- **InconsistencyMaintenance**, an inference engine which supports divergent models for multiple perspectives and multiple participants.
- **BoundaryLanguage**, an experiential (visual, auditory, manipulative) programming language based on spatial representation of formal program semantics.
- **SystemsAssessment**, including display, resource and time management; and history and statistics accumulation.
- **ActivityTracking**, for keeping records and histories of virtual events.

EXPERIENTIAL COMPUTING

Generation of, management of, and interaction with virtual worlds is deceptively difficult. Our intuitions, which were developed within the physical world, are not as reliable in the virtual world, since the virtual is defined by potential, by expectation, and by imagination, just as much as by action and reaction. Future designers of virtual worlds will need sophisticated software tools that are intended to address the specific issues of virtual environment design and construction, tools that evolve out of intensive research into human-computer interaction in inclusive digital environments.

Experiential Computing is a software toolkit designed and developed over ten years in research and industrial contexts specifically for VR. The scope and the capabilities of the suite define a new computational metaphor. Like the desktop metaphor, which defines the current generation of user interface techniques, the entity/environment metaphor has the potential to define the next generation of participant interface techniques. The proposed technology will provide Human-Computer Interface (HCI) tools based in real-world conceptualization, natural interaction, situated spatial understanding, and a unified behavioral metaphor for both programming and simulation.

A user of the completed Experiential Computing suite will be able to call the system up within a Unix environment and encounter a graph-based configuration language (**Express**) which would allow easy configuration of the hardware components of a VR system. Once the system is linked and initialized, **Core**, **EntityManager**, and **PS** (with **Sensors**, **CIG** and **Sound**) would load automatically. Core would manage the dynamics of the virtual environment and the objects in it, EntityManager would manage communication between entities, and PS would provide rapid coordinated display and tracking.

Core and EntityManager extend 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.

Should the user wish to enter an existing world, he or she would indicate the particular files to be loaded within Express. The **EntityLibrary** tool would permit exploration of the graphic content of files, if needed. The user would enter the VR through the standardized **VirtualCommons** interface, which would provide a gentle transition between physical and virtual experiences, and would serve as an organizational hub for worlds to be visited. All worlds would provide generic interaction tools such as **Voice**, **InconsistencyMaintenance**, and the **Wand**.

Should the user wish to construct a virtual world, he or she would call appropriate construction tools from the Express interface. The **EntityEditor** would provide templates, forms, and languages for modeling entities, in particular for assigning attributes, perceptions, reactions, and persistent behaviors. Specialized tools would provide extended capabilities. The **DynamicsEditor**, for example, would be used for surface integrity, physical dynamics, and collision detection. **ActivityEditor** would be used for scripting time related events. The **SpaceEditor** would be used for assigning environmental attributes such as gravity and spatial metric. The **RelationEditor** would be used to establish relationships between attributes within and across entities. And the **RecursiveFormer** would be used to generate expansive generic environments, and to construct forms from within a virtual environment using the generalized sweep constructor. Of course, static modeling of object geometries would be accomplished using a commercial static modeling package not included in this toolkit. (We elected not to build a static modeler because the functionality of existing commercial tools is adequate.)

Finally particular generic maintenance and programming functions are provided by generic tools. The **SystemsAssessment** provides monitoring and management tools for the computational infrastructure. **ActivityTracking** provides scripted and filtered accumulation of participant behavior. **BoundaryLanguage** provides an experiential programming language.

An example of the tools in use follows.

Educational Scenario: MathWorld includes embedded geometric problems for students to practice problem solving. Imagine an outdoors virtual environment and several embedded tasks requiring the use of trigonometry. One such task might be for a student to measure the height of a tree. Of course, there are several methods to do this, including 1) measuring the length of the tree's shadow and the angle of the sun, 2) sighting a protractor set at 45 degrees to the top of the tree, or even 3) climbing the tree and dropping a rope to the ground. More exotic techniques that do not transfer to physical reality also exist in the virtual world. A student could lift up the tree and lay it on the ground to measure, or a student could grow in size to match the tree and then measure his or her own height.

These capabilities must be constructed. The objects (tree, shadow, ruler, protractor, rope) can be constructed with the EntityEditor, viewed in the EntityLibrary, assigned physical attributes with the DynamicsEditor, and assigned relationships using the RelationEditor. Global effects, such as the presence of gravity and the lighting effect of the sun, are established using the SpaceEditor. And ActivityEditor can enforce sequences of behavior such as requiring that the student use the protractor in a potential solution.

The virtual body PS and the Wand provide navigation and object interaction. One potential would be to point the Wand at the top of the tree and ask for the ray length, emulating a laser range finder. As an interaction tool, the Wand would be used to pick up and lay a tree on the ground. With different functionality, it could be used as the student's ruler, emulating a measuring tape. To change scale, a designer/student would associate the scale attribute of their virtual body with a command attribute in the environment (such as Voice recognition of "double"), using RelationEditor. Within the RelationEditor the meaning of "double scale" is established by an algebraic expression ("double <measure>" = 2 * <measure>) or by drawing a graph between the scale attribute and the command attribute.

Should a teacher wish to monitor or track the progress of a student, ActivityTracking provides templates for filtered history accumulation.

TECHNICAL DESCRIPTION

The motivation to build a VR interaction toolkit includes

- control of software complexity,
- an interface based in natural behavior and multiple intelligences,
- enhancing of native human abilities of spatial understanding and experiential learning,
- support of concurrent multiple participants and cooperative work,
- direct, non-symbolic communication, and
- a potential dominant market position in an explosive industry

VR 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 behavior. VR uses spatial interaction as a mediation tool. The prevalent textual interface of command lines and pull-down menus is replaced by physical behavior within an environment. 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.

Complex information systems must present information dynamics to the human in a manner which calls upon the native, intuitive capabilities of our entire sensorium. Non-linear information is simply too complex to be understood solely by intellectual abstraction. In particular, we must call upon visualization, audio cueing, kinesthetic feedback, and the full dynamics of *participation within the complex environment* in order to develop a deep understanding of today's information systems.

VIRTUAL REALITY SOFTWARE TOOLS

The suite of twenty-one software tools is grouped broadly into three categories:

Infrastructure:
Core, EntityManager, PS, Sensors, CIG, Sound, Wand, Voice

Construction:
RelationEditor, EntityEditor, SpaceEditor, ActivityEditor, DynamicsEditor, EntityLibrary, RecursiveFormer

Environment:
VirtualCommons, Express, SystemsAssessment, ActivityTracking, InconsistencyMaintenance, BoundaryLanguage.

INFRASTRUCTURE TOOLS

Core: The Core manages interprocess communication and distributed processing, enforcing timing and consistency across multiple participants. It is independent of applications. Core provides process integration and coherency across multiple seats and multiple systems.

EntityManager: The EntityManager manages the customized functions for each entity and module in the virtual environment. It coordinates database filtering, entity perspective, and behavioral loops.

PS, 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.

PS is a participant’s interface to the virtual world supported by Core. PS monitors the sensors that track the user's physical state, sends this information to the database, and displays the data in the database appropriately. Previous virtual reality systems have assigned all or most of these tasks to a single computer. In doing so, the interface elements such as displays and sensors are reduced to running at the speed of the database, and the database is itself limited by the interface processes running on the same CPU. PS lets each system do what it does best, enabling the interface to run as fast as possible while communicating with the database as fast as it can keep up.

This approach provides several major advantages. PS can be compiled and run to work with any display system on any processor with the proper communications capabilities. Since the protocol for communicating with PS is standard, a flavor of PS that takes advantage of its specific hardware can be used with any database that speaks the standard PS protocol. PS takes the hardware and software tasks that are common to all virtual worlds and packages them neatly for use in any virtual world running on any system.

Sensors: Sensors 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.

CIG: The CIG is a fast graphical rendering system which provides consistent, hardware-independent imaging for the PS Virtual Body. Its task is to take graphics-related attributes from a virtual environment and present them to the virtual body's graphics hardware, taking advantage of any capabilities present in the hardware for speed or realism. The CIG 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 when stereo is either not desired or not available.

Sound: The Sound renderer is the auditory counterpart to the CIG. It provides spatially localized (3D) and stereo sound to virtual environments. Sound could employ two Crystal River Beachatrons, yielding eight individually spatialized sound sources, each with 256 point head-related transfer functions (HRTFs) and controllable attributes such as intensity, pitch, duration, roll-off, and Doppler effects. Or it could be simpler, using stereo digital sound technology.

Voice: Natural Language Understanding is used to generate a dialog between the user and the computer. The grammar of this system is restricted to the description of a small environment. The user interacts with the system through phrases that indicate an action or query the system status. The voice recognition system would be implemented on a digital signal microprocessor (DSP); the Natural Language Understanding element, using LISP.

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. 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:** Display the length of the ray vector, expressed in the metric of the intervening space.

- **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 patron issuing the command is linked to the ray vector.
- **Move faster/slower:** Move the viewpoint of the patron along the ray vector.

CONSTRUCTION TOOLS

Seven virtual world construction tools are proposed for development.

RelationEditor: The RelationEditor is a virtual environment development tool that provides a graphical means of specifying entity behaviors. Using this system, we program entity attributes by drawing graphs that specify relationships between attributes. For example, a simple color controller can be built by relating the color of one object to the coordinates of a hand tracker.

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 interaction 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. For example, gravity can be attributed to space when it acts on every object within it. Other properties of space include metric structure (grids, orderings, sets, reals), gradients (gravity, wind, electromagnetic forces), continuity, grain-size and coordinate systems.

DynamicsEditor: A software library of dynamical functions that one can easily incorporate into the behavior of entities. These dynamical functions will 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.

ActivityEditor: Humans use stories to structure their interaction with knowledge. ActivityEditor 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.

EntityLibrary: A large collection of graphics files will be assembled so that world designers will not have to redesign common objects. Grids, cubes and polyhedron collections, etc., will be available, along with representative natural world objects. A method of browsing and selecting from the library is also included.

RecursiveFormer: A specification and construction capability based on generalized Lindermeyer systems (recursive term-rewriting over graphics specifications), permitting top-down design of environments and providing control of the level of detail and refinement in a graphical object. Models well suited for this technique have many repetitions of small groups of basic elements, such as trees in a forest, buildings in a city, or clouds in the sky. RecursiveFormer provides fine grained control over the coherence and juxtaposition of elements and the probabilistic ranges of their form using a diversity of techniques. Both the geometry and the topology of forms can be specified by equations in an algebra of form, by production rules, by form abstraction, or by direct model modifications. RecursiveFormer provides rapid modeling capabilities which bridges the gap between photographic images and solid modeling with polygons.

A virtual environment can be abstracted into the composition of three sets: the individual objects or entities, the space these entities jointly occupy, and the spatial relationship between each entity and the origin of the space, as expressed in the metric of that space. The advantages of *form abstraction* include:

- positional information is separate from the geometric form of an entity
- space can be subdivided recursively
- entity interactions can be treated as pair wise
- entities can be locally independent of the space they occupy

The RecursiveFormer includes a generalized sweep capability We believe that sweeps, initiated by hand and arm gestures are the most natural way to generate solids in virtual spaces. The generalized sweep construction tool has the following properties:

- hand and arm gestures trigger the creation of solid objects
- sweeping a point will generate a one dimensional line; sweeping a line will generate a plane; sweeping a plane will generate a solid

- translational, rotational, and scaling sweeps
- sweep curve may be freehand or specified by a function
- freehand sweeps will have a TIDY option which will smooth it to a function of specified degree
- object libraries will include the basic solids

ENVIRONMENTAL TOOLS

Express: Express is a simple interface which allows novices to access and configure the Core system. It provides a simple WIMP interface to the complex operating system and design tools in a VR system. The current design of Express incorporates a visual control-flow language which structures control over data, processes, and the interrelation between the two. It provides experiential cueing that helps the designer decide which data to use, how to integrate it, what behaviors and filters to use, how to structure timing and frequency of events, and in general how to define causality in the virtual environment.

VirtualCommons: A continually-running multi-participant virtual world to serve as a gateway to other worlds. The VirtualCommons will also serve as a central meeting place, hopefully providing a social center and a venue for exploring social issues of multi-participant virtual interaction. A design objective is to allow all VR worlds to be loaded and entered from within the VirtualCommons, by an arbitrary number of participants, from a number of different hardware platforms.

Inconsistency Maintenance: The mathematical tools which coordinate actions of multiple participants in virtual worlds. Virtual worlds permit mutually inconsistent models across multiple participants. Each participant can maintain a separate personal environment concurrently in the same virtual space. Communality of mutually shared perspectives is negotiated rather than assumed.

Using multiple-valued logics, InconsistencyMaintenance provides tools to maintain inconsistent views and interpretations, and tools to negotiate differences when consistency is desired. Differences can be resolved by presenting each participant with personal views, by providing a mezzo-space where communalities can be openly negotiated, or by maintain inconsistency through mathematical techniques.

The negotiation of inconsistencies can be sensory (sharing viewpoints), knowledge based (sharing memories), or rule based (sharing dispositions). Maintenance of contradiction in virtual worlds requires merging inconsistent knowledge without disabling action or inference. Our approach is similar to a hypothetical worlds approach, but splits at the variable level (bottom-up) rather than at the model level (top-down).

BoundaryLanguage: BoundaryLanguage provides a functional experiential (visual, auditory, tactile and behavioral) programming language. We have been able to demonstrate that mathematics itself (in particular 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, rockwalls that respond to gravity, the things of everyday life. String-based symbolic representations of mathematical concepts are typographically convenient, but tokens are not at all essential to mathematical expression. VR makes it convenient to express abstract ideas using spatial configurations of familiar objects. One benefit of this approach is that we can build visual programs, set them on a virtual table, and watch them work. We can experience programs as other entities rather than as dumps of text. Bugs would manifest as structural anomalies, as visual irregularities. It is but a quirk of typography that we have ignored the experiential semantics of computational languages. More fundamentally, experiential computing unites our spatial and our symbolic cognitive skills, permitting mathematical visualization and whole body processing.

SystemsAssessment: The Core requires testing and debugging tools, independent of the application. SystemsAssessment includes management and optimization of display, memory, communication, and timing across the entire system.

ActivityTracking: ActivityTracking uses the Core/EntityManager infrastructure to collect experimental, historical, and tracking data about participants in a virtual environment. Performance history and statistics is also measured and stored by this tool.