Virtual Reality in Computational Neuroscience

Jason Leigh (spiff@eecs.uic.edu), Christina A. Vasilakis, Thomas A. DeFanti
Electronic Visualization Laboratory, University of Illinois at Chicago

Robert Grossman
Laboratory for Advanced Computing, University of Illinois at Chicago

Chris Assad, Brian Rasnow, Alex Protopappas, Erik De Schutter, James M. Bower
Dept. of Computation and Neural Systems, California Institute of Technology

April 20, 1994

Abstract

Computational Neuroscience is a branch of neuroscience that investigates how the nervous system works, by developing comprehensive simulation models of single neurons or networks that are based on experimental findings. In this paper we describe our efforts in designing a scalable virtual reality system and heuristic-based interface library for use by computational neuroscientists to visualize simulated data from their research[1].

1 Introduction

The last several years have seen a dramatic increase in the number of neurobiologists building or using computer-based models as a regular part of their efforts to understand how different neural systems function[Bow92]. As experimental data continues to amass, it is increasingly clear that detailed physiological and anatomical data alone are not enough to infer how neural circuits work. Experimentalists are increasingly recognizing the need for the quantitative approach to exploring the functional consequences of particular neuronal features which is provided by modeling. This combination of modeling and experimental work has led to the creation of the new discipline of computational neuroscience[EB93].

Typically neurally based simulations generate enormous amounts of complex data. The computational neuroscientist is often faced with the daunting task of interpreting ths data in relation to real experimental data. As an aid in interpreting this data we have utilized advanced visualization techniques that have, most recently, included virtual reality (VR). This exploratory work was demonstrated at the Neural Simulation Software Demonstration exhibit at the Neuroscience'93 conference in Washington DC. From this demonstration we learned that many neuroscientists were completely unaware of the technology available for advanced visualization. We are presently implementing a subset of this work for display in the CAVE VR system[CSD93] at the VROOM exhibit at SIGGRAPH'94.

[1]This work is supported in part by the Human Brain Project of the NIMH, grant # Mh52145; NSF grant # IRI-9213822, and # CDA-9303433 which is also supported by ARPA. Support for PTool- a scalable persistent object manager, comes from NASA grant NAG2-513 and DOE grant DE-FG02-92ER25133

In this paper, we will first identify a practical VR hardware configuration for neurobiologists. We will then describe the "V" VR interface library[2], currently under development, which is designed to work with this VR configuration and with the CAVE VR system. Thirdly, we will describe a persistent object-store used to manage large collections of neuroscience data for efficient access in VR environments/worlds[3]. Finally, we will describe applications of our VR system to the visualization of neuroscience data.

2 A VR Visualization System for Computational Neuroscientists

Over the years a significant amount of VR hardware and software has evolved from VR research[CNSD+92, CSD93, KWA93, SLGS92]. Many of these systems are undergoing continual improvement. These improvements, however, are generally not available to the larger VR public either due to a lack of commercial availability, or that the system is prohibitively expensive. Many non-VR scientific disciplines are interested in exploring the use of VR in assisting their research, but the large selection of possible VR configurations makes the process of identifying a practical system difficult. From the computational neuroscientist's point of view the system must be affordable, scalable and practical for regular and prolonged use. In addition, there needs to be enough software support so that they can readily use the system for their research rather than spend most of their time developing low-level visualization code[KEL+92].

For the last several years, computational neuroscientists have been developing general purpose neural simulation system software as platforms for the construction of complex simulations[DS92]. The initial VR design work described here was performed in relationship to GENESIS (GEneral NEural SImulation System) which has been developed in Dr. Bower's laboratory at Caltech. GENESIS was developed as a research tool to provide a standard and flexible means of constructing realistic simulations of biological neural systems at many different levels, from sub-cellular components, to whole cells to networks of cells [BH91]. The system is object oriented, user extensible, and places a significant emphasis on the support of advanced user interfaces. Construction of both simulations and GUIs within GENESIS is accomplished by linking objects in simulation libraries via a high level scripting language. GENESIS is currently in use in many laboratories around the world who share GENESIS objects and code via the internet based users group BABEL. BABEL also provides a means for designing, testing, and using the VR system described here. Ideally the VR system should be fully integrated with GENESIS.

2.1 Hardware Configuration

Our VR system, which is essentially a Fish Tank VR system [KWA93], consists of an SGI Indigo XS24; a Logitech ultrasonic 6D mouse and head tracker; and a pair of CrystalEyes stereographics glasses and synchronizing infra-red emitter (Figure 1). This configuration was chosen primarily because it provided reasonably good performance for a low cost.

We decided to use the stereographics glasses rather than head-mounted displays (HMD) because many affordable HMDs have resolutions (generally NTSC) that are too low for viewing small details in scientific data[CNSD+92]. The problems of visually induced nausea caused by HMDs and the general discomfort, experienced by users, over prolonged use of the head gear, were also strong determining factors against their use. Also, by using the stereographics glasses and a color monitor, all interactions with the virtual environment are made in the line of sight of the monitor. This affords the use of the sonic tracker and mouse as a low cost 3-space tracking and pointing device. The only disadvantage is that the virtual environments created in our system are not as immersive as environments rendered in HMD systems.

[2] "V" is inspired by "X".
[3] "Environment" and "world" are used interchangeably in this paper.

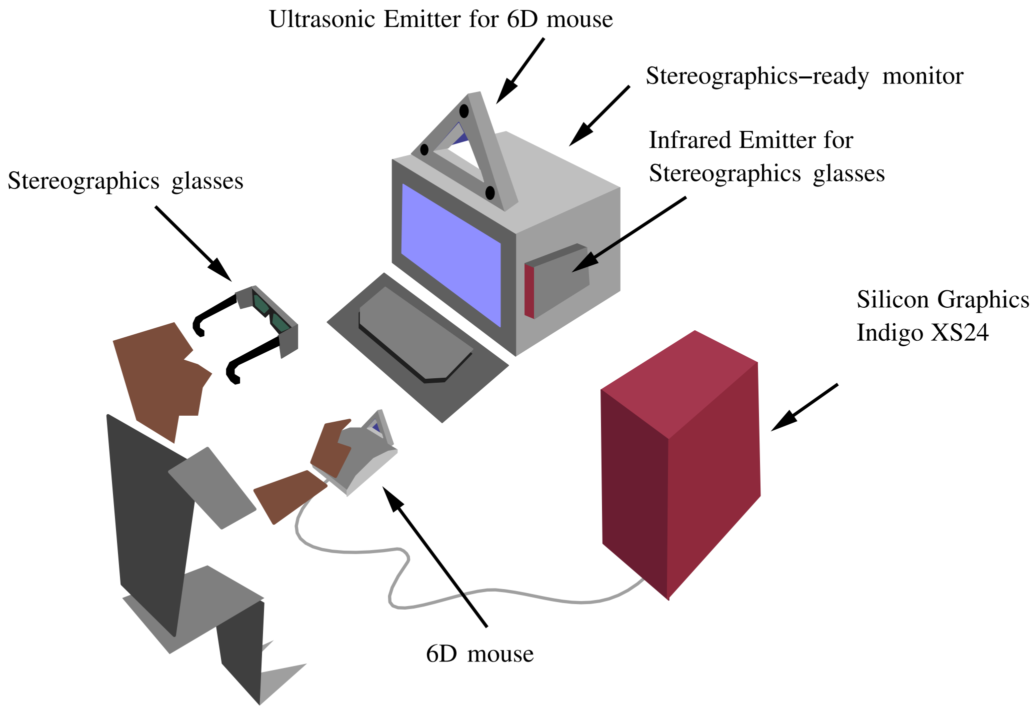

Figure 1: Hardware configuration for a Fish Tank VR system.

The BOOM did not suit our needs due to the excessive cost for color systems. Finally, although we do not anticipate neurobiologists being able to afford the CAVE system, we invited them to use the CAVE at the Electronic Visualization Lab, for their research. To encourage this, our VR software library was developed to be compatible with both our Fish Tank system and the CAVE.

2.2 Software Configuration

In evaluating possible software libraries for the development of our neuroscience virtual environments, three main requirements had to be met. First the software had to provide support for interfaces to our VR devices, and basic virtual widgets (menus, sliders) for interaction. Secondly, the software had to provide high-level widgets (graph plotters, isosurface viewers, 3D neuron morphology viewers) that could be used for the kind of data analysis to which the neurobiologists were accustomed. Thirdly, the software had to be scalable in order to handle large amounts of data for visualization.

2.2.1 VR Software Libraries

A number of software libraries are available for the development of interactive VR environments; most of these have been summarized by Shaw [SLGS92]. In general these libraries provide device-independent interfaces to virtual devices and general support for virtual widgets. In most of these libraries however, no heuristic support for the design and management of the interfaces exist. Essentially, the unaided application developer is left to consider all of the interface options, such as optimal placement and manipulation schemes of menus, buttons and potentiometers. This means that the design-space of possible implementations is potentially enormous.

Figure 2 illustrates a subset of the design-space for interacting with a flat virtual menu. The tree sketches the possible mechanisms of interaction in three phases of menu use:

- Menu Invocation- where the decision is *which* 3D mouse button to press to bring up the menu.
- Menu Placement- where the decision is on *where* the menu is to be placed upon invocation. The menu could appear in front of the viewer; it could appear in some preset position; it could appear where the 3D mouse is pointing; or even follow the user as they moved around the virtual space.
- Menu Selection- where the decision is on *how* the user chooses one of the menu items. Possibilities include: using the user's head orientation to point a virtual laser pointer (raycasting[JE92]) at the desired menu item; using the 3D mouse to point at the menu item; or even using the buttons on the mouse to iterate through each menu choice until the desired one is reached.

The list of possibilities is virtually endless and we have not even begun to consider the design-space of possible graphical representations of the menu[4]. These are the kinds of possibilities presented by most of the currently existing VR toolkits. In many ways MIT's X-windowing system suffered the same problem on its first introduction to the UNIX world. It was soon evident that some form of consistent user-interface management system was needed. This consequently resulted in the creation of numerous public-domain and commerical window managers.

The problems of interface management (as we have illustrated in the example above) are considerably more complex in 3D. Clearly, some form of interface management system for VR is needed. Shaw points out that the VR style of interaction is not well enough understood to consider the development of more intelligent user-interface management systems[SLGS92]. While this may be the case currently, it offers little help to the neuroscientists (and scientists in general) who are interested in building their own virtual worlds. In fact,

[4] In fact it is not clear whether flat menus are the best means for presenting multiple choices to users in virtual environments.

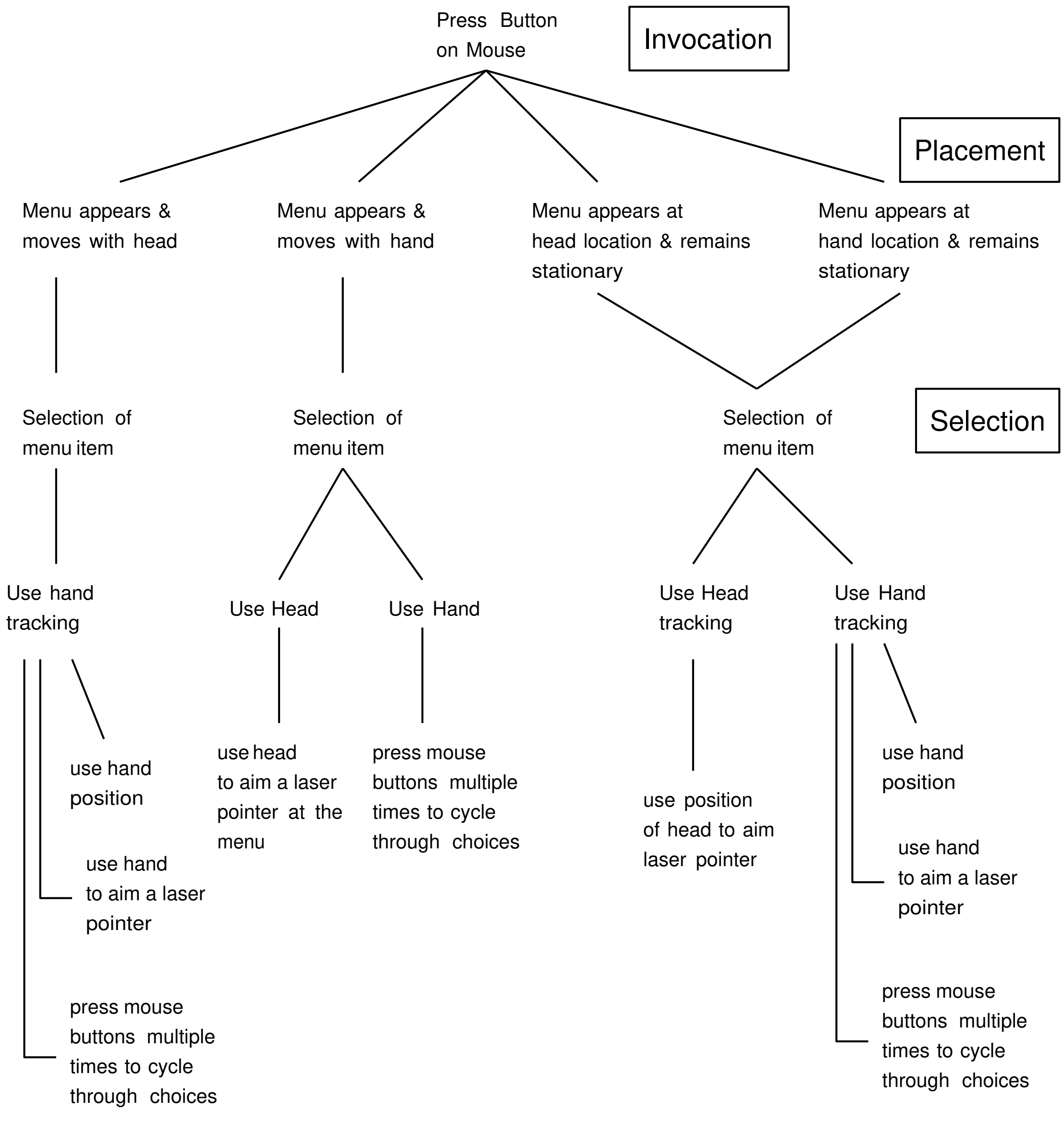

Figure 2: A design-space for the possible modes of operation of a flat VR menu.

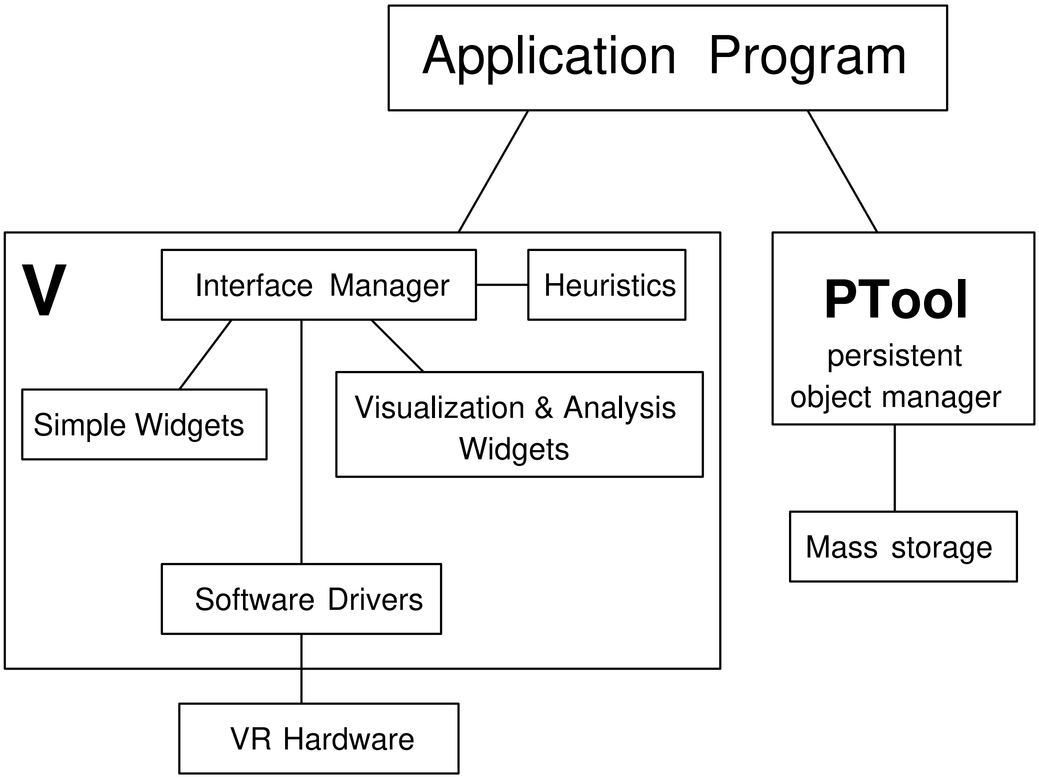

Figure 3: Interfacing an application program with "V" and PTool (a persistent object-manager designed to manage large scientific datasets for real-time VR environments). The application may be a program written in C or part of a neural modeling simulator.

even VR software developers are not always aware of what constitutes a good interface design in a virtual world[Vas94]. While the solutions to their interface problems may come by exhaustive experimentation, neuroscientists are generally not interested in pushing the bounds of VR research. They are only (and rightfully so) interested in using tools that directly enhance their own research. Therefore, it would be very useful for a VR interface management system to embody a set of heuristics that are based on the most recent research on VR interfaces[KWA93, JE92, SLGS92, SZ93, Vas94]. Furthermore, to enable more extensive use of VR for scientific research, additional libraries of high-level scientific visualization tools, such as graphs and isosurface builders, that are routinely used for data analysis by scientists[RRS92], should be provided.

2.2.2 "V"

The "V" interface library (Figure 3) is an attempt at providing a heuristic-based interface management system and high-level scientific widgets for interaction in virtual worlds. Like currently existing VR toolkits, "V" provides a set of drivers to control input and output VR devices (including the CAVE) through a consistent application programmer's interface (API).

"V" provides a simple interface to virtual menus, sliders and graph tools which is controlled by a user-iterated event loop much like the CAVE library. This simplicity allows open exploration of VR interface issues. We intend to use this as a testbed for VR experiments to help us devise heuristics needed for "V"s interface manager. Thus far, "V"'s interface tools and their style of operation are designed by following a small set of heuristics. While these heuristics are by no means definitive, they provide a means of reducing the design-space of possibilities so that the neuroscientists will not have to unduly concern themselves with such issues; this allows them to concentrate on their visualization task. The following is a brief description of the heuristics embodied in the current version of "V"'s interface manager:

1. The tools in "V" were designed in collaboration with an artist so that they maintain visual simplicity, while providing sufficient cues to communicate their intent. We believe most of the graphics workstation's power should be spent visualizing the data rather than drawing elaborate interfaces.

2. Z-buffering is turned off when tools are invoked, so that they will not be occluded by other objects in the space. In the CAVE, "V" tracks the position of the viewer's head and automatically places the tool three feet in front of the head position for comfortable viewing and interaction[5].

3. After performing an operation with a tool, it disappears to reduce the complexity of the visuals presented in the virtual world, while also reducing the rendering load on the graphics workstation. The disadvantage of using this scheme is that it does not allow easy management of multiple tools simultaneously. This limitation will be removed as we further develop "V" to incorporate more elaborate management techniques.

4. Raycasting is the mechanism by which tool selections are made. It has been shown that this technique is effective in reducing arm fatigue ("gorilla-arm") when making numerous selections from virtual menus [SLGS92, JE92].

These heuristics are currently organized as methods in the C++ classes used to build "V". As the set of heuristics grows, it is likely that we will have to employ a knowledge-based management approach. Thus we tend to follow the general approach used by Lewis[LKL91] in using a rule-based system to drive a user-Interface Management system for Virtual Worlds, and Feiner[BF92] in using a rule-based system to automatically design virtual worlds for visualizing multivariate data.

2.2.3 Managing Visualization Data for VR Environments Using A Persistent Object-Store

As mentioned earlier, another important component to VR world construction for scientific visualization, is the management of large amounts of data that are the basis for the visualization. This is particularly important for data intensive fields like computational neuroscience. Virtual Reality systems, and the environments that are created as a result of their application, have in the past relied on the ability of the system to maintain all the data being visualized, in the main memory of the system. Unfortunately, this strategy does not lend itself to the exploration of more complex and data-rich environments, such as those occupying hundreds of megabytes to terabytes of data (characteristically generated by many scientific disciplines). These massive amounts of data are generally stored in large databases, which are rarely interfaced with real-time VR systems [CLB+93]. Instead, the traditional approach is to extract small subsets of the data from the database and then view it in the VR environment.

The choice of using this approach stems from a number of obstacles that are encountered when interfacing databases with VR systems. First of all, the rate at which data from a database can be imported into a VR environment is significantly slower than what the graphics is capable of animating [GGR93]. Secondly, commercial databases are designed with the additional overhead of providing transaction management that can significantly reduce the performance of the database[GQ93, GLQ93, BG91]. Thirdly, VR developers are rarely database experts and would prefer to store the data in flat files rather than use esoteric APIs to a commercial database system.

[5] This was empirically found to be a comfortable distance.

PTool [GQ93, GLQ93, BG91], is a persistent object manager designed to address these issues. This comes as a result of understanding the nature of scientific data, which is generally characterized by high volume, low update frequency, and indefinite retention[FJP90]. This means that we do not necessarily have to provide transaction management as in most commercial database systems, hence lowering the overhead of accessing data. PTool is designed to work with approximately a gigabyte of data; a companion tool called VTool is designed to work with hundreds of thousands of PTool volumes, supporting approximately a terabyte of data. Comparisons of PTool against other commercial, relational and object-oriented databases, on approximately 100 megabytes of data, has shown PTool to out-perform a relational database by a factor of ten; and an object-oriented database by factor of at least four[GQ93]. Also comparisons of PTool against storing data in RAM and in memory mapped files have shown little performance loss, which makes it well suited to manage data in real-time VR environments.

PTool supports the uniform creation, storage and access of complex objects, regardless of their lifetimes (persistent objects). In other words, a mechanism is provided so that persistent objects out-live the processes which create them and can be accessed in a uniform manner by other processes. In C++, transient objects are created at run-time using the "new" operator; with PTool, persistent objects are created by overloading "new" and indicating to which data-store the objects belong. The data-store is implemented as a virtual memory mapped file. This is based on similar implementations described by Skekita [ES91] and Williams[IW91]. The advantage of using this scheme is that, what is traditionally known as the database schema is stored as a C++ class; and the API is simply an overloading of existing C++ operators that are already familiar to application programmers.

3 Example Applications

Three applications were prototyped. They involved visualizing simulation data generated by the GENESIS neuronal network simulator developed at Caltech[LDB+94]. The first application visualizes the electric fields of the Apteronotus Leptorhynchus weakly electric fish[BR93]. The second visualizes a network of pyramidal cells in the piriform cortex[WB92]. The last visualizes a complex spike in the cerebellar Purkinje cell[DB94].

3.1 Visualization of Electrical Fields in the Apteronotus Leptorhynchus

Apteronotus Leptorhynchus, or the brown ghost, is a South American freshwater species, generally nocturnal and living in turbid waters. It has a long organ lying ventral to the spinal cord, running from behind the head to the tip of the tail, which emits a continuous and periodic potential with a fundamental frequency from 600 to 1000 Hz [BR93].

The visualization (Figure 4) consists of a model of the fish surface and midplane in which the electric organ discharge potential was simulated by a number of current sources. These sources are current monopoles- red spheres are used to indicate that positive current is flowing out from them, while blue spheres are used to indicate negative current, flowing into them. The relative size of each sphere indicates the absolute magnitude of the source. Each frame of the animation is treated as a static field problem, and total current from the sources always sums to zero. Based on what we know of the real electric organ, we can reconfigure these simple sources at each frame of the animation in order to match the simulated potentials with fields actually measured from the fish. The goal is to understand how the fish uses the phase and amplitude information from the electric organ discharge for electrolocation and communication.

In real life, when the brown ghost swims by an object, the object perturbs the current distribution, thereby casting an "image" onto the fish's surface, which the electroreceptor array can detect. In our visualization, a virtual object is interactively inserted into the midplane where the perturbation to the electric field is calculated (with some analytical approximations). The object image is then displayed on the fish's surface,

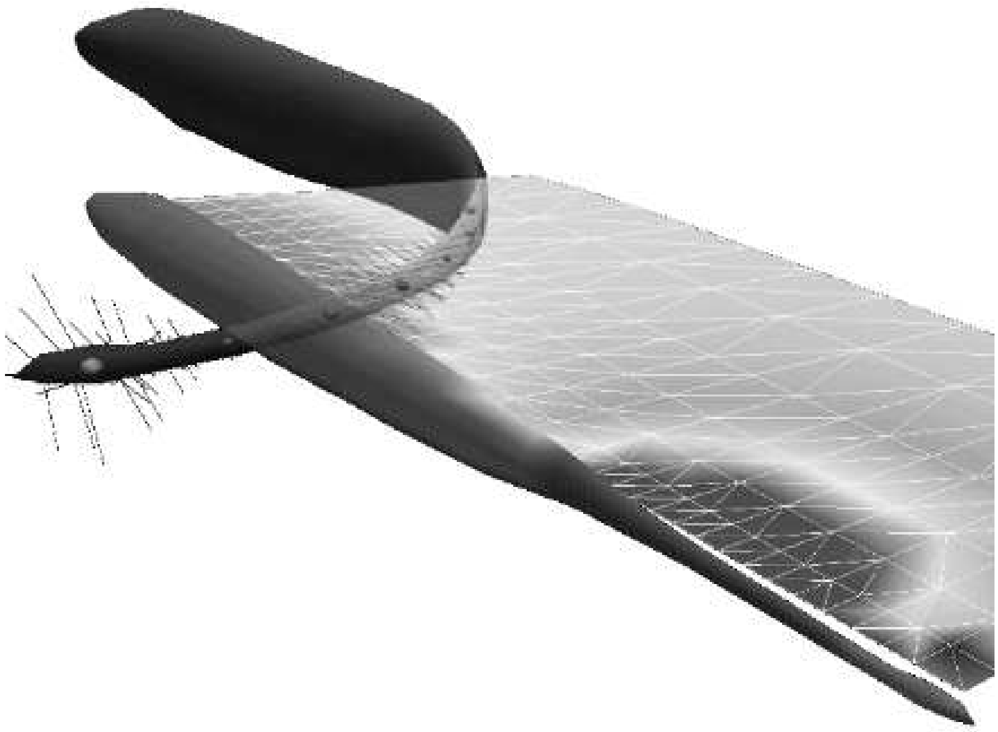

Figure 4: Visualization of two Brown Ghosts showing potential at the fish's surface and the electric field on a midplane of water around the fish (bottom). The spheres inside the fish and along the tail are current sources (top).

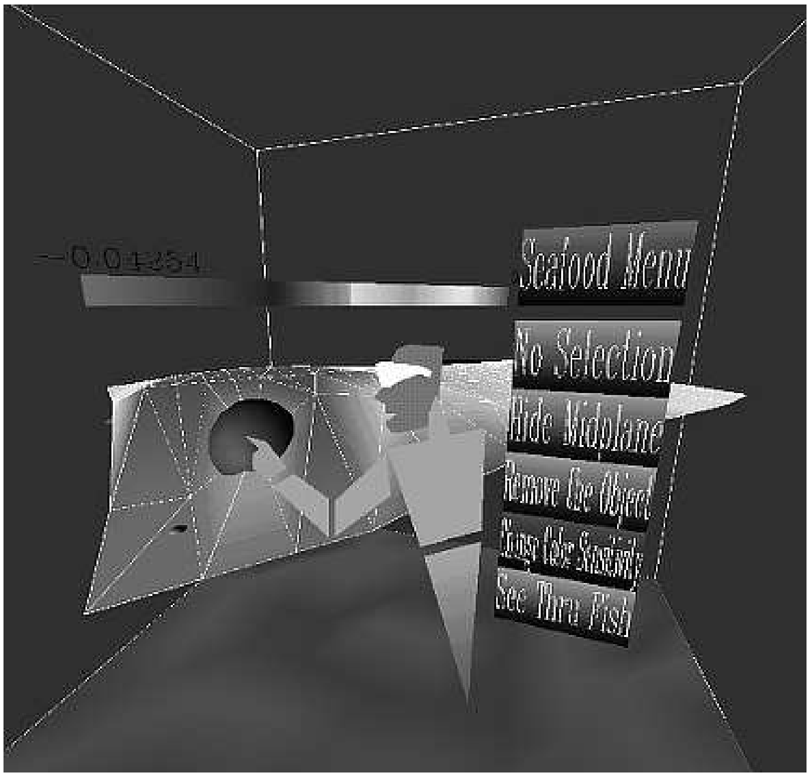

Figure 5: Visualization of the electric field perturbated by a virtual object dropped into the midplane. This is a schematic taken from a display in the CAVE.

representing the difference between the potential on the skin with and without the object present (Figure 5).

We have extended this simulation for visualization in the CAVE system. In the CAVE viewers are able to watch the monopoles by physically inserting their heads inside the body of the fish. The idea here is that we can take advantage of an immersive VR system to provide a more natural interface to the data. In this case, the most obvious way of seeing the sources inside the fish, is by sticking ones head into it!

3.2 Visualization of a Piriform Cortex Simulation

The piriform cortex is a three layered cortical area which receives its input from the olfactory bulb, which in turn receives input from the nose [She76]. The principal neuron of the piriform cortex is the pyramidal cell which receives afferent input from the bulb and makes connections with other local and distant pyramidal cells within the piriform cortex[Pro93].

The previously published simulation[WB92] consists of 135 pyramidal cells, feedforward inhibitory cells and feedback inhibitory cells all in a 15×9 array (Figure 6). The pyramidal cells are modeled as five compartments where each compartment receives a distinct kind of synaptic input and is located in a different sub-layer of the cortex. The inhibitory cells are modeled as one compartment. Input from the bulb is modeled as random activity entering the cortex via the lateral olfactory tract (LOT) in much the same way it does in the real system.

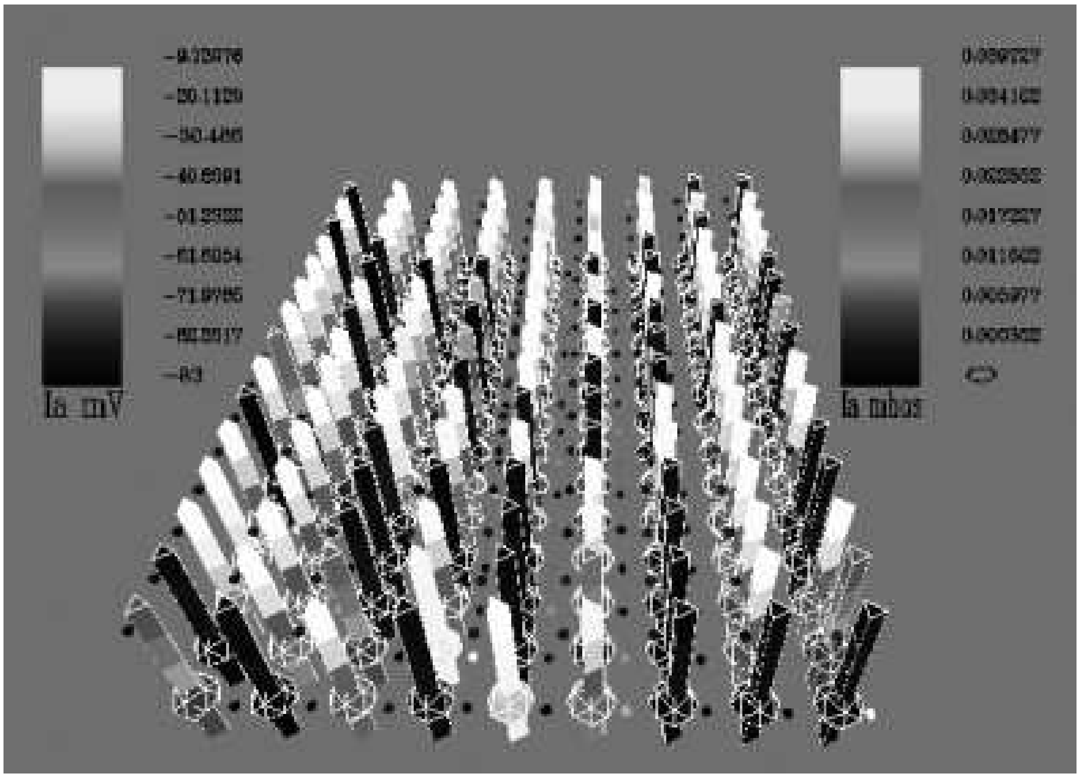

Figure 6: View of a 15×9 array of pyramidal cells in a piriform cortex network simulation model. Each pyramidal cell is composed of five simulation compartments modeled within GENESIS.

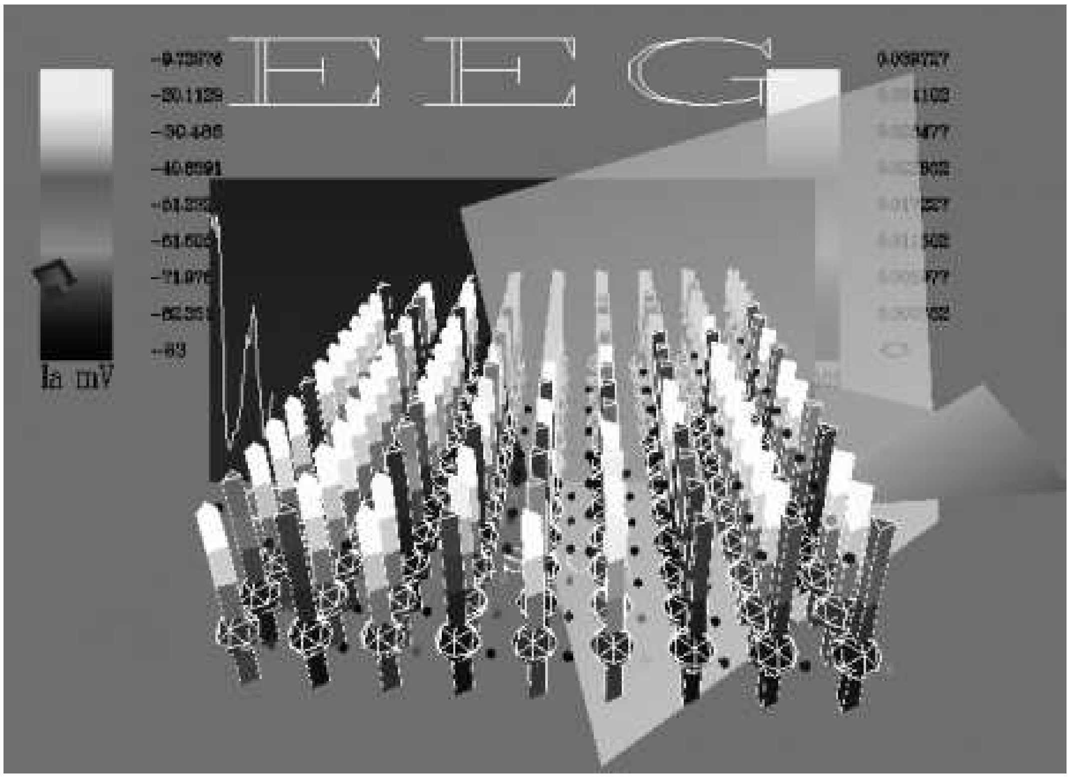

Figure 7: View of a virtual slicing tool that can be used to cut out pieces of the network for isolated viewing. In the background a graph shows the averaged electrical activity of the neurons as an EEG.

In the visualization, a virtual menu can be invoked from which either the viewing of membrane potential or synaptic conductance, can be chosen. Figure 6 shows the membrane potential at each compartment of the cells. As random input from the LOT fibers are generated, a wave of activity can be seen propagating through the network. By showing synaptic conductance rather than membrane potential, this wave can be seen more prominently. This is because by viewing conductance we are observing synaptic activity alone, whereas when viewing membrane potential, we are looking at a combination of both the active and passive properties of the cell.

The visualization also allows you to peel off each layer of compartments as well as take arbitrary slices out of the network to view isolated regions. A graph plot can be invoked to view the averaged electrical activity of all the neurons as a simulated EEG (Figure 7). From this the user can see how the EEG oscillations correlate with the waves of activity.

3.3 Visualization of a Complex Spike in a Cerebellar Purkinje Cell

The Purkinje cell is the largest neuron in the cerebellar cortex and is its only output element[Ito84]. Purkinje cells have two features which make them unique. First, their dendrites have a large number of Calcium channels, with intradendritic recordings showing typical dendritic Calcium spikes. Second, the cell receives over 150000 parallel fiber synapses; more than any other neuron in the brain, but only a single climbing fiber synapse (Figure 8) [Ito84]. This single synapse originates from the inferior olivary neuron and makes

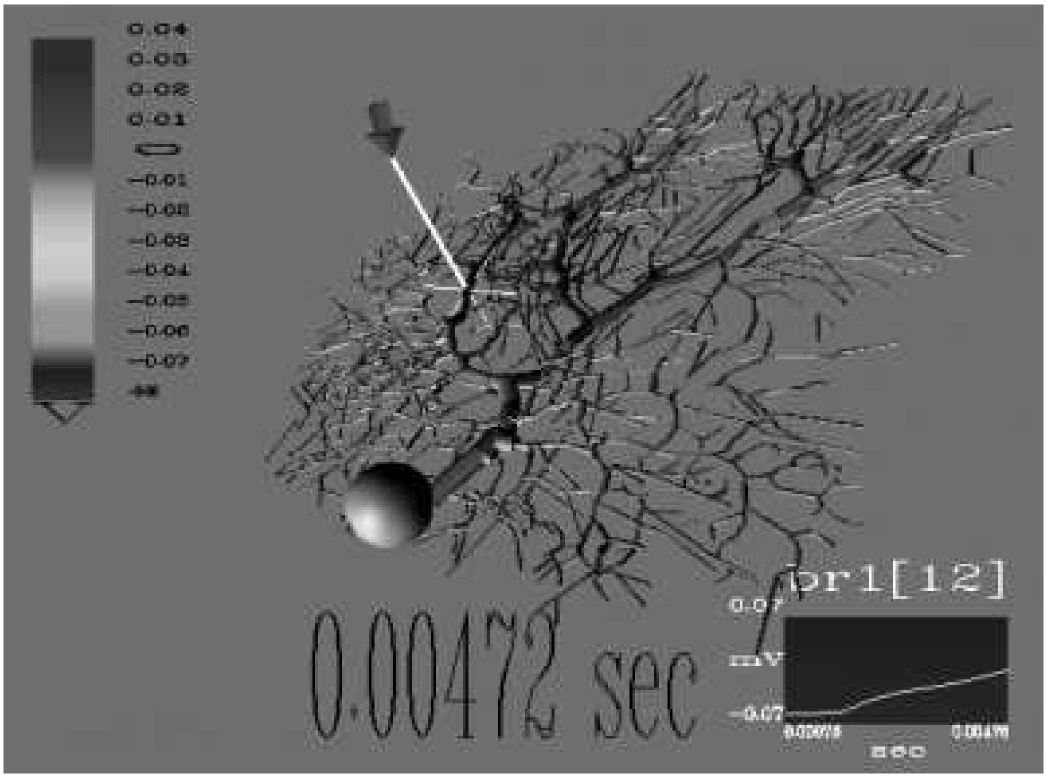

Figure 8: Compartmental model of a Purkinje cell showing the complex spike by cycling through precomputed simulation data stored on a PTool data store. The virtual electrode is used to query individual components of the neuron for its membrane potential.

contacts on the soma (the spherical object), the main dendrite, and the thick dendrites that eminate from the main dendrite. Activation of the climbing fiber synapse results in a massive response, called the complex spike. The complex spike is caused by the activation of Calcium channels in the dendrite during a prolonged dendritic spike. The complex spike changes the immediate sensitivity of the Purkinje cell to other synaptic inputs and can induce long term depression of the synaptic conductance of parallel fiber inputs[LDA+93].

We developed a detailed computer model of a rat purkinje cell which we ran on an Intel Touchstone Delta supercomputer[DSB92, DSB93, DB94]. Both the morphology and the ionic channels were modeled as accurately as available experimental data allow. The results of the simulation were stored in a PTool data store and visualized in our virtual reality system.

The visualization (Figure 8) shows the model's response to the climbing fiber synaptic input. Displayed is the membrane potential at each of the compartments that comprise the neuron. Blue represents approximately -80mV and red approximately +40mV. While viewing the complex spike, the user can activate a virtual electrode to probe any compartment in the neuron. This system allows the user to load other neuron structures and their corresponding firing data.

4 Conclusions and Future Work

We have described a low-cost, high performance fish tank VR system to support research in computational neuroscience. The VR development library "V" will continually be improved as we amass increasing knowledge about interaction in virtual environments. This has come in part from work by other researchers[KWA93, JE92], but we will soon be engaging in a series of quantitative evaluations of the effectiveness of various interface paradigms in VR. The goal is to build a heuristic-based virtual user-interface management system for virtual environments. In addition, we will continue to build high-level analysis tools for viewing computational neuroscience data.

PTool has been found to be effective in managing data for real-time VR visualization. By applying database technology to support VR, we not only improve the VR performance, but we also develop new and interesting interfaces for databases[JFL94]. For many years researchers in the database community have voiced the need for more intuitive interfaces to databases[BK90]; we believe VR holds some interesting possibilities.

The application of virtual reality in computational neuroscience is still in its infancy. This preliminary work has opened new possibilities for neuroscientists to visualize their research data.

It is quite typical during experiments and simulation to generate enormous amounts of multidimensional data. One of the problems today with being able to generate this data is that it is still extremely difficult to make any sense of it. Numerous pattern searching techniques may be devised to analyze this data but the human visual system is still the most advanced pattern matching system we possess. We are therefore greatly interested in further using virtual reality to produce visualizations that can assist this process.

References

[BF92] C. Beshers and S. Feiner. Automated design of virtual worlds for visualizing multivariate relations. In *Proceedings of IEEE Visualization'92*, pages 283–290, 1992.

[BG91] A. Baden and R. Grossman. Database computing in high energy physics. In Y. Watase and F. Abe, editors, *Computing in High-Energy Physics 1991*, pages 59–66, Tokyo, 1991. Universal Academy Press, Inc.

[BH91] J.M. Bower and J. Hale. Exploring neuronal circuits on graphics workstations. *Scientific Computing and Automation*, pages 35–45, March 1991.

[BK90] F. Bancilhon and W. Kim. Object-oriented database systems: In transition. *SIGMOD RECORD*, 19(4):49–53, December 1990.

[Bow92] J. M. Bower. Modeling the nervous system. *Trends Neurosci.*, 15:411–412, 1992.

[BR93] J. M. Bower B. Rasnow, C. Assad. Phase and amplitude maps of the electric organ discharge of the weakly electric fish, apteronotus leptorhynchus. *Comparative Physiology A*, (172):481–491, 1993.

[CLB+93] C. Cruz-Neira, J. Leigh, C. Barnes, S. M. Cohen, S. Das, R. Engelmann, R. Hudson, M. Papka, T. Roy, L. Siegel, C. A. Vasilakis, T. A. DeFanti, and D. J. Sandin. Scientists in wonderland: A report on visualization applications in the cave virtual reality environment. In *Proceedings IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, pages 59–65. IEEE Computer Society Press, October 1993.

[CNSD+92] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The cave automatic virtual environment. *Communications of the ACM*, 35(2):64–72, June 1992.

[CSD93] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In J. T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 135–142, August 1993.

[DB94] E. DeSchutter and J. M. Bower. An active membrane model of the cerebellar purkinje cell: I simulation of current clamps in slice. *Journal of Neurophysiol.*, 71:375–400, 1994.

[DS92] E. De Schutter. A consumer guide to neuronal modeling software. *Trends Neurosci.*, 15:462–464, 1992.

[DSB92] E. De Schutter and J. M. Bower. Purkinje neuron simulation on the intel touchstone delta with genesis. In T. Mihaly and P. Messina, editors, *Proceedings of the Grand Challenge Computing Fair*, pages 268–279, Caltech, Pasadena, CA, 1992. CCSF publishing.

[DSB93] E. De Schutter and J.M. Bower. Integration of synaptic inputs in a model of the cerebellar Purkinje cell. In F. H. Eeckman and J. M. Bower, editors, *Computation and Neural Systems*, pages 355–362. Kluwer Academic Publishers, Boston, 1993.

[EB93] F. H. Eeckman and J. M. Bower, editors. *Computation and Neural Systems.* Kluwer Academic Publishers, Boston, 1993.

[ES91] M. Zwilling E. Skekita. Cricket: A mapped, persistent object store. In S. B. Zdonik A. Dearle, G. M. Shaw, editor, *Implementing Persistent Object Bases: Principles and Practice*, pages 89–102. Morgan Kaufmann, San Mateo, California, 1991.

[FJP90] J.C. French, A. K. Jones, and J. L. Pfaltz. Summary of the final report of the nsf workshop on scientific database management. *SIGMOD RECORD*, 19(4):32–40, December 1990.

[GGR93] J. D. Mackinlay G. G. Robertson, S. K. Card. Information visualization using 3d interactive animation. *Communications of the ACM*, 36(4):57–71, April 1993.

[GLQ93] R. Grossman, D. Lifka, and X. Qin. An object manager utilizing hierarchical storage. In *Twelth Symposium on Mass Storage Systems.* IEEE, IEEE Press, 1993.

[GQ93] R. Grossman and X. Qin. Ptool: A software tool for working with persistent data. Technical Report 93-5, Laboratory for Advanced Computing, University of Illinois at Chicago, 1993.

[Ito84] M. Ito. *The Cerebellum and Neural Control.* Raven Press, New York, 1984.

[IW91] M. Wolczko I. Williams. An object-based memory architecture. In S. B. Zdonik A. Dearle, G. M. Shaw, editor, *Implementing Persistent Object Bases: Principles and Practice*, pages 89–102. Morgan Kaufmann, San Mateo, California, 1991.

[JE92] R. H. Jacoby and S. R. Ellis. Course notes 9. In *Using Virtual Menus in a Virtual Environment, in Implementation of Virtual Environments*, pages 12.1–12.9. SIGGRAPH, July 1992.

[JFL94] A. E. Johnson, F. Fotouhi, and J. Leigh. S.a.n.d.b.o.x.: an interface to scientific data based on experimentation. In *Proceedings of Eurographics '94 - in press*, 1994.

[KEL+92] S. Karin, M. Ellisman, R. Langridge, K. Schulten, and J. Wooley. Visualization in computational biology. In *Proceedings of SIGGRAPH*, pages 404–405, 1992.

[KWA93] K .S. Booth K. W. Arthur. Evaluating 3d task performance for fish tank virtual worlds. *ACM Transactions on Information Systems*, 11(3):239–265, jul 1993.

[LDA+93] J. Leigh, T. A. DeFanti, C. Assad, B. Rasnow, A. Protopappas, E. De Schutter, and J. M. Bower. Neural simulation software demonstration exhibit. Neuroscience 93 Conference, November 1993.

[LDB+94] J. Leigh, T. A. DeFanti, B. B. Blumenthal, R. G. Grossman, D. Bilitch, J. M. Bower, and D. Beeman. The genesis simulation-based neuronal modeling database. *Information Systems-submitted for review*, 1994.

[LKL91] J. B. Lewis, L. Koved, and D. T. Ling. Dialogue structures for virtual worlds. In *Conference Proceedings of SIGCHI'91*, pages 131–136, 1991.

[Pro93] A. Protopappas. Genesis tutorial on the piriform cortex, 1993.

[RRS92] N. L. Max R. R. Springmeyer, M. Blattner. A characterization of the scientific data analysis process. In *IEEE Visualization*, 1992.

[She76] G. M. Shepherd. *The Synaptic Organization of the Brain.* Oxford University Press, New York, 1976.

[SLGS92] C. Shaw, J. Liang, M. Green, and Y. Sun. The Decoupled Simulation Model for Virtual Reality Systems. In *Proceedings of CHI '92*, pages 321–328. ACM, May 1992.

[SZ93] D. J. Surman and D. Zeltzer. A design method for whole-hand human-computer interaction. *ACM Transactions on Information Systems*, 11(3):219–238, July 1993.

[Vas94] C. Vasilakis. User studies for toolkit development in virtual reality. In *Proceedings of Psychology of Programming Interest Group*, 1994.

[WB92] M. Wilson and J. M. Bower. Cortical oscillations and temporal interactions in a computer simulation of piriform cortex. *J. Neurophysiol.*, 67:981–995, 1992.