By Chris R. Johnson, Rob MacLeod, Steven G. Parker, *and* David Weinstein

Biomedical Computing and Visualization Software Environments

Problem-solving environments and advanced visualization take on the complexity of biomedical computing, improving its utility to scientists and clinicians alike.

Over the past two decades, computer simulation and visualization have significantly influenced and improved the field of biomedicine, as they have other areas of science and engineering. While some images and visualizations facilitate medical diagnosis, others help physicians plan surgical procedures. For example, visualization tools have been critical in planning recent brain surgeries to correct large aneurysms in patients being treated by neurosurgeons at the University of Utah Medical Center. Biomedical computing and visualization techniques are used for medical training. Computer simulation lets biomedical researchers subject increasingly sophisticated quantitative and qualitative conceptual models of biological behavior to rigorous quantitative simulation and analysis. Visualizing the results of these simulations helps scientists explore and understand the simulation behavior and results.

The desire to understand biological systems drives researchers to create increasingly sophisticated computational models. While such sophistication is essential to good research, the resulting complexity of the scientific computing process has become a major hindrance to further use of advanced computing tools in biomedical research and patient treatment. Sources of this complexity include the number of equations and variables required to represent realistic functions, the size of the resulting systems and data sets, and the diversity of the

Side view of cortical imaging and dipole source localization. Electric scalp potential recordings (color-coded on the disks on the scalp surface) are used to recover the potential distribution on the cortical surface, as well as the equivalent dipole current sources (shown as arrows embedded in the cortex). Potentials are color-coded: red corresponds to positive values; blue to negative values; and green to values near zero. *Image generated with the SCIRun/BioPSE software system provided by the Scientific Computing and Imaging Institute.*

MANAGING LARGE-SCALE COMPUTATIONS REQUIRES POWERFUL HARDWARE AND EFFICIENT, TRANSPARENT SOFTWARE THAT *frees the user to engage the complexity of the scientific problem,* RATHER THAN THE COMPLEXITY OF THE TOOLS THEMSELVES.

computational resources required to support significant understanding of biomedical phenomena and medical treatment [2]. Biomedical computing researchers collect multichannel and multimodal data from real-time collection instruments, access large distributed databases, and rely on simulation and visualization systems for navigating biomedical systems and models.

Managing these large-scale computations requires powerful hardware and efficient, transparent software that frees researchers and clinicians alike to engage the complexity of the scientific problem, rather than the complexity of the tools themselves. Unfortunately, such biomedical computing software does not exist today.

The current selection of computational tools is increasing so quickly that navigating and choosing from among them has become its own challenge (see the article by Homa Javahery et al. in this issue). The need to integrate software is especially great when scientists seek to build models that span spatial or temporal scales or that cross physical systems. No individual researcher has the skills required to master all the computational and biological knowledge needed to create geometric and mathematical models, map them to numerical algorithms, implement them efficiently in modern computers, visualize the results, and then understand them as they pertain to the specific biological system under investigation. Modeling complex systems requires a multidisciplinary team of specialists, each with complementary expertise and supported by a software infrastructure that integrates that expertise into a complete software system.

Problem-solving environments (PSEs) provide a natural platform for integrating and leveraging multidisciplinary expertise to create complete systems for biomedical computing [2, 3, 7, 9]. Such systems solve the challenge of interfacing disparate elements while also providing a level of functional abstraction researchers need in dealing with complex software systems. PSEs also provide infrastructure for integrating computational knowledge. PSEs may incorporate a number of specific elements, including knowledge of the relevant discipline(s); the most useful computational techniques, algorithms, and data structures; the associated programming techniques; the relevant human-computer interface design principles; the applicable visualization and imaging techniques; and methods for mapping the computations to various computer architectures [2].

PSEs can also consolidate knowledge from a range of experts into a system providing end users a powerful set of computational tools. The Scientific Computing and Imaging (SCI) Institute at the University of Utah has a long history (since 1994) of research involving PSEs for scientific computing, including SCIRun, BioPSE, and Uintah [5, 6, 8].

SCIRun and BioPSE

The SCIRun PSE supports the interactive creation, investigation, and steering of large-scale scientific computations. It has been enhanced through the efforts of two large research centers that use it as their core software system: the Center for the Simulation of Accidental Fires and Explosions (C-SAFE), a U.S. Department of Energy Level 1 Center; and the National Institutes of Health's National Center for Research Resources (www.ncrr.nih.gov) Center for Bioelectric Field Modeling, Simulation, and Visualization. Due largely to these efforts, SCIRun has now evolved into a comprehensive software environment for scientific computing applications.

The SCIRun component model is based on a generalized dataflow paradigm that allows the tight integration of computational components and visualization components. A dataflow model involves three processes in sequence: data is sent to a software component; the component manipulates the data in some manner; and the new data then passes downstream to the next component for further manipulation.

SCIRun can be viewed as a computational workbench in which scientists design and modify simulations interactively via a component-based visual programming model. SCIRun also facilitates interactive debugging and steering of large-scale, typically parallel, scientific simulations by, for example, enabling scientists to modify geometric models and interactively change numerical parameters and boundary conditions. An especially important aspect of SCIRun is that it combines the setting of input parameters, computation, and visualization into one tool that is used in real time, as opposed to the tradi-

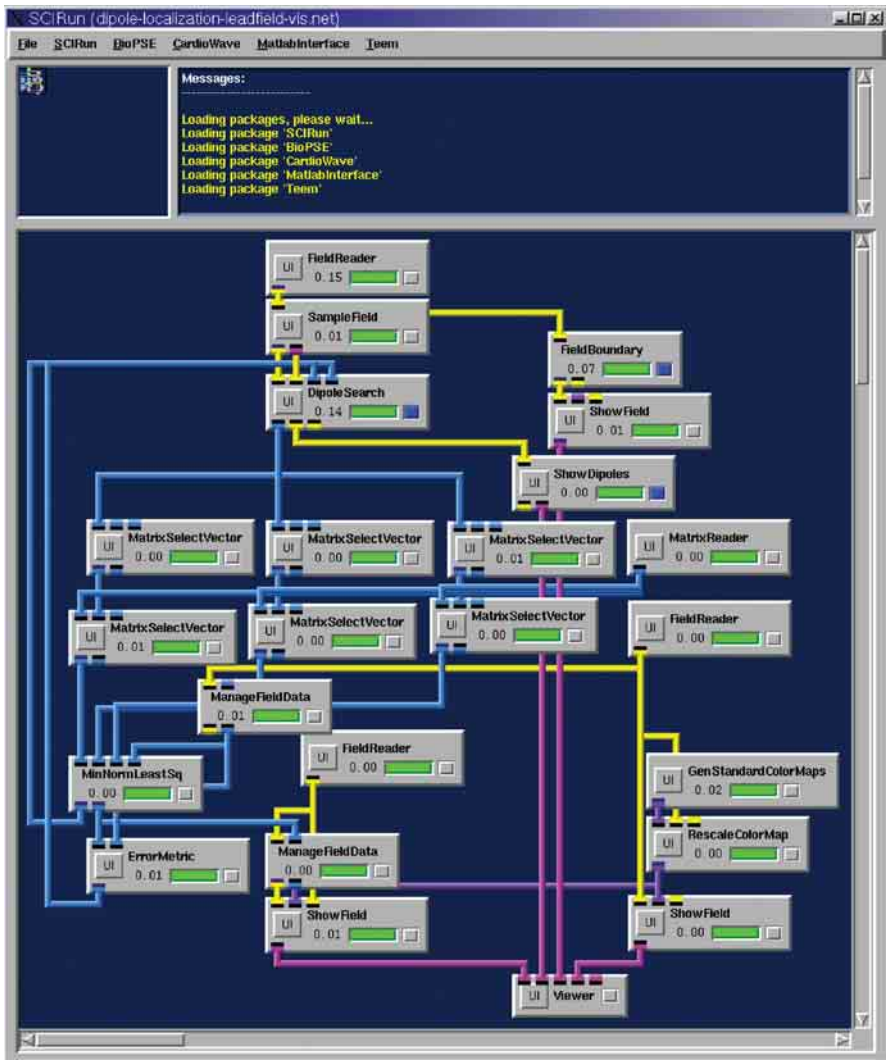

Figure 1. SCIRun/BioPSE modules combined for EEG modeling, simulation, and visualization.

tional practice of doing these three tasks separately and in batch mode.

While SCIRun provides the framework and software infrastructure needed to support this extensive functionality, the individual components encapsulate the actual science one might wish to carry out. The modules are standalone pieces of software designed by various individuals or groups and contributed to the system. Interesting problems are solved through the combined functionality of a number of such modules.

SCIRun consists of a layered set of libraries. Though it uses an object-oriented design, its developers have carefully avoided overuse of the object-oriented paradigm, which could threaten efficiency. In implementing the SCIRun kernel and modules, we have leveraged a powerful toolbox of C++ classes within SCIRun, tuning them for scientific computing and operation in a multithreaded environment, and have employed an efficient reference-counting scheme with smart pointers to allow different modules to efficiently share common data structures.

The BioPSE package introduced a generic Field data structure to SCIRun that is used to represent any geometric domain and a set of data values defined over that domain. The design challenge for Fields was to create a general and flexible data structure without sacrificing system performance. Since Fields are the predominant primitive in the SCIRun dataflow architecture, they must be designed to meet the various needs of our users. Fields have greatly reduced the complexity of many of the existing modules while simultaneously enhancing their functionality.

The ShowField module in BioPSE/SCIRun, for example, is a generic tool for visualizing a Field. It takes an arbitrary Field and a colormap as input, and as output generates geometric scene-graph primitives for subsequent rendering. For different types of Fields the geometric primitives might have elements in common but might also have elements that are distinct. Before BioPSE/SCIRun was available, for every possible combination of geometry, user-selected rendering parameters, colormap options, and data value types, a specific piece of code would be necessary to handle that particular set of options. Maintaining such code is prohibitively complicated. In contrast, by implementing a consistent interface to all of our Field types, we have abstracted away the details of the different geometry types in the ShowField module. This approach to abstracting geometry greatly reduces the number of lines of code required for ShowField, makes the code much easier to understand for both developers and users, and reduces the likelihood that the code will contain bugs. Moreover, it dramatically simplifies the process of adding a new geometry or data value type to BioPSE/SCIRun; a new type just needs to support the required interface in order to be renderable by the ShowField module.

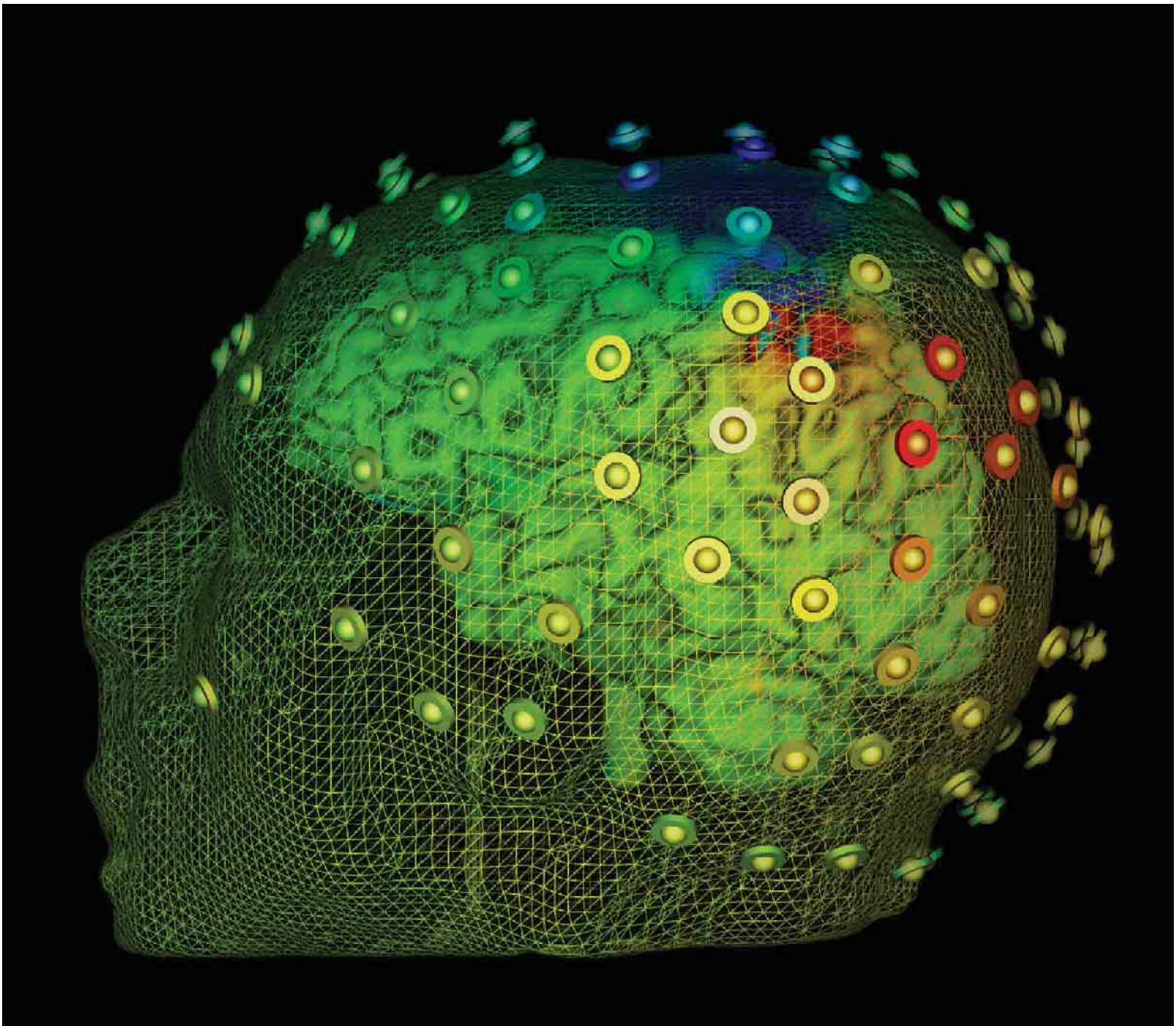

Figure 2. Visualization of results of an EEG simulation localizing a neural source.

A common drawback to generalized interfaces is that generalization typically comes at the price of performance. In order to support a general interface, the code cannot know in advance what type of Fields might arrive as input; optimization is therefore difficult to do. Dynamic compilation and loading is used to permit optimization based on the input types. It compiles optimized code for each specific case (as soon as a module knows which specific types of Fields have arrived as input); the system then dynamically loads the resulting new object and executes it. For users, these operations are all transparent, and for developers they are simple to use. The advantage of this technique is that SCIRun generates only specific optimized code for the types of Fields about which a particular user is interested. (For a complete discussion on dynamic compilation, see [4].)

The next generation of SCIRun (tentatively called SCIRun2, possibly available in 2005) is a framework combining the SCIRun infrastructure with common component architecture (CCA) [1] and connections to other commercial and academic research-oriented component models. SCIRun2 [11] will utilize parallel-to-parallel remote method invocation to connect components in a distributed memory environment and be multithreaded to facilitate shared memory programming. It also will have an optional visual-programming interface. Overall, SCIRun2 will provide a broad approach to component-based biomedical simulation to allow scientists to combine a selection of tools for solving problems of interest. The overarching SCIRun2 design goal—giving computational sci-

WHILE VISUAL PROGRAMMING *is natural for computer scientists* AND FOR SOME ENGINEERS ACCUSTOMED TO WRITING SOFTWARE AND BUILDING ALGORITHMIC PIPELINES, *it is overly cumbersome for many application scientists.*

entists the ability to use the right tool for the right job—is motivated by the diverse needs of biomedical and other scientific users.

EEG simulation and visualization. Figures 1 and 2 illustrate the functionality of SCIRun and BioPSE with an example of electroencephalographic (EEG) source localization. Figure 1 outlines the dataflow network implementing the solution. The input data files loaded at the top of the network include the finite element mesh (FEM) that defines the geometry and conductivity properties of the model and a precomputed lead-field matrix that encodes the relationship between the electric sources in the domain and the resulting potentials that would be measured at the electrodes. Farther down in the network, a set of modules optimizes the dipole location in order to minimize the lack of conformance between the measured potentials from the electrodes and the simulated potentials due to the dipole. Finally, visualization and rendering modules provide interactive feedback to the user.

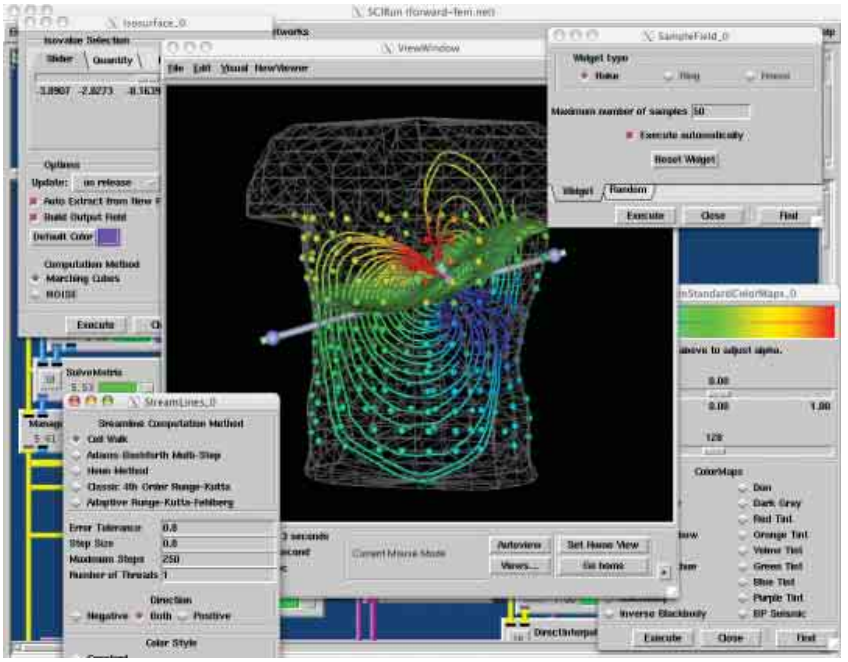

Figure 3. User interfaces compared: (a) BioPSE dataflow interface for a bioelectric field application; (b) BioFEM custom interface. Though the BioFEM application is functionality equivalent to the dataflow version for BioPSE, the BioFEM version provides an easier-to-use custom interface.

A major hurdle to SCIRun becoming an everyday tool for scientists and engineers has been its dataflow interface. While visual programming is natural for computer scientists and some engineers accustomed to writing software and building algorithmic pipelines, it is overly cumbersome for many application scientists. Even when a dataflow network implements a specific application (such as the bioelectric field simulation network provided with BioPSE and detailed in the BioPSE Tutorial), the user interface components of the network are presented to the user in separate user-interface windows, without any context for their settings. For example, although SCIRun provides a generic file-browser user interfaces for reading-in data, many such user-interface elements exist in the dataflow network, each with the same generic presentation, and are thus indistinguishable. The user cannot tell which type of file each browser expects.

One approach to improving user-interface support for SCIRun is to create PowerApps, or a customized interface built atop a dataflow application network; see [10] for examples. The dataflow network controls the execution and synchronization of the modules comprising the application, but SCIRun developers have consolidated the generic user interface windows, replacing them with an application-specific interface.

Included with the version 1.22 release of BioPSE (July 2004) is a PowerApp called BioFEM based on a FEM network; Figure 3 shows the striking difference in user interfaces between BioPSE and BioFEM. In the dataflow version of the application, users have separate interface windows for controlling different aspects of the simulation and visualization, but nothing about these windows indicates the context for the associated controls. The PowerApp version of the same functionality, by contrast, has a

THE FUTURE OF BIOMEDICAL COMPUTING DEPENDS ON AN INTEGRATED SOFTWARE ENVIRONMENT *that allows multidisciplinary specialists to contribute specific tools to it without having to be expert in all other fields.*

much more streamlined and context-associated interface and a much simpler appearance. The full flexibility of the more general dataflow interface is hidden but still available should the researcher require functionality not provided by the specialized interface.

Future Biomedical Software

The future of biomedical computing depends on an integrated software environment that ideally combines software from multiple sources into a computational workbench, and that allows multidisciplinary specialists to contribute specific tools to the workbench without having to be expert in all other fields. Creating a dramatically better computational workbench requires five major advances in software engineering:

- Allowing biomedical software tools to seamlessly interoperate with one another;
- Uniting state-of-the-art tools created by different biomedical researchers to perform state-of-the-art computational experiments;
- Making tools accessible to other researchers through Web-based and/or grid-based mechanisms;
- Making tools more powerful through parallel and distributed implementations and through performance improvements; and
- Making tools easy to use, accurate, and reliable through sophisticated user-interface development and robust software engineering.

Leveraging the lessons we've learned creating the SCIRun, BioPSE, and Uintah PSEs, we are now developing a new component-software architecture to create a biomedical software environment that addresses the features required to make advanced PSEs an everyday tool for researchers and clinicians. C

REFERENCES

1. Armstrong, R., Gannon, D., Geist, A., Keahey, K., Kohn, S., McInnes, L., Parker, S., and Smolinksi, B. Toward a common component architecture for high-performance scientific computing. In *Proceedings of the Eighth IEEE International Symposium on High-Performance Distributed Computation* (Redondo Beach, CA, Aug. 3–6). IEEE Computer Society, 1999.
2. Bramley, R., Char, B., Gannon, D., Hewett, T., Johnson, C., and Rice, J. Enabling technologies for computational science: Frameworks, middleware, and environments. In *Proceedings of the Workshop on Scientific Knowledge, Information, and Computing,* E. Houstis, J. Rice, E. Gallopoulos, and R. Bramely, Eds. Kluwer Academic Publishers, Boston, 2000.
3. Bramley, R., Gannon, D., Stuckey, T., Villacis, J., Balasubramanian, J., Akman, E., Breg, F., Diwan, S., and Govindaraju, M. Component architectures for distributed scientific problem solving. *IEEE Comput. Sci. Engineer. 5,* 2 (1998), 50–63.
4. Cole, M. and Parker, S. Dynamic compilation of C++ template code. *Scientific Programming* (2003), 321–327.
5. de St. Germain, J., Parker, S., McCorquodale, J., and Johnson, C. Uintah: A massively parallel problem-solving environment. In *Proceedings of the Ninth International Symposium on High-Performance Distributed Computing* (Pittsburgh, PA, Aug. 1–4, 2000), 33–42.
6. Johnson, C., Parker, S., and Weinstein, D. Component-based problem-solving environments for large-scale scientific computing. *Concurrency and Computation: Practice and Experience 14* (2002), 1337–1349.
7. Johnson, C. and Parker, S. Applications in computational medicine using SCIRun: A computational steering programming environment. In *Proceedings of Supercomputer '95* (San Diego, Dec. 4–8), IEEE Press, 1995, 2–19.
8. Parker, S., Weinstein, D., and Johnson, C. The SCIRun computational steering software system. In *Modern Software Tools for Scientific Computing*, E. Arge, A. Brauset, and H. Langtangen, Eds. Birkhauser Press, Cambridge, MA, 1997, 1–44.
9. Parker, S. and Johnson, C. SCIRun: A scientific programming environment for computational steering. In *Proceedings of Supercomputer '95* (San Diego, Dec. 4–8), IEEE Press, 1995.
10. Scientific Computing and Imaging Institute. SCIRun and BioPSE PowerApps; available from www.sci.utah.edu.
11. Zhang, K., Damevski, K., Venkatachalapathy, V., and Parker, S. SCIRun2: A CCA framework for high-performance computing. In *Proceedings of the Ninth International Workshop on High-Level Parallel Programming Models and Supportive Environments* (Santa Fe, NM, Apr. 26–30). IEEE Computer Society Press, 2004.

CHRIS R. JOHNSON (crj@sci.utah.edu) is director of the Scientific Computing and Imaging Institute and Distinguished Professor of Computer Science and Director, School of Computing, the University of Utah, Salt Lake City.

ROB MACLEOD (macleod@cvrti.utah.edu) is an associate director of the Scientific Computing and Imaging Institute and associate director of the Cardiovascular Research and Training Institute at the University of Utah, Salt Lake City.

STEVEN G. PARKER (sparker@cs.utah.edu) is a research assistant professor in the School of Computing and the Scientific Computing and Imaging Institute at the University of Utah, Salt Lake City.

DAVID WEINSTEIN (dmw@sci.utah.edu) is technical manager of the National Institutes of Health NCRR for Bioelectric Field Modeling, Simulation, and Visualization in the Scientific Computing and Imaging Institute at the University of Utah, Salt Lake City.

The authors gratefully acknowledge support from National Institutes for Health NCRR and Biomedical Information Science and Technology Initiative, National Science Foundation Partnerships for Advanced Computational Infrastructure, and from the Department of Energy Advanced Simulation and Computing Initiative and Scientific Discovery through Advanced Computing program. SCIRun, BioPSE, and the PowerApps software are all available as open source from the SCI Institute's Web site (www.sci.utah.edu). SCIRun, BioPSE and the PowerApps are supported on three different platforms: Linux (Red Hat 8.0/9.0 and Mandrake 9.0), Macintosh (OSX 10.3), and SGI (IRIX 6.5).

Physical model of simulated coral produced using a 3D printing technique (selective laser sintering). Thousands of independent virtual polyps grew this structure themselves in a simulated reef environment over 24 hours on a 16-processor Linux cluster.

Simulations: Roeland M.H. Merks, Alfons G. Hoekstra, Jaap A. Kaandorp, and Peter M.A. Sloot, Computational Science, University of Amsterdam, The Netherlands.
Photography: Ronald van Weeren, Artis Zoo, Amsterdam, The Netherlands.