Electronic “How Things Work” Articles: Two Early Prototypes

Franz G. Amador, Deborah Berman, Alan Borning, Tony DeRose, Adam Finkelstein, Dorothy Neville, David Notkin, David Salesin, Mike Salisbury, Joe Sherman, Ying Sun, Daniel S. Weld, and Georges Winkenbach

Abstract—The Electronic Encyclopedia/Exploratorium (E²) is a vision of a future computer system—a kind of electronic “How Things Work” book. Typical articles in E² will describe such mechanisms as compression refrigerators, engines, telescopes, and mechanical linkages. Each article will provide simulations, three-dimensional animated graphics that the user can manipulate, laboratory areas that allow a user to modify the device or experiment with related artifacts, and a facility for asking questions and receiving customized, computer-generated English-language explanations. In this paper, we discuss some of the foundational technology—especially focusing on topics in artificial intelligence, graphics, and user interfaces—needed to achieve this long-term vision. We describe our initial prototype system and the technical lessons we have learned from it, as well as our second prototype currently under construction.

Index Terms—Electronic books, electronic encyclopedia, hypermedia, interactive CAD systems, interactive graphics, interactive simulation, model-based reasoning, user interface.

I. MOTIVATION

HYPERMEDIA systems have recently enjoyed considerable attention, both within the research community and in the popular press. Few question the technology’s promise, but current systems offer only a glimpse of the potential of interactive documents. Current hypermedia documents, such as those produced by HyperCard or MacroMind Director, are in essence finite automata in which a graph of actions is traversed in response to the user’s inputs. This approach has the drawback that authors must anticipate and script for all possible user interactions.

We concentrate instead on richer kinds of interaction. In particular, we are interested in understanding how to design and construct articles for an intelligent “How Things Work” electronic book. Articles might describe mechanisms such as engines, telescopes, and mechanical linkages. We envision these articles as having a collection of “simulators” and “laboratory areas” that the reader can interact with in flexible, unanticipated ways. Our goal is to allow readers to learn both how and why specific behaviors occur in the domain described by the article.

To focus our search for general-purpose tools, we are concentrating on our first article, which covers compression refrigeration and requires describing, simulating, and explaining material in the domains of thermodynamics, electricity, and magnetism. A first-time reader of the article might run a simulation of the heating and cooling process that occurs when the refrigerator door is open. A more advanced reader might change parameters, such as the thermodynamic properties of the refrigerant, to learn more about how this thermal process works. Even more advanced readers might wish to experiment further, crafting their own refrigerators by combining predefined components such as compressors and evaporators. In addition, readers of the article will be able to ask questions, ranging from the simple “What devices are connected to the expansion valve?” to the complex “What happens to efficiency as the boiling point of the refrigerant is increased?” or “How is the compressor removed from the chassis?”

Our models will ideally include the information needed for animated graphical depiction of the compressor, refrigerant flow, phase changes, control system, and so forth. Both qualitative and quantitative descriptions will be used: detailed thermodynamic equations will drive the numeric simulation, and qualitative descriptions will be used to produce causal, English-language explanations.

In the long term, the ability to simulate the behavior of reader-defined assemblies and to receive customized explanations of them will allow readers to explore a vastly richer document than is possible with existing systems. The author’s effort will be combinatorially amplified by the use of simulation and automatic explanation.

Our first two prototypes, which we describe here, make progress toward the underlying technology of this vision. Developing these foundations—as opposed to building a comprehensive knowledge base—is the primary objective of our Electronic Encyclopedia/Exploratorium (E²) effort. In particular, problems need to be solved in (at least) artificial intelligence, graphics, and user interfaces.

* In artificial intelligence, the two key problems are automated model management and the automatic generation of explanations. Automated model management is used to dynamically choose simplifying assumptions and perspectives that are appropriate to the task at hand (for instance, an appropriate simulation). Automatic generation of explanations is needed to overcome the impossibility of anticipating and storing all queries.

Manuscript received June 1, 1992; revised December 1, 1992. This research was funded in part by the National Science Foundation under Grants IRI-8 957 302 (Weld), CCR-9 113 367 (Notkin), CCR-8 858 804 (Notkin), IRI-9 102 938 (Borning), and CCR-8 957 323 (DeRose); by the Office of Naval Research under Grant 90-J-1904 (Weld); by the Research Institute in Software Engineering (DeRose, Notkin); and by the Xerox Corporation (DeRose, Notkin, Weld).

The authors are with the Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195.

IEEE Log Number 9209927.

1047-4347/93$03.00 © 1993 IEEE
In graphics, the key problem is the automatic generation of presentations, since it will not be possible to store all images that might be useful to display. This problem is analogous to the automatic generation of explanations in AI.

In user interfaces, the key problem is defining a metaphor that gives readers powerful yet intuitive ways of manipulating the mechanisms, simulations, and toolkits. For example, it must be simple for the reader to connect components (such as compressors and pipes) as well as to place instruments (such as thermometers and strip chart recorders) on appropriate devices. A particular focus of the E3 user interface is facilitating the viewing and manipulation of multiple models at the UI level, in addition to their use in the underlying AI subsystems.

To make systems like E3 feasible, it is also essential to integrate solutions to these problems. Our first two prototypes are increasing our insight into the kinds of software architectures that will give us the power and the flexibility needed to construct such systems.

II. First Prototype

Our first prototype, developed in about six months, served two functions. First, it gave us a firmer understanding of the technical challenges of our long-term E3 vision. Second, it provided a quick look-and-feel prototype that helped us refine that vision.

The prototype emulated the style of Van Amerongen's book, The Way Things Work [15], which describes the composition and function of mechanical and electrical devices. In the prototype, the reader can view and query several relatively simple models of a compression refrigerator. One model describes the structural relationships among the refrigerator's parts, and is useful for answering queries such as "What parts comprise the refrigerator?" and "What parts are connected to the compressor?" Another model describes the geometry of the refrigerator using constructive solid geometry (CSG) expressions [12], allowing the user to view the refrigerator from any angle. A thermal model allows the generation and simulation of some of the heating and cooling properties of the refrigerator, and can be used for queries such as "How long does it take for the compressor to turn on when the refrigerator door is opened?"

In addition to the above two goals of the prototype, we had several lower-level objectives. First, we wanted to investigate the usefulness of a knowledge representation system for storing and manipulating E3 models. Second, we wanted to investigate powerful schemes for rendering 3D CSG representations. And third, we wanted to investigate the usefulness of user interface management systems in producing E3 articles.

2.1. Architecture

At the core of the first prototype is a declarative structural model written in LOOM [1], [8], a well-known knowledge representation system. The structural model describes a device's components and the connections between them. There is also a dynamics model, which encodes the underlying physics of devices. It includes differential-equation descriptions of all possible physical processes (such as boiling) and the conditions necessary for their occurrence. These may depend on the status of the refrigerator; for instance, an external cooling process occurs when the refrigerator door is opened. The simulation component of the prototype assembles the active processes, collects their equations, and simulates them via a Runge-Kutta numeric integration package. This simulation gives the behavior of the refrigerator's internal temperature over time, as well as the times when the compressor switches on and off.

Three-dimensional objects, such as the refrigerator itself, are described using a CSG representation, which defines scenes of composite objects in terms of three-dimensional geometric primitives, transformations, and set operations. CSG has the advantage that it can represent compactly and exactly very complex models. It is also relatively easy for an author to create complex objects with CSG expressions because set operations parallel manufacturing operations.

The user interface encompasses the structural, dynamics, and CSG models, and it invokes the renderer and simulation components. Using a mouse, the reader navigates through the device and reads descriptions of each part. Invoking the renderer allows the reader to view 3D images of the device, using the mouse to change the eye point and viewing direction. The reader can also use a dialog menu to query selected state attributes of the refrigerator's components using the structural model. The user interface window appears in Fig. 1.

The prototype was developed on a Sun Sparcstation running UNIX and X11. The structural and dynamics models were written in Common Lisp and used the LOOM knowledge representation system. The rendering software was written in C. The user interface was written in Common Lisp using the CLIM user interface manager.

2.2. Experience

We learned a variety of lessons from the first prototype, many of which formed the basis for the design of the second prototype (Section III).

We found that that a knowledge representation system, such as LOOM, is an appropriate way to describe both device/component hierarchies and the interconnections between components. However, the abstractions in the first prototype were quite simple, and the challenge now is to model the structure and behavior of more complex models that accurately portray real mechanisms. To manage this complexity our models need to support multiple device views. For example, we may wish to view only the electrical, thermal, or fluid-flow aspects of a real refrigerator, or a composite view of several aspects.

One weakness in the first prototype is that the connections among parts in the domain model are untyped. That is, the structural model does not specify whether a compressor can be attached directly to an evaporator, or if there needs to be a pipe in between, thus potentially allowing the assembly of nonsensical devices. This motivated our decision to support typed connections in the second prototype.
There are several weaknesses in the graphic component of the first prototype. Most importantly, the 3D images are not interactive. In particular, the renderer cannot respond to user actions other than moving the viewing position, since backmappings from the graphical representation to the CSG (and LOOM) representations are not retained. However, the startup of the scene is very fast compared to ray tracing, and redisplay of a rotated scene can be performed in near-real time.

We found several strengths and weaknesses in using CLIM as our user interface manager. CLIM does provide sufficient power to build the 2D schema of the refrigerator. However, it does not provide a way of incorporating icon bitmaps, and it is insufficient for handling 3D images. In addition, CLIM does not provide much leeway in tailoring the look and feel of the user interface.

III. SECOND PROTOTYPE

Our second prototype, which is nearing completion, improves significantly upon the first prototype, and includes enhancements to the artificial intelligence, graphics, and user interface components. Its user interface is based upon a "laboratory" metaphor, in which the user can construct and simulate refrigeration devices. This prototype also handles a greater variety of models than did the first.

Each primitive component has a typed structural description (Section 3.1), an iconic representation (Section 3.2), a dynamics description (Section 3.3), and a geometric description (Section 3.4). A compressor's structural description, for example, states that it has two ports for pipe connections and two for electrical connections; its dynamics description specifies the processes relating voltage, fluid flow, and pressure; and its geometric description defines its 3D appearance.

There are potentially many structural, dynamics, and geometric descriptions for each primitive component. For example, there might be both a stylized and a detailed geometric model; similarly, there could be both simple process models and ones that account for complex behavior such as turbulence. Depending on the interests and questions of the user, different models will be selected.

3.1. Structural Descriptions

Structural descriptions model devices, components, and their interrelations. Because of our successful experience with LOOM in the first prototype, we continue to use it as the central repository for these structural descriptions.

There are two basic relationships among devices: composition describes how one device is built out of other devices, and connection describes how devices are connected within a containing device. Thus, composition defines a hierarchy of devices; that is, the components of a device are themselves devices.

To describe multiple views of devices, we define connection types and constrain how one device component connects to others. Thus, components may have one or more typed terminals that define where and how a component may connect with other components. Terminals are connected together by nodes, which enforce type compatibility.

Encoding the structural description in this way in the knowledge base makes it simple to answer queries such as "What are the components of a device?" "What components are connected to a device?" "What connections are available for a given component?" "What types of connections exist for a component?" "What state variables exist for a component,
3.2. User Interface

Because simulation is at the heart of E³, we have designed the user interface for the second prototype around a "laboratory metaphor," in which readers can explore and manipulate simulations of refrigerators.

The laboratory defines two basic sets of entities. The first set is a "kit" of useful components that can be easily assembled in different ways to build a variety of devices and demonstrations. The components for the refrigerator are a compressor, an expansion valve, an evaporator, a condenser, and a collection of pipes for attaching them together. The second set is a collection of measuring tools that can sense attributes such as temperature, pressure, and voltage. This set also includes strip chart recorders and other tools for displaying changes of variables over time. Both device components and measuring tools are structurally modeled as components with terminals that allow their interconnection.

Overview: At the bottom of the interface display is a set of component icons, each of which has a set of possible visual behaviors (e.g., movements of the needle on a gauge dial). These icons represent component types that the user graphically instantiates by dragging into the central laboratory work area, where they can be connected together to create devices (see Fig. 2). The devices can be saved and reloaded later. A set of sample devices with which the user can experiment can be provided.

Components are connected by means of terminals, which appear as small plugs on each icon. Terminals come in a variety of types depending on their function (e.g., electrical, thermal, physical). When the user drags a terminal (and associated component), it "sticks" to nearby compatible terminals, thereby providing dynamic feedback to the user as to whether the connection is permitted. Terminals are connected using special components that typically represent physical, real-world connectors such as wires or pipes. When sufficient connections have been made, the components' behavior is simulated, and the simulation results drive animation.

User Interface Toolkits: As noted earlier, we found several weaknesses in using CLIM in constructing the first prototype. After careful consideration, we decided to adopt Garnet [10] as a user interface support system, for several reasons. First, it is much easier to build new look-and-feel styles in Garnet than in CLIM, making such experimentation easier in E³. Second, Garnet provides an interface builder, in most cases saving the developer from having to write significant amounts of Lisp code. Third, a Garnet interface is described using multi-way constraints, which is an approach we have found useful in constructing user interfaces [9]. Finally, we have Common Lisp source code available for Garnet, allowing us to experiment with different UI support system architectures.

3.3. Dynamics Modeling and Simulation

This section describes our architecture for flexible simulation. We start by briefly describing the Quantified Modeling Language (QML), our language for specifying the components of a device and the physical laws that these components obey. We then describe how simulations are constructed automatically.

The QML Language: QML is a high-level language that allows a model builder to specify a set of ordinary differential equations (ODE's) that model a device. The E³ system compiles QML models at run time into a set of ordinary, algebraic, and differential constraints, which are passed to a constraint
The solver passes control back to the simulation engine when one or more of the bounding constraints has been violated. Many of these constraints were generated by the simulation engine as a result of quantitative preconditions in the active and inactive model fragments. For active model fragments, constraints are generated to ensure that quantitative preconditions remain true during simulation. For inactive model fragments, the simulation engine generates constraints that will be violated when preconditions are newly satisfied, causing new model fragments to become active. In any case, when the solver terminates, the simulation engine must re-evaluate the model fragments to determine the new active set. Thus the overall simulation process forms a cyclic pattern of determining active model fragments, compiling the ODE constraints from the model fragment effects, and simulating until bounding constraints are violated.

There are several benefits from constructing the ODE equational model dynamically. First, it allows modular model specification. Second, E3 can provide extra information during simulation. For example, a user can ask which processes are influencing any state variable at any time, since this information is easily derived from the set of model fragments that are active at that time. Finally, E3 will be able to display the equations it is using for simulation, even as those ODE’s change over time. These aspects can be combined with a flowchart style of display that illustrates how the model fragments interact.

### 3.4. Geometric Descriptions

The second prototype makes several improvements in the 3D viewer, which is responsible for producing three-dimensional images of varying quality. One key improvement is the development of several more powerful tools, including a more flexible viewer. Another is the ability to map selections of objects in the 3D view back into the CSG and structural representations. This supports user operations such as graphically opening the refrigerator door or selecting devices to expand into components.

The CSG definition language remains unchanged from the first prototype. However, we take advantage of more of its features. First, when CSG objects are created, each object is associated with a unique name. This enables us to keep information allowing backmapping from the rendered image to the CSG description. Second, each object also carries a set of attributes such as hide/show, transparent, and selected. Viewing commands modify the attributes associated with a specified object. For instance, the main body of the refrigerator can be made partially transparent to reveal the components within.

```prolog
(define-model-fragment contained-liquid(?container ?liquid)
  (preconditions (> (weight-of-in ?liquid ?container) 0)
    ...
  )
  (effects (= (pressure-at-bottom ?container)
    (/ (weight-of-in ?liquid ?container)
      (area ?container))))
)```
In addition, objects can be dynamically added or removed from the scene. Hence, the view can be dynamically updated, which is useful when the model must be refined in response to some user query. For example, the compressor can be represented initially by a single object. Should the user ask a question about its functioning, it can be replaced by a more detailed representation extracted from the LOOM structural hierarchy.

One shortcoming of the graphics support in both prototypes is that the 3D graphical representation is less connected to the 2D schematic representation than is desirable. In particular, the 3D view currently exists in a window separate from the rest of the interface. (Neither CLIM nor Garnet is especially helpful in handling graphics produced through systems not written in Common Lisp.)

IV. RELATED WORK

In one of the earliest electronic encyclopedia projects [17], readers were presented with multimedia articles on a variety of topics, including articles with simulations of physical phenomena such as Hook's Law and resonance behavior. The system was a significant advance at the time. However, the simulations were hand-coded in Lisp, and the scope of manipulation the user could perform was fixed and limited. The E³3 system adds flexibility and semantic knowledge to electronic encyclopedias.

The ARIA system of [14] had a responsive, real-time, qualitative dynamic simulator, designed to explain the underlying physics of xerography and copier processes. However, like the Weyer and Borning system, ARIA was limited to a fixed set of simulations that could not be constructed dynamically in response to user actions. Similarly, the STEAM project [7] produced an impressive interactive simulator/tutor for a naval propulsion system, though all the simulations and explanations were hand-crafted in advance.

Forbus and Falkenhainer [5] pioneered the idea of custom generation of self-explanatory simulations, but their approach did not scale to devices of an interesting size. They have since created a second implementation [6], but their simulation generation algorithm still has exponential time and space complexity because of its reliance on an assumption-based truth maintenance system (ATMS) [2]; as a result, their SIMGEN program takes hours to construct simulations, ruling out the real-time operation necessary for E³3. Our QML simulation construction algorithm (Section 3.3) is much faster; initial experiments suggest that E³3 can create custom simulations in seconds. We note that the "How Things Work" project at Stanford [4] is addressing issues similar to those tackled by E³3 (with the exception of graphics); however, the Stanford work is too preliminary to discuss extensively.

COMET (COordinated Multimedia Explanation Testbed) [3] and IBIS (Intent-Based Illustration System) [13] are experimental systems for interactively generating multimedia instruction for equipment maintenance and repair. These systems coordinate to take the user through a series of steps of a repair or troubleshooting procedure. The user can place a query at any time, in response to which the systems will generate a customized explanation and accompanying illustrations. COMET and IBIS are built around a diagnostic expert system and several knowledge bases. Although E³3 shares many aspirations with these systems, E³3 does not emphasize generation of natural language and its coordination with graphical images. However, by allowing users to assemble new devices and by constructing custom simulations for them, E³3 allows a more varied style of interaction than COMET and IBIS, which are specific to troubleshooting the behavior of a single device.

V. CONCLUSION AND FUTURE WORK

Building articles for intelligent, electronic books is challenging. It requires solving a variety of technical problems in areas as diverse as artificial intelligence, graphics, and user interfaces. In addition, it requires a sense of the kinds of readers and authors that will use such a system. During the first 18 months of the E³3 project, we have focused on making progress in both of these directions through the rapid design and construction of prototype articles on compression and refrigeration.

We have made progress in each of the technical areas. In artificial intelligence, we have supported the automatic management of models of devices, as well as the automatic generation of responses to queries about the devices. In graphics, we have produced high-quality 3D images whose underlying structural representations can be manipulated interactively. In user interfaces, we have defined a laboratory metaphor that appears to be rich enough to support many of the kinds of tools and articles we anticipate.

Several of these and other results in the E³3 project to date are contributions within their own research areas. In addition, the E³3 project provides a large, demanding, practical application that makes sure we are effective in integrating these pieces into a cohesive whole.

Our medium-term approach is to design, build, and evaluate prototypes at the rate of roughly one every 12 months. Soon we expect to include articles on topics other than compression and refrigeration. We intend, at the very least, to determine how successful we have been in making the system sufficiently general to handle many different types of devices and scientific phenomena.

In addition to improving on the results discussed throughout the paper, there are several technical areas in which we plan to focus.

• While our modeling and simulation system automatically generates and explains device simulations, it does not yet allow for coordinated use of multiple models for a single device. We are working to add this capability so as to synchronize multiple simulations of a device from various ontological perspectives, e.g. fluid, thermal, or electrical. This capability is needed to support new user-interface tools, such as the "molecular magnifier" described below.

• There are (at least) two central graphics research challenges in E³3 that we have not explored in detail yet. One is the development of algorithms that dynamically synthesize geometric models from their structural descriptions. This problem is difficult because the structural
description only partially specifies the details necessary to produce an image; the choice of perspective, illumination, and transparency of surfaces must be made after analyzing the query. The other is generating non-photorealistic illustrations that convey information better than traditional photorealistic rendering. Thus, some of our future research will be devoted to the development of mechanisms to support the automatic generation of effective illustrations from geometric models.

- We plan to include two sets of tools for the user interface that act on the simulation, rather than within it. One set of tools helps the user manage the models and views of the simulation. The other set of tools are used to help explain the workings of the simulation by such means as annotating the device schematic or displaying animations of microscopic phenomena within the device. For example, we envision a "molecular magnifier" tool, which, when used to view the macroscopic behavior of a fluid, simulates the molecular kinetics of a portion of the fluid and displays the behavior of a representative population of the fluid's molecules.

- We must also address how authors construct $E^3$ articles, in addition to considering how readers interact with them. We expect domain experts to fill the authoring role, and we do not anticipate that they will be expert programmers. Providing an effective interface for authoring is necessary for our vision of a large collection of articles to be realized.

Achieving our five- to ten-year vision of a collection of intelligent How Things Work articles would have considerable relevance for several application areas, including not only encyclopedias of the future, but also education in science and technology, training and retraining, self-documenting systems, and the automatic creation of intelligent manuals for manufactured products.

ACKNOWLEDGMENT

Richard Anderson contributed to the initial $E^3$ concept and provided help with this paper. Craig Chambers advised on QML design. Gretchen Karonji motivated us to think about the issues of combining macro and thermodynamic issues in the refrigeration. Michael Sannella helped significantly in understanding the power and weaknesses of Garnet. David Vogt helped produce the graphics viewer on the HP.

REFERENCES


Franz G. Amador received the B.A. degree in 1985 from Swarthmore College and the M.S. degree in computer science in 1990 from the University of Washington.

He is currently working toward the Ph.D. degree in computer science at the University of Washington. His current research interests include automated modeling, simulation, and explanation of physical systems.

Debbie Berman is a second-year Ph.D. student at the University of Washington.

Her interests include user interfaces and graphics. She spent last summer at Apple Computer designing and prototyping a user interface. Prior to graduate school, Debbie worked at Bellcore in the Multimedia Communications Research.

Alan Borning received the B.A. degree in mathematics in 1971 from Reed College and the Ph.D. degree in 1979 from Stanford University in computer science.

He joined the University of Washington in 1980, where he is currently an Associate Professor in the Department of Computer Science and Engineering. Previously, he spent a year as a post-doctoral fellow at the University of Edinburgh, Edinburgh, Scotland. His principal research interests are in computer languages and systems based on constraints. He has also done work on object-oriented languages, logic programming languages, user interfaces, and computers and society.
Tony DeRose received the B.S. degree in physics in 1981 from the University of California, Davis, and the Ph.D. degree in computer science in 1985 from the University of California, Berkeley.

He is currently an Associate Professor of Computer Science at the University of Washington. He received the Presidential Young Investigator award from the National Science Foundation in 1989. He has served on numerous technical program committees, including ACM SIGGRAPH, and he is an associate editor of *ACM Transactions on Graphics*.

From September 1991 to August 1992 he was on sabbatical leave at the Xerox Palo Alto Research Center and at Apple Computer.

Michael Salisbury received the B.S. degree in physics and computer science in 1990 from the California Institute of Technology and the M.S. degree in computer science in 1992 from the University of Washington. He is currently working toward the Ph.D. degree in computer science at the University of Washington.

His current research interests include user interfaces and graphics.

Joseph Sherman received the B.A. degree in 1980 from Carroll College and the M.S. degree in computer science in 1992 from the University of Washington.

He is currently working towards the Ph.D. degree in computer science at the University of Washington. His current research interests include interactive user interfaces, computer generated explanations, and computer-based learning.

Ying Sun received the B.S. degree in computer science in 1989 from Zhejiang University, P.R. China, and the M.S. degrees in electrical engineering (in 1991) and computer science (in 1992) from the University of Washington.

She is currently a Ph.D. student at the University of Washington. Her areas of interest include artificial intelligence, knowledge-based systems and model-based diagnosis.


He is an Associate Professor in the the Department of Computer Science and Engineering at the University of Washington. His research interests are primarily in software engineering, especially evolution and environments, with secondary interests in parallel and distributed systems. He is the program chair for the ACM SIGSOFT ’93 Symposium on Foundations of Software Engineering and the program co-chair for the 17th International Conference on Software Engineering.

David Salesin received the Sc.B. degree from in 1983 Brown University and the Ph.D. degree in 1991 from Stanford University.

He is an Assistant Professor in the Department of Computer Science and Engineering at the University of Washington. He spent the 1991–1992 academic year on leave as a Visiting Assistant Professor in the Program of Computer Graphics at Cornell University. His primary research interest is in computer graphics, especially techniques for image synthesis. He also has interests in user interfaces and computational geometry.

Georges Winkenbach received the diploma in electrical engineering from the School of Engineering de Canton de Neuchatel in Switzerland, and the B.S. and M.S. degrees in computer science from the University of Washington, where he is a Ph.D. student.

His research interests include high quality computer graphics, geometric and solid modeling, and modeling systems for novice users.