

Wizard of Oz Support throughout an Iterative Design Process

A case study of a location-aware audio experience using the authors' DART design environment shows how the Wizard of Oz prototyping method can work throughout an iterative design process.

The Wizard of Oz prototyping approach, widely used in human-computer interaction research, is particularly useful in exploring user interfaces for pervasive, ubiquitous, or mixed-reality systems that combine complex sensing and intelligent control logic. The vast design space for such nontraditional interfaces provides many possibilities for user interaction through one or more modalities and often requires challenging hardware and software implementations. The WOz method helps designers avoid getting locked into a particular design or working under an incorrect

set of assumptions about user preferences, because it lets them explore and evaluate designs before investing the considerable development time needed to build a complete prototype.

As with other throwaway tools, most WOz interfaces aren't conceived to evolve with the system. So, designers tend to use WOz studies once (or perhaps twice) during a system's evolution, in sharp contrast to other evaluation methods that might be used repeatedly as the system evolves. This is unfortunate, as WOz studies can help designers evaluate partially complete systems and advance the underlying technology's design.

One reason for the method's limited use is the effort required to engineer a successful WOz interface and integrate it with an incomplete sys-

tem. To address this problem, we built explicit WOz support into the Designer's Augmented Reality Toolkit,¹ a design environment for augmented and mixed-reality systems. Because designers using DART will find it easier to create and reuse WOz interfaces throughout the design cycle, they'll be more likely to use this evaluation method.

Based on our work and our survey of the literature (see the sidebar), we believe opportunities exist to better exploit the WOz strategy. This article expands on our initial work² by introducing a preliminary framework for wizard roles, providing more detail about wizard tool support in DART, and supporting post-WOz data analysis. If WOz tools become a fluid component of a natural design process, wizards can take on various roles with differing levels of responsibility for facilitating interaction between computing systems and users.

WOz roles in iterative design

Interactive system design typically involves an iterative process of brainstorming, prototyping, development, user testing, and evaluation. This isn't a clear-cut process; it often iterates through many cycles before reaching a final system. Figure 1 illustrates how we can use WOz throughout the design process, as a system evolves. In practice, the development cycle is much more complicated, including the likelihood that the vision for the system will change at various points, altering the technology development curve.

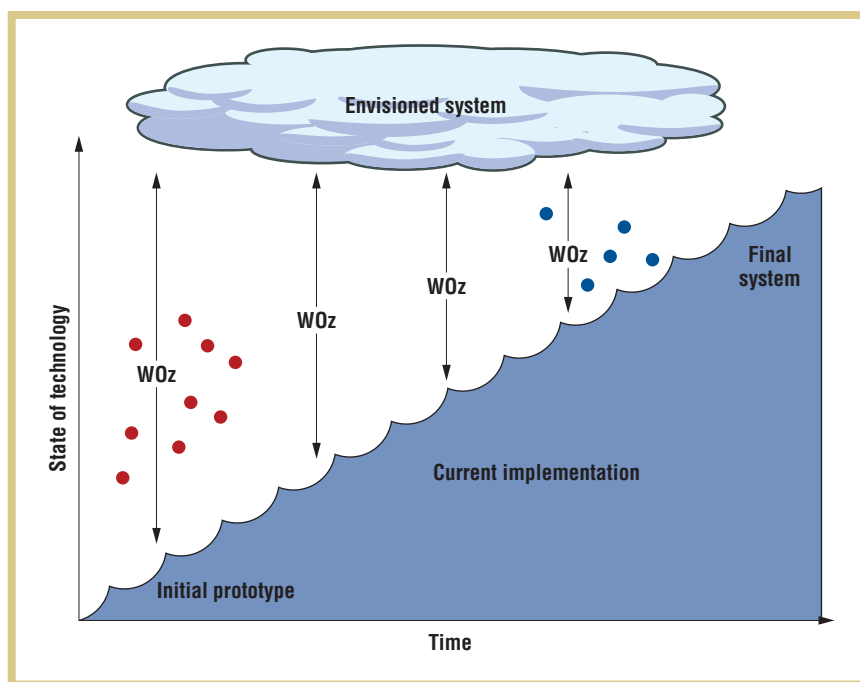
Steven Dow, Blair MacIntyre,
Jaemin Lee, Christopher Oezbek,
Jay David Bolter, and
Maribeth Gandy
Georgia Institute of Technology

Figure 1. One general view of design evolution and the role WOz simulation can play in facilitating evaluation of an envisioned experience. The WOz method has traditionally been used early to simulate undeveloped technology (examples indicated as red dots) or near the end of design, when the wizard can supervise a full implementation or simulate a small piece of the technology (blue dots). We can also use WOz simulation in the middle ground as a means for transitioning to a final design.

In WOz studies, a *wizard operator* generally plays some role in a work-in-progress computer system—he or she might simulate sensor data, contextual information, or system intelligence. We use the complementary word *puppet* to refer to the mocked-up user experience that the wizard controls through the wizard interface. During a WOz-based evaluation, the wizard essentially becomes a crutch for simulating the envisioned interface and interactions before the system works entirely. As technology development progresses, the gap filled by the wizard shrinks.

Traditionally, a wizard operator plays one of two roles. A *controller* fully simulates an unbuilt system component (perhaps a sensor or system intelligence), whereas a *supervisor* oversees a working system and, if necessary, overrides decisions that the system or user makes. The blue and red dots in figure 1 represent examples of previous work with the WOz method.

A less common role, that of *moderator*, lies between the controller and supervisor roles. When a technology or system component is working but is still not trusted, instead of hooking it into the system, you can take on the moderator role to observe this component's output and then send that output as input to the rest of the system. However, because the moderator can override the system component or sensor before the output reaches the rest of the system, a moderated evaluation can still give you the



Our goal was to make it easier for designers to integrate live video, tracking technology, and other sensor data within new media applications.

envisioned user experience. This lets you obtain useful evaluation feedback while collecting a rich set of metadata for evaluating the development of the system component or sensor in question (assuming the system logs the evaluation session, and the wizard operator implicitly tags the log when he or she either follows the action the component suggests or overrides it). By using the moderator role to ensure that the user receives the envisioned experience, you can perform a much more realistic evaluation despite the underlying system behaving unexpectedly (perhaps behaving erratically or completely failing). By analyzing what was happening when the wizard diverged from system decisions, the evaluation might also uncover false design assumptions. Conducting WOz studies at intermediate stages of system development helps refine the user interaction while informing the technology development, especially if you collect user

interaction and wizard operator input data during the moderated experience.

Although we'd like to claim that the wizard's role evolves naturally from controller to moderator to supervisor with a progressively diminishing cognitive load, the design process isn't this straightforward. Nor do these descriptions represent an exhaustive list of specific roles; rather, we intend them to reveal the spectrum of possibilities for WOz prototyping.

WOz support tools in DART

We built DART for prototyping a wide range of applications in augmented reality, mixed reality, and ubiquitous computing. Our goal was to make it easier for designers to integrate live video, tracking technology, and other sensor data within new media applications. One key decision was to integrate these emerging technologies into an established and familiar com-

The Wizard of Oz in HCI

WOz techniques have been used in various ways, depending on the technologies available and the project's specific goals.

Its most common role in human-computer interface development places responsibility on the wizard to fully control a missing piece of the technology infrastructure (from simulating the entire system to simulating one sensor). Used early in design as a lightweight prototyping method, the WOz method presents users with rough sketches of interface ideas, even when it's unclear what the underlying technology should be. In their study, Scott Hudson and his colleagues wanted to understand users' willingness to be interrupted in an office environment.¹ Their WOz study helped them identify appropriate sensors for the space. Stephen Voida and his colleagues used WOz to study basic interaction techniques for a projector/camera-based augmented-reality environment.² They wanted to understand user behavior and preferred modes of interaction unconstrained by technology-imposed limitations, such as special gloves or highly constrained movements to aid a computer vision subsystem.

Further in the design process, after identifying appropriate technologies, you still might find them too difficult to implement, especially for testing speculative user interface issues. In the tool Topiary, the wizard plays the role of a location sensor (for example, a GPS) during the design of location-based applications.³ In Kent Lyons' evaluation of dual-purpose speech, the WOz method enabled him to explore a larger portion of the design space by analyzing unrestricted speech input for novices.⁴ Quan Tran and his colleagues used a wizard to simulate vision technology during the development of the Cook's Collage, a memory aid for elders in a kitchen environment.⁵ Using WOz, prototypes such as this can mature into sophisticated applications that help researchers test

interaction theories without spending time over-engineering a complex underlying system that might not be needed in the final product.

Even in finished applications where the system does most of the work, a wizard can play a variety of roles. In Alice's Adventures in New Media, a wizard operator simulates a gesture recognizer as input to a sophisticated, agent-based narrative engine as part of an augmented-reality experience.⁶

All these examples position the wizard as a controller, whether it's early or late in the design process. In the mixed-reality performance Desert Rain, the wizard plays the supervisor role, helping participants through the experience on a case-by-case basis.⁷ Wizards can also play a dedicated role, add intelligence beyond the current possibilities of computing, or simply monitor an experience to help when problems arise, much like an amusement park ride operator.

Other authors have pointed out general considerations for WOz studies. An early article by Nils Dahlback and his colleagues revealed a need for careful design of WOz simulations in natural-language dialogue systems (although their observations are generally applicable); researchers must pay attention to nonsimulated parts of the system, the constraints of the user task, and guidelines for the wizard.⁸ As Lennart Molin indicates, user awareness of correct and incorrect wizard behavior can taint an evaluation and compromise the results.⁹ Reflecting on their use of the WOz method, David Maulsby and his colleagues state that a designer benefits greatly by becoming the wizard operator and that formal models for the system's behavior are necessary for keeping the simulation honest.¹⁰

In the Suede system for prototyping and evaluating speech user interfaces, Scott Klemmer and his colleagues reveal the need for

mercial media design environment, Macromedia Director.

DART comprises

- a Director extension called the DART Xtra, written in C++, to handle low-level interaction with sensors, trackers, and cameras; and
- behaviors, written in Director's custom scripting language Lingo, for manipulating common high-level design features.

DART behaviors are abstractions of system entities that can be configured using simple parameter windows or customized by modifying the Lingo code

directly. We seek to support learnability through these accessible high-level behaviors that help familiarize designers with the environment and its constraints, and we hope to sustain learning by letting designers peel away layers of abstraction and work directly with code and raw data. At its core, DART imbues the philosophy of rapid prototyping and design iteration by enabling designers to flexibly coevolve application logic and media content.

Event architecture and networking

DART uses a simple event broadcast/subscription model (we use the abstractions of cues and actions) to com-

municate between behaviors. A cue fires when changes happen in the system: a high-level user interaction (for example, a position cue—"user near site A") or a change in internal state (for example, a time cue—"2 minutes elapsed"). DART executes an action in response to a particular cue being fired, typically changing media content or the application state. The system labels cues and actions using unique event names such as myEvent1; the designer can set up cues to respond to sensor values (for example, when the GPS device returns a value within some range) and link them to output actions (for example, playing sound file A). An action must subscribe to a cue, using the

simulating system error to realistically evaluate user performance during WOz studies.¹¹ The work on Suede supports our argument that WOz studies should simulate the envisioned interaction, even when generating the envisioned system requires modification to the wizard input. These considerations all point to the need for careful and appropriate experimental design in WOz simulations. We should plan WOz user studies to answer existing design questions, and the wizard's role in the experimental design should be well defined and consistent.

Several research projects emphasize WOz simulation in development environments, but generally limit the design space to one interface domain (such as speech recognition in the Suede tool or location simulation in Topiary^{3,10}). The Neimo project developed a WOz testing environment for studying multimodal systems where one or more wizard operators can supplement missing system functions.¹² Our work on DART focuses on the rapid prototyping of more general applications in mixed-reality and ubiquitous computing.¹³ While DART isn't appropriate for all pervasive computing systems, it illustrates how a judicious choice of programming model can enable WOz tools to be integrated into a general-purpose design environment.

REFERENCES

1. S. Hudson et al., "Predicting Human Interruptibility with Sensors: A Wizard of Oz Feasibility Study," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI 03)*, ACM Press, 2003, pp. 257–264.
2. S. Vaida et al., "A Study on the Manipulation of 2D Objects in a Projector/Camera-Based Augmented Reality Environment," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI 05)*, ACM Press, 2005, pp. 611–620.
3. Y. Li, J. Hong, and J. Landay, "Topiary: A Tool for Prototyping Location-Enhanced Applications," *Proc. ACM Symp. User Interface Software and Technology (UIST 04)*, ACM Press, 2004, pp. 217–226.
4. K. Lyons, *Improving Support of Conversations by Enhancing Mobile Computer Input*, doctoral dissertation, College of Computing, Georgia Inst. of Technology, 2005.
5. Q. Tran and E. Mynatt, *What Was I Cooking? Towards Deja Vu Displays of Everyday Memory*, tech. report GIT-GVU-TR-03-33, Georgia Inst. Technology, 2003.
6. B. MacIntyre et al., "Augmented Reality as a New Media Experience," *Proc. Int'l Symp. Augmented Reality (ISAR 01)*, IEEE CS Press, 2001, pp. 197–206.
7. B. Koleva et al., "Orchestrating a Mixed Reality Performance," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI 01)*, ACM Press, 2001, pp. 38–45.
8. N. Dahlback, A. Jonsson, and A. Lars, "Wizard of Oz Studies: Why and How," *Proc. Int'l Workshop Intelligent User Interfaces (IUI 93)*, ACM Press, 1993, pp. 193–200.
9. L. Molin, "Wizard of Oz Prototyping for Cooperative Interaction Design of Graphical User Interfaces," *Proc. SIGCHI Nordic Conf. Human-Computer Interaction (NordiCHI 04)*, ACM Press, 2004, pp. 425–428.
10. D. Mulsby, S. Greenberg, and R. Mander, "Prototyping an Intelligent Agent through Wizard of Oz," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems (CHI 93)*, ACM Press, 1993, pp. 277–284.
11. S. Klemmer et al., "Suede: A Wizard of Oz Prototyping Tool for Speech User Interfaces," *Proc. ACM Symp. User Interface Software and Technology (UIST 00)*, ACM Press, 2000, pp. 1–10.
12. S. Balbo, J. Coutaz, and D. Salber, "Towards Automatic Evaluation of Multimodal User Interfaces," *Proc. Int'l Workshop Intelligent User Interfaces (IUI 93)*, ACM Press, 1993, pp. 201–208.
13. B. MacIntyre et al., "DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences," *Proc. ACM Symp. User Interface Software and Technology (UIST 04)*, ACM Press, 2004, pp. 197–206.

cue's event name to complete the broadcast/subscription connection. Multiple cues might trigger one action, or multiple actions might subscribe to one cue (see figure 2a). This loose coupling facilitates rapid prototyping; for example, you can replace an action that's initially triggered using a keyboard cue with an external device button cue or a more sophisticated cue. Our earlier work details DART's networking and distributed shared memory system.¹

WOz prototyping tools

The WOz tools in DART leverage the event broadcast/subscription architecture. To enable WOz communication,

the designer adds the "Puppet of Oz" behavior to the DART application on the machine that is delivering the user experience and adds the "Wizard of Oz" behavior with the puppet machine's IP address to the DART application on a remote wizard machine. The two high-level behaviors establish the networking connection and enable the wizard machine to trigger any actions currently available in the user application by simply firing actions locally using the same cue names. The WOz tools employ two forms of communication: cue broadcast (from wizard to puppet) and action list notification (from puppet to wizard) (see figure 2b). By using common naming

conventions, the designer can trigger cues locally or through a remote wizard.

Leveraging the continuous notification of available actions on the puppet, DART can automatically generate a WOz interface consisting of GUI buttons that correspond to the list of possible actions on the puppet. DART maintains an action subscription list on the puppet during runtime, forwarding subscription list changes to the wizard (events can be added and removed at runtime). The wizard generates a generic button interface (using a simple layout algorithm) and labels each button with the unique event name. As the puppet application runs, the wizard automatically refreshes the correspond-

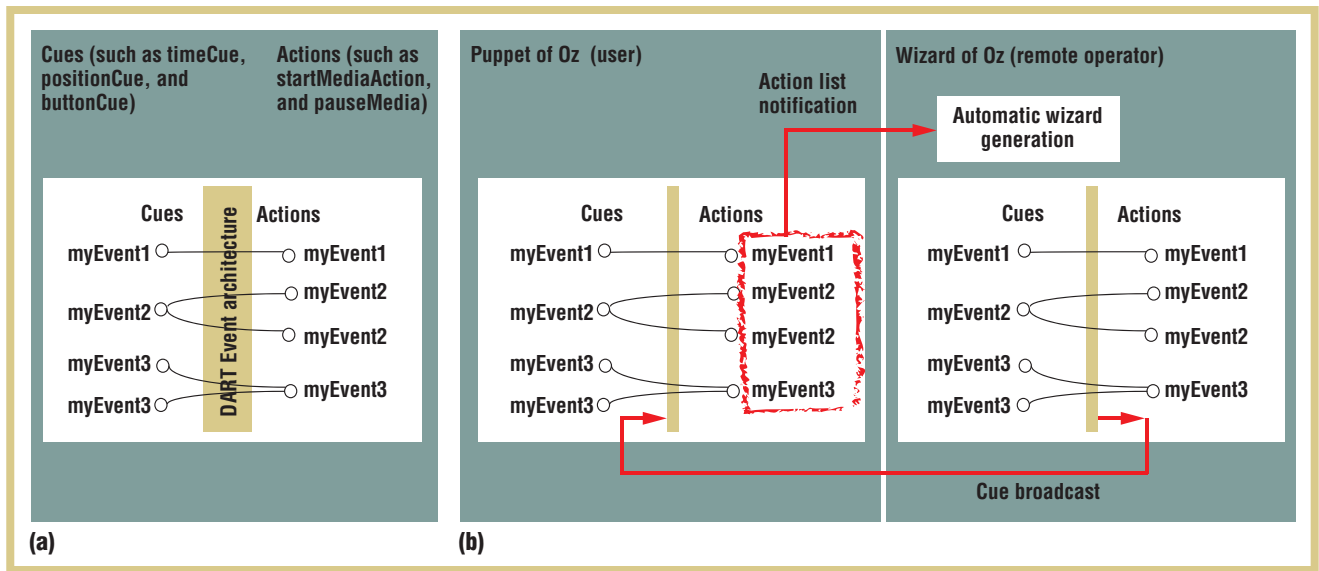


Figure 2. (a) A standard event-based architecture in DART. (b) Integration of WOz tools into the event-based model. Cues fired on the wizard machine are broadcast locally and to the puppet. The puppet machine notifies the wizard about available actions at runtime to automatically generate a basic button UI on the wizard interface.

ing set of buttons. By supporting automatic wizard generation in a high-level behavior (WizardButtonAuto), we aim to lower the barrier for using WOz prototyping as an evaluation strategy.

You can also customize the wizard interface to control the puppet using built-in Director widgets and DART behaviors. For example, you can create a custom button in Director and then place a WizardButton behavior on top of the graphical element (one parameter of the behavior is a specific action name to launch on the puppet). Or, you can place an overhead map image in the application and attach a MapTracking behavior that generates synthetic GPS reports, which appear to the puppet application to be real GPS reports. You can program your own behaviors in Lingo, taking advantage of the networking infrastructure and DART architecture to tailor the wizard interface. The wizard can send cues locally to subscribers or broadcast them only to the puppet.

One strategy we've used is to integrate part of the puppet user interface into the wizard interface so that the wizard operator will experience the same application

state as the user. In The Voices of Oakland project, described later, we use this strategy to let the wizard listen to the same audio segments as the user, following along with the user and deciding what content to display. Because you develop both interfaces in DART, and because DART uses a model of independent actors communicating via events, you can easily port any code developed on either the wizard or the puppet interface to the other.

Visualization tools

DART supports the ability to capture data from video cameras, trackers, analogs, buttons, as well as wizard input for later analysis and playback. DART stores the time-synchronized data in Director casts (collections of Lingo scripts, text files, and other media in Director), which can be imported later into any DART application. Using integrated playback facilities in DART, you can replay the sensor data as it happened during the experiment. The application logic responds to replayed data just as it would to live data, so in playback mode, you'll perceive exactly what the user perceived during the experience.

Replaying an experience is one visualization strategy; DART also includes tools for visualizing data textually and graphically. Observers simply show a text print-out of the data based on current time. DataGraphs behaviors format the replay data into a graphical image. These high-level behaviors require you to specify certain parameters, such as the specific replay data set and the graph boundaries (such as axis assignment, maximum values, and size of data points). You can attach multiple DataGraphs behaviors to the same image so that you can visualize multiple data streams on one graph. DART's visualization tools can show both static, cumulative data as well as dynamic data values at any particular time on the abstract clock. The abstract clock can be controlled independently from Director's clock by DART's TimeSlider behavior or using custom widgets. For the Voices of Oakland experience, we visualized each participant's GPS data on the same image to get an overview of user movement (described in more detail later).

Although you can move the numbers collected from WOz experiments to Matlab, Excel, or another common graphing tool, integrating these facilities

Figure 3. A participant experiencing the Voices of Oakland system in Atlanta's Oakland Cemetery.

in the DART environment has certain benefits. One advantage is the ability to visualize live data in parallel with previously collected data, enabling real-time analysis of user performance. Embedded visualization takes advantage of DART's abstract clock controls. DART replays data on an abstract clock that you can pause, unpaue, set to a particular time, rewind, fast-forward, and so on. By supporting lightweight visualization tools within the design environment, you can perform new sorts of analysis while maintaining consistency with the rest of the design environment.

Case study: A mixed-reality experience in the Oakland Cemetery

A case study of the Voices of Oakland project, an audio location-based experience in Atlanta's historic Oakland Cemetery, illustrates the value of flexible wizard prototyping and analysis tools. Visitors wander through the space and hear an unfolding drama about Atlanta's history through its inhabitants' voices (see figure 3). We describe our design considerations, including issues of story style (dramatic vs. commentary), story arc (linear vs. nonlinear), agency (user vs. system control), medium (visual vs. nonvisual media), and technology (for instance, location tracking) elsewhere.³

As we described earlier, a wizard can have three roles or levels of responsibility during a design process. In our first iteration of the Voices of Oakland experience, the wizard fully controlled the audio segments the participant heard. In the second iteration, we included buttons for the participant to select content. We routed button presses to a graphical display in the wizard interface, but the moderator wizard still had to choose the appropriate audio segments on the basis of this feedback. In the third iteration, we handed over full responsibility to the



In our first iteration of the Voices of Oakland experience, the wizard fully controlled the audio segments the participant heard.

system for presenting audio based on location and user button interaction, with the wizard supervising the experience. The DART WOz tools facilitated easy transition between each change in the wizard interface.

First design iteration: Wizard as controller

In the first generation of the Voices of Oakland project, the wizard operator was integral for simulating several participants' experience. We started with rough audio content and a vague idea of where and when we wanted those stories to be presented as the user moved through the space. To facilitate the controller wizard role, we developed two modes of interaction:

- a map-based interface, where the wizard tracks the user's position and triggers content when the user moves near a red content zone (see figure 4a, left); and

- a button-based interface, where each button triggers a specific audio segment to play (figure 4a, right).

The map-based interface took several days to create because we had to configure the map image to correspond with the correct GPS coordinates and link the audio content to particular locations in the cemetery. We generated the button-based interface quickly using DART's WizardButtonAuto behavior.

In our pilot study, we evaluated the wizard interface as well as the participant experience. Our evaluation involved two wizards (script writers) and two users. Aside from demonstrating how the wizard interface worked, we gave the wizard operators no specific instructions on how to create the visitor's experience; they could simulate the GPS tracking or directly trigger the media content by pushing buttons. During the study, we found that the wizard operators preferred the buttons rather than the map, osten-

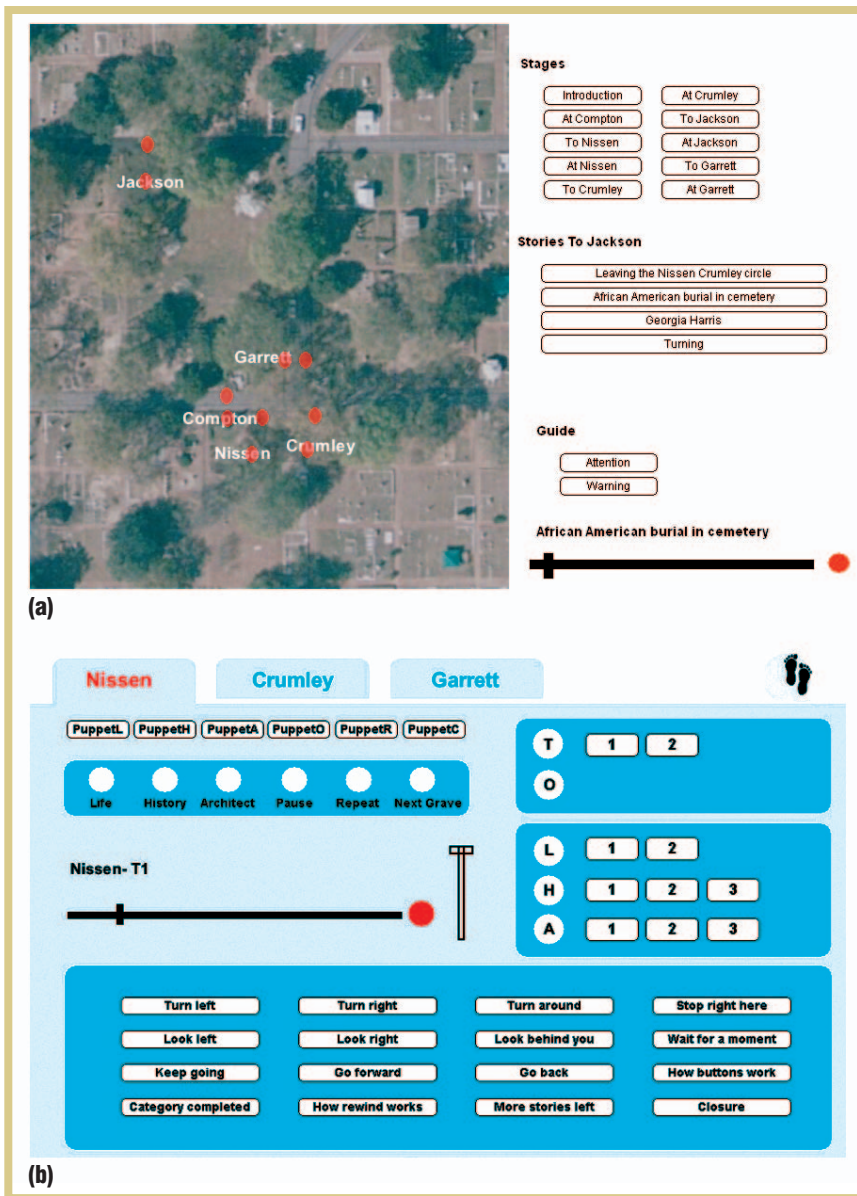


Figure 4. (a) Wizard interface used for the first design iteration. The controller wizard could simulate user position on a map (left) or trigger audio content directly using buttons (right). (b) Wizard interface used in iterations 2 and 3. The wizard observes the user's button interaction in the six circles in the upper left and activates audio content using buttons (story segments on the right, navigational segments at the bottom).

primary instructions were inadequate. We chose to route the user's button interaction through the wizard interface and situate the wizard as a moderator. We had questions about the users' expectations about the interaction, and we wanted to control for interface issues such as multiple button pressing while providing an engaging experience. The moderator could see the button interaction in the wizard interface (and the user's position in the physical space) and determine the appropriate content. Although it took longer to create this custom wizard interface, the DART WOz tools enabled us to easily coevolve the wizard application to match the new user interaction and to support the moderator wizard role.

During an outdoor Fall festival in the cemetery, we informally tested the experience with about 15 members of the public. For the most part, our four different wizard operators simply relayed user button presses, indicating that buttons were sufficient for controlling the audio content at each grave. We didn't record raw GPS data or try to visualize the results, in part because we weren't doing a formal evaluation—although we realized later that it would have been useful.

Third design iteration: Wizard as supervisor

For the third version of the Voices of Oakland, we allowed participant interaction with the button device to directly affect content changes. The wizard interface remained the same (figure 4b), but the wizard's role shifted to supervisor.

sibly because it put them closer to the content and enabled them to create a more compelling visitor experience. (However, because the location-based implementation was fairly crude at this point, this observation should be taken with a grain of salt.) The tour participants provided feedback about the type of content and length of audio segments, and we incorporated many ideas into the next iteration.

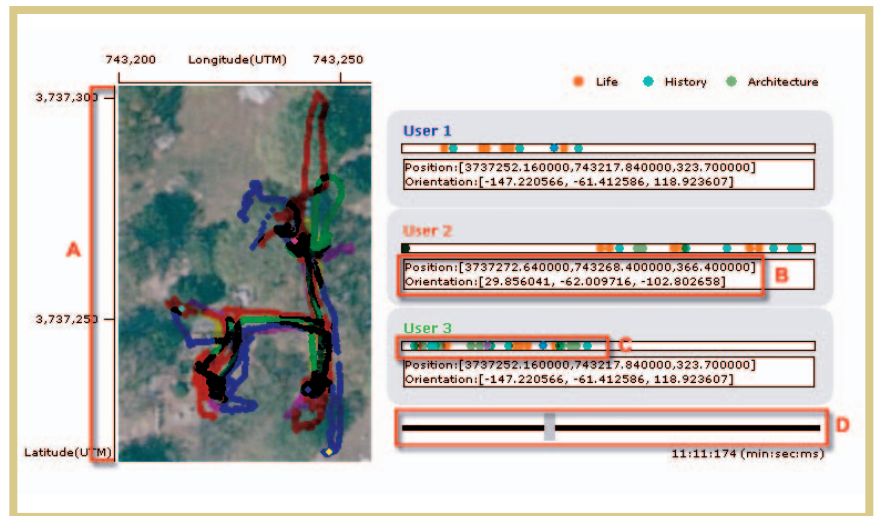
Second design iteration: Wizard as moderator

After the initial pilot testing, we made

several changes to the experience and the wizard's role. We divided the script into short audio dialogues and organized them into categories—life stories, history, and architecture. To simulate user control, we gave the user a handheld controller with buttons corresponding to the content categories. The visitor could listen to short audio stories within a category and then dig deeper, or choose a different category while at any particular grave site.

The new wizard interface (figure 4b) incorporated Director's built-in GUI widgets and a set of buttons (near the bottom of 4b) for guiding the visitor if the

Figure 5. Visualizing the Voices of Oakland experience with a few DART behaviors (DataGraphs, Observers, TimeSlider) inside Director: (a) GPS data for five participants, with dynamic circles showing the users' position at a particular time; (b) textual representations of GPS location and head rotation data; (c) graph of button interaction over time; (d) TimeSlider for controlling DART's abstract time.



The wizard could observe the user's button presses and override the user's content choices, if necessary. The wizard still communicated with the system when a participant was in range of a particular grave and controlled the ancillary navigational content as needed.

We used DART's capture facilities to record all the sensor data (GPS, head rotation, and button presses) and wizard actions during a formal user evaluation. Even though we weren't using the GPS and head rotation data in the application, we collected the data to help us close the gap between the wizard-simulated experience and a working application. Using DART's replay and visualization tools, we created interactive graphs to help analyze the WOz study (see figure 5).

The DART visualization tools are an initial attempt at supporting WOz study analysis. Figure 5 shows multiple participants' GPS data above a satellite image of the cemetery. For each participant, we could print the head rotation values and graph the button interaction over time. We included the TimeSlider in our visualization so that we could scrub the time values and see dynamic updates in the graphs. Our visualization helped us understand how GPS and head rotation data might be useful in our system design. (While a graphical view of the rotation values might have been useful, we opted for a simpler textual representation until we clearly needed the graphical view.) More details about the study results are available elsewhere.³ During future WOz evaluations, we also

plan to explore visualizing prior participant data during the study of live participants for real-time predictive feedback on user behavior.

Explicit support for the WOz simulation method is certainly important in pervasive computing prototyping environments. By providing easy-to-use high-level tools for wizard interface creation and data visualization, we hope to lower the barrier for designers and researchers to explore a variety of potential uses of WOz simulation throughout the design cycle. We also hope this leads to a more comprehensive framework of possible wizard roles.

We learned many lessons during our experimentation with WOz simulation in the Oakland project. Clearly, the nature of the application will dictate what facilities are needed. The ability to easily try different WOz approaches was useful, given the design space we were exploring. If we were doing visual augmented reality or spatialized audio where accurate location was integral, the map-based wizard interface might have been more useful, but we might not have had as many other useful options for wizard controls.

Automatic wizard interface generation appears to be most useful for early

design exploration. We used the automatically generated interface during our early informal studies but not during our formal WOz studies, because customized WOz features were more useful. The embedded visualization tools are well suited for intermediate stages of design and novel wizard roles, where we can learn a lot about users' preferred interactions and the technology's limitations. The rapid-prototyping philosophy of DART and our WOz tools encourages designers to explore various roles with different degrees of responsibility throughout a design process.

Over the past decade, researchers have raised many significant considerations that designers must be conscious of when designing WOz experiments. In particular, they should design the wizard interface with the wizard operators' perceptual, cognitive, and motor skills in mind, just as with any user interface. When there's less demand on the wizard, the wizard can pay more attention to the user and the environment, perform necessary activities more easily, and contribute observations to the evaluation of the experience. For the WOz method to be effective, designers must be able to bridge the gap between the wizard's role and actual system implementation. We believe that integrating WOz prototyping tools into the programming environment is vital to supporting this transition. ■

DON'T RUN THE RISK.

BE SECURE.

IEEE
SECURITY & PRIVACY

Ensure that your networks operate safely and provide critical services even in the face of attacks. Develop lasting security solutions, with this peer-reviewed publication.

Top security professionals in the field share information you can rely on:

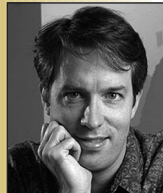
- Wireless Security
- Securing the Enterprise
- Designing for Security Infrastructure Security
- Privacy Issues
- Legal Issues
- Cybercrime
- Digital Rights Management
- Intellectual Property Protection and Piracy
- The Security Profession
- Education

Order your subscription today.

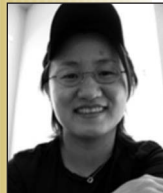
www.computer.org/security/



Steven Dow is a PhD student in human-centered computing in the College of Computing at the Georgia Institute of Technology. His research interests include design tools, human-computer interaction, ubiquitous computing, mixed reality, and experience design. He received his MS in human-computer interaction from the Georgia Institute of Technology. Contact him at the College of Computing, GVU Center, Georgia Inst. of Technology, Atlanta, GA 30332-0280; steven@cc.gatech.edu.



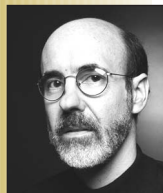
Blair MacIntyre is an assistant professor in the Georgia Institute of Technology's College of Computing and the Graphics, Visualization, and Usability Center. His research interests include understanding how to create highly interactive augmented-reality environments, especially those that use personal displays to augment a user's perception of his or her environment. He received his PhD in computer science from Columbia University. Contact him at the College of Computing, GVU Center, Georgia Inst. Technology, Atlanta, GA 30332-0280; blair@cc.gatech.edu.



Jaemin Lee is a freelance usability specialist. Her research interests include HCI, ubiquitous computing, and context-aware computing. She received her MS in human-computer interaction from the Georgia Institute of Technology. Contact her at 1550 Iron Point Rd., Apt. 221, Folsom, CA 95630; jaemin.lee@gmail.com.



Christopher Oezbek is a PhD student in software engineering at the Free University Berlin. His research interests include documentation processes, API usability and discovery, and the open source development process. He received his MS in computing from the Georgia Institute of Technology. Contact him at the Working Group Software Eng., Free Univ. Berlin, Takustr. 9, D-14195 Berlin, Germany; oezbek@inf.fu-berlin.de.



Jay David Bolter is the Wesley Chair of New Media at the Georgia Institute of Technology. Along with the Augmented Environments Lab at Georgia Tech, he is helping to build augmented-reality systems to stage dramatic experiences for entertainment and education. He also wrote several books on hypertext, media theory, and digital design. He received his PhD in classics from the University of Toronto. Contact him at the School of Literature, Communication, and Culture, Georgia Inst. Technology, Atlanta, GA 30332-0165; jay.bolter@lcc.gatech.edu.



Maribeth Gandy is a research scientist with the Interactive Media Technology Center at the Georgia Institute of Technology and a doctoral student there with a focus on augmented reality and HCI. Her other research interests include mobile, wearable, and ubiquitous computing; universal design; and computer audio. She received her MS in computer science from the Georgia Institute of Technology. Contact her at the Interactive Media Technology Center, Georgia Inst. of Technology, Atlanta, GA 30332-0280; maribeth@imtc.gatech.edu.

REFERENCES

1. B. MacIntyre et al., "DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences," *Proc. ACM Symp. User Interface Software and Technology (UIST 04)*, ACM Press, 2004, pp. 197–206.
2. S. Dow et al., "Wizard of Oz Interfaces for Mixed Reality Applications," *Extended Abstracts SIGCHI Conf. Human Factors in Computing Systems (CHI 05)*, ACM Press, 2005, pp. 1339–1343.
3. S. Dow et al., "Exploring Spatial Narratives and Mixed Reality Experiences in Oakland Cemetery," *Proc. ACM SIGCHI Conf. Advances in Computer Entertainment (ACE 05)*, ACM Press, 2005, pp. 51–60.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.