Topiary: A Tool for Prototyping Location-Enhanced Applications

Yang Li[1], *Jason I. Hong*[1], *James A. Landay*[2, 3]

[1]Group for User Interface Research
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720-1776 USA
{yangli, jasonh}@cs.berkeley.edu

[2]Intel Research Seattle
1100 NE 45th Street
Suite 600
Seattle, WA 98105
james.a.landay@intel.com

[3]DUB Group
Computer Science and Engineering
University of Washington
Seattle, WA 98105-4615 USA
landay@cs.washington.edu

ABSTRACT

Location-enhanced applications use the location of people, places, and things to augment or streamline interaction. Location-enhanced applications are just starting to emerge in several different domains, and many people believe that this type of application will experience tremendous growth in the near future. However, it currently requires a high level of technical expertise to build location-enhanced applications, making it hard to iterate on designs. To address this problem we introduce Topiary, a tool for rapidly prototyping location-enhanced applications. Topiary lets designers create a map that models the location of people, places, and things; use this active map to demonstrate scenarios depicting location contexts; use these scenarios in creating storyboards that describe interaction sequences; and then run these storyboards on mobile devices, with a wizard updating the location of people and things on a separate device. We performed an informal evaluation with seven researchers and interface designers and found that they reacted positively to the concept.

Categories and Subject Descriptors: H.5.2 [**User Interfaces**]: Prototyping, Evaluation / methodology; D.2.2 [**Design Tools and Techniques**]: User interfaces

Additional Keywords and Phrases: Ubiquitous computing, prototyping, informal user interface, Wizard of Oz, location-enhanced, context-aware

INTRODUCTION

Ubiquitous computing has been an active area of research for over a decade, and has opened many new possibilities for human-computer interaction [42]. One especially promising branch of ubiquitous computing that has begun to see commercialization is location-enhanced computing, services and applications that can use one's current location as well as the location of other people, places, and things. One example is AT&T's Find Friends service, which lets mobile phone users find the current location of a friend [3]. Another is E911, which transmits a mobile phone user's current location when making emergency calls.

However, while there is some support for building such applications [14, 15, 20], it currently requires a high level of technical expertise to do so, making it hard for designers to prototype, evaluate, and iterate on designs. Furthermore, developers must deal with relatively low-level sensing technologies such as GPS, active badges [41], and Cricket location beacons [33]. These obstacles make it difficult to iterate on a design, as well as test designs with real users until the actual application is fully completed, by which time it is often too late to make major changes.

As one step towards addressing this problem, we have developed Topiary [1], a prototyping tool for location-enhanced applications. Topiary is aimed at supporting interaction designers in the early stage of design rather than in creating full-fledged systems. Topiary allows designers to demonstrate scenarios depicting location contexts, to storyboard location-enhanced behaviors using these scenarios, and then "run" the storyboards using Wizard of Oz techniques to fake location information.

Why Prototyping Tools

User interface prototyping tools have been developed for several other domains [2, 4, 22, 23, 25, 27]. Prototyping tools offer three significant benefits. First, they lower barriers to entry, making it easier for interaction designers to take part in development. Second, they can help speed up iterative design cycles by making it easier to design, prototype, and evaluate ideas. Third, they make it easier to get user feedback early in the design cycle, when it is still cheap and relatively simple to make major changes.

There is, however, a question of timing here. Location-enhanced applications are still emerging, and there are

UIST '04, October 24–27, 2004, Santa Fe, New Mexico, USA.

[1] Topiary can be downloaded at http://dub.washington.edu/topiary.

consequently few experts in their design and few best practices that can be embodied in a prototyping tool. Despite this, we argue that *now* is the right time to develop such a tool. Many people believe location-based computing will see tremendous growth in the near future, especially given the adoption of E911 in the United States. Market research firm Gartner has made the optimistic prediction that there will be around 42 million American businesses and consumers using location-enhanced applications in 2005 [32]. There is a remarkable opportunity to influence how location-enhanced applications are designed for the better, leading practice rather than simply following it. We believe that enabling developers and designers to more easily iterate while we are still in the early stages of adoption will lead to more high-quality applications in the future.

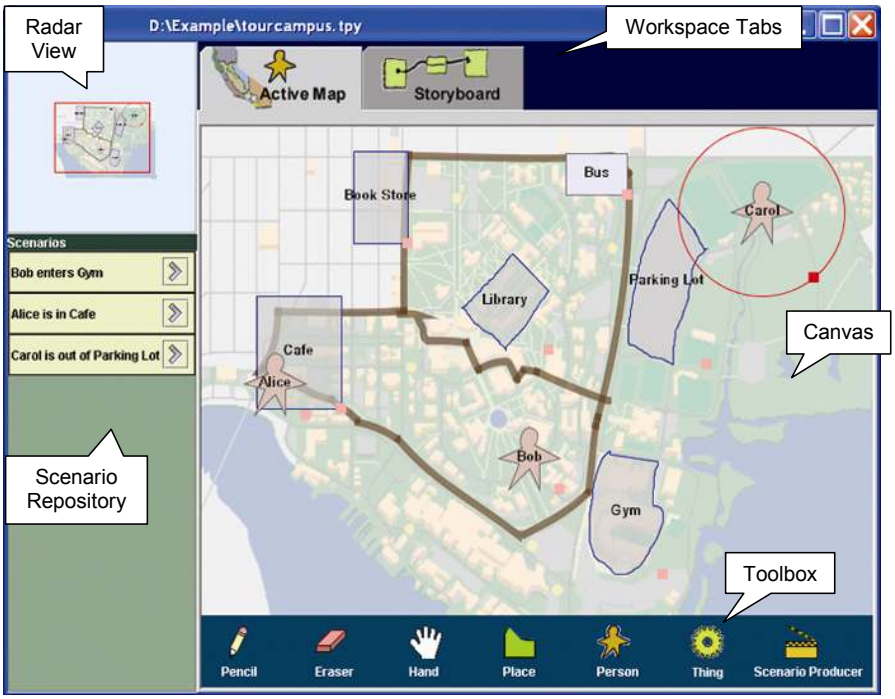

Figure 1. Topiary's Active Map workspace allows designers to create models of people, places and things and to demonstrate scenarios specifying location contexts. The bold lines on the canvas indicate roads drawn using the *Pencil tool*. This figure shows the Active Map design for a Campus Tour Guide.

The rest of this paper is organized as follows. First, we provide an analysis of requirements for such a prototyping tool. Next, we show how Topiary can be used to prototype these applications. Then, we describe our evaluation of Topiary, followed by related work. Finally, we wrap up with a discussion of results and our conclusions.

REQUIREMENTS FOR A PROTOTYPING TOOL

In this section, we describe how prototyping location-enhanced applications is different from prototyping GUI applications. We also examine several location-enhanced applications and provide an analysis of what features need to be supported in a prototyping tool.

How Prototyping Location-Enhanced Apps is Different

An important question to ask here is, why is prototyping a location-enhanced application different from prototyping traditional GUIs? Why can existing prototyping tools not be used? In this section, we outline three key reasons.

Modeling Location Contexts. One way that location-enhanced applications differ from GUIs is that they also use contextual information implicitly perceived by sensors, such as one's location and one's proximity to other people. Contextual information enriches interactions with a much wider input space than mouse or keyboard events. A prototyping tool for location-enhanced applications needs to make it easy for designers to model these location contexts, letting them quickly explore this input space without having to deal with low-level issues such as sensors or programming logic.

Specifying Location-Enhanced Behaviors. Location-enhanced applications also have more complicated interaction sequences than traditional user interfaces. In addition to explicit interactions, e.g., pressing a button, they must also support implicit interactions based on sensed input. One example of an implicit interaction is automatically displaying a page describing the user's current location when they enter a new place. Another example is tailoring manual input to the current situation, such as having a "Show Map" button that shows a floor plan when inside a building and a city map when outside. A prototyping tool for location-enhanced applications needs to make it easy for designers to specify interaction sequences that integrate both implicit and explicit interactions.

Testing and Analyzing a Design. It is important to evaluate applications with real end-users to obtain feedback and refine a design. However, location-enhanced applications are more difficult to test than traditional graphical user interfaces, because location-enhanced applications need to incorporate the current location context and because end-users of these applications are often mobile. A prototyping tool needs to make it easy for designers to quickly test and analyze their designs.

Common Features in Location-Enhanced Apps

The primary metric of success for any prototyping tool is if it can be used to prototype a useful and non-trivial subset of the full design space of applications. Ideally, we could observe and interview designers of location-enhanced applications to learn their processes and best practices and design a tool around those practices. However, location-based computing is still emerging, and there are consequently few, if any, experienced designers out there.

As an alternative, we analyzed several different location-enhanced applications and identified common location functions. We used these common interaction techniques as the basic feature set for Topiary. We also limited the scope to applications that have display-based visual output. For example, Topiary can be used to prototype applications like Cyberguide [1] because the output is limited to a single PDA. However, Topiary does not have explicit support for prototyping a room that automatically turns on its lights when a person walks into it. Topiary does not prohibit the creation of the latter; it simply does not provide hooks for non-display output or explicit wizard support for doing so. Nonetheless, there is still a very large and useful subset of applications that exclusively use visual output.

The genres we have identified include guides for exploration and navigation [1, 9]; finders for finding people, places, or things [3, 18]; group awareness displays [14, 18]; augmented-reality games [17]; information tagging and retrieval, including personal memory aids [7, 34] and notes associated with places [8, 16, 31]; message routing [29, 30]; and safety [28]. We examined these applications and found the following common functions:

- *Location status*, simply displaying someone's or something's location
- *Finders* for a specific or nearest person, place, or thing
- *Active Maps*, dynamically updated maps that show the location of people, places, and things
- *Triggers*, arbitrary functions that activate when something is *in* or *near* something else
- *Wayfinding*, textual or visual descriptions of how to get to a place or how far away something is
- *Resource allocation*, adapting infrastructure resources as people move around (e.g., network packet routing)
- *Tagging*, associating location data to another arbitrary piece of data (e.g., to a photograph)

Topiary currently supports all of the above features except for resource allocation and tagging. We made this decision because these features have been used in relatively few applications and because tagging and resource allocation are more about internal computation than user interaction, which is the main goal of a UI prototyping tool.

THE TOPIARY SYSTEM

Based on this requirements analysis, we spent two months iterating on paper prototypes, getting feedback from researchers familiar with location-enhanced applications and prototyping tools. Our early studies led us to split the tool into three parts: the *Active Map*, the *Storyboard*, and the *Test* workspaces. Each of these workspaces addresses a separate challenge for a prototyping tool for location-enhanced applications, as described in the previous section.

Here we give an overview of how Topiary is used (see Figure 1). First, designers use the *Active Map* workspace to create a model of the location of people, places, and things. Then, they demonstrate scenarios describing location contexts, such as "Alice is in the Gym" or "Bob is entering Room 525". It should be noted that Topiary can model indoor or outside locations, as it is independent of any specific sensing technology. Afterwards, designers can sketch pages and links to create interface mockups in the *Storyboard* workspace, using scenarios as conditions or triggers on a link. For example, the designer can specify that clicking a button goes to one page if "Alice is in the Gym", or automatically go to another if "Bob is entering Room 525". After a few mockups have been created, a designer can let real users try out the design in the *Test* workspace, by "running" the sketches on a mobile device like a PDA. A user can interact with these sketches, while a wizard follows the user and updates the location of people and things on a separate device. Optionally, a sensor infrastructure can be used to update location information, if available. We describe each of these workspaces in more detail using a running example of a Campus Tour Guide.

Active Map Workspace

The *Active Map* workspace lets designers model the spatial relationships between people, places, and things. Designers can either sketch a map or load a GIF or JPEG image of a map into the *Active Map* workspace as a background image to help with positioning of these entities. For example, Figure 1 shows a map of a university campus. There are currently no semantics associated with these images, they are simply meant to help designers get a better understanding of a geographical area and they can also be displayed to the end-users in the resulting interface.

The designer can use the *Pencil tool* to draw paths, which are used for the wayfinding feature. Topiary parses these informal sketches into a road network (see Figure 1) that can be dynamically searched for the shortest path. This network consists of a graph where the vertices are stroke intersections and the edges are segmented strokes.

Creating Entities. The *Place tool* is used to create places (see Figure 2). To use it, the designer outlines the boundaries of a place. Inspired by the selection techniques of ScanScribe [36], we apply this technique to make it easy to create rectangular places while not excluding arbitrarily-shaped ones.

The *Person tool* is used to create a person. The designer selects the Person tool and then clicks on where he wants

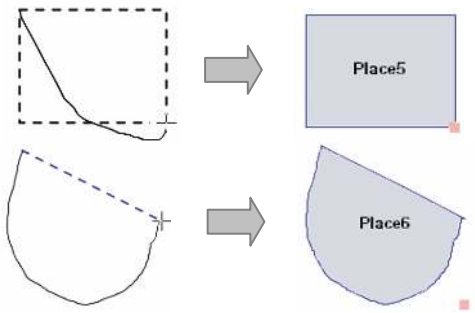

Figure 2. The *Place Tool* can recognize rectangles and polygons based on the overall shape drawn.

the person to be. The *Thing tool* works in the same manner, but is used for creating things such as cars and printers.

Each of these entities (place, person, or thing) is given a unique default name such as "Place5". This name can be replaced with typed text. Places can also be created within other places, creating hierarchies.

Capturing Location Contexts through Scenarios. In Topiary, *scenarios* are a collection of location contexts that can be used for specifying location-enhanced behaviors. Location contexts in Topiary are binary spatial relations of the form *[entity] [relationship] [entity]*, for example "Alice is in the Gym", where "Alice" and "Gym" are the two entities and "in" is the relationship.

Scenarios are captured with the *Scenario Producer tool*, located on the far right side of the toolbox (see Figure 1). The Scenario Producer is a simple form of programming by demonstration [11, 24]. Like a screen capture tool, selecting the *Scenario Producer* brings up a recording window that can be positioned over entities of interest (see Figure 3a). This window can be resized to include or exclude entities.

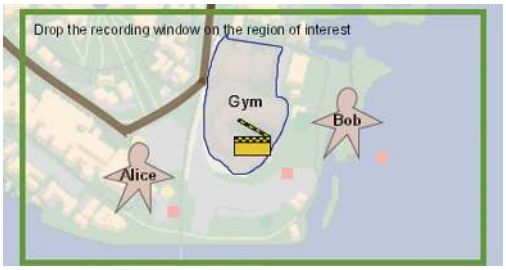

Figure 3a. The *Scenario Producer tool* uses a green window for selecting entities of interest. Here, the recording window is positioned over three entities, Bob, Alice and the Gym.

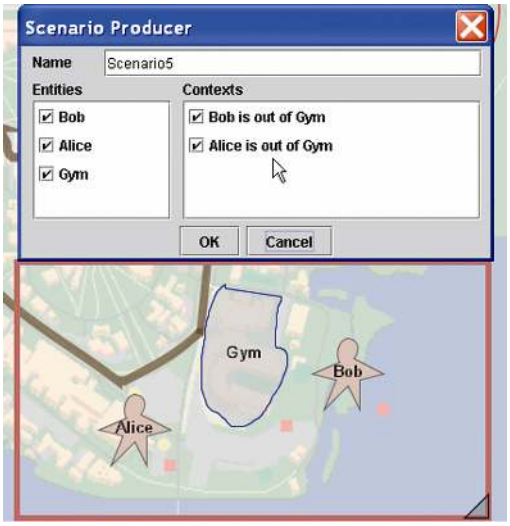

Figure 3b. Once the recording window is dropped, a dialog box is brought up which lets designers select which contexts they are interested in. The list of Entities on the left side lets designers filter out Contexts on the right.

Once the recording window is dropped, Topiary will distill location contexts from the spatial relations of the included entities. A dialog box is then brought up that lets designers select contexts of interest (see Figure 3b). The left side of this dialog box contains a list of entities that can be used for filtering contexts. Unchecking an entity removes all contexts associated with that entity. Designers can also demonstrate transitions by moving entities within the recording window. For example, dragging Bob into the Gym changes the event "Bob is out of Gym" into "Bob enters Gym" (see Figure 3c). New scenarios are added to the Scenario Repository (see left side of Figure 1).

The Scenario Producer as described so far only supports the spatial relationships "in", "out", "enters", and "exits", but not proximal ones such as "near" and "moves near". To specify these, designers can create a proximity region by dragging the proximity handle of an entity to reflect various application-specific definitions of "near" (see Figure 4).

Topiary represents these contexts internally via the spatial relations of containment and intersection of graphical objects (see Table 1).

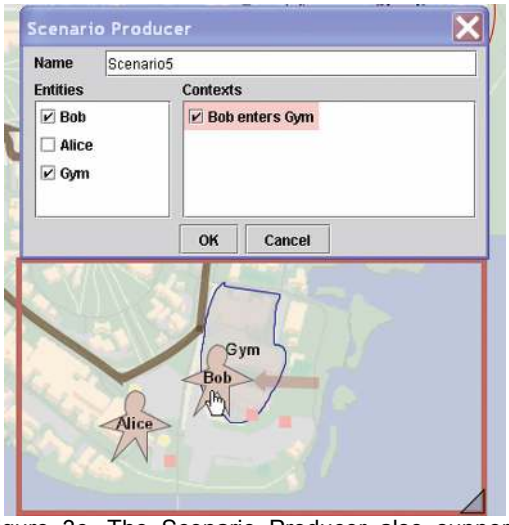

Figure 3c. The Scenario Producer also supports context transitions. The designer drags Bob into the Gym, with the context changing from "Bob is out of Gym" to "Bob enters Gym". This figure also shows an example of filtering. Entity Alice is unchecked, and all related contexts are filtered out.

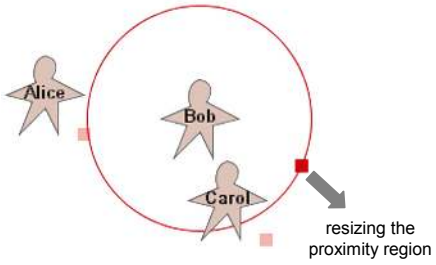

Figure 4. Proximity regions can be specified by dragging the pink proximity handles around each entity. This picture specifies that "Bob is near Carol" and "Bob is far from Alice".

Contexts	Spatial Relations		
near / far	*P* (place) place	*contains* *intersects*	person | thing *P* (person | thing)
moves near / away	*P* (place) place	*contains** *intersects**	person | thing *P* (person | thing)
in / out	place	*contains*	person | thing
enters / exits	place	*contains**	person | thing

Table 1. Set of basic location contexts supported by Topiary. Each of these contexts is represented by spatial relations between graphical objects. *P(X)* represents the proximity region of *X*. Spatial relationships labeled with * indicate they carry temporal information as well. The containment is calculated based on whether the center of a graphical object is contained by another object's bounds.

Combined, these relations allow Topiary to support two basic kinds of location contexts: presence and proximity. A *presence context* describes whether a person or thing is *in* a place, e.g., "Bob is in the meeting room". A *proximity context* describes whether two entities are near one another, e.g., "Alice is near Bob". Topiary also supports *transitions* from one context state to another, e.g., "Alice enters her office". In this case, the location context carries temporal information in addition to spatial information.

As defined at the beginning of this section, scenarios are a *collection* of location contexts. Thus, scenarios can also be used to model complex situations. For example, the scenario "Bob and Alice are in the Meeting room" can be represented using the two location contexts "Bob is in the Meeting room" and "Alice is in the Meeting room".

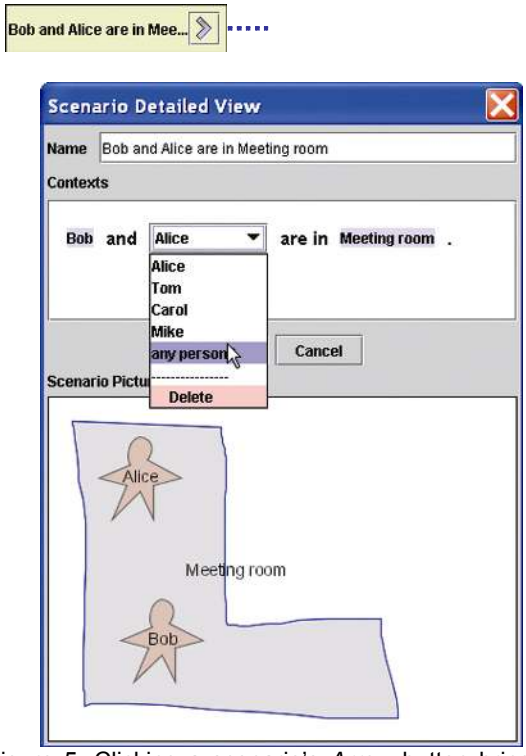

Figure 5: Clicking a scenario's *Arrow* button brings up a detailed view which shows a textual and graphical description of the scenario. Scenarios can be modified by replacing a specific entity with another, or generalized by replacing it with "any person", "any place", or "any thing".

Generalizing Scenarios. Scenarios in Topiary can also be generalized from concrete examples. Figure 5 shows how the scenario "Bob and Alice are in the Meeting Room" can be generalized to "Bob and *Any Person* are in the Meeting Room". The internal representation for this is below (where *B* stands for *Bob* and *M* stands for *Meeting Room*.):

$$\exists p, In(B,M) \wedge In(p,M) \wedge \neg Is(p,B) \wedge IsPerson(p)$$

In Test mode (described below), Topiary checks whether *Bob* and any other *people* are in the *meeting room* simultaneously. A Back Tracking search with Minimum Remaining Values and Degree heuristic [35] is used to match wildcards to concrete entities. Besides matching spatial relations, we also need to match temporal information. For example, for the location context "Any Person enters Any Place", the algorithm keeps observing whether there is a transition from *Out* to *In* between any pairs of persons and places.

Storyboard Workspace

Designers can use the *Storyboard* workspace (see Figure 6) to create interface mockups, creating *pages* that represent screens and *links* that represent transitions between pages. Conceptually, Topiary's Storyboard workspace is similar to the storyboard in tools such as SILK [23]. The key innovations in Topiary's storyboards are that scenarios created in the *Active Map* workspace can be used as conditions or triggers on links, and *context components* specialized for location-enhanced applications can be embedded into pages.

To create a *page*, the designer uses the *Pencil* and draws a rectangle, which is recognized as a page. Pages allow freeform ink, which are processed only for smoothing and grouping. To minimize distractions for designers, there are no other forms of recognition. To create a *link*, the designer draws a line from one page to another. Topiary has two kinds of links (see Figure 6). *Explicit links*, denoted in blue, start on ink within a page. Explicit links represent GUI elements that users have to click on, e.g., buttons or hyperlinks. *Implicit links*, denoted in green, start on an empty area in a page. Implicit links represent transitions that automatically execute when scenarios associated with that link occur. Explicit links model actions taken by end-users, whereas implicit links model sensed data.

One or more scenarios can be added to a link by dragging them from the Scenario Repository onto a link[2]. Multiple scenarios represent the logical AND of the scenarios.

[2] Two kinds of built-in scenarios, namely movement speed and temporal conditions (times, time intervals and elapsed times), can be directly inserted into a link by bringing up a Pie Menu on the link via a right click.

Scenarios can be removed by dragging them out of the link or by using the *Eraser* tool, and copied by holding the *Ctrl* key when dragging. Again, these scenarios let designers place conditions or triggers on links, letting different pages be displayed depending on the state of the location contexts.

Detecting Conflicts in Links. A link cannot be activated if it contains multiple scenarios that cannot be satisfied simultaneously (e.g., "Bob is in the Gym", "Alice is far from the Gym", and "Alice is close to Bob"). When Topiary detects a conflict, it shows a dialog reporting the problem and asks the designer to resolve it.

Detecting these conflicts is a Constraint Satisfaction Problem (CSP) [35]. Each location context is a spatial constraint (see Table 1) and Topiary simply detects whether a link is over-constrained. However, since shapes in Topiary can be freeform polygons, and since spatial relationships are nonlinear, existing methods for solving geometric constraints cannot be directly applied.

Instead, since we only need to know whether these spatial constraints can be satisfied rather than finding geometric solutions, we approximate by converting spatial constraints to Boolean constraints and then solve the Boolean CSP. Our algorithm converts each spatial constraint into a set of propositions based on predefined knowledge of spatial relations. For example, the location context "Bob is far from the Kitchen" generates the following predicates:

$\neg Close(B,K)$: "Bob is far from the Kitchen"

$\neg In(B,K)$: "Bob is out of the Kitchen"

$\forall e, In(e,K) \rightarrow \neg Close(B,e)$: "Bob is far from any entity in the Kitchen"

$\forall p, In(p,K) \rightarrow \neg In(B,p)$: "Bob is out of any place inside the Kitchen"

Conflicts can also arise from ambiguity between links. If two links with the same scenarios originate from the same page element (for explicit links) or the same page (for implicit links), it is ambiguous as to which link should be activated. By simply comparing two groups of scenarios, Topiary finds these ambiguities and marks these links in red.

Built-in Context Components. Topiary provides five built-in context components for displaying spatial and temporal information. These context components make it easy to prototype features common in many location-enhanced applications that are hard to specify using storyboards alone.

The *Active Map component* lets designers embed a view of the Active Map workspace into a page (see top-left page of Figure 6), letting end-users see either part of or the entire map, the current location of people and things, as well as the shortest path to a destination (if one is specified). A dialog box lets the designer show a fixed region of the Active Map workspace or a region around an entity, e.g., a 180 foot square around Bob. Topiary lets the designer set up a mapping from pixels to physical measurements. The designer can also choose to show directional information, as well as a path by specifying a starting point and an end point, both of which can be fixed points or an entity. This path is dynamically generated based on the road network drawn in the Active Map workspace.

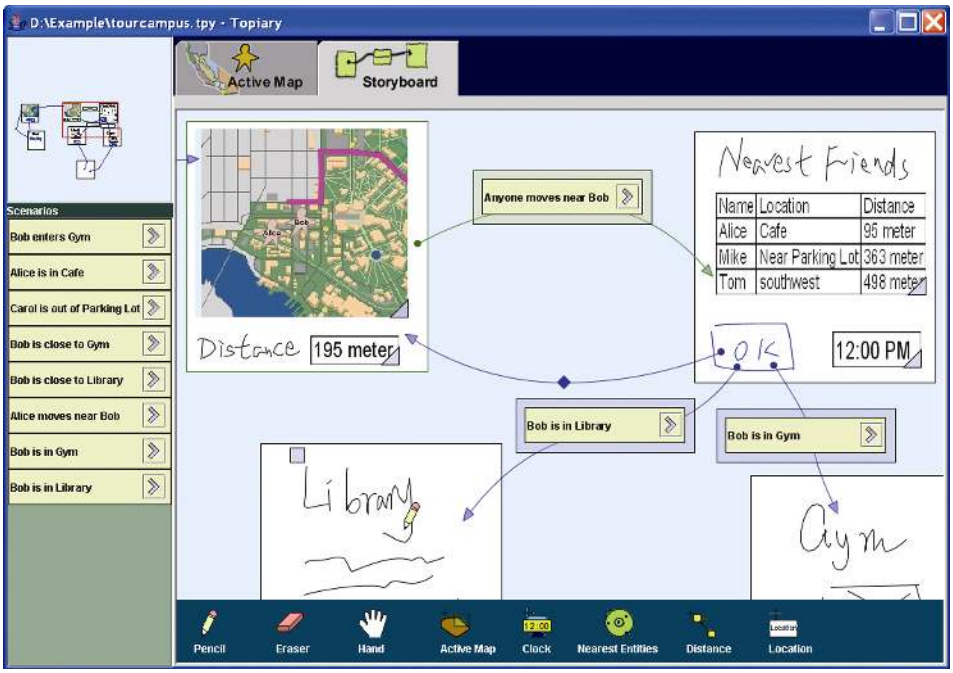

Figure 6: Topiary's *Storyboard* workspace lets people create mockups of applications. Explicit links (the lower three links, in blue) represent things like buttons and hyperlinks. Implicit links (the top link, in green) represent automatic transitions. Here, the implicit link is: automatically go from the *Map* page to the *Nearest Friends* page when "Anyone moves near Bob". The three explicit links originating from the *OK* button make the behavior of that button change depending on which scenarios are true.

The *Nearest Entities component* displays a table of the nearest N entities from a set of entities (see the *Nearest Friends* page in Figure 6). When this component is used, a dialog is brought up that lets the designer select the set of entities to choose from. The designer can also choose to show the name, location, and distance of these entities. Figure 6 shows the names, locations and distances of the three friends who are nearest to Bob. Topiary will show the most accurate location information that it can get at test time. For example, it can show Alice's location as "Café", "near Café" or "Northwest (to Bob)".

The *Clock component* is used to display either the current time or the time when a scenario with context transitions happened, e.g., the time when Bob entered the bookstore. The *Distance component* shows the distance between two entities. The *Location component* displays an entity's location by name.

The values for these components are automatically updated based on the simulated locations of people, places, and things in the Test workspace, described in the next section.

Test Workspace

After several pages have been created, designers can try out their designs in the *Test* workspace. To enter the Test workspace, the designer opens a pie menu by right clicking on the desired start page and then selecting the Test option.

The Test workspace has two major parts: the Wizard UI and the End-User UI (see Figure 7). The End-User UI is what end users will see and interact with. The Wizard UI is where the designer can simulate location contexts, while observing and analyzing a test. These UIs can be run on the same device (to let a designer try out a design) or on separate devices (one for the Wizard, the other for the user).

The Wizard UI has four parts. The *Wizard Map* is a copy of the Active Map workspace, with the key difference being that it represents the current location of people and things. The designer can simulate location contexts by moving people and things around to dynamically update their location. If moving a person or a thing causes an implicit link to activate, then the End-User UI will automatically transition to that page. On the bottom-left is the *End-User Screen*, a copy of the End-User UI, which also updates in response to end-user input on a PDA. The designer can click on the same links that a user could as well for test purposes. A *Radar View* of the map area is provided for navigation. The *Storyboard Analysis window* shows a simplified view of the storyboard workspace with the current page and the last transition highlighted, which can help designers to figure out interaction flows.

To connect the End-User UI to the Wizard UI, the designer starts a special Topiary client on a separate device. This client then searches for a network connection to the Wizard UI. The End-User UI is active once it is connected. A user can click on any explicit links, and any implicit links activated by changes in location will also fire. The designer only needs to update the location of people and things.

Topiary can also use real location data if it is available, enabling more realistic testing at larger scales. A designer can use sensor input by checking the Sensor checkbox (see center top of Figure 7). Topiary currently acquires location data through Place Lab [37], which allows a WiFi-enabled device to passively listen for nearby access points to determine its location in a privacy-sensitive manner.

To analyze a design, designers can record a test and replay it later. Topiary can capture users' actions, like mouse movements and clicks, as well as physical paths traveled.

Implementation

Topiary is implemented in Java 2 SDK v1.4.2 on top of SATIN [21], a toolkit for informal, pen-based applications. Topiary currently has 398 Java classes with approximately 26,000 lines of source code. Communication between the Wizard UI and the End-User UI is done through Java object serialization and network sockets. The Topiary client has only 18 Java classes and is compatible with JRE 1.1, so it can run on a wide variety of PDAs and phones. All processing of pages and links is done on the Wizard UI. We

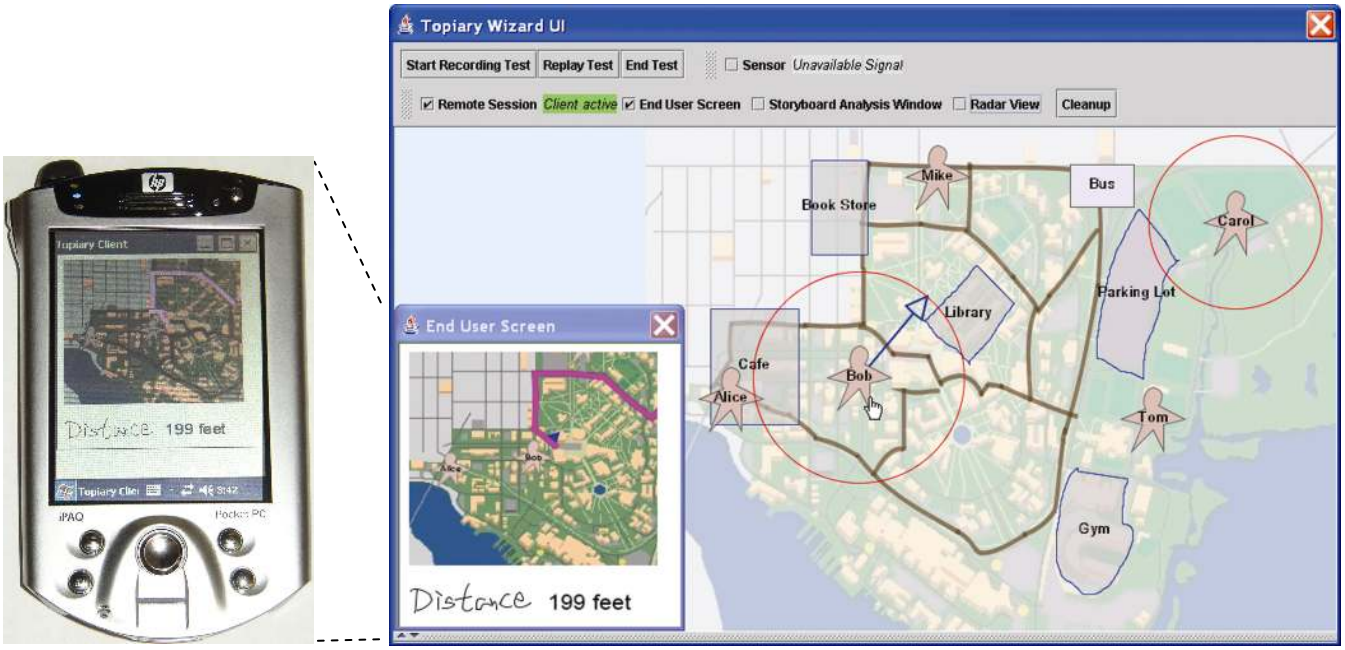

Figure 7. The Test workspace consists of two major parts: the Wizard UI and the End-User UI. The left figure shows an End-User UI running on a PDA and the right figure shows a Wizard UI. Here the Storyboard Analysis Window and the Radar View are turned off. Designers can turn on the sensor input by checking the Sensor checkbox if any available.

have tested Topiary with the wireless connection established over IEEE 802.11 with both access point and peer-to-peer connection modes.

EVALUATION

We ran an informal evaluation on an early implementation of Topiary with 7 participants. Two were researchers familiar with location-enhanced applications and five were interface designers (2 students and 3 professionals). Participants were offered $50 plus $100 for the best design. We used an IBM Thinkpad with a 700MHz CPU, 512MB RAM, and 14.1 inch display, and a Wacom Graphire tablet. Participants were shown a 15 minute demo of Topiary, and were coached in completing three tasks during a 30 minute tutorial that showed how to use all of the features. The three tasks were designing an In/Out board, a Find Nearest Printers application, and a Find Nearby Friends application.

The final task, on which participants were judged, was to create a tour guide for either the Berkeley campus or San Francisco, whichever they preferred. There were three requirements: show an area map, display information about interesting spots, and support finding friends. Participants could add any other features they thought would be useful.

After finishing, we asked our participants to rate the understandability, ease-of-use, and usefulness of various aspects of Topiary on a seven-point scale (7 is the best).

Observations

For the most part, all of the participants could accomplish all of the tasks. Our participants did not encounter any serious problems with the Active Map workspace or the Wizard UI. However, our participants did encounter two common problems with the Storyboard workspace. First, participants had some difficulties understanding the interaction flow after a dozen or so pages, due to the large number of crossing links, as well as to space management in that pages were often packed tightly together.

Second, participants often had difficulties in ensuring that all of the necessary scenarios were covered properly on a page. The best way to describe this problem is to contrast Topiary with other prototyping systems. Many prototyping systems use a page metaphor. However, in these systems, the only way to transition from one page to another is by explicit user interaction. The number of transitions and possible actions are limited by what makes sense for that page. However, in Topiary, even if users are on one page, the background state of location information can change dramatically. The challenge for designers is that they need to account for what explicit actions the user might take, as well as what changes in spatial relationships might occur.

The storyboard analysis view and the test recording/replay in the Test workspace have been built since our evaluation to address these issues and to help designers debug their designs. Our participants also requested a feature for showing a region around an entity rather than only a fixed region of the Active Map workspace. As mentioned earlier, we have added this feature to the Active Map component.

Feedback

Participants generally liked the Topiary concept and the design process it supports. Some participants familiar with ubicomp thought Topiary was much easier to understand and use than dealing with sensors and logic-based rules. Overall, Topiary was rated 5.7 of 7 for understandability, 5 for ease-of-use, and 5.9 for usefulness. We believe the ease-of-use rating was due more to bugs rather than the interaction. One participant summarized it best: “although there are some bugs, I think it’s very smart … it is really fast to prototype.” Some participants enjoyed using the tool, saying “this is fun” and “this is neat”. However, the storyboard had the lowest ratings, with an understandability of 4.9, ease-of-use 5.6, and usefulness 5.6.

Our Experience

We have also used Topiary to prototype other applications, such as a context-aware reminder system and a car-based navigation system. These examples can be found at http://dub.washington.edu/topiary/examples, and embody many features that can be found in existing research and commercial location-enhanced systems.

To further validate Topiary, we iterated on the design of a tour guide application and then implemented it. We focused on the UI for finding the path to a specific place. It took us three hours to make four prototypes using Topiary, each using a different navigation technique. The first design shows a map of the entire campus. The second design shows an area centered on the user and lets the user manually zoom in and out. The third design uses the user’s current location to show different regions of the campus. The last design is similar to the second, except it automatically zooms in or out based on the user’s current speed. All of these designs showed the users’ current location, and distance and shortest path to the target.

We then had three people try all four designs on a PDA in the field, with a wizard updating their location on a Tablet PC. We were able to make some changes to the design instantly in response to their suggestions. Interestingly, our participants did not realize their location was being updated by a wizard rather than by real sensors. We also used sensor input in part of the test to see how sensor accuracy affected users. One person suggested showing a region for the possible location instead of just a point. Based on participant feedback, we spent an hour creating a new design combining designs 2 and 4, letting users switch between automatic and manual zooming. In addition, we added a feature to highlight the target when it was nearby.

We also tried several techniques for helping users go in the right direction, including rotating the map, showing orientation, and showing trajectory arrows. Our participants gave us many useful comments. For example, two of them suggested showing a movement trail.

After testing, we started to consider implementation issues. For example, since Place Lab does not provide precise orientation, we decided to show a movement trail instead of showing potentially inaccurate directional arrows. Building

the application took about 2 weeks. Topiary provided us a lightweight way of getting early feedback from users and to quickly figure out the design issues in the early stages of design, which is much cheaper and less risky than directly building the real application and then testing with users.

RELATED WORK

There have been many prototyping tools for various domains [4, 22, 23, 25]. Topiary is the first prototyping tool for location-enhanced apps, representing a first step towards prototyping tools for ubicomp. Topiary's storyboard is based on these previous systems, extending the concept to location-enhanced interactions. DENIM [26] uses the state of visual elements as conditions on explicit links, e.g., if a checkbox is checked. Topiary, in contrast, lets designers create custom scenarios based on spatial relationships and use those scenarios in conditioning links. Topiary also vastly expands the notion of implicit links, in this case links activated from implicit location input. Previously, the only kind of implicit link in DENIM was for time.

UbiWise [5] is a desktop-based virtual environment for simulating ubicomp devices, environments, and interactions. In contrast, Topiary supports designers in rapidly prototyping and testing out designs in the actual physical environments that users live, work, and play in.

iCAP is a tool for prototyping context-aware applications [39]. iCAP uses sensors as the key abstraction, letting designers link input to output using Boolean logic and a rule-based system. In contrast, Topiary uses a much higher level of abstraction, letting designers think visually in terms of people, places, things, maps, and scenarios.

The *a CAPpella* [13] system looks at end-user configuration of a pre-deployed sensor environment via machine learning. In contrast, Topiary allows interaction designers to *design*, instead of *customize* existing, location-enhanced applications. This occurs at the very early stages of design without requiring any sensor infrastructures to be deployed.

Topiary's Scenario Producer is a simple domain-specific version of end-user-triggered behaviors (e.g., [11, 24, 43]), though it focuses less on the actual behavior and more on capturing events. Topiary is the first informal prototyping tool to use this approach for specifying events of interest.

The term "Active Map" is taken from Schilit and Theimer [38]. The idea of using a map in location-enhanced applications is a common one (e.g., [1, 10]). The idea of overlaying places on top of images was inspired by HTML image map tools (e.g., [6]). Topiary introduces the use of maps as an aid for designers in informal prototyping.

The Wizard of Oz technique is often used for simulating speech recognition systems [12, 22]. The idea of having a Wizard follow a user around to update location information was inspired by a prototype we built for helping users find things [40]. Topiary can be used to make a rough version of this same application, obviating the need for custom software. Topiary also takes a step towards more involved wizard interfaces for prototyping tools, having wizards fake sensor information, in this case location information.

DISCUSSION AND FUTURE WORK

Topiary's map representation of location contexts has several benefits over other approaches, such as textual or logical representations. First, it is easy to understand because it has a direct mapping to spatial relationships in the physical world. Second, it allows designers to quickly understand the overall situation. Third, location contexts can be simulated simply by moving entities on the map in Test mode, rather than having to type anything.

Topiary currently does not have explicit features for handling sensor ambiguity, primarily because Topiary is intended for early-stage interaction design rather than dealing with the vagaries of sensing technologies. Topiary indirectly supports ambiguity in three ways. The sketched location model is inherently ambiguous, the wizard can deliberately generate noise in the positions of the icons while testing, and turning on the sensor infrastructure tests a design with the ambiguous input of real sensors. However, since ambiguity is an essential aspect of sensors, we are planning on adding basic support features in future work. One idea is to include an error model which can generate various sensing errors, similar to that used in Suede [22].

Another direction for future work is to support a wider range of contextual information. Currently, Topiary only supports spatial and temporal contexts, but one could imagine other kinds, such as activity.

We are also investigating approaches to make the storyboard scale better. One possible solution is to organize storyboards hierarchically as in StateCharts [19]. In our case, a set of pages that represent an interaction sequence can be grouped together as one composite page. We are also looking at giving designers better awareness of which scenarios have been covered on a page and which have not.

CONCLUSION

In this paper, we described Topiary, the first tool for prototyping location-enhanced applications. Topiary introduces active maps for prototyping, which lets designers model the location of people, places, and things and demonstrate scenarios depicting location contexts. Topiary also introduces a richer visual language for storyboarding, letting designers create links that are active only when scenarios associated with that link are true. Topiary supports explicit links, ones that end-users interact with, and implicit links, ones that automatically take place depending on the state of current spatial relationships. Topiary's Test workspace lets designers try out their designs by "running" them with real users and analyze them by capturing and replaying a test. Designers can update the location of people and things on the Wizard UI, which can be linked with an End-User UI running on a separate device.

ACKNOWLEDGMENTS

We would like to thank Jimmy Lin, Richard Davis, Scott Klemmer, Scott Lederer, Alan Borning, and Gaetano Borriello for their feedback. This work was supported by NSF under Grant No. IIS-0205644.

REFERENCES

1. Abowd, G.D., et al., *Cyberguide: A Mobile Context-Aware Tour Guide.* Baltzer/ACM Wireless Networks, 1997. **3**(5): pp. 421-433.
2. Apple, *HyperCard User's Guide*. 1987: Apple Computer, Inc.
3. AT&T, AT&T Wireless mMode - Find Friends. http://www.attwireless.com/mmode/features/findit/FindFriends/.
4. Bailey, B.P., Konstan, J.A., and Carlis, J.V. *DEMAIS: Designing Multimedia Applications with Interactive Storyboards*. in *ACM Multimedia 2001*. pp. 241-250.
5. Barton, J.J. and Vijayaraghavan, V. *UBIWISE, A simulator for Ubiquitous computing systems design*. 2003, HP Labs. Tech. Rep. HPL-2003-93.
6. Boutell, T., Mapedit. http://www.boutell.com/mapedit/.
7. Brown, P.J. and Jones, G.J.F., *Context-aware Retrieval: Exploring a New Environment for Information Retrieval and Information Filtering.* Personal and Ubiquitous Computing, 2001. **5**(4): pp. 253-263.
8. Burrell, J., et al. *Context-Aware Computing: A Test Case*. in *Ubicomp 2002*. Göteborg, Sweden. pp. 1-15.
9. Cheverst, K., et al., *Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences.* CHI Letters, 2000. **2**(1): pp. 17-24.
10. Cheverst, K., et al. *Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences*. in *Human Factors in Computing Systems: CHI 2000*. pp. 17-24.
11. Cypher, A., et al., *Watch What I Do: Programming by Demonstration*. 1993, Cambridge, MA: MIT Press.
12. Dahlbäck, N., Jönsson, A., and Ahrenberg, L. *Wizard of Oz Studies - Why and How*. in *IUI 1993*. pp. 193-200.
13. Dey, A.K., et al., *a CAPpella: Programming by Demonstration of Context-Aware Applications.* CHI Letters, 2004. **6**(1): pp. 33-40.
14. Dey, A.K., Salber, D., and Abowd, G.D., *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications.* Human-Computer Interaction, 2001. **16**(2-3): pp. 97-166.
15. Ericsson, Mobile Positioning SDK. http://www.ericsson.com/mobilityworld/sub/open/technologies/mobile_positioning/tools.html.
16. Espinoza, F., et al. *GeoNotes: Social and Navigational Aspects of Location-Based Information Systems*. in *Ubicomp 2001*. Atlanta, GA. pp. 2-17.
17. Falk, J., et al. *Pirates: Proximity-Triggered Interaction in a Multi-Player Game*. in *Human Factors in Computing Systems: CHI 2001 (Extended Abstracts)*. pp. 119-120.
18. Griswold, W.G., et al. *ActiveCampus - Experiments in Community-Oriented Ubiquitous Computing*. 2003, Computer Science and Engineering, UCSD. CS2003-0765.
19. Harel, D., *Statecharts: A visual formalism for complex systems.* Science of Computer Programming, 1987. **8**(3): pp. 231-274.
20. Hong, J.I. and Landay, J.A. *An Architecture for Privacy-Sensitive Ubiquitous Computing*. in *Mobisys'04*. Boston, MA. pp. 177-189.
21. Hong, J.I. and Landay, J.A., *SATIN: A Toolkit for Informal Ink-based Applications.* CHI Letters, 2000. **2**(2): pp. 63-72.
22. Klemmer, S.R., et al., *SUEDE: A Wizard of Oz Prototyping Tool for Speech User Interfaces.* CHI Letters, 2000. **2**(2): pp. 1-10.
23. Landay, J.A. and Myers, B.A., *Sketching Interfaces: Toward More Human Interface Design.* IEEE Computer, 2001. **34**(3): pp. 56-64.
24. Lieberman, H., *Your Wish Is My Command: Programming by Example*. 2001: Morgan Kaufmann.
25. Lin, J., et al., *DENIM: Finding a tighter fit between tools and practice for web site design.* CHI Letters, 2000. **2**(1): pp. 510-517.
26. Lin, J., Thomsen, M., and Landay, J.A., *A Visual Language for Sketching Large and Complex Interactive Designs.* CHI Letters, 2002. **4**(1): pp. 307-314.
27. Macromedia, Director. http://www.macromedia.com/software/director/.
28. Mayor, M., New Wireless Device Could Rescue Firefighters. http://www.wirelessnewsfactor.com/perl/story/9134.html.
29. Nagel, K., et al. *The Family Intercom: Developing a Context-Aware Audio Communication System*. in *Ubicomp 2001*. Atlanta, GA. pp. 176-183.
30. Navas, J.C. and Imielinski, T. *Geocast - geographic addressing and routing*. in *MobiCom 1997*. pp. 66-76.
31. Pascoe, J. *The Stick-e Note Architecture: Extending the Interface Beyond the User*. in *IUI 1997*. pp. 261-264.
32. Pfeiffer, E.W., *WhereWare.* MIT Technology Review, 2003: pp. 46-52.
33. Priyantha, N.B., Chakraborty, A., and Balakrishnan, H. *The Cricket Location-Support System*. in *MobiCom 2000: The Sixth Annual International Conference on Mobile Computing and Networking*. Boston, Massachusetts. pp. 32-43.
34. Rhodes, B. and Starner, T. *The Remembrance Agent: A Continuously Running Automated Information Retrieval System*. in *PAAM*. London, UK. pp. 487-495.
35. Russell, S. and Norvig, P., *Constraint Satisfaction Problems*. Second ed. Artificial Intelligence: A Modern Approach. 2003: Prentice Hall. 137-160.
36. Saund, E., et al., *Perceptually-Supported Image Editing of Text and Graphics.* CHI Letters, 2003. **5**(2): pp. 183-192.
37. Schilit, B., et al. *Challenge: Ubiquitous Location-Aware Computing and the Place Lab Initiative*. in *WMASH 2003*. San Diego, CA. pp. 29-35.
38. Schilit, B.N. and Theimer, M.M., *Disseminating Active Map Information to Mobile Hosts.* IEEE Network, 1994. **8**(5): pp. 22-32.
39. Sohn, T. and Dey, A.K. *iCAP: An Informal Tool for Interactive Prototyping of Context-Aware Applications*. in *CHI 2003 (Extended Abstracts)*. pp. 974-975.
40. Takayama, L., et al. *You're Getting Warmer! How Proximity Information Affects Search Behavior in Physical Spaces*. in *CHI 2003 (Extended Abstracts)*. pp. 1028-1029.
41. Want, R., et al., *The Active Badge Location System.* ACM Transactions on Information Systems, 1992. **10**(1): pp. 91-102.
42. Weiser, M., *The Computer for the 21st Century.* Scientific American, 1991. **265**(3): pp. 94-104.
43. Wolber, D., *Pavlov: an interface builder for designing animated interfaces.* ACM Transactions on Computer-Human Interaction (TOCHI), 1997. **4**(4): pp. 347-386.