Visually Interactive Location-Aware Computing

Kasim Rehman, Frank Stajano, and George Coulouris

Computer Laboratory
University of Cambridge
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
{kr241,fms27,gfc22}@cam.ac.uk

Abstract. The physical disappearance of the computer, associated with Ubicomp, has led to a number of interaction challenges. Due to the lack of an interface users are losing control over applications running in Ubicomp environments. Furthermore, the limited ability for these applications to provide feedback makes it difficult for users to understand their workings and dependencies. We investigate whether an interaction paradigm, based on the visualising location-aware applications on a head-mounted display, is feasible and whether it has the potential to improve the user experience in the same way graphical user interfaces did for the desktop. We show the feasibility of the idea by building an Augmented Reality interface to a location-aware environment. Initial user trials indicate that the user experience can be improved through in-situ visualisation.

1 Introduction

Long-term use of indoor location-aware applications, has brought to light a number of usability problems. The disappearance of the traditional "interface" in the Ubicomp paradigm has resulted in users not being able to control or understand such applications, to an extent that makes them feel comfortable. This research proposes one solution to this problem.

Our group's research into indoor location-aware applications in the course of the Sentient Computing Project [1] has examined how we can support office workers in their daily interaction with computing, communication and I/O facilities by letting applications adapt to changes in location of users and things. Over the past years users have been supported in having phone calls forwarded automatically to their current location; having videos, notes and documents recorded along with the user's current context; being notified about events in the physical world etc. Notably, these applications have been designed for spontaneous walk up and use.

Contrary to what you might expect the user experience relating to such applications has remained suboptimal. For example, automatic actions often occur without users knowing why. Sometimes expected actions are not performed by the system for no apparent reason. What characterises such breakdowns in location-aware applications is that they are entirely unintelligible for most users.

These problems are not accidental but at the root of context-aware computing. Bellotti and Edwards [2], starting from the point of view that complex machine inferencing

M. Beigl et al. (Eds.): UbiComp 2005, LNCS 3660, pp. 177–194, 2005.

based on human context is a "difficult proposition", recapitulate on four design principles that need to be adhered to. More specifically, they recommend that context-aware systems inform the user of *system capabilities and understandings*, and provide *identity/action disclosure* ("Who is that, what are they doing and what have they done?), *feedback* and *control*.

A number of other Ubicomp researchers have pointed out problems along these lines such as Rehman et al. [3], Bellotti et al. [4], Edwards and Grinter [5], Shafer et al. [6], Dourish [7], Dey et al. [8] and Odlyzko [9].

The more interesting part, however, is how to solve these problems. Location-aware walk-up-and-use applications in particular offer little facilities for feedback and control as opposed to PDA-supported location-aware applications.

In our attempt to tackle this challenge we decided to introduce visualisation into the location-aware environment. One of the research questions we are interested in is, can we reap benefits from visualisation in Ubicomp[1] in the same way the desktop benefited from its introduction in the form of graphical user interfaces (GUIs). Amongst the benefits of desktop GUIs has been the provision of a good mental model [10], the ability to achieve your goals through a number of predictable interaction steps, due to a small set of standard interaction facilities; and, very importantly, it *shows* us the system state at any one point in time. Each of these features seems relevant to the Ubicomp interaction problem. In the following we will present a platform for building location-aware applications that exhibit these features.

A head-mounted display (HMD) combined with Augmented Reality (AR) [11] makes it possible to give users the illusion that visualisations are co-located with devices, people and physical spaces: objects on which location-aware applications operate. We will show how location-aware applications can make use of such a facility to convey a mental model and provide feedback, referring directly to the objects of interest. Combining this with a personal interaction device, we can create a new visual interaction paradigm which allows for control as well.

Our main result is that introducing visualisation into Ubicomp, firstly, allows users to make a better mental model of the application, secondly, reduces the cognitive load associated with the application and, thirdly, gives them a more empowering user experience.

2 System Description

Before we present the platform we created to build visually interactive location-aware applications we will briefly introduce AR.

2.1 Augmented Reality

In its widest sense any system that connects the real and virtual world can be labelled "Augmented Reality". As such, even tangible interfaces are examples of AR. A narrower definition involves a system that uses an HMD, a tracker and 3D graphics. The

[1] We regard location-aware computing, or the more general context-aware computing as flavours of Ubicomp.

tracker continuously measures the position and orientation of the head. The system responds to the user's head movement by continuously shifting the virtual viewpoint it uses to display a 3D graphics scene on the *see-through* HMD. This makes the virtual scene *appear co-located with the physical world*. Achieving a good alignment, also called registration, is notoriously difficult and depends on good calibration of the HMD [12]. AR requires trackers with high accuracies. Even though these are often tethered there have been successful mobile indoor AR systems [13]. Figure 1 shows the equipment that needs to be carried around by the user in a typical mobile AR setup.

Fig. 1. Equipment required for the tetherless version of our system: a laptop, a helmet with an HMD and a camera, and the HMD electronics unit.

2.2 Architecture

Figure 2 shows the architecture of our system. The Ubicomp backend that runs our location-aware environment is called SPIRIT [14]. It stores a virtual world model in a database. This virtual world can be regarded as a mirror image of the real world. Every real world "smart" object and person has a virtual CORBA [15] proxy that provides an interface to their virtual state and capabilities. SPIRIT gives application developers access to these interfaces. SPIRIT's crucial property, however, is that the world model is a spatial model of the physical environment that is managed by SPIRIT. As smart objects and people move in the real world their locations and spatial relationships in the world model are updated. SPIRIT can naturally only update locations if the real counterparts are tracked by the Active Bat system.

The Active Bat [14] system used by SPIRIT is an indoor ultrasound location system. With this system Active Bats (Fig. 3) can be located anywhere in our laboratory within 3 cm 95% of the time. The Bat is a small trackable tag that is about 85 mm long. Two small buttons are located on the left side.

SPIRIT allows applications to subscribe to events. Using the spatial model of the physical environment, low-level Bat sightings are abstracted to high level sentient events

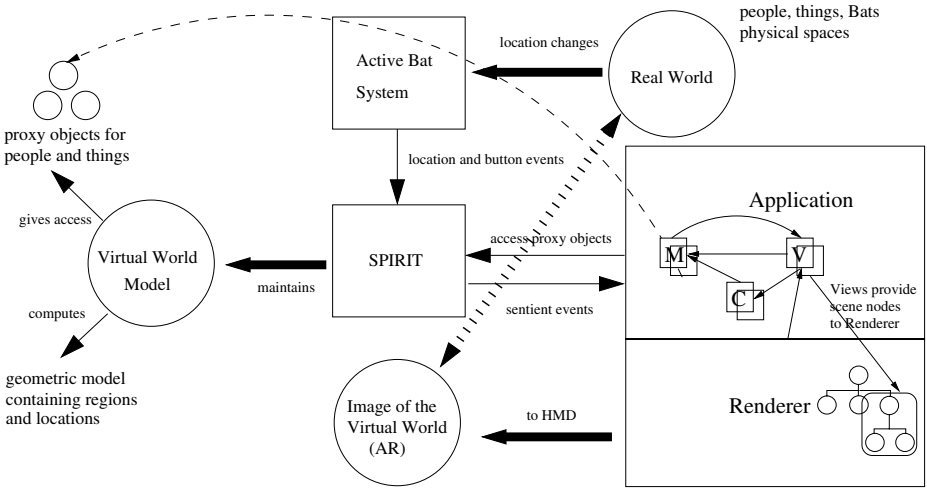

Fig. 2. How to bring an image of the virtual world into the real world. Solid arrows indicate system interactions. Dashed arrows show conceptual relationships between the virtual world, real world and the image of the virtual world. The important arrows have been made thick.

Fig. 3. The Active Bat

to be used by application developers. Example high-level events are: “Andy is close to Pete”, or “Andy is in the same room as John”. By using Bats to tag objects and people we can let our applications reason about the spatial relations of objects and/or users. Each lab member is equipped with a personal Bat that not only tracks them but can be used to initiate actions in the location-aware environment (by pressing its two buttons). Button press events are forwarded to SPIRIT in the same way as Bat movement events,

which allows SPIRIT in turn to generate events such as "Andy's Bat's side button was pressed while being held close to the monitor". This allows application developers to introduce interaction into their location-aware applications.

The most important abstraction of the SPIRIT system is a physical region. For each object or person a set of regions is stored. These are predefined around the particular person or object, having different sizes and shapes. High-level events are generated by evaluating the overlap and containment of such regions. More on how to employ regions to compute spatial relationships can be found in [14].

The question we faced was how would a visually interactive location-aware application interface with the existing SPIRIT system. There are a number of issues here but our main concern was that we wanted to visualise parts of the virtual world, i.e. we wanted the user to see what is happening inside the SPIRIT system, rather than building an interface that receives messages from a particular location-aware application in the course of its execution. In the first case, visualisation and application are accessing the same data structures, in the second case the visualisation is a client of the application.

The architecture devised to fulfill this requirement was an object-oriented Model-View-Controller (MVC) architecture [16]. This implies that the application is modelled as a set of triples, each containing a Model, a View and a Controller[2]. Each domain object the application operates on is mapped to one such triple.

The visualisation essentially consists of constructing a 3D world on the user's HMD. This is achieved through a scene-graph-based 3D graphics package [17]. A component called Renderer provides a platform to build visually interactive location-aware applications on. It takes care of all AR-related operations in order to separate them from the core application logic. It also maps the view hierarchy constructed in the application to the scene graph; views are organised entirely hierarchically in MVC.

Models in the application are images of the actual virtual proxies. These Models are able to access the state and capabilities of the virtual proxies and can be regarded as equivalent from the application's point of view. The important connection is as following: Models are representatives of objects living in the virtual world and Views merely visualise them. The Views make up the 3D world the user sees through the HMD. Hence, the user sees an image of the virtual world overlaid on the real world. We can now relate the virtual state of a smart object directly to its physical embodiment.

We have conveyed a very high-level view of our system since in this paper we are mainly interested in studying the effects of our system on users; a more accurate description of our system for the purpose of reproduction can be found in [18].

3 Introducing Visual Interaction into a Location-Aware Application

In order to put our interaction paradigm into practice we chose following approach: We ported a location-aware application already deployed and used in our lab to our platform so that it could provide feedback via the user's HMD. The ultimate aim was to compare two versions of essentially the same application.

[2] The original MVC allows models to have multiple Views, but we only need one View per Model to generate the AR overlay.

3.1 A Typical Location-Aware Application

The application we chose to port for our user trial is the *Desktop Teleport* application already deployed in our lab. Many GUI environments allow you to have different Desktops, each containing a particular set of applications, documents and settings. In this location-aware teleport application, users can *walk up* to a computer, press a button on their Bat and have a Desktop that is running on a different computer "teleported" onto the current computer. VNC [19] is used in order to achieve this. VNC stands for Virtual Network Computing and allows users to access their GUI Desktop remotely from any computer. The computer running the Desktop locally contains a VNC Client that is listening to "connect Desktop" events from the middleware, which are initiated by the Bat button press. When it receives such an event it connects to a VNC server which then sends bitmapped images showing its current screen to the client. The server receives mouse and keyboard events in return. It is important to note that users can have multiple Desktops running simultaneously.

One use for this application would be to walk up to a random computer in the lab, click the Bat button and bring up your personal Desktop that contains your email inbox. After checking your email you can disconnect. All of this is done without having logged in or out.

The teleport application makes use of "active" regions defined around computers in our lab. When users enter one of these active regions, the Bat buttons invisibly gain functionality. The upper button cycles through the Desktops of the user, since she can have more than one running. The user can see a different Desktop on the screen every time this button is pressed. It is possible for the user's Bat to be in two teleport regions simultaneously. This could be the case if the user is, say, working on two computers that are next to each other and their teleport regions happen to overlap. The lower Bat button will then cycle through available machines.

Sometimes users choose to turn off teleporting, say, because they want the buttons to have some other functionality. Currently, this is being done by holding your bat at a specific location in space and pressing one of the Bat buttons.

The description of this application will immediately reveal a number of potential usability problems.

- One problem is that the Bat can *invisibly* take on different functionalities according to where in the physical space it is located (inside or outside a teleport region). With many applications running simultaneously this can become a considerable problem; in general, applications can turn any part of the physical space into an "active" region.
- Another problem is the different concept of the active region that system and user have. In the teleport application the design idea is that the user's Bat starts controlling the Desktop when she is standing in front of the computer. The SPIRIT system evaluates this by testing for a region overlap as described in Sect. 2.2. The user, on the other hand, does not use regions in order to understand the concept of "in front of a computer". The result is that user and computer will have slightly different ideas of where the teleport region is.
- Finally, we face the usual problems of applications without a feedback path. The "nothing happened" syndrome is notorious in our lab. Basically, error diagnosis by

the user is impossible and the only advice given to users is to try again and email support.

In many ways the application is typical for what we might expect from location-aware applications, should they become pervasive. It contains a mix of implicit (user location) and explicit (button press) interaction . It needs to deal with user settings (teleporting on or off). Furthermore, it involves a networked service (teleporting service). Finally, it uses location contexts that are more fine-grained than rooms.

3.2 Interaction Prototypes

One of our aims when introducing this interaction paradigm was to supply a set of widgets with it as well. The question was what kind of widgets will AR-based visual interaction in Ubicomp environments require? In a large survey of Ubicomp applications we found a number of patterns of interaction. The set of widgets [18] we built for these patterns is centred around the Active Bat as a personal interaction device. The concept of a personal interaction device that is always carried around by the user has been suggested in previous Ubicomp literature [20,21]. A big advantage such a device has is that you can use it to address the system (in Bellotti's terms [2]).

In the next section we will discuss two of our widgets in use in a real application: *Bat Menu* and *Hot Buttons*.

3.3 The First Interactive Application in Space

Using object-oriented analysis we identified all objects of interest and split them up into Models, Controllers and Views. The Bat buttons that had previously gained functionality invisibly were now labelled using AR. We employed our *Hot Buttons* widget which is similar to the way "hot buttons" work on mobile phones. Their descriptions change according to the context.

The teleport region, also previously invisible, was read from the world model and visualised as a polygon in space. The current machine was indicated by an outline around the actual monitor. Figure 4 shows what the user sees through her glasses when walking up to a computer. Users can now see where to place or where not to place their Bat in space in order to achieve what they want. We use stereoscopic see-through glasses in order to support depth perception, which is necessary when you are visualising regions in "thin air".

When the user walks into a teleport region, new labels appear on her Bat buttons, signifying the relevant functionality. They disappear when the user leaves the region. Inside the region the user has the ability to switch through her Desktops. As previously, this is accomplished by the user pressing the upper button on the Bat. We decided to visualise this interaction using our *Bat Menu*. A menu appears overlaid next to the Bat with each item indicating a Desktop by name. As the user presses the upper Bat button, Desktops switch through on the computer as before, but now she sees a red outline on the menu jumping from item to item. The current Desktop on the computer and the current menu item always match. The augmented Bat with the menu is shown in Fig. 5. The menu of Desktops is controlled by the button marked by the overlay "Desktop>>".

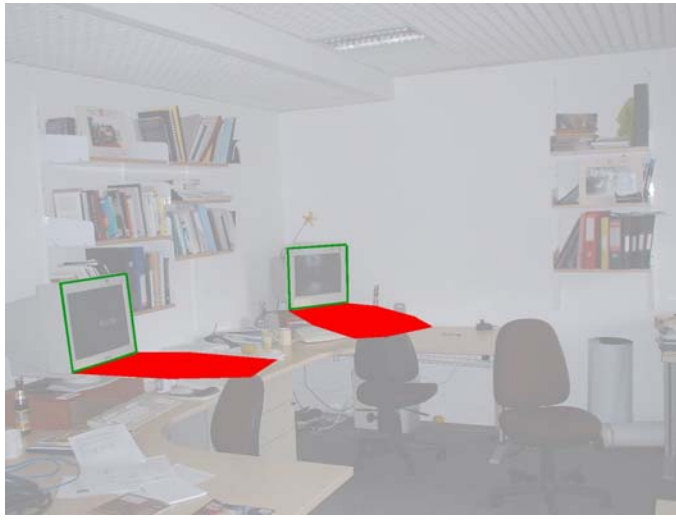

Fig. 4. Users see the teleport regions through their glasses. The regions are shown just below both monitors (The HMD is see-through, so the real-life scene is not present in the computer-generated image. Therefore, the composite must be simulated in order to show it here.).

Fig. 5. The augmented view of a Bat while inside a teleport region. Square menu items show desktop names (here too, the composite picture is simulated).

Teleport-able machines (computers) will have a green outline overlaid around their actual monitor. Using the lower Bat button, labelled in green, the user can cycle through the machines, a bright green outline jumping from monitor to monitor indicating the current machine.

A big influence in designing the interaction was Norman's conceptual model [10] methodology. Its idea is that as a designer you create a model for the user by communicating to her (visually) how to use your product; in essence the designer is translating a user manual into visual design.

Applying it to our design meant that we had to make sure that in each use case the application always shows the user what is possible, how to achieve it and how to evaluate whether you have achieved it. As an example, users of our Bat Menu can instantly identify each of these.

4 User Evaluation

The visual and non-visual version of the teleport application were now compared against each other in a user trial. One feature not included in the trial was the ability to switch between machines that are located next to each other; this feature is generally not used in the lab. Instead you can now control the state of the teleporting service using the lower Bat button, no matter where you are standing.

The visualisation shows the teleport regions around the computer in which teleporting can be initiated using the Bat button labelled with the AR overlay "Desktop>>". The second button has an overlay label that reads "Teleporting is on" or "off", depending on whether you are inside our outside of the teleport region. Pressing it will toggle the label from one state to the other.

4.1 Method

The number of test subjects was chosen to be ten. Five of the test subjects can be regarded as novices and five as experts depending on their familiarity with location-aware applications.

The trial consisted of 5 parts. Each of the two experimental parts was preceded and followed by an interview part. The first experiment was a test involving the non-augmented teleport application, the second, the augmented teleport application. One might argue that performing these two parts in sequence will result in users being more familiar with the application when they use the augmented version of the application, but the experiment had to be done in one order and this was the most sensible one. Furthermore, the test subjects were given time to familiarise themselves with the application before they were tested.

The aim was not just to get their answers to questions but also find out *why* they gave particular answers or performed in a certain way. Therefore, we used a combination of short answer questions, and open questions that encouraged the test subject to talk; for the experimental part we employed user observation and thinking aloud techniques. The tasks consisted of giving answers to *what if* questions while using the application. Interviews were flexible with the evaluator drilling down to detail if the test subject

had something interesting to say. The tasks and initial interview questions, however, remained the same for all. The guide used for the experiments/interviews is shown in Appendix A. We are only presenting our most important results. Details can be found in [18].

One premise we use for the interpretation of our observations is the mental model theory. Mental model theory [22] assumes that humans form internal representations of things and circumstances they encounter in everyday life in order to explain how they work. One important aspect is that these representations are "runnable" in the head, i.e. they can be used in order to predict the result of a particular interaction with the world. They are not always accurate, which is why humans can have misconceptions about the effects of their interaction. Nevertheless, a mental model can be updated to a more accurate one when a situation occurs where a misconception becomes obvious.

4.2 Lessons Learnt

Users Can Be Provided with a Conceptual Model of a Location-Aware Application The conceptual model methodology [10] briefly introduced in Sect. 3.3 assumes that humans make mental models about applications or products we design and that designers can influence the formation of these[3]. Two questions we were interested in were:

1. What do the internal representations that users make of location-aware applications look like?
2. Can we, through our visualisations, influence the mental model they make of the location-aware application?

The basis for eliciting the mental models users built of the application are the *what if* questions (Appendix A), the explanation of how the application worked and an additional task given to the test subjects. The additional task was described as following:

> Imagine you want to provide a manual of the application for other users. Instead of a description can you draw a diagram for this purpose. Try not to use text if you can.

First of all, we can say that the mental model theory is suitable to explain our findings. Users are able to answer questions about what the effects of particular actions are using their mind only. When building these models users make certain assumptions about how things should work. For example, one test subject thought you need to point the Active Bat at the monitor in order to teleport. This, even though neither the Active Bat nor the monitor show any clues that a directional signal is used between them. Another test subject thought the teleport regions exactly coincide with the extent of the desks on which our computers stand.

Interestingly, we observed that such misconceptions were not at all limited to novices. In fact every test subject had some kind of idea of where teleporting would be active.

[3] We are using a broad definition of conceptual model here.

Especially, the case of the person who associated the desk extent with the teleport region for no logical reason, shows that users might need to have some visual idea of where this region is. So, *by trying to aim for invisibility we leave a gap in the user's mental model that is filled by self-initiative.*

Another observation is that mental models about the application can vary a lot. For example, one of the test subjects, in his explanation employed no metaphors at all. The drawing produced by him even includes a reference to a variable and a lot of text. So, in general we can say that this is a non-visual person. As a contrast another person produced a drawing in which he visualises the on/off button as a light bulb. His depiction is fairly concrete, like an image. This by the way was the only fully correct "manual" we received. Another person seemed to have a more procedural model. His "manual" includes a number of different cases that "work" or do "not work". He depicted four cases, varying the distance and position of the Bat to the monitor and also the teleport setting. Two other notable metaphors that were employed by the users were, viewing the Bat as a remote control and viewing the application as a state machine.

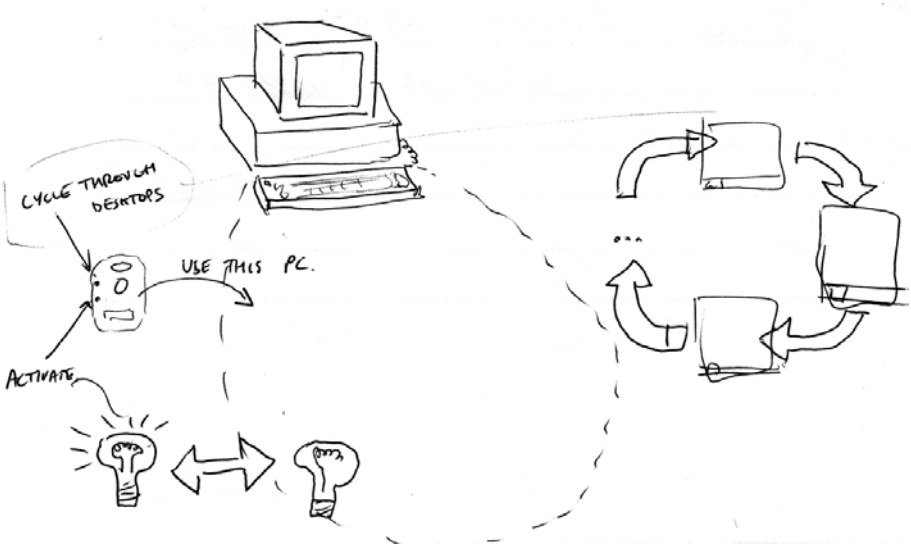

Fig. 6. Full marks for this diagram. The test subject has a good mental model of the application

We shall now examine how the visual interface affected the user's mental model of the application. Two "tricky" bits can be identified in this application. Firstly, the fact that teleporting only works if the user is standing in a particular region and, secondly, the fact that the teleporting state (on/off) influences the function of the first Bat button. Teleporting will not work outside the region but will only work inside it if teleporting is enabled. On the other hand, turning teleporting on or off will work independently of the

location. This makes sense, since users want to turn teleporting on or off independently of their location.

It was found that the overall understanding of the application was much better during and *after* the use of the visualisation. When users were asked to explain how the application works before and after using the visual interface, in general their second explanation was much deeper and more detailed than the first one, especially with respect to the two above-mentioned non-straightforward concepts. The answers obtained in the interviews corresponded to the observations made during the experiments. Seven test subjects had problems working out the *what if* questions whereas nobody had problems with the visual version.

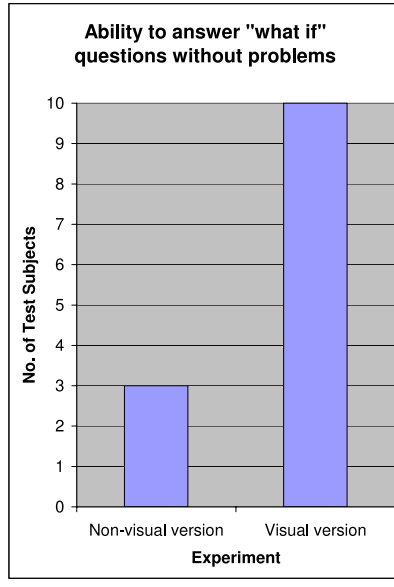

Fig. 7. All test subjects using the visual version could work out all answers.

Visualisation Reduces the Load Location-Aware Applications Pose on the User's Working Memory We stated earlier that users were able to answer all *what if* questions during the visual experiment. Partly, this is due to the increase in user understanding we identified afterwards.

However, the fact that the interface shows you your context, i.e whether you are inside a teleport region or not, we found was somehow disproportionately helpful in answering the *what if* questions. It seemed that thinking about whether you were at the right location "blocked out" thinking about whether teleporting was actually on or off, i.e. visualising *where* something will work, freed cognitive resources for other processing. Remember, that in order for the user to evaluate whether a teleport will

be successful two conditions need to be fulfilled: the user needs to be in the teleport region *and* teleporting needs to be enabled. The fact that this error was so consistently performed struck us as odd.

After consulting some research on *systematic errors* the most plausible explanation is that what we had witnessed was a working memory overload. According to one theory [23], systematic errors are performed when the working memory goes beyond a threshold, but are not performed at all when it is below that threshold. This is one of the more unexpected results of this user trial. Even though we have been using location-aware applications for years in our lab the load some of them pose on the working memory is not mentioned when users are asked about usability problems.

Let us enumerate the items that need to be kept in the user's short term memory for our application: which Active Bat button to use for teleporting, where to stand, whether teleporting is enabled, how to enable it and whether the machine has a listening VNC client running on it; and all of this is just for one application. Looking at it from this perspective it becomes clear how a memory overload could occur.

Another observation was that only expert users could remember how many Desktops they have running at the beginning of the experiments. Many users in the lab have Desktops running for months because they forget about them. Since Ubicomp is supposed to support unstructured, often interrupted tasks, offloading memory requirements is desirable.

Visualising the Ubicomp System Could Create a New Kind of User Experience We shall now examine the effects of introducing visualisation on the general user experience. This is not a full evaluation of how the user experience changes if you are experiencing all location-aware applications through an AR interface. Many more experiments with a non-prototypical system would be required for that. Nevertheless, we can obtain hints as to how the user experience will change if visualisation becomes more widely used.

Users were generally very happy with the visual interface. Nine out of ten test subjects made positive or very positive statements in this respect. One of the test subjects said that the Augmented Reality interface lets you know that "the application is not broken". She was an experienced user of location-aware applications and this seemed to be her biggest problem with location-aware applications in general. The remark says more about the user experience users currently have with "invisible" location-aware application than it applies to the visually enhanced version.

Interestingly, providing users with a better kind of location-aware application made clear to us what users had been missing, or rather been putting up with so far:

- Especially experienced users appreciated the fact that the Active Bat could give visual feedback. The only feedback received currently from our Bats is audio in the form of beeps of different pitches. One test subject explained that when she hears a beep or a sequence of beeps she has "no idea of what is going on".
- Another test subject said he would not rely on the teleport application currently deployed in our lab and would always have a backup if he planned to use it in order to teleport his desktop containing presentation slides to a presentation room (a popular use of the application).

– Finally, one misconception a user had of the existing teleport application was that he had thought the teleporting region was only a small region around the monitor. What was peculiar, was that he was a frequent user of the existing teleport application. He did not realise that the teleport region was a lot bigger, simply because he only uses the small region in front of the monitor.

What these examples show is a particular attitude towards location-aware applications. Apparently, users hardly explore them. They are conservative in the sense that they only use what they know works and even then they are in a constant state of uncertainty as to whether it is performing or not. This is, of course, not the attitude we as designers can allow the users to have. What needs to be done is to work on changing this user experience. We need to spend time thinking how we can give users the feeling that they can rely on, even play with, the applications without breaking them.

In this context, what was mentioned again and again was a kind of "coolness" factor experienced by users using the Augmented Reality interface to the location-aware application. Possibly, by generally introducing more enjoyable features into location-aware applications we can influence the user experience.

4.3 User Feedback

At the end of the experiments test subjects were asked what was the most desirable and worst feature of the system. The following gives a list of the most desirable features mentioned: Feedback, predictability, "coolness", explicit showing of location contexts and visualisation of the Desktops as a menu.

Most of these points have already been discussed in the previous sections. There is no clear cut definition for "coolness", but it is the adjective used by several test subjects.

The most undesirable features can be listed as: Calibration, bulkiness of hardware, slow update rate and the limited field of view.

Calibration relates to a short (10 s on average) process to be performed once for the experiment by each user. Test subjects had to adjust the HMD until they would see a virtual cube on a particular location. The slow update rate is not a property of the head tracker but comes from the Active Bat system (around 2 to 3 Hz). Hence, only location updates of the Active Bat overlay suffered from this problem. The rest of the application was running at 30 Hz, the update rate obtained by the tracker. The limited field of view is due to the HMD. Since it contains mini-monitors it uses to generate the virtual images, its field of view cannot wrap around the user's head.

Initially, we had expected that achieving accurate enough overlay for interaction with a location-aware application might be difficult. However, we were able to resolve this issue by careful pre-calibration of the HMD to an extent that misalignment was not mentioned as an undesirable feature by a single user. In general, we found that users had no problems fusing the virtual and real image, i.e the visualisations were in deed regarded as being co-located with the real world and not interfering with it.

It has to be said that undesirable features were hardly ever mentioned in the interviews. It was only after prompting test subjects that these were mentioned. This does not mean that these features are negligible. In fact most of these will become bigger problems when users have to wear HMDs for a longer period than a few minutes. On

the other hand, we are not proposing to deploy this system at the present time in its present form. This is only a starting point and the future will show in which direction this research will develop.

5 Outlook

The limited display facilities we have in Ubicomp can prove to be a potential stumbling block in developing this research further. Not everyone feels comfortable using an HMD. Also, most real world locations are not fitted with an accurate tracking system. Before exploring a number of alternatives we would like to make clear to what extent our system is independent of the Active Bat system and the use of an HMD.

As Fig. 2 shows, our system receives updates about location changes and button presses in the form of sentient events. Such events can be generated using any location technology. Also, MVC makes no assumption about the display technology used. Any display technology can be used provided that a View object has access to the display. It makes architecturally no difference whether the display technology used is a PDA, projector, LCD display or an HMD. In each case a View object will be receiving updates from the Model and display-specifically mapping these to visualisations.

Let us look at the practical implications of different display and sensing technologies. Overlaying visualisations on movable objects (such as the Bats) is not possible without an accurate tracking system. However, if real world objects themselves provide a small (colour) display there is no need to track them to such a high accuracy[4]. The power consumption of such displays can, of course, be a limiting factor for years to come.

Nonetheless, other opportunities to visualise Ubicomp applications exist. Projector-based visualisation such as the Everywhere Display [24] appears promising. By combining a camera with a projector, interactive visualisations can be created on real world objects. Most notably, it allows us to visualise "active" regions as long as there is a display surface available, such as the floor.

PDAs can also be used to render visualisations. One of its more sophisticated uses would be to use it as a "portal" [13] to the virtual world. Wagner et al. [25] recently presented AR on a PDA. The PDA uses a camera and overlays virtual images on the live feed. The PDA could show users the same visualisations we have used.

Finally, less obtrusive displays than the one we used, almost indistinguishable from normal eyeglasses and better suited for day-long wearing, have been on the market for a couple of years now.

This discussion, however, should not distract us from the main point of the paper. The study of the effects visualisation has on users is to a large extent independent from what technology we use. The user experience after introducing visualisation did not improve because users were impressed by AR visualisations, but because they were feeling much more in control.

[4] The objects still need to be tracked in order to create location-aware behaviour, but the accuracy required can be far less, depending on the application.

6 Conclusion

The physical disappearance of the computer in the Ubicomp paradigm will continue to lead to usability problems. Users have difficulties in using hidden features of "smart" objects and in understanding what virtual implications their actions have, or in fact don't have.

Our hypothesis has been that we can increase application intelligibility by visualising such smart applications.

In order to test our hypothesis we built a system to upgrade location-aware applications with Augmented Reality visualisation capabilities. We stress that our prototype, while not of production quality or particularly comfortable is not a demo but a complete, usable system. We applied it to an application that is already deployed and used in our lab: Desktop Teleporting. For the first time a location-aware application had a chance to present its inner workings to the user. Most notably, we were able to show users spatial aspects (such as regions) of an application that operates in the physical world.

We then carried out a small-scale but carefully conducted user trial whose outcome validated our hypothesis. In nearly all test subjects we witnessed an increase in user understandability. On the basis of the mental model theory we were able to establish a link between our hypothesis and the result.

Most importantly, using Augmented Reality we were able to give our test subjects, for a limited time, a novel and much more empowering user experience. We believe that visualisation will be a fundamental component in making Ubicomp applications easier to understand and use. We hope and expect that this paradigm will be widely adopted in future computing environments.

7 Acknowledgements

The work described was conducted at the Laboratory for Communication Engineering (LCE, now part of the Computer Laboratory as the Digital Technology Group). We thank its leader Andy Hopper for his vision, encouragement and support. Our research has greatly benefited from the LCE's unique infrastructure, the cooperation with other LCE members and the group's constant drive towards inventing the future.

The psychology-oriented UI work done at the Computer Laboratory's Rainbow Group (current affiliation of the first author), especially by Alan Blackwell who also offered valuable comments on this paper, has been inspiring in drawing some important conclusions. A big "thank you" to our test subjects. Furthermore, the first author thanks Tom Rodden for an enlightening and challenging discussion on Ubicomp Design.

The first author was generously supported by AT&T Research Laboratories, Cambridge; Cambridge University Board of Graduate Studies; Cambridge European Trust; Cambridge University Engineering Department and St. Catharine's College, Cambridge.

A Guide Questions to Be Used by the Evaluator

1. How many Active Desktops do you have?
2. Is your Teleporting on or off? Would you prefer to control it from your bat or a "SPIRIT Button" on the wall?
3. What do you know about Teleporting?
4. How does it work? (For novices delay this question, until they have explored the application)
5. *Evaluator*: identify concepts, conventions and prompt user
6. Can you Teleport to the broadband phones? Our phones are embedded computers. The aim of this question is to find out whether they believe that every computer in the Lab "affords" [10] teleporting.
7. *Evaluator*: Explain experiment.
8. Experimental Part I begins. *Evaluator*: Let user play with invisible application, observe difficulties.
9. *Evaluator*: Ask "what if" questions involving one Bat press, user movement and a Bat button press and combinations of the two. Experimental Part I ends.
10. Imagine you had to give another user a manual for this application. Can you make a drawing instead?
11. Experimental Part II begins. *Evaluator*: Let user play with visible application, observe difficulties.
12. *Evaluator*: Ask "what if" questions involving one Bat button press, user movement and a Bat button press and combinations of the two. Experimental Part II ends.
13. How does it work?
14. *Evaluator*: identify concepts, conventions and prompt user
15. Teleporting is best described as a property of: Space, Bat, Machine, "System", Bat System, other:

References

1. Andy Hopper. The Royal Society Clifford Paterson Lecture, 1999. Available at: `http://www.uk.research.att.com/pub/docs/att/tr.1999.12.pdf`.
2. Victoria Bellotti and Keith Edwards. Intelligibility and Accountability: Human Considerations in Context-Aware Systems. *Human-Computer Interaction*, 16(2, 3 & 4):193–212, 2001.
3. Kasim Rehman, Frank Stajano, and George Coulouris. Interfacing with the Invisible Computer. In *Proceedings NordiCHI*, pages 213–216. ACM Press, 2002.
4. Victoria Bellotti, Maribeth Back, W. Keith Edwards, Rebecca E. Grinter, D. Austin Henderson Jr., and Cristina Videira Lopes. Making Sense of Sensing Systems: Five Questions for Designers and Researchers. In *Conference on Human Factors in Computing Systems*, pages 415–422, 2002.
5. W. Keith Edwards and Rebecca E. Grinter. At Home with Ubiquitous Computing: Seven Challenges. In *Proceedings of the 3rd international conference on Ubiquitous Computing*, pages 256–272. Springer-Verlag, 2001.
6. Steven A. N. Shafer, Barry Brumitt, and JJ Cadiz. Interaction Issues in Context-Aware Intelligent Environments. *Human-Computer Interaction*, 16(2, 3 & 4):363–378, 2001.

7. Paul Dourish. What We Talk About When We Talk About Context. *Personal and Ubiquitous Computing*, 8(1):19–30, 2004.
8. Anind K. Dey, Peter Ljungstrand, and Albrecht Schmidt. Distributed and Disappearing User Interfaces in Ubiquitous Computing. In *CHI '01 Extended Abstracts on Human factors in Computing Systems*, pages 487–488. ACM Press, 2001.
9. Andrew Odlyzko. The visible problems of the invisible computer: A skeptical look at information appliances. *First Monday*, 4, 1999. Available at: `http://firstmonday.org/issues/issue4_9/odlyzko/index.html`.
10. D.A. Norman. *The Design of Everyday Things*. The MIT Press, 1989.
11. Steven K. Feiner. Augmented Reality: A New Way of Seeing. *Scientific American*, April 2002.
12. H. Kato and M. Billinghurst. Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. In *Proceedings 2nd International Workshop on Augmented Reality.*, pages 85–94, 1999.
13. J. Newman, D. Ingram, and A. Hopper. Augmented Reality in a Wide Area Sentient Environment. In *Proceedings ISAR (International Symposium on Augmented Reality)*, 2002.
14. A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The Anatomy of a Context-Aware Application. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom-99)*, 1999.
15. Object Management Group. *The Common Object Request Broker: Architecture and Specification, Revision 2.0*, July 1995.
16. G. Krasner and S. Pope. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 system. *Journal of Object Oriented Programming*, 1(3):26–49, 1988.
17. Josie Wernecke. *The Inventor Mentor*. Addison-Wesley, 1994.
18. Kasim Rehman. Visualisation, interpretation and use of location-aware interfaces. Technical Report UCAM-CL-TR-634, University of Cambridge, Computer Laboratory, May 2005.
19. T. Richardson, Q. Stafford-Fraser, K.R. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1):33–38, Jan/Feb 1998.
20. R.W. DeVaul and A. Pentland. The Ektara Architecture: The Right Framework for Context-Aware Wearable and Ubiquitous Computing Applications. Technical report, The Media Laboratory, Massachusetts Institute of Technology, 2000.
21. Gregory Finn and Joe Touch. The Personal Node. In *Usenix Workshop on Embedded Systems*, 1999. Available at: `http://www.usenix.org/publications/library/proceedings/es99/full_papers/finn/finn.pdf`.
22. Donald A. Norman. Some observations on Mental Models. In Genter and Stevens, editors, *Mental Models*, pages 7–14. Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.
23. Michael D. Byrne and Susan Bovair. A Working Memory Model of a Common Procedural Error. *Cognitive Science*, 21(1):31–61, 1997.
24. Claudio S. Pinhanez. The everywhere displays projector: A device to create ubiquitous graphical interfaces. In *Ubicomp*, pages 315–331. Springer-Verlag, 2001.
25. Daniel Wagner, Thomas Pintaric, Florian Ledermann, and Dieter Schmalstieg. Towards massively multi-user augmented reality on handheld devices. In *Third International Conference on Pervasive Computing (Pervasive 2005)*, Munich, Germany, May 2005.