Extending Aura with an Augmented Reality Interface

CHRISTOPHER PRIDE
ADVISOR: DAVID GARLAN

In a ubiquitous computing environment augmented reality would be an ideal choice for a display for the user. An augmented reality display assists users by adding computer generated information to their perception of reality, thus making it ideal for ubiquitous computing. Unfortunately augmented reality is technically difficult and costly to implement even when the application is designed for its use from the ground up. However, many of the necessary devices for a low fidelity implementation of augmented reality are readily available in a ubiquitous computing environment. Our research focuses on using Aura, a ubiquitous computing framework, to marshal the available devices in the environment. These devices can then be connected within the framework to provide an augmented reality display to the user at the best fidelity possible, given the available resources in a user's environment.

1. INTRODUCTION

In a ubiquitous computing environment ideally a user's devices should vanish into the background. An augmented reality display is traditionally thought of as a head mounted display that adds real time computer generated information to the user's view of reality. A more general description of an augmented reality display is that it adds computer generated information to the user's perception of reality through whatever devices are available. In this way it is not limited to obtrusive head mounted displays. This makes an augmented reality display ideal for a ubiquitous computing environment.

With today's technology designing a program to use an augmented reality display is expensive and restrictive. The program would need to be unique to the devices used and would need to be built knowing ahead of time the requirements of each device you will be working with. Additionally, as devices change or as new ones are released, the program would need to be retooled to interface with these new devices. However in a ubiquitous computing environment the necessary devices for this type of a display method are already available in the environment. In a ubiquitous computing environment, in order to provide context aware features, there is generally a device for location detection, which is something that can be used by an augmented reality display to determine when to update information being given to the user. Frequently a user is carrying a device with a screen, such as a PDA or cell phone; these could also be used to provide information to the user.

There are many challenges in bringing together an augmented reality display. Some of the questions that arise include: How to design input methods for an augmented reality display; how best to actually display information to the user; what method to use to provide output on a head mounted display; how applications change in such an environment; and, what level of detail of contact information is necessary? However the problem that we find most intriguing, and that seems to supersede all others is how to bring together multiple devices currently in use to establish an augmented reality display.

Some of these questions have been addressed, in part, by existing research. The Tinmith project [1] has designed a wearable computer system for high fidelity augmented reality. Their system, while it provides an excellent solution to providing a good augmented reality display that is not limited by the environment, is still expensive and requires development targeted at their platform. iCrafter, a service framework for ubiquitous computing environments [2], addresses some of the problems of how to deal with user interfaces in this type of environment using their framework.

We propose to use Aura [3], a ubiquitous computing framework, to marshal available hardware into the best possible augmented reality display. Aura communicates with hardware in the environment and selects from devices and applications available a subset that can perform the services necessary to accomplish the user's tasks. Thus if we give the user a task that requires the use of an augmented reality display we can use Aura to bring together the available resources to provide such a display.

The contributions of this thesis are threefold: We develop a new abstract view of augmented reality appropriate for retargeting in heterogeneous pervasive computing environments. We demonstrate that Aura can be used to marshal necessary services to provide an augmented reality interface. We identify new capabilities needed in the Aura system for the selection of suppliers and for the configuration of tasks to use complex service connections.

In the remainder of this paper we review the topics of Pervasive/Ubiquitous Computing, Augmented Reality. We introduce the Aura framework in detail. We then discuss the design and implementation of an Augmented Reality interface under Aura. Finally, we discuss the consequences of what we found and suggest further work.

2. CONTEXT/RELATED WORK

2.1 Ubiquitous Computing

2.1.1 Weiser's Vision. [4] Ubiquitous or Pervasive Computing is a model for computation where the computers have been integrated into everyday life often in a transparent way. This vision of computing was introduced in the 1980s by Mark Weiser.

Weiser envisioned a future of computing where computers fade into the background. His vision included three unique computer interfaces: tabs, pads, and boards. Where tabs are effectively post-it notes, pads are sheets of paper, and boards are whiteboard sized. The user interacts with the interfaces as they would interact with non-computing devices, requiring no unusual or new skill-sets. The devices all work together in a seamless manner and fade into the background; thus their ubiquity.

2.1.2 Applications. The field of ubiquitous computing research topics is very broad so there are many different applications. Some well known applications in ubiquitous computing are the smart home projects. [5] The applications of RFID tags are often in the realm of ubiquitous computing. [6]

2.2 Augmented Reality

The concept of augmented reality is that a user's experience of an environment can be augmented with computer generated information. The most commonly discussed form of augmented reality is overlaying 3D information using some form of head mounted display. In this paper we consider a more general form of augmented reality.

2.2.1 Applications. Much research has been devoted to very specific applications of augmented reality. There is a large body of research on using augmented reality visualizations in medicine. For example, this class of application often uses an augmented reality display to add visualizations of sensor data on patients (Ultra sound 3D overlays [7] and similar applications [8]). Another large field of research involves the use of augmented reality displays for manufacture and repair. These applications provide schematic overlays for technicians (Airplane engine plans [9], building schematics [10], and printer maintenance [8]). There are a large number of projects that work on outdoor modeling and exploration [10] [11] [12] [13].

2.2.2 Implementation. There are many different styles of implementation. The style of augmented reality that is most often discussed, where 3D images are overlaid on top of regular vision, is implemented using a head mounted display. There are two main methods to implement this. In one method a camera is coupled with a closed pair of goggles where the overlay is added to the camera images and then displayed on the goggle's lenses. The second method makes use of a semi-transparent head-mounted display and the computer generated information is displayed directly on that. [8]

Recently several toolkits have been released for the implementation of augmented reality applications. Some of the toolkits work with the use of markers. They use visual recognition for markers that are placed in and can be recognized in the environment. [14] [15] Other solutions use cameras which must be calibrated to the environment. This requires programming to establish objects that are in the physical world and then allow for the addition of other virtual objects. [16] The Tinmith project uses orientation and location sensors to adjust the overlay which they generate in a variety of ways. [17]

2.3 Aura [3]

The Aura ubiquitous computing framework is designed with the idea that the user's attention is the most valuable resource in a system [18]. Since advances in computing technology have not changed a user's ability to pay attention, the framework attempts to eliminate extraneous distractions of computer interaction as much as possible.

Aura is a task-oriented system. It is built around the concept that the user wants to accomplish tasks rather than run programs. To accomplish a task a user must employ a number of services. A service is something like text editing; this could be provided by Emacs, MSWord, or any number of programs. Each service is supplied by a supplier. More than one service could be provided by one supplier.

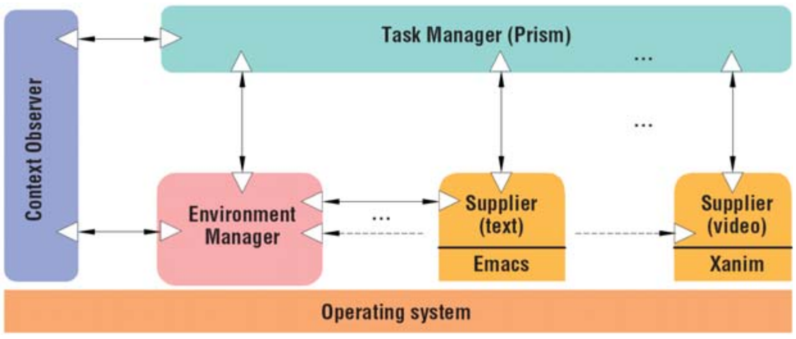

Fig. 1. Aura Architecture

Displayed in Figure 1 is a diagram of the Aura Architecture. Described in the continuing sections are many of the principal components in the Aura Architecture.

2.3.1 Prism. Prism is the component that the user utilizes for task management. The Environment Manager is what keeps track of the available suppliers and marshals the correct suppliers for each task that Prism asks for.

The Prism component of Aura allows the user to configure his or her tasks. It allows the user to select tasks and setup what services are needed for each task. It allows for multiple configurations of services for each task and maintains a feasibility rating for each task. It also contains some context awareness control for switching between tasks.

Prism requests a set of suppliers for a task from the Environment Manager and when it returns a list it considers the preferences of the user about what type of suppliers they prefer and uses that to choose what set of suppliers to use. Once it has a set of suppliers it can set the configuration of the suppliers to the correct settings for the user's task. [19]

2.3.2 Environment Manager. The Environment Manager controls and/or selects from available suppliers. Prism requests a set of services for a task from the Environment Manager. The Environment Manager considers the available suppliers and then proceeds to choose the best available suppliers for the task. It then contacts the suppliers telling them to activate and sets up the connections. It keeps track of the status of the suppliers and will notify Prism if there are problems with a supplier. [19]

2.3.3 Supplier. With Aura, a supplier is the front end for a device or application. The supplier needs to be able to manage (or manipulate) the settings of the device or application for the user and communicates with Prism and the Environment Manager. [19]

2.3.4 Connections. The connections infrastructure establishes and maintains the connections between suppliers, the Environment Manager, and Prism. There is also a specification for connections between suppliers. Suppliers are connected to each other by the Environment Manager and manage the connections on their own. [19]

2.3.5 Aura Operations. So to give an idea of how this works, we will go through the operations involved in a user reviewing a video. The user would create a task, review video. This task would need a text editor and video player. The user could then set up preferences for those services including the video to play and the file to save the review to. Then the user would focus the task and Prism would request suppliers for those services from the Environment Manager. The Environment Manger would check the suppliers in the environment and choose suppliers for the text editor and video player services and start these services. The Environment Manager will tell these suppliers to start and tell Prism who they are. Prism will send the initial configuration to the suppliers and then the user can interact with the suppliers.

3. WHY AURA?

In a ubiquitous computing environment augmented reality is an amazing tool. It is, however, still difficult to marshal the necessary devices, and Aura is, by design, very good at this type of support. In particular; the Environment Manager is very good at selecting the most preferable suppliers in the environment; the concept of suppliers meshes very well with assembling an augmented reality display in a heterogeneous scenario; and Prism allows the user to specify preferences for one device over another.

In our design we discuss: How to model an augmented reality display as a collection of services; How an augmented reality display can be degraded to different fidelities; The reason we focused on thinking through only one type of content to display; and how suppliers can be set up to support degrading to different fidelities of augmented reality.

4. DESIGN

To extend Aura with an augmented reality display we needed to break down the concept of an augmented reality display into a set of services. In the end we decided on breaking it down into three classes of services: input services, content services, and output services.

When working on this design, rather than focusing just on implementing augmented reality at the highest fidelity, we focused on designing a solution that would allow a best attempt augmented reality interface. Thus if the necessary suppliers for high fidelity augmented reality are not available then we would employ the best suppliers available to deploy the best quality augmented reality possible in the situation, even if it is not ideal. Thus if we don't have sensors that can accurately track head movement to keep an overlay aligned, we can instead use a lower fidelity augmented reality display.

While designing this it became clear to us that to work this through properly we needed to focus on a single type of application. We chose to work with path guidance because this is one application where it is easy to consider different levels of fidelity. For example the highest fidelity level could be done with 3D arrows interwoven with the landscape, mid level fidelities could include: compass based navigation, map with a marker, or text step–by-step instructions. The lowest fidelity level could be a map with the path overlaid or textual instructions from the beginning to the end.

One of the major challenges was determining how to set up the display components such that it was capable of degrading in this fashion. We considered a number of different methods for determining this. The method of output selected would depend on what output methods are available and what types of input devices will be utilized. One possible method was to have a different supplier for each display at each level of augmented reality fidelity. This would necessitate a large overhead to support so many different fidelities for each class of display. Another possible solution was to have a translation supplier, which would be informed of the different types of input suppliers and the output suppliers in use and then would determine the correct type of output. This turned out to be unfeasible under the current Aura architecture, as Aura lacks a method for a supplier to be informed of the different types of suppliers available. The method that we adopted involved a mix between the two types of suppliers. The design we settled on was to have one translation supplier for each type of augmented reality fidelity due to limitations in the current design of Aura.

5. IMPLEMENTATION

We decided to simulate the suppliers as the effort of assembling programming the actual physical devices necessary for augmented reality was beyond the scope of this project.

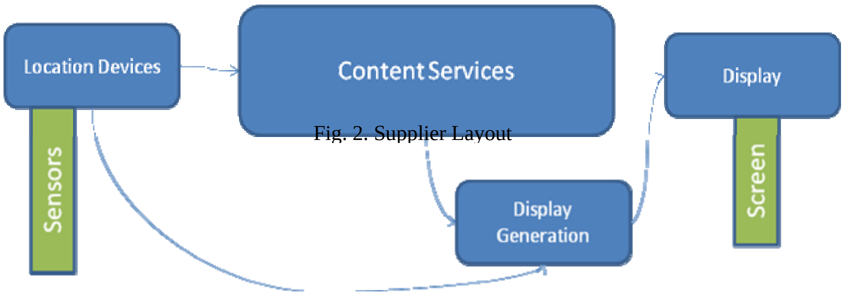

Fig. 2. Supplier Layout

Aura already has a provision for simulating an application as has been demonstrated in other applications. To meet our needs for this simulation we extended the “stubs,” as they are called, to support connections between suppliers. The current implementation of stubs in Aura run a script that has three types of actions: send, receive, and sleep. What is received is based on Aura message types. The send action uses prebuilt messages that were sent either to Prism or the Environment Manager based on message type. We extended this to allow the supplier stub to send messages to other suppliers.

Implementing a set of stubs based on the design, where each augmented reality fidelity level has its own display generator stub was not difficult. However since Aura only offers a static service selection in the configuration we had to write a configuration for every compatible type of input and output stub for each display generation. Each configuration was assigned a weight based on the perceived value of the configuration.

Currently Aura lacks the ability for the user to configure some of the features that we used. In particular the user cannot set up connections between suppliers. We worked around this by adding the connections to underlying configuration files not intended for editing by a user.

6. DISCUSSION

In this section we revisit some of the design decisions that we mentioned in Section 3 in more detail and explain our design choices. We also discuss the available alternatives and introduce some suggested changes to Aura, which are discussed further in Section 7.

We were considering the augmented reality interface to be a single composite service, which is defined in the Aura Software Architecture specification [19]. This however was not actually implemented in the current version of Aura. We worked around this by bringing the three classes of services to the top level. The abstraction of the three classes or services (input, content, and output) worked very well with Aura as it is currently implemented. A different service grouping might have worked better if the composite service architecture was working. A possible configuration with that functionality would be having all the support services for an augmented reality display bundled into one composite service and having the content separate, with hooks for the content to get location information.

A major challenge in the development of our solution was the fact that only a static configuration of service could be selected. The language that Prism uses to request suppliers from the Environment Manager could be extended to allow for a more dynamic supplier selection where some suppliers are chosen based on the other suppliers available. Even a simple language that only allows for "if/ then" clauses would have made service selection a useful tool for configuration. A principle design decision in Aura is the concept of "Quality of Service" parameters for each supplier. In the Aura design two suppliers of the same service that have different quality of service parameters have to be completely interchangeable. What would have been more useful would be to control the selection of suppliers via quality of service parameters or by feature set. For example MS Word is a text editor, yet it has features in addition to text editing. It provides spell checking and formatting as well as text editing. In contrast to this, MS Notepad only has text editing. Despite the obvious differences in the available features of each editor, if they were both registered as providing a text editor service then Aura would consider them the same. Having some type of feature set in addition to the service type and then having the ability to dynamically pick which services or feature sets you want would have made the selection of a display generator much easier.

Another solution that would be useful for this application would be to have a query to ask about the other suppliers that are currently in use. Then the display generator could use the quality of service information from the other suppliers to choose which type of output to use. This would permit (or support) a much more general service that fits many suppliers. The display generation service would then choose what type of display to use based on the quality of service of the other suppliers. This would increase the flexibility of the service by permitting suppliers to be totally interchangeable.

In our implementation we assumed that the data was sufficiently rich to be able to choose different levels of display. A very interesting question along these lines is what is there about a content supplier that could change the type of content it provides based on the type of display that will be in use. Currently this would require a huge number of additional configurations to be generated. There are a number of ways this could be made more convenient. The method of having different display generation services and a service selection language that allows for conditional service selection statements would allow for such a content supplier to also select that based on the display generation service in use. The other solution of the display generation supplier taking in the quality of service terms could be that there is a connection available for a supplier to query the display generation supplier about what style of display will be used.

Possible future work could include replacing some of the stubs in our work with real applications. For the sensor input a solution like the context information service [20] could be used. Replacing the display stub with a real supplier is another option. Looking at this from a human-computer interaction standpoint would also be interesting. Questions to consider are how the user best provides input to the system and at what point is the system more obtrusive than ubiquitous? The challenge of data representation between the components and the content is another interesting problem.

Other future work could include developing a full package of suppliers for different fidelity levels. This could be put together with work on the data representation between the content and the display to provide an augmented reality tool kit for the development of content that would work with an augmented reality interface on Aura. If developing augmented reality applications for Aura were easy, then work could also be done to look into using Aura task management capabilities and context aware functionality to mix tasks, and switch between them. For example a user could be performing one task using an augmented reality display and need to receive some notification about another task. How best Aura could be used to give the user the notification without interrupting his existing task.

7. AURA CHANGES

In my work with Aura I found a number of features that were either unimplemented or underspecified. There were several features that I identified that would allow for development of more powerful and flexible applications. There were some problems with the Prism user interface lacking some of the necessary capabilities for advanced features. Connections between suppliers lacked robustness. The task descriptions language lacked power.

In Prism there was no way to configure a task that needs connections between services. There are a couple ways that this could have been represented in the user interface. A section could be added to the task description where there is an option to add a connection. Then a configuration screen could be displayed for the connection where the user could select the components to connect and the details of the type of connection. Then back on the task description panel the user could select which configurations would use that connection.

The specification for connections between suppliers is robust but the implementation of connections in the existing Aura codebase was incomplete. Currently since there was no user interface for the addition of connections there was difficulty configuring the connections behind the scenes. This could be solved by a more complete implementation of the specification.

Currently there is no specified protocol for inter-supplier communication. I feel that having some specification for the format in which they should send messages between components would encourage development of interoperable suppliers. I also feel that having some set of defined message classes would be of great value. Specifically a standard for requesting details about the other supplier would help the development of suppliers designed to communicate. Depending on what type of suppliers are connected together the suppliers may want to send very different types of information based on what the other suppliers is. Having a standard in place for this either at the supplier level or between the suppliers and the Environment Manager would ease the development of such suppliers.

An extension of the task description language is something that would help development of the type of application of Aura that I desired. As it stands two suppliers of a service must be completely interchangeable though one may be more preferable than the other. An extension of the task description language that allowed specification of configurations based on the supplier’s quality of service would have made development of an application such as mine much easier. Another possible way of implementing this would be to add a new concept. Each supplier of a service would have its quality of service and also a feature set. A feature set would be additional features above and beyond those indicated by the type of service. So Microsoft Word and Notepad could both be services of type “text editor” but Microsoft Word could also have features: formatting, and spell checking. This would allow for some applications to have all services of the same type be interchangeable. This could then be used to implement a task description language that allows logic statements that limit configurations based on the feature set of the suppliers available.

8. CONCLUSION

The expense of developing an application explicitly to use an augmented reality interface means that the applications are very specific and limited. The Aura framework can be used to help, by marshaling the necessary devices to make a best effort attempt at an augmented reality interface. The contributions of this thesis are threefold: We presented a conceptual breakdown of augmented reality into components suitable for use in a diverse ubiquitous computing environment. We showed that Aura is capable of initializing a mix of components into a configuration appropriate for an augmented reality display. We specified capabilities in the Aura system that currently lack the necessary sophistication for development of this style of application.

9. ACKNOWLEDGMENTS

I would like to thank the many people that supported my work on this project. Thanks go to: David Garlan, Vahe Poladian, Bradley Schmerl, and the ABLE Research Group. I would like to give additional thanks to David Garlan as he has been a great source of direction and guidance through the course of the project. I would also like to thank Vahe Poladian for his immeasurable help in patching bugs in Aura, guidance about the inner workings of Aura, and helping determining what parts of the Aura architecture are not yet implemented as laid out in the Aura Software Architecture [19] document. I am grateful for the questions and insight I received at the mid-semester presentations from other Carnegie Mellon School of Computer Science undergraduate senior thesis students and their advisors. I would also like to thank Cyndi Pride for her assistance in proofreading and revision.

WORKS CITED

[1]. *Designing Backpacks for High Fidelity Mobile Outdoor Augmented Reality.* **Piekarski, W., Smith, R. and Thomas, B. H.** Arlington, VA : s.n., October 2004, 3rd Int'l Symposium on Mixed and Augmented Reality.
[2]. *ICrafter: A Service Framework for Ubiquitous Computing Environments.* **Ponnekanti, Shankar R., et al.** Atlanta, Georgia : Springer Berlin/Heidelberg, 2001. Ubicomp 2001: Ubiquitous Computing: Third Internation Confrernce. Vol. 2201/2001, p. 56.
[3]. Project Aura. [Online] http://www.cs.cmu.edu/~aura/.
[4]. **Weiser, Mark.** The Computer for the 21st Century. *Pervasive Computing.* January-March 2002.
[5]. *A Survey of Research on Context-Aware Homes.* **Meyer, Sven and Rakotonirainy, Andry.** [ed.] Chris Johnson, Paul Montague and Chris Steketee. Adelaide, Australia : s.n., 2003. Conferences in Research and Practice in Information Technology. Vol. 21.
[6]. *Bridging Physical and Virtual Worlds with Electronic Tags.* **Want, Roy, et al.** s.l. : ACM Press, 1999. Proceedings of CHI'99.
[7]. *Merging Virtual Objects with the Real World: Seeing Ultrasound imagery within the Patient.* **Bajura, Michael, Fuchs, Henry and Ohbuchi, Ryutarou.** 2, Chicago : s.n., July 1992, Computer Graphics, Vol. 26, pp. 203-210.
[8]. **Azuma, Ronald T.** A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments.* August 1997, Vol. 6, 4, pp. 355-385.
[9]. *A mobile application of augmented reality for aerospace maintenance training.* **Haritos, T. and Macchiarella, N. D.** 2005. The 24th Digital Avionics Systems Conference, 2005. Vol. 1, pp. 5.B.3 - 5.1-9.
[10]. **Azuma, Ronald, et al.** Recent Advances in Augmented Reality. *IEEE Computer Graphics and Applications.* November/December 2001, pp. 34-47.
[11]. *Integrating Virtual and Augmented Realities in an Outdoor Application.* **Piekarski, W., Gunther, B. and Thomas, B.** San Francisco, Ca : s.n., Oct 1999. 2nd Int'l Workshop on Augmented Reality. pp. 45-54.
[12]. *A Wearable Comput System with Augmented Reality to Support Terrestrail Navigation.* **Thomas, Bruce, et al.** Pittsburgh, PA : IEEE Computer Society, 1998. 2nd Internation Symposium on Wearable Computers.
[13]. *A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment.* **Feiner, Steven, et al.** Cambirdge, MA : IEEE, 1997. Proceedings International Symposium on Wearable Computing 1997. pp. 74-81.
[14]. **Fiala, Dr. Mark.** ARTag marker system and SDK. [Online] http://www.artag.net/.
[15]. Augmented Reality Toolkit. [Online] http://artoolkit.sourceforge.net/.
[16]. Mixed Reality Toolkit (MRT). [Online] University College London. http://www.cs.ucl.ac.uk/staff/r.freeman/.
[17]. **Piekarski, Wayne.** Tinmith Augmented Reality Project. [Online] University of South Australia. http://www.tinmith.net/.
[18]. *Project Aura: Toward Distraction-Free Pervasive Computing.* **Garlan, D., Siewiorek, D., Smailagic, A., Steenkiste, P.** April-June 2002, IEEE Pervasive Computing, pp. 22-31.
[19]. **Sousa, J. and Garlan, D.** *The Aura Software Architecture: an Infrastructure for Ubiquitous Computing.* s.l. : Carnegie Mellon Technical Report, CMU-CS-03-183, 2003.
[20]. *An Integrated Contextual Information Service for Pervasive Computing Applications.* **Judd, Glenn and Steenkiste, Peter.** Dallas-Fort Worth, TX : s.n., 2003. First IEEE Internation Conference on Pervasive Computing and Communications.

cpride@andrew.cmu.edu