Moving Beyond Appliance Computing on Embedded Displays

Author: Christopher Ackad, cack7320@mail.usyd.edu.au
Supervisor: Prof Judy Kay
School of Information Technologies

1. Problem Statement

Collaborative tabletops, or Single Display Groupware, enable people to collaborate at an interactive table or wall. To date, such tabletop computing research has been dominated by an appliance model, exploring new interaction approaches within a single application. Our aim is to move beyond this, to provide support for users to *switch between arbitrary applications*, to *configure the environment*, *alter the facilities* available and *switch between file sets*. These capabilities are commonly available on the desktop. However the affordances of the tabletop create different design constraints. So we need to explore new ways to create interface elements that can support this functionality.

2. Our Approach

Our approach was to design and build an application manager for the Cruiser Tabletop framework [3] through an iterative design process. We explored the design space, by creating a series of low fidelity prototypes and evaluated them using discount usability techniques [2]. This led to a design which was implemented as the Cruiser Application Manager (CAM) which enables users to manipulate and configure the environment, either before starting an application, or while it is running.

3. User View

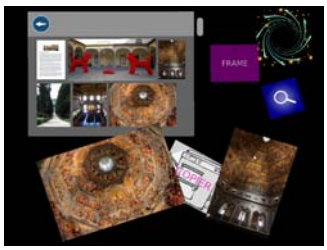
Figure 1a. *CAM minimized.*

Figure 1b. *CAM expanded.*

We illustrate the user view of CAM in the series of "table-shots" [1]. Figure 1a shows the minimized interface object. This is always present on the tabletop. It was designed to be small enough to avoid cluttering the screen but distinctive enough to be found easily when needed. It can be manipulated like a regular interface object in Cruiser. Notably, it can be easily resized and moved to any tabletop location. When the user enlarges it, the icons for available applications become visible, as in Figure 1b, which has three applications.

To start one of these, the user drags out the relevant icon and enlarges it to see the launch options for that application. Figure 2a shows the tabletop after the user pulled the Focus icon to the right and enlarged it.

A little later, the table appears as in Figure 2b. It is running Focus with the Florence file-set and the CAM element is minimized to reduce clutter and to hide the controls when not in use. Once the user manipulates the controls, in this case selecting London, and then touches Update, the table appears as in Figure 2c with the Focus browser updated while the rest of the table is unaffected.



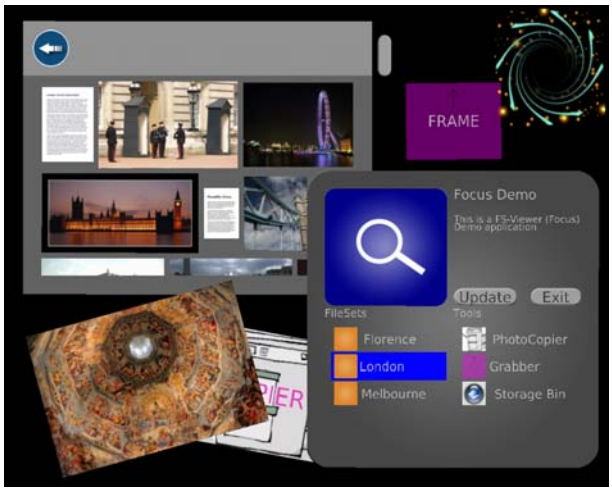


Figure 2a (Top Left). *Starting Focus application with Florence file-set*
Figure 2b (Bottom Left). *Using Focus*
Figure 2c (Right). *Switching file-set to London file-set*

4. The Architecture

The CAM architecture (Figure 3) is designed around the premise that each application is a collection of widgets that interact in a single environment. CAM enables each application developer to configure the environment and define which file sets are manipulated, both prior to loading the application and as it runs. Each application and its options are defined by an XML file, which provides links to configuration files that are loaded when the application starts and instructions that are used when switching file sets on runtime.

When loading an application, CAM first generates a list of configuration files from the selected components of the XML file which then are forwarded to Cruiser as shown in Figure 3. Cruiser applies the configuration while it is loading the application. This level of modularity is backwards compatible with previous Cruiser applications.

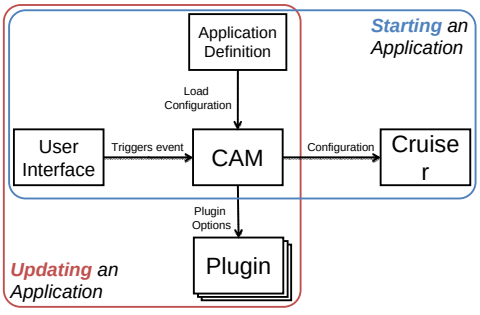


Figure 3. *Overview of how CAM Starts and Updates an Application on the tabletop*

To support switching between file sets within an application, the plugins register themselves with CAM so that when a user selects an option that updates the current environment, CAM calls the registered function with the configurations defined in the XML file.

5. User Evaluation

A Think Aloud user study and questionnaire assessed the learnability and ease of use for core functionality of CAM. This began with a tutorial about Cruiser, without CAM. Then participants were invited to explore CAM, to complete 10 tasks, without assistance. The eight participants from diverse backgrounds quickly discovered how to use CAM, completing all tasks and easily recovered from errors.

6. Contributions

- Designed a new set of interface elements to support application management in the tabletop environment.
- Designed architecture to support CAM functionality for Cruiser applications.
- Implemented CAM architecture that enables users to switch applications and update the environment.
- Reviewed several prototype user interface approaches.
- Conducted a usability study demonstrating the learnability and ease of use of CAM.

7. References

[1] Christopher James Ackad, Anthony Collins, and Judy Kay. Moving beyond the tabletop as an appliance. In Adjunct Proceedings of ITS '09, the ACM International Conference on Interactive Tabletops and Surfaces (to appear), 2009.

[2] Trent Apted, Anthony Collins, and Judy Kay. Heuristics to support design of new software for interaction at tabletops. In CHI '09 Workshop on Multitouch and Surface Computing, 2009.

[3] Trent Apted, Judy Kay, and Aaron Quigley. Tabletop sharing of digital photographs for the elderly. In CHI '06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 781–790, New York, NY, USA, 2006. ACM Press.

The University of Sydney