Moving Beyond Appliance Computing on Tabletop Systems

Christopher Ackad
SID: 306 135 949
Supervisor: Professor Judy Kay

A thesis submitted in partial fulfilment of the requirements for the degree of Bachelor of Information Technologies (Honours)

November 2009
I certify that:

1. I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure;
2. I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to the University commencing proceedings against me for potential student misconduct under Chapter 8 of the University of Sydney By Law 1999 (as amended);
3. This Work is substantially my own, and to the extent that any part of this Work is not my own I have indicated that it is not my own by Acknowledging the Source of that part or those parts of the Work.

Name: Christopher Ackad

Signature: Date: 1 November 2009
Abstract

Collaborative tabletops, Single Display Groupware devices, enable people to collaborate at an interactive table. To date, such tabletop computing research has been dominated by an appliance model, exploring new interaction approaches within a single application.

Tabletops have created the need to explore new ideas for interaction in this design space. Considerable research has explored aspects of this design space ranging from computer human interaction techniques, facilitating collaboration between people on the tabletop to user interface design. Tabletops have the potential to be integrated into a ubiquitous environment to provide a seamless interaction with other technologies. However, little work has been conducted in providing core functionality to the tabletop that is common to desktop computers and other devices.

This thesis explores opportunities to provide a set of core operating systems interface facilities to the tabletop environment: the goal is to move beyond single application appliance computing that has dominated the research area towards a multipurpose device. The 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.
Acknowledgements

I would like to first thank my supervisor, Professor Judy Kay, for her invaluable motivation, insight, guidance, patience and dedication throughout this project.

Significant thanks must be given to Anthony Collins without his hard work and help putting up with my constant barrage of technical questions for the technical implementation and help with in the design process. I also would like to thank Paul Steaza for his help in the design and evaluation process of this project.

I would like to thank the members of the CHAI research group who have given up time to evaluate my low fidelity prototypes and the anonymous testers who have given time to evaluate the implementation.

Finally I would like to thank my parents for their constant support and determination to help me through my honours year.
Contents

Chapter 1 ................................................................. 17
   1.1 Definitions ..................................................... 17
   1.2 Motivation ..................................................... 18
   1.3 Context ......................................................... 19
   1.4 Challenges .................................................... 20
   1.5 Thesis Goals .................................................. 21
   1.6 Thesis Contributions ....................................... 21

Chapter 2 ................................................................. 23
   2.1 Tabletops ....................................................... 23
       2.1.1 Tabletop System Design Considerations ........... 25
   2.2 Operating Systems .......................................... 26
       2.2.1 The Unix Shell .......................................... 26
       2.2.2 The Xerox Star .......................................... 27
       2.2.3 Plan 9 Acme .............................................. 29
       2.2.4 Operating System Summary ......................... 30
   2.3 Tabletop Systems ............................................. 31
       2.3.1 Cruiser .................................................... 31
       2.3.2 Microsoft Surface ....................................... 33
   2.4 Tabletop Primitives .......................................... 33
D.2 Tutorial Sheet ........................................................................................................................................119
D.3 Cruiser Guide Usability Study Main Task ..........................................................................................120
D.4 Post-Experiment Questionnaire .........................................................................................................122
List of Figures

Figure 1.1: A map of the specialisation and user space of every day technology .................. 19
Figure 2.1: The Xerox Star User Interface (Johnson et al., 1989)........................................ 28
Figure 2.2: Icons of files, folders, devices and applications (from left to right) (Smith et al., 1982)
........................................................................................................................................... 29
Figure 2.3: A sample Acme screen displaying its unique command line window management
(Pike, 1994).................................................................................................................................. 30
Figure 2.4: Cruiser running SharePic (Apted et al., 2006)...................................................... 32
Figure 2.5: A comparison of menus on a cluttered tabletop. (Leithinger and Haller, 2007)...... 33
Figure 2.6: The Focus Viewer (Collins and Kay, 2008)......................................................... 34
Figure 3.1: The Chosen Heuristics ........................................................................................... 39
Figure 3.2: Scatter Prototype Start Screen............................................................................... 41
Figure 3.3: Scatter Prototype File-Set Selection ...................................................................... 41
Figure 3.4: An Open Cruiser Application.................................................................................. 41
Figure 3.5: Scatter Prototype in Application File-Set Selection.............................................. 41
Figure 3.6: Scatter Prototype in Application New Application Selection............................... 42
Figure 3.7: Scatter Prototype in Application File-Set Selection for a new Application.............. 42
Figure 3.8: Drawable Menu Prototype Start Screen............................................................... 44
Figure 3.9: Drawable Menu Prototype File-Set Selection........................................................ 44
Figure 3.10: Drawable Menu Prototype in Application File-Set Selection............................... 45
Figure 3.11: Drawable Menu Prototype in Application New Application Selection................. 45
Figure 3.12: Zipper Prototype Start Screen............................................................................. 47
Figure 3.13: Zipper Prototype Application Selection Menu.................................................... 47
Figure 3.14: Zipper Prototype File-Set Selection ..................................................................... 48
Figure 3.15: Zipper Prototype Control Mode .................................................................48
Figure 3.16: Zipper Prototype in Application File-Set Selection ..................................48
Figure 3.17: Zipper Prototype in Application New Application Selection ..................48
Figure 3.18: Panel Box Starts Screen Application Box in icon form ............................51
Figure 3.19: Panel Box Starts Screen Application Box in expanded form .......................51
Figure 3.20: Panel Box Starts Screen, Application Icon ................................................51
Figure 3.21: Panel Box Starts Screen, Application Configuration Panel .........................51
Figure 3.22: Panel Box in an Application in its Icon form ............................................51
Figure 3.23: Panel Box in an Application in its expanded form .....................................51
Figure 4.1: The Focus Browser Widget ............................................................................58
Figure 4.2: The Applications container holding individual Application panels. On the left is the Application container in its icon form. On the right is the expanded Application Container. ...59
Figure 4.3: Application Panel in the Start Screen On the left is the Application Panel in its icon form. On the right is the expanded Application Panel. ..............................................59
Figure 4.4: The CAM Application Panel for Cruiser Demo ..............................................60
Figure 4.5: The Travel Guide Launched Application .......................................................62
Figure 4.6: The Application Panel In Application Mode ..................................................63
Figure 4.7: Changing Focus Fileset on the Travel Guide. ..................................................64
Figure 5.1 An Overview of CAM’s Architecture, only new additions are shown here. ........67
Figure 5.2: Shows the process of loading configuration files to start an application. ...........68
Figure 5.3: Overview of the XML Application Definition file .........................................69
List of Tables

Table 2.1: A list of Interface Characteristics ..........................................................24
Table 2.2: Comparison of the user representation of key operating system services.......30
Table 3.1: Flaws found from the Cognitive Walkthrough of the Scatter Prototype........42
Table 3.2: Heuristic results for the Scatter Prototype .............................................43
Table 3.3: Flaws found from the Cognitive Walkthrough of the Drawable Menu........45
Table 3.4: Heuristic results for the Drawable Menu ................................................46
Table 3.5: Flaws found from the Cognitive Walkthrough of the Zipper .....................49
Table 3.6: Heuristic results for the Zipper prototype .............................................49
Table 3.7: Flaws found from the Cognitive Walkthrough of the Panel Box ...............52
Table 3.8: Heuristic results for the Panel Box .......................................................53
Table 5.1: List of XML Elements used in the Application Definition File ..................70
Table 6.1: Error Recovery Solutions in CAM .......................................................75
Table 6.2: Summary of Tasks vs Function .............................................................77
Table 6.3: Participants Demographics .................................................................82
Table 6.4: Observation of Task Execution for Scenario 1 .......................................84
Table 6.5: Observation of Task Execution for Scenario 2 .......................................85
Table 6.6: Quantitative results from Questioners ..................................................88
Chapter 1

Introduction

This thesis explores opportunities to provide a set of core operating systems interface facilities to the tabletop environment: the goal is to move beyond single application appliance computing that has dominated the research area towards a multipurpose device. The aim is to move towards the generality offered by general purpose computing via a suitable Operating System interface that enables a person to use many different applications in a flexible way.

1.1 Definitions

The following is a brief description of some terms used in this thesis that the reader may be unfamiliar with.

Ubiquitous Computing is a term that describes an environment where computer technology is seamlessly integrated into everyday life, to the point that it is invisible to us.

Single Display Groupware describes devices where multiple users simultaneously and collaboratively interact with a single display (Stewart et al., 1999).

Surface Computing is a general research area focusing on computer human interaction facilitated by an interactive surface.

Tabletops are a specialisation of surface computing focusing on the computer human interactions through a horizontal surface. Users can interact with the surface through touching the digital display to manipulate physical objects on the display.
**Appliance Computing** is a term used to describe computing devices that specialise in a single function.

**Cruiser** is an application framework designed for the rapid and flexible development of immersive tabletop applications built in C++ using OpenGL. It is a multi-user, multi-touch collaborative tabletop environment developed by Trent Apted (Apted, 2008).

### 1.2 Motivation

Tabletops, unlike conventional desktop computers and other everyday devices such as multifunctional gaming consoles and touch-based mobile phones, provide unique forms of interaction with computer environments. Collaborative tabletops are commonly classified as Single Display Groupware devices, where multiple users collaborate simultaneously with a single display mechanism. Tabletops introduce a unique design space for interaction by providing a *horizontal* display and replacing the traditional input devices such as the keyboard and mouse with forms of *gestural input*, to interact directly with the display or through an augmented environment.

Tabletops have created the need to explore new ideas for interaction in this design space. Considerable research has explored aspects of this design space ranging from computer-human interaction techniques, facilitating collaboration between people on the tabletop to user interface design. Tabletops have the potential to be integrated into a ubiquitous environment to provide a seamless interaction with other technologies. However, little work has been conducted in providing core functionality to the tabletop that is common to desktop computers and other devices.

Currently Tabletop software is immature in comparison to its desktop counterpart. Most of the work available on tabletops has been restricted to research organisations. The way applications are presented on tabletops varies depending on the environment where the table is situated, hardware capabilities and supported interaction modes (Wallace and Scott, 2008). There is a variety of ways that applications can be displayed on a tabletop surface. These include a widget structure, a form of modular design where small application primitives (components) are combined to produce an application with a set functionality, to the standard application interface that is all too familiar from the desktop running in a standalone environment. Given the immaturity of the field of tabletop design, and the diversity of applications for tabletops, little research has been conducted in *application management* on a surface environment. Due to the limited input on a tabletop, something like a Unix shell becomes infeasible. Indeed, the tabletop's limited input provides unique design challenges that need to be addressed.
1.3 Context

This thesis explores application management on table tops, reviewing their context in a pervasive environment. There has been little work in exploring aspects of activity and application management on tabletop technology. Most of the literature focuses on elements of interface design, interaction paradigms, facilitating social interaction and environment specialisation, all implemented in the context of a single activity, leaving an important area of tabletop and ubiquitous computing research untouched.

Figure 1.1 illustrates the place of tabletops against other major computing devices currently available. The horizontal axis shows the degree of specialisation these devices exhibit. It shows that tabletops have a role that is quite specialised, though somewhat less specialised than a typical information kiosk such as those found at tourist attractions. They are also less specialised than emerging appliance computers such as Keep in Touch (Langdale et al., 2006) a special purpose digital appliance that allows family members with special needs from young children to grandparents to keep in touch. Smart phones and other mobile platforms have become more general purpose, for example the Apple iPhone (Apple, 2009) provides users with access to a variety of applications and functionality, such as interactive map applications and web browser but these capabilities are limited by the device size and context of use. Desktop computers can be seen as a multipurpose digital device providing users with a large variety of functionality through the use of applications, each within their own environment accessed through the computer’s windows manager.

![Figure 1.1: A map of the specialisation and user space of every day technology](image.png)
1.4 Challenges

The vertical axis describes the user space from the multiuser public spaces to the restricted single user personal spaces that these devices are used in. Smart phones and mobile devices are inherently personal, as each mobile phone provides a direct link to its owner. Desktop computers are generally single user devices and provide each user with their own personal and private space to work in. However digital appliances, interactive kiosks and tabletops are intended for use in public interfaces and by multiple users.

Tabletop systems are a multiuser collaborative platform, and to date have been specialised to a single activity. Examples include: stylised interactive information terminals (Microsoft, 2008); museum exhibits (Geller, 2006); collaborative work spaces for information sharing (Apted et al., 2006, Collins and Kay, 2008). When compared to common day desktop computers, tabletops provide very little in terms of flexibility and generality, but offer an easy to use collaborative platform.

Tabletop research and development has been dominated by an appliance model. There has been little research into support for core facilities for general computing: starting an application, changing the files used and altering application settings. Providing these capabilities will expand its scope and provide users with a multifunction environment on the tabletop with the functionality and freedom that is present in desktop computers, moving away from the appliance model.

1.4 Challenges

Tabletop systems provide a number of unique challenges from interaction techniques to facilitating social activities. Interactions with tabletops are limited by the absence of the keyboard and mouse, relying primarily on gestural input. Types of input vary from device to device; they include touch, stylus and augmented object recognition as well as a combination of these input types. The way we interact with tabletops moves away from the conventional desktop metaphor to a more natural and physical design space.

In developing user interfaces for tabletop environments, a number of design constraints need consideration. These design constraints include facilitating collaborative interaction, supporting multiple table sizes, user’s location relative to the table, using table space affectively and reducing clutter (Apted et al., 2009). Tabletops are becoming more focused on collaborative interaction and multiuser environments. Table sizes vary depending on the application, introducing human reach issues where items on larger displays are not reachable by people. Often there is more than one user on a single tabletop. They are not always looking at the table from the same orientation, therefore interface design should take into account the users’ location and have orientation independence. Another challenge in developing in a
Chapter 1. Introduction

1.5 Thesis Goals

The primary goal of this project is to explore ways of moving beyond appliance computing on tabletop systems. This consists of:

- Exploring interfaces that provide application management capabilities for tabletop systems incorporating:
  - A flexible way of starting and switching between applications;
  - The ability to customise the user environment;
  - The use of different file sets within a running application.
- To create a software architectural framework which can be incorporated into the chosen software framework, Cruiser

1.6 Thesis Contributions

This thesis explores the interface and implementation for application management on tabletop systems moving away from table tops as appliance devices. The contributions are:

1. Exploration of several approaches to designing the interface for application management, based on prototyping and discount usability evaluations; (Chapter 3)
2. Design of a new tabletop user interface that allows users to load and switch between individual and sets of plug-ins (application) in Cruiser; (Chapter 4)
3. An architecture that supports this functionality to operate within Cruiser by adding support to allow Cruiser to load and unload plug-ins on the fly; (Chapter 5)
4. Evaluation of the user interface with user study; (Chapter 6)
5. Demonstration of the architecture interims of its power, flexibility and effectiveness. (Chapter 6)
Chapter 2

Background

This thesis aims to explore a new way to provide a selection of the general facilities offered by operating systems to a tabletop environment. This chapter provides an overview of the previous work on existing operating system user interfaces, tabletop systems, with a particular focus on tabletop design considerations, previous work on tabletop primitives and other existing systems.

2.1 Tabletops

Tabletops are a new and emerging technology, which provide a different perspective on the way we interact with computer systems. Tabletops are considered a form of Single Display Groupware (SDG) devices (Stewart et al., 1999) where multiple users interact collaboratively sharing a single display. The first Tabletop system was conceived by William Newman and Pierre Wellner (Newman and Wellner, 1992) in an attempt to extend the desktop metaphor from a visual representation on a computer display (Johnson et al., 1989) to an interactive digital surface. This provides a unique design space that moves away from conventional computing use to a more ubiquitous environment.

Generally desktop computers are seen as a single user environment rather than a collaborative medium. Desktops do not provide the same capabilities as a SDG device because desktops are designed in the context of a single user per display and often personal. An overview of the main interface characteristics from desktop computing and counterparts in tabletop computing are shown in Table 2.1.
Tabletops, single display groupware devices as shown in Table 2.1 are multi user by nature and usually designed to promote collaboration over a single device, compared to the single user analogy of desktop computing. They are generally used in public spaces to promote collaboration and social activities. Being horizontal displays and interaction devices, tabletops tend to have a series of unique design constraints. Tabletops are orientation independent of where multiple people stand around a table at any given time. This alludes to a series of design problems such as text orientation effect on readability of the user interfaces (Wigdor and Balakrishnan, 2005). Table size can vary depending on its implementation; this introduces human reach limitation where on very large tables reach is limited to the length of the user’s arms. This is not a problem on desktop interfaces where reach is limited to the physical placement of the mouse and its vertical display provides a single orientation. Tabletops tend not to use conventional input technologies such as the keyboard or mouse, rather relying on physical interactions to enact a given function. Gestural interactions are used to represent the way people interact with physical tabletops (Terrenghi et al., 2007) to mirror the physical environment. Tabletops often suffer from file clutter where large number of files are displayed on screen. There are a number of solutions for handling file clutter on tabletops ranging from tangible draws (Hartmann et al., 2006) to special objects like the Black Hole (Apted et al., 2006) to hide files and reduce clutter, where as in desktops a hierarchical file system using folders provides a way to organise and hide files.

These characteristics present a unique set of design challenges, varying from text input (Hirche et al., 2008), object display (orientation, table clutter, human reach) to user identification (Apted et al., 2009). These design challenges are non-existent in the desktop environment. A large number of tabletop hardware exists with varying and unique input
modalities such as touch and multi-touch, stylus to vision based systems that additionally support object recognition in tangible interfaces.

User interfaces presented on tabletops are often simple and designed so that a user can use it with no prior knowledge or experience on the system. Individual files on tabletops are often represented pictorially, where the amount of detail shown is in relation to the objects size (Apted et al., 2006). One would interact and enlarge the object to provide more information and shrink to reduce. Compared to desktops, file representation and organisation on a tabletop environment is still being explored. This has resulted in a variety of ways in which files are accessed on a tabletop displays from hierarchical file access common to desktops to associative file access (Collins, 2006).

Application management on tabletop systems is still a relatively unexplored aspect of tabletop computing compared to desktop computing. Currently the Microsoft Surface (Microsoft, 2009), the first generic tabletop consumer device provides a simple solution that only allows the user to launch and switch between applications¹. When comparing this to window and application managers in desktop systems (Gnome, MS Windows, Mac OSX) the system presented on the Surface provides very limited functionality. It does not support manipulating file sets outside an application, nor loading an application with a specific configuration. All of these are available in desktop computers and in some consumer devices.

### 2.1.1 Tabletop System Design Considerations

The design space provided by tabletops provides unique challenges in building applications and their user interfaces. Some of the key characteristics that are particularly important for to the tabletop environment include; collaborative interaction, gestural input and orientation independence (Apted et al., 2009). Therefore work has been conducted on defining guidelines and design consideration for the tabletop environment.

Research conducted by Stacey D. Scott (Scott et al., 2003) provides an important base for designing user interfaces for tabletop environments. They presented a series of guidelines that tabletops must support to provide a seamless and usable interface. These guidelines include supporting: interpersonal interaction, fluid transitions between activities, transitions between personal and group work, transition between tabletop and external work, the use of physical objects, shared access to physical and digital objects, and consider the appropriate arrangements of users and simultaneous user actions.

¹ Microsoft does not mention or talk about their application manager interface for surface. However this behaviour has been observed from demonstration videos and personal experience with the surface emulator.
The other aspect of design should consider the usage context. Work by Wallace and Scott (Wallace and Scott, 2008) presents 5 key contextual areas that need to be considered in designing tabletop hardware and applications for a specific environment. These include:

- Social/Cultural: social and cultural norms that govern social behaviour in an environment;
- Activity: the types of tasks or activities the group is engaged in;
- Temporal: how often and long the table will be used (including time pressures);
- Ecological: the environment the table is situated in, including objects and devices nearby;
- Motivational: the personal and professional needs that motivate the activities that the table is used for.

These guidelines are structured to provide tabletop and application designers with a basis of understanding the tabletop design space and its intended use.

### 2.2 Operating Systems

This thesis explores ways of incorporating application management facilities provided by an operating system into a tabletop environment. The key facilities that are available to the user by an operating system include program execution and management, computer communications, file system manipulation, I/O operations and system status (Elmasri et al., 2009, Silberschatz et al., 2004). A number of ways are used to present these facilities to the user from the command line to the windowed environment in desktop computers.

The previous work conducted on operating system user interface design gives insight on how people view an operating system and how to provide a familiar environment to the potential users. This section reviews the user interfaces available in desktop operating systems by reviewing three influential operating systems interfaces; the pure Unix Shell command line interface, the Xerox Star “desktop metaphor” Graphical User Interface (GUI) and the Hybrid windowed command line interface Acme from Plan9.

#### 2.2.1 The Unix Shell

The Unix shell is one of the oldest operating system interface that is still in wide use, with the earliest version dating back to 1969 (Ritchie and Thompson, 1974). It is a command line interface and viewed as a “tools metaphor” in contrast with the “desktop metaphor” (Johnson et al., 1989). The tools metaphor captures the notion that a user first invokes an application and appends a file to manipulate, using the application as a tool to achieve the end goal. One of the unique features of the Unix Operating System is that many aspects are treated as files, from the system applications down to the I/O interfaces, giving a large amount of
freedom and consistency when interacting with operating system components. One of the features of the Unix Shell that is rarely seen in graphical user interfaces is its ability to use filters and pipes. This is where the standard output of one command is passed into the standard input of another command in a sequence, allowing users to chain smaller commands together to complete a job, thus providing the user with great flexibility and control of the environment.

2.2.2 The Xerox Star

The desktop metaphor originated from the “Xerox Star Office Automation System” designed in the late 1970s (Johnson et al., 1989). The Xerox Star was designed to provide the functionality common to operating systems of the time for people who were not technically savvy enough to use computers running a command line interface such as the Unix Shell in an office environment. It was released on the market in the early 80s introducing new concepts and infrastructure into the office environment. The Xerox Star employed bitmapped point and click display and exemplary design to introduce the concept of desktop computing (Figure 2.1). By using the office desktop as an analogy of user interaction, it helped users to easily and quickly pick up and use computers without needing an extensive technical background. As shown in Figure 2.1 it introduced icons to represent file systems and applications, arranging them on the desktop for easy access. Users used the mouse to manipulate and interact with the icons on screen such a clicking on them to open the file, folder or run a program. It also introduced the first freeform windows manager to arrange applications on the screen allowing users to manipulate the windows, like resizing, moving and overlapping the windows.
Figure 2.1: The Xerox Star User Interface (Johnson et al., 1989)

The Xerox Star implemented a graphical user interface that metaphorically represented an office desktop. The system used an iconic file management and direct manipulation to allow users to interact with the applications and files directly (Smith et al., 1987). It implemented generic commands such as copy, paste and delete which are common functionalities for interacting with the file system and within applications providing continuity between operations. The Xerox Star user interface provided a high level of consistency across applications from user interaction to the application on screen representation. From these generic commands there were very few modes where common user interaction would have a different meaning. The Xerox Star employed techniques such as the notion of the desktop metaphor which only required the user to open the file to edit a document instead of invoking a text editor with a file as an argument. The Xerox Star allowed users to interact with applications, files and devices directly though its icon representation. Figure 2.2 shows a sample of icons in the Xerox Star; (from left to right) files, folders, devices and applications. File types were associated with an application. When the user selected a file (the icon to left) and pressed an open button on the keyboard, its relevant application loaded with the file opened. Folders behaved as applications where the user selected a folder and opened a browser showing the files within. Users could interact with devices by simply copying the file onto the device icon (ie printing), selecting the file, pressing the copy button on the keyboard, then selecting the device and pressing paste (Smith et al., 1982). Programs could be opened using the same...
method as with files. Each icon shown was distinctively different representing the functionality it provides.

![Icons of files, folders, devices and applications](image)

Figure 2.2: Icons of files, folders, devices and applications (from left to right) (Smith et al., 1982)

The Xerox Star was not a commercial success but has revolutionised how we design user interfaces for modern day computing (Blackwell, 2006). Operating systems from Mac OS to Windows to the various flavours of Linux use the core ideas and designs introduced by the Xerox Star as the foundation in designing their operating system user interfaces. This makes it very important for this thesis both in terms of the demonstrated power of the user interface concepts and because it is very familiar to most potential tabletop users.

### 2.2.3 Plan 9 Acme

The Acme interface is a graphical shell and text editor for the Plan 9 operating system (Pike, 1994). It moved away from the conventional graphical and command line interfaces. Building from the Unix operating system, it presents a fundamentally different user interface from that of the Unix Shell or the Xerox Star, by providing a windowed command line interface. Acme is designed to optimise the use of the mouse as well as to remove menus by adding the ability for users to write commands and execute them from a click of the button on the fly. This is done by treating every textual element on the screen in such a way that it can be selected and executed. When a command is executed in an active window, the window manager creates and tiles a new window with the results displayed. Figure 2.3 displays a typical Acme session, on the right is a mail program, directory listing some open files and a window displaying compiler errors. To the left three source files are opened including the dat.h, acme.l and time.l as well as a window displaying a stack trace. By right clicking on the line in the stack trace window, Time.l was opened with the offending line highlighted. After clicking on the compiler message in the left column, the cursor moves over the offending line in acme.l. The Acme interface provides a unique perspective to command line interface design, the idea of using any onscreen text as an executable command and its window management presents a dynamic and customisable environment.
2.2.4 Operating System Summary

These three operating system environments the Unix Shell, Acme 9 and the Xerox Star present quite different views of the ways users might interact with computer systems. The two extremes presented here are the command line and the windowed graphical user interface from the Unix Shell to Xerox Star respectively. Each of these has varied levels of power and control traded against ease of learning.

The differences and similarities of each of these operating systems are shown in Table 2.2. The Unix Shell uses a command line interface with a minimalistic user interface providing power and flexibility to the user. The Acme Plan 9 utilises a hybrid windowed command line interface to extend the command line to the mouse to provide an efficient and streamlined environment. While Xerox Star used a graphical user interface to display elements on screen to improve the ease of use and learnability of the interface. To execute a programme; in the Unix Shell user types in a command with arguments and presses enter, in the Acme interface they type their command and click on it to open a new window running the command, and in the Xerox Star the user would click on the program or file icons and press the open button on the keyboard. To manipulate files in both the Unix Shell and the Acme user interface, the users execute a series of standard commands including cd (change directory), ls (list directory), mv (move), cp (copy) etc.

```
#endif

Figure 2.3: A sample Acme screen displaying its unique command line window management (Pike, 1994)
```
and special generic keyboard buttons such as open, copy, paste etc. In all three operating systems, devices are shown in a similar manner as files. The Unix Shell and Acme interact with these like any other file or program while the Xerox Star presented the users with uniquely identifiable icons for which they used generic commands like copy and paste to interact with devices such as printers.

### Table 2.2: Comparison of the user representation of key operating system services

<table>
<thead>
<tr>
<th>Operating System Services</th>
<th>Unix Shell</th>
<th>Acme Plan 9</th>
<th>Xerox Star</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface</td>
<td>Command Line</td>
<td>Hybrid Windowed Command Line</td>
<td>Windowed Graphical User Interface</td>
</tr>
<tr>
<td>Program Execution</td>
<td>Typed Command with arguments</td>
<td>Typed and Text Selection</td>
<td>Icon Selection</td>
</tr>
<tr>
<td>File System Manipulation</td>
<td>Typed Command: cd, ls, mv, cp etc.</td>
<td>Selected Command: cd, ls, mv, cp etc</td>
<td>Windowed Browser click on listing Icons</td>
</tr>
<tr>
<td>I/O Representation</td>
<td>As Files</td>
<td>As Files</td>
<td>Iconic</td>
</tr>
</tbody>
</table>

The Unix Shell requires a steep learning curve but provides the most control to the user, displaying the core textual information needed without any stylisation. The Plan 9 Acme interface provides the power of the command line and the multitasking capabilities of a windowed environment. Its design provides a flexible environment that is geared towards efficiency, at some cost in learnability and ease of use. The Xerox Star interface demonstrated a fluent and easy to use interface, utilising visual elements that metaphorically represent the office desktop environment. It traded the user’s speed and efficiency for learnability and ease of use. This has resulted in the Xerox Star user interface being the bases of all the mainstream graphical user interfaces used in modern day desktops.

## 2.3 Tabletop Systems

Tabletop system environments vary based on the hardware capabilities and software design of these systems. Good examples of these are the Cruiser Framework (Apted, 2008) and the Microsoft Surface (Microsoft, 2009).

### 2.3.1 Cruiser

Cruiser is an extensible framework for the rapid and flexible development of immersive tabletop applications built in C++ using OpenGL. It is a multi-user, multi-touch collaborative tabletop environment developed by Trent Apted (Apted, 2008), initially designed as a framework for the collaborative photo sharing application SharePic (Apted et al., 2006).
2.3 Tabletop Systems

Cruiser's primarily input is the use of physical gestures facilitated through a touch or stylus interface. Cruiser supports user identification as well as simultaneous input, provided that the tabletop hardware supports it. Cruiser also supports a large number of devices, from input devices to displays and all the mainstream operating systems (Linux, Windows, Mac OSX). Early work with Cruiser was conducted using a Diamond Touch display utilising multi-touch and allowing for user identification, later implementing a Mimeo stylus system that uses digital pens to identify users and their input.

Cruiser implements a widget interface where each element is an independent object (Figure 2.4). Cruiser consists of seven core tabletop interaction primitives, which are the ability to select, move, rosize (rotate, resize), copy, group, delete and capture. With these interaction primitives users are able to manipulate items on screen as well as capture an image of the current state.

![Figure 2.4: Cruiser running SharePic (Apted et al., 2006)](image)

**Framework**

Cruiser utilises a plugin framework for the development of applications. This architecture primarily consists of four layers:

- **The Core**: This is the core environment providing tabletop interaction functionality;
- **The Plugin Libraries**: These provide the application components;
- **The Plugins**: These provide the application logic including the front end viewable to the users;
- **The Utilities Libraries**: These provide the Core and Plugins access to utilities outside of core such as thread management.

This plugin architecture allows researchers and developers to build libraries and plugins for Cruiser without modifying the core architecture and provides a large amount of flexibility by
allowing Cruiser to run on multiple platforms. This plugin architecture also permits 3rd party members to add capabilities and support for newer hardware.

### 2.3.2 Microsoft Surface

The Microsoft Surface is the first commercial tabletop device released in 2007 by Microsoft (Perenson, 2007). It utilises vision-based motion detection and capture technology allowing the detection of multi-touch and object recognition enabling the use of tangible objects to work with Surface. It is a complete self-contained system, reducing its flexibility but providing full support of the hardware capabilities. It is a 30-inch rear projection display contained within a table 22 inches high, 21 inches deep and 42 inches wide. It has a maximum resolution of 1024 by 768 and support up to 52 simultaneous points of touch (Microsoft, 2009).

Surface applications are built using either the XNA framework or Windows Presentation Foundation (WPF). Each application is given an independent environment to run in. The Surface provides the ability to move between applications using The Guide, accessible through the corners of the screen. However a limitation to this interface is that the user cannot choose a specific configuration nor move items from one application into another.

### 2.4 Tabletop Primitives

One of the user interface primitives that is not often seen on tabletop systems is the ability to call up a secondary functionality in the form of menus. Traditional menus like text on tabletops often suffer from orientation problems as it is very difficult to determine a user’s location to orientate the text properly. It is difficult to interact with menus on a cluttered table. Work by Leithinger and Haller (Leithinger and Haller, 2007) introduces the idea of user drawn menus as a way of displaying menus on the tabletop. Their work compared a traditional dropdown menu and a pie menu in a cluttered tabletop environment then presented a user drawn menu as a solution to this (Figure 2.5). However, they did not address the issue of tabletop orientation but presented a novel way for users to control the display of object’s properties and secondary functions.

![Figure 2.5: A comparison of menus on a cluttered tabletop. (Leithinger and Haller, 2007)](image)

To the left is a traditional drop down menu, centre a pie menu and to the right a user drawn menu.
2.5 File System Access

One of the core components of any operating system is file manipulation. Presenting a file system on tabletop environments effectively is a challenge. Due to the restrictions of its design space very little can be used from a desktop file system in a tabletop environment. Recent work has presented a series of solutions on how file systems should be displayed to the user, from Tangible Draws (Hartmann et al., 2006) to Bubble Clustering (Watanabe et al., 2007) to Associative file systems (Collins et al., 2007).

Focus, an associative file system browser for the Cruiser tabletop (Collins and Kay, 2008) functions by presenting the user with files; when a user dwells on a one of these files, associated files are displayed on screen and irrelevant files are removed. It uses a context based search to find files associated to the focus file as a way of browsing a file system. Files that are more relevant appear larger than those that are less relevant, addressing issues of clutter management and screen space as shown in Figure 2.6. It includes a history bar so that users can back track to previous Focus Searches as well as restart the search, by selecting the blue recycle icon. Focus also has no files stored on the tabletop but remotely accesses multiple file systems simultaneously to retrieve files from a query. This satisfies the need to easily navigate and access files on the tabletop and other file systems seamlessly.

Figure 2.6: The Focus Viewer (Collins and Kay, 2008)
2.6 Summary

This chapter provides the background research for the thesis, across multiple fields. Very little research has been conducted in investigating ways of providing application management functionality to the tabletop environment, specifically the ability to seamlessly switch between tasks. Literature on operating system design and development is essential in grasping the design space presented to the operating system interface designers and to gain an understanding of the design concepts and constraints of the tabletop environment. The research outlined in this chapter provides a foundation for developing an application manager as well as describes the key design considerations of the tabletop environment.
Chapter 3

Interface Design Overview

This chapter describes the prototyping approach taken to design the user interface for an application manager on the tabletop environment. The only other such interface that provides some limited level of application management on a tabletop is the Microsoft Surface (Microsoft, 2009) and it only provides limited capabilities; it can only switch between arbitrary applications. So it goes just one step beyond the appliance model. The goal of this project is to expand beyond this to provide the capabilities to configure an application prior to launch and while it is running, alter the facilities available and switch between filesets. Our approach is based upon a combination of parallel and iterative prototyping using discount usability methods to evaluate for usability and to inform refinement to the design.

3.1 Design

Low fidelity prototyping was used to explore the design space early in the design cycle. This allowed for the creation of multiple low fidelity prototypes each exploring different aspects of the tabletop design space with modest effort for the benefit of useful feedback to guide the design process.

3.2 Evaluation Methodology

The discount usability methods chosen to evaluate the developed prototypes were a Heuristics Evaluation and Cognitive Walkthrough (Nielsen, 1992). Heuristics Evaluations were chosen because they provide good feedback on the designs based on established and validated
design principals and Cognitive Walkthroughs focus on the learnability aspects of the design and identify learnability or usability flaws.

### 3.2.1 Heuristics

When conducting the evaluations of the low fidelity CAM prototypes, a set of evaluation heuristics is needed. Figure 3.1 list the combination of Nielsen Ten (Nielsen, 1994), Large Display Information Exhibits (Somervell et al., 2003) and Tabletop Heuristics (Apted et al., 2009) that were chosen to evaluate these designs. The Nielsen Ten heuristics were chosen because they describe the general usability concepts that are applicable to many user interface designs including those for desktop computers as well as other computing device user interfaces. Importantly, they have been experimentally evaluated and are widely accepted.

The Large Display Information Exhibits heuristics give general design guidelines for the limited input and interaction types used in designing interfaces for large interactive displays. Although tabletops are often small table displays, these heuristics have the potential to be valuable when designing user interfaces for systems that have different interaction paradigms similar to those available at a tabletop and other devices which have input restricted to touch and stylus. We studied them and identified four that directly apply to tabletop:

- **Appropriate colour schemes can be used for supporting information understanding**: is applicable to any nonconventional display especially one with unique interaction and representation paradigms.

- **Layout should reflect the information according to its intended use**: looks at information displayed on large displays with limited input. This is especially applicable when designing a tabletop interface that needs to display a large amount of information without overwhelming the user.

- **Judicious use of animation is necessary for effective design**: with tabletops and other systems, animation can guide user’s actions and highlight areas of importance.

- **Show the presence of information, but not the details**: states that the information displayed to the user must not go into great detailed as this is better suited to desktops. As tabletops also have restricted resolution and clutter as well as limited forms of interaction, this approach is also important to limit the amount of information presented.
The Tabletop Software heuristics were chosen since they directly apply to designing and building applications on tabletops. As they also help the designers and evaluators consider common problems that are associated with tabletop user interfaces. These heuristics were designed for systems that are similar in nature to Cruiser and focus on the key usability problems on tabletop environments. The detailed list of heuristics can be found in Appendix A.

3.2.2 Evaluators

Five evaluators from the field of computer human interaction with experience with tabletop or touch systems where chosen to conduct the Heuristic Evaluation and four for the Cognitive Walkthrough. These experts vary in experience with the field of tabletop computing and the use of discount evaluation methodologies.

Each evaluator was presented with a 30 minute overview of the low fidelity prototypes and evaluation tasks. At the end of the overview they were given a list of the heuristics, the cognitive walkthrough material and copies of the low fidelity prototypes to evaluate in their own time. The evaluators were requested to assess the results from the heuristics evaluation using a 0 to 4 severity rating, where 0 is no problem and 4 is catastrophic. Based on the feedback from the evaluators one prototype was chosen, refined and implemented (See Chapter 4 and 5). A sample result of both the heuristics and cognitive walkthroughs can be found in Appendix B and Appendix C.

3.3 Prototype Overview

A combination of parallel and iterative prototyping methodologies was chosen to effectively explore the design space. This resulted in 4 distinct design streams: Scatter,
3.3 Prototype Overview

.Drawable Menu, Zipper and Panel Box prototypes. For each of the design streams a PowerPoint prototype was built which explored 3 scenarios: starting an application on table, switching applications and changing filesets of an active application. The following sections will present the design rational, overview and results of the discount usability evaluations for each prototype.

### 3.3.1 Scatter Prototype

#### Design Rational

The Scatter prototype was one of the first low fidelity prototype designs explored. This prototype explores an interface approach where each application is represented by an individual icon. The drivers for this design are the goals of simplicity as well as matching widely the known tabletop interaction conventions. It also draws on conventions from Cruiser including dwell and scatter. This design was themed from the early work on Focus and Cruiser where items are scattered around the screen. The dwell action from the navigation was based on the paradigm used by Focus, whereby to conduct a search a user would dwell on an image of interest. The reason for scattering images on the screen randomly is that the user’s orientation is not known, each item is provided with a random orientation which the user can pick up and reorientate for better viewing.

#### Overview

Figure 3.2 shows the start screen presented to the user. Each application is represented by an icon (since this was a low fidelity prototype the icons are all green boxes), with its name rendered underneath the icon. The icons are scattered randomly on the table, similar to the effect of throwing a set of pictures onto a physical table.

The next step is to select a file set; this is done by dwelling on the application icon to reveal the file set selection screen. Figure 3.3 show the fileset screen where the user dwelled on “Application 2” and now 3 file sets are available. These filesets are represented by an icon as with the applications in the start screen and their names are rendered on the bottom of the icon. To launch the application with a file set, the user dwells on the chosen file set icon. If there are no file sets available to the user, dwelling on the application in the start screen starts the application (Figure 3.4). To progress back to the start screen (Figure 3.2) from the fileset selection screen (Figure 3.3) a user dwells on the Application icon, in Figure 3.3 this would be “Application 2”. 

Figure 3.2: Scatter Prototype Start Screen

Figure 3.3: Scatter Prototype File-Set Selection

Figure 3.4 shows a loaded application in Cruiser, an wormhole is displayed near the centre of the screen. This wormhole can be moved and manipulated like any other object. When a user wishes to change the fileset, they first dwell on the wormhole to switch into control mode. The screen dims and the application object becomes unavailable. The icon of the current application appears attached to the wormhole which acts as an anchor in both modes and the filesets are scatter around the table as shown in Figure 3.5. To change a fileset the user dwells on the fileset icon to update the current environment. To return to application the user dwells on the wormhole object.

Figure 3.4: An Open Cruiser Application

Figure 3.5: Scatter Prototype in Application
File-Set Selection

When a user wishes to switch to another application from control mode (Figure 3.5) the user dwells on the current application icon (the green box) to move to the application selection screen as shown in Figure 3.6. From here the users dwell on the new application to bring them to Figure 3.7 from there they dwell on the fileset to launch a new application.
Cognitive Walkthrough

The results of the Cognitive walkthrough as shown in Table 3.1 revealed that this design has learnability flaws. The flaws identified by all evaluators is that dwelling on the current application to move back to the application selection screen is not obvious and confusing as users will not understand the task. The second flaw identified is that users can get lost with the wormhole analogy when trying to switch to control mode as this is not obvious and users may confuse the wormhole for the black hole. The concept of dwelling may not be familiar to users. They may instinctively tap instead of dwell on an object.

Table 3.1: Flaws found from the Cognitive Walkthrough of the Scatter Prototype

<table>
<thead>
<tr>
<th>Flaw Identified</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dwelling on current application to go back to application menu is not clear.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dwelling on wormhole to switch to ‘control mode’ is not obvious.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dwelling on an application icon to select application is not immediately obvious, tapping (clicking) is more natural.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dwelling on a fileset icon to select application is not immediately obvious, tapping (clicking) is more natural.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Key: ✓ show evaluators that identify the flaw

Heuristic Evaluation

The Heuristics evaluations found as shown in Table 3.2, that the scatter prototype contain a number of design flaws. Three out of the five evaluators viewed the lack of a clear exit as major usability flaw scoring a rating of 3 across all evaluators. Clutter also was perceived as a serious design flaw where 3 of the evaluators have it the same rating of 3. Reachability was also identified by three of the evaluators but this time there was a wide distribution in severity ratings. A lack of documentation was identified as a flaw in the design with three evaluators giving it a rating of 3 and 2. No indication of currently selected file set and remembering to dwell on wormhole to switch applications were identified by two of the five evaluators with a severity rating of 2 and 3. The reaming flaws in the design were identified by only one evaluator although scoring a severity rating between 2 and 3.
3.3 Prototype Overview

### Table 3.2: Heuristic results for the Scatter Prototype

<table>
<thead>
<tr>
<th>Flaw Identified</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>No real exit or back button defined</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clutter is a big issue with this interface</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reachability can be a problem</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No help, instructions or documentation given to user</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No indication of currently selected file set</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User must remember to dwell on wormhole to switch applications</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orientation problem</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No loading indicator</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application icons in different locations each time screen shown - confusing to user</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User must remember to dwell on application to view list of other applications</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wormhole could be blocked</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>No container symbology</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Interface has no shortcuts for power users</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

#### 3.3.2 Drawable Menu

**Design Rational**

The Drawable Menu prototype was inspired by the work of Leithinger and Haller (Leithinger and Haller, 2007) involving user-drawn path menus. This prototype works on the premise that when the user draws a curved arc, causing menu items to appear on the curve orientated by the focal point of that arc. This prototype is separated into two modes: the Application and Control. A Control Object is used in this prototype to prevent stealing the gesture of drawing the curve from the application that is running. Since drawing an arc is a simple gesture, placing the Drawable Menu component in the application space would prevent the application from using this gesture and any derived gestures; this would be a serious limitation so it were avoided.

**Overview**

The user starts with blank screen, here they draw an arc. The icon of applications appears on the drawn arc orientated towards the focal point of the arc Figure 3.8. Each application is represented by an icon and has its name rendered below it. The arch itself is visible to the user and behaves like a scroll bar where the scroll button (the blue dot) can be moved side to side revealing more icons. This allows for more than 3 applications to be accessed at the same time keeping the number of visible applications to a minimum to avoid
unnecessary crowding on the menu. This design assumes that the user would draw the arc orientated towards their location and the menu would be orientated within an acceptable limit.

The next step is to select a fileset; this is done by dwelling on the selected application icon to move to the fileset menu. Figure 3.9 shows the fileset menu; here the selected application moves down below the curve, in this case “Application 1”. The remaining application icons fade away, replaced by the fileset icons on the drawn path. To launch the application with a file set the user dwells on the chosen file set icon. If there are no file sets available to the user, dwelling on the application icon in the application menu starts the application. To move back to the application selection menu the user dwells on the application icon in this case “Application 1”.

While an application is running a wormhole object is displayed on screen (refer to Figure 3.4). The wormhole object can be moved around and manipulated as with any other object. When users wish to change file sets, they first dwell on the wormhole to switch into control mode, then draw an arc. This results in a menu appearing containing the filesets of current application as shown in Figure 3.10. To switch filesets the user only dwells on the new fileset to update the current application. When a user wishes to switch to another application from control mode (Figure 3.10) the user dwells on the current application icon (underneath the menu) to open the application selection menu as shown in Figure 3.11. Here they dwell on an application to view its filesets, then dwell on the selected fileset to launch the application with that fileset.
Cognitive Walkthrough

The Cognitive walkthrough, as shown in Table 3.3, revealed that this design has some learnability flaws. The first flaw identified is that the user has to draw a menu to be able to launch an application. This action is not very obvious and can result in first time users getting lost. Users can also get lost with the wormhole analogy when trying to switch to Control Mode. Finally it is not obvious how to exit or switch between applications. All four evaluators were unanimous in identifying these flaws.

<table>
<thead>
<tr>
<th>Flaw Identified</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drawing a path to access the menu is not obvious</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dwelling on wormhole to switch to ‘control mode’ is not obvious.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>It is not initially obvious how to go back to the application menu, dwelling on the current application is not clear.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Key: ✓ show evaluators that identify the flaw
Heuristic Evaluation

As shown in Table 3.4 the Heuristics evaluation revealed that the drawable menu prototype contained a number of design flaws. Three out of the 5 evaluators viewed that the lack of a clear exit and a lack of documentation as flaws in the design with the three evaluators giving it a rating of 3’s and 2’s. Three evaluators indicated that no loading indicator or feedback is a flaw with severity ratings of 3 and 2. One evaluator noted a critical flaw where it is not clear how to close the menu. The remaining flaws in the design were identified by only one evaluator although scoring severity ratings of 2 and 3’s.

Table 3.4: Heuristic results for the Drawable Menu  

<table>
<thead>
<tr>
<th>Flaw Identified</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>No real exit or back button defined</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No help or instructions given to user to draw the menu</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No loading indicator / feedback</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interface has no shortcuts for power users</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not clear how to close menu after it is drawn</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not clear that (dot) is slideable</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The path drawn to restrictive</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Wormhole may not be reachable by all at the tabletop</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The control mode might not be necessary</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User must remember that to view the list of other applications they need to dwell on the application icon</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>The path - drawing method not intuitive</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Unclear if information in menu can be scaled easily</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Menu cannot be re-orientated once it open</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interface is geared towards a single user</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
3.3.3 The Zipper

Design Rational

The Zipper concept explores utilising the control object more effectively than in the previous prototypes by centralising the menu around it. This prototype works with the user dragging out a zipper from the control object to reveal a menu.

Overview

Figure 3.12 shows the start screen presented the user with a wormhole object and a animated blip. The user pulls down on the blip like a zipper to reveal a menu as shown in Figure 3.13. This design assumes that the user is dragging the zipper towards them as such it orientates the text vertically relative to the zipper’s position to the wormhole object. Also a scroll bar appears on the left hand side to denote the number of items in this list.

![Figure 3.12: Zipper Prototype Start Screen](image)

![Figure 3.13: Zipper Prototype Application Selection Menu](image)

The next step is to select a fileset, this is done by dwelling on the application icon to move to the fileset menu as shown in Figure 3.14. The selected Application icon is moved to the right of the menu, in this case “Application 1” the remaining application icons fade away, replaced by the fileset icons on the menu. To launch the application with a file set the user dwells on the chosen file set icon. If there are no file sets available to the user dwelling on the application icon at the start screen launches the application. To move back to the application selection menu the user dwells on the application icon in this case “Application 1”.

While an application is running a wormhole object is displayed on screen (refer to Figure 3.4). The wormhole object can be moved around and manipulated as with any other object. When a user wishes to change file sets they first dwell on the wormhole to switch into control mode, here the wormhole centres itself and reveals it zipper (blue dot) as shown in Figure 3.15. Figure 3.16 the user drags out the zipper to show the fileset menu. To switch filesets the user dwells on the new fileset which updates the environment and exits out the
control mode moving the wormhole object back to its position prior to switching to control mode.

When a user wishes to switch to another application from control mode (Figure 3.16) the user dwells on the current application icon (on the right of the wormhole) to open the application selection menu as shown in Figure 3.17. Here they dwell on an application to view its filesets, then dwell on the selected fileset to launch the application with that fileset.

**Cognitive Walkthrough**

The Cognitive walkthrough as shown in Table 3.5 revealed that even though users are attracted to the animated blip, they would not know what to do with it. Users could understand the wormhole analogy a bit better when trying to switch to control mode since they may acquire a sense of familiarity from the start sequence but still it is not obvious to them how to exit or switch between applications. All four evaluators were unanimous in identifying these flaws.
Table 3.5: Flaws found from the Cognitive Walkthrough of the Zipper

<table>
<thead>
<tr>
<th>Flaw Identified</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>There is no indication that the zipper is a menu, not clear how to use zipper.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dwelling on wormhole to switch to ‘control mode’ is not obvious.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>It is not initially obvious how to go back to the application menu, dwelling on the current application is not clear.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Key:** ✓ show evaluators that identify the flaw

**Heuristic Evaluation**

The Heuristics evaluation revealed as shown in Table 3.6 that elements of the user interface in this prototype had some flaws. Four out of the five evaluators found Reachability as a serious flaw with rating ranging from 2 to 4. Also the same evaluators viewed that this design is geared towards a single user with severity rating from 2 to 4. A lack of documentation was identified as a flaw in the design with three evaluators giving it a rating of 3 and 2. Three out of the five evaluators have stated that no indication of currently selected file set is a design flaw with severity ratings of 2 to 4. No real exit or back button defined was also identified by three of the evaluators as a design flaw with severity ratings of 3, 2 and 1. Wormhole and Black Hole to closely related, the scroll bar not intuitive, dwelling on application icon to view list of other applications not intuitive and dwelling on wormhole to switch applications not intuitive were identified as design flaws by two out of five evaluators with severity ratings of 2 to 4 and a median rating of 3. The remaining flaws in the design were identified by only one evaluator although scoring severity ratings of 2 and 3’s.

Table 3.6: Heuristic results for the Zipper prototype

<table>
<thead>
<tr>
<th>Flaw Identified</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reachability can be a problem</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>This interface is geared towards a single user</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>No help, instructions or documentation given to user</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No indication of currently selected file set</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No real exit or back button defined</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wormhole and Black Hole to closely related</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The scroll bar is not intuitive</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dwelling on application icon to view list of other applications is not intuitive</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dwelling on wormhole to switch applications is not intuitive</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No loading indicator / feedback</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zipper itself (and application icons) do not appear to be resizable</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zipper does not look like a zipper</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not clear what to do with zipper</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Menu cannot be reorientated once opened</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.3.4 Panel Box

Design Rational

After reviewing the 3 previous prototype designs, it became apparent that the limitations to the interface designs reduced their ability to provide advance functionality to the user. The Panel Box interface embodies elements common to desktop computing such as buttons and selections into a user interface for the tabletop. It works on the premise that the panel is an object on screen until it reaches a set size. When resized it reveals a configuration panel from which the user selects options and uses buttons that are familiar to them to configure and launch the application.

Overview

Figure 3.18 shows the start screen presented to the user. Here there is a single distinct icon presented to the user. When the user resizes this it reveals a container containing icons representing installed applications as shown in Figure 3.19. The user then drags out the icon (Figure 3.20) and resizes it to reveal the Application Panel (Figure 3.21). Here the user taps on each of the available options to configure the application, such as selecting filesets, after the user dwells on the “Go” button to start the application. This loads the application with the Application Panel associated to that application loaded in its icon form as shown in Figure 3.22.

At any time within the application the user can enlarge the Application Panel to update the environment like changing filesets as shown in Figure 3.23. When a user changes the configuration they can update the environment by dwelling on the “Go” button in the panel. To switch to another application, the user dwells on the “Exit” button in the panel to quit the application and go back to the start screen. Here they follow the same steps as in starting an application.
Chapter 3. Interface Design Overview

3.3 Prototype Overview

Figure 3.18: Panel Box Starts Screen Application Box in icon form

Figure 3.19: Panel Box Starts Screen Application Box in expanded form

Figure 3.20: Panel Box Starts Screen, Application Icon

Figure 3.21: Panel Box Starts Screen, Application Configuration Panel

Figure 3.22: Panel Box in an Application in its Icon form

Figure 3.23: Panel Box in an Application in its expanded form
Cognitive Walkthrough

The Cognitive walkthrough revealed, as shown in Table 3.7, that the application box is familiar to the user however lacks indicators to resize the container. Resizing the container and dragging out the application icon is not apparent, users will tend to tap or dwell on these items. There are no defaults, so it is unclear if the user will need to select the fileset and options before starting the application. Also users may confuse pulling out the application icon as starting the application as well as resizing the menu box is not apparent. From the table it is apparent that there is a wide distribution in the perception of faults in this prototype.

Table 3.7: Flaws found from the Cognitive Walkthrough of the Panel Box

<table>
<thead>
<tr>
<th>Flaw Identified</th>
<th>Examiner</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resizing application icon to reveal list of filesets is not obvious.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>It is unclear whether the user has to select a fileset and/or configuration before pressing the ‘Go’ button.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Dragging application out of box may be thought of as launching the application.</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Resizing menu box to reveal list of applications is not obvious.</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Key: ✓ show evaluators that identify the flaw

Heuristic Evaluation

The results of Heuristics Evaluation for this prototype (summarised in Table 3.8) shows Three out of the five evaluator identified reachability as a design flaw with a rating of 4, 1 and 1. The word "exit" needing careful consideration was identified as a design flaw by two of the five evaluators with a severity rating of 3 to 4. Interface is geared towards a single user and interface has no shortcuts for power users were also identified as flaws with a severity rating of 2 to 1. All other flaws were only identified by one evaluator with severity ratings of 4 to 1.
Table 3.8: Heuristic results for the Panel Box

<table>
<thead>
<tr>
<th>Flaw Identified</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reachability can be a problem</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The word &quot;exit&quot; needs careful consideration. Are we switching applications or exiting them? Likewise &quot;go&quot;.</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Interface is geared towards a single user</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interface has no shortcuts for power users</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>The menu inside the application icon should be proportionally shown, otherwise the resizing action may be not clear.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No loading indicator / feedback</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No indication of currently selected file set</td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scroll bar might not be intuitive</td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No help, instructions or documentation given to user</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support both drag and touch for selection</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clutter - When user has expanded the Application icon, the application box is left littering the table</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>There is quite a lot of information on screen at once.</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unclear what is optional to select on screen</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If there is no other action the user can do, the application icon should be in large-enough scale that would not require the user to resize.</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Confirmation before exit useful</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>It is not obvious to user’s that they should shrink the application icon to cancel their changes</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Might not need so much detail (text) in the application information section</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.3.5 Conclusion

Review of the evaluators’ comments on the four prototypes revealed that each prototype has some flaws. When looking at aspects of Learnability and Usability, the Panel Box Prototype appeared to be the easiest to use with a simple and familiar user interface. Although it also has its drawbacks such as reachability on large displays, the Panel Box prototype provides flexibility and power to the user without any confusing and lengthy navigation structures. The Cognitive Walkthrough (Table 3.5) and Heuristic Evaluations (Table 3.6) revealed that the Panel Box prototype had less severe flaws and the perception of the flaws was diversely distributed between the evaluators.

The Panel Box prototype was found to provide a much more streamlined user interface and improved usability. This design was refined with the feedback from the evaluation and was implemented in Cruiser to form the core user interface for CAM outlined in Chapter 4.

The Drawable Menu prototype showed promise but this menu design cannot be used by itself in all contexts and the concept of the Control Mode increases confusion to the user. However the Drawable Menu could be combined with other interface types applicable to scenarios where the hiding the application icon is important.
Chapter 4

User View

This chapter describes CAM Cruiser Application Manager for the Cruiser Tabletop Framework. This is a tabletop application manager user interface allowing users to configure, load and switch between applications on the tabletop. CAM also provides users the ability to specify what files to load before an application started and allows switching filesets during the runtime of an application. It also behaves as a toolbox for users, allowing them to load tabletop primitives such as a storage bin to store files from the current session on the tabletop in one self-contained user interface element. A detailed outline is provided of the user view of CAM, how the interface is presented to the user and how users typically interact with it.

4.1 Main Goals

The main goal of CAM is to provide tabletop users with some of the core application management capabilities commonly found on desktop computers user interfaces. Our aim is to provide the following capabilities within the tabletop environment, without the need to exit the framework. These include:

- Loading and unloading applications,
- Loading an application with a specific configuration,
- Specifying filesets prior to load and during the usage of an application, and
- Providing access to tabletop tools (such as a copier object, storage bin, etc.)
4.2 Cruiser

CAM is designed and built as an extension to the Cruiser Tabletop Framework. Cruiser is a multiuser, multi-touch, gesture-based tabletop framework to support rapid and flexible development of immersive tabletop applications (Apted, 2008). It was initially designed as a collaborative photo sharing application, SharePic, and later evolved into a framework for developing tabletop applications. This system is designed to support and promote collaborative social interactions around a table. It is designed to support a variety of hardware configurations and currently is implemented using a MIMIO Stylus pens to track user movements. A number of gesture types have been utilised in CAM which will be discussed in later sections of this chapter, but first we need to provide an overview of Cruiser.

4.2.1 Interaction

Cruiser was design to facilitate interaction with a variety of hardware setups ranging from touch to stylus. Currently users can interact with Cruiser through the use of digital pens utilising the Mimio capture hardware (Ink, 2009). Cruiser has a multiuser architecture that supports user identification through the use of different pens. A series of primitive interaction gestures are available on the tabletop; these range from taping, moving, resize (rotate/resize), dwelling, flipping and flicking. Tapping is where a user taps on an object. This is commonly used to select it. Moving occurs when a user holds and drags an object to another location. Each object is given momentum to provide a sense of realism, the behaviour of which is similar to that seen on an Air Hockey Table. The rotate and resize gesture named resize is a combined gesture that rotates and resizes the object around its centre; this is triggered by selecting and dragging the corners of an object round its centre. Dwelling is where a user selects and holds down on an object. This is often used to trigger an event such as search on a file in Focus. Flipping an object in Cruiser is done by selecting the edge of an object and dragging it over its centre. This often is used to look at the metadata of a file such as filenames of images or to turn pages in a multipage PDF document in Focus. The flick gesture is when a user adds momentum to an object; it is used to pass objects to other users on the table.

At the moment the capture hardware is limited to tracking a single pen at any one time preventing the full utilisation of the simultaneous multiuser capabilities of Cruiser.
Chapter 4. User View

4.2.2 Presentation

Cruiser is designed around a widget paradigm, where applications are considered as a combination of these widgets that function together as a single unified application. Cruiser consist of a series of primitives and smart objects. The primitives in Cruiser include files such as documents, images and videos. Smart objects such as the Black Hole, Copier, Storage Bins, Frame and Remote Frame Buffer provide additional functionality and different behaviours at the tabletop. The Black Hole behaves as a recycle bin when items are thrown into it. The Storage Bin provides a location where objects can be placed and stored to reduce clutter. Copier is an object that allows users to make multiple copies of any object swiped over the copier; copies are then dragged out from the Copier. The Frame allows the user to create a screen shot of the section of the screen below the frame. The Remote Frame Buffer provides a remote connection to a desktop computer on the tabletop.

4.3 Focus

Focus is Cruiser’s associative file system browser. It provides remote file system access on to the tabletop. This was described under previous work outlined in Section 2.5 File System Access. To demonstrate the fileset manipulation capabilities of CAM we used a new version of the Focus Bowser (Figure 4.1) that is held in a container, aimed at reducing clutter. Objects in the widget can be dragged out from the container, which prevents them from being removed from screen when a new Focus search is conducted. These files can be searched by dwelling on them either in the widget or on the table. A thumbnail of each Focus search item appears in the search bar to the top of the widget. Users can go back to the start page by dwelling on the home button (the blue arrow). When the Focus widget is resized below a certain threshold it hides the images making it easier for users to manipulate the widget compensating for the Fat Finger syndrome associated with touch displays.
File system access is a core functionality of desktop computers and operating systems alike. Focus is important since it provides these abilities to view remote file systems effectively on the tabletop environment.

### 4.4 Cruiser Application Manager - CAM

CAM utilises many of the gestures derived from Cruiser in its interface design. This is important to provide consistency between both the tabletop paradigms and ones from desktops; it utilises all of the gestures described in section 4.2 for aspect of its user interface. CAM also introduces some behaviours, in order to overcome usability problems identified in the design process, which we describe in the next chapter. Observations of users indicate they have an instinctive response to objects on screen: they tap or dwell on them rather than dragging and resizing. The new behaviour introduced in CAM is the dwell-to-resize behaviour: when a user dwells on an object, it enlarges. In the case of CAM, when a user dwells on it, it enlarges to reveal its configuration panel.

#### 4.4.1 Walkthrough

This section provides a walkthrough of CAM for some basic scenarios, each demonstrating one of its core design goals as stated in the Introduction. There are two distinct modes for CAM, which we refer to as Launch Mode and In Application Mode.

**Start Screen**

The Launch Mode describes the screens that are presented to the user when they first turn on the table, as shown in the table-shot at the left of Figure 4.2.
Here the user is presented with an icon labelled Applications. The user can manipulate this in the same way as typical objects on Cruiser: it can be moved, resized but not flipped. When the user resizes this icon beyond a certain threshold, it reveals the available collection of installed applications, as shown at the right of Figure 4.2 which has the applications, Cruiser Demo, Museum Guide and Travel Guide. The user can also dwell on the icon to resize it to the state showed at the right of Figure 4.2 without the need to resize the object. The dwell-to-resize behaviour was created to overcome problems observed and feedback from the design evaluations where concern was raised that users will not instinctively resize an object on screen but rather dwell on it.

Each icon in the Application Container (in Figure 4.2) is a Launch Application Panel (Launcher). When either a user dwells on one or drags it out and resizes this icon beyond a set size it displays the Launcher providing access to the filesets and controls (Figure 4.3).
This panel is separated into four zones, the information display (Figure 4.4, A-C), control section (Figure 4.4, D), the fileset panel (Figure 4.4, E-F) and options panel (Figure 4.4, G-H).

A) The Application Icon (for visual consistency with the panel in icon mode)
B) The Application Title (for ease of identification)
C) The Application Description
   (additional information that can be used for documentation)
D) Start Button (on dwell launches the application)
E) FileSets Selection Panel (selection of filesets with their icons)
F) Fileset predefined icon (for ease of identification)
G) A Selected Option (coloured in blue to easily highlight)
H) A Unselected Option

Figure 4.4: The CAM Launcher Panel for Trip Planner

The information section consists of three parts the Application Icon (A), Application Title (B) and Application Description (C). The application icon (A) is for consistency and to reduce the cognitive load by promoting recognition over recall. This complies with Nielsen design Heuristic of recognition rather than recall. This is to reassure users that this is the panel of the icon they just enlarged. To the left, the application title provides the user with textual information that is relevant to this application and the control buttons provided to launch the application. The textual information displayed gives the name of the application (B) and a description area (C) that allows the application designer to place a small amount of textual information that may help or provide documentation to the users.

Underneath this is the Start button (D), unlike other elements on the panel that react to taps the Start button reacts on dwell. This is to minimise the errors from inadvertent taps on the
Start button when a user is attempting to move the panel. This complies with the Nielsen’s heuristic Error Prevention.

Below these are the fileset selections (E, F) and option selections (G, H). Unselected options appear with a grey background (H), a selected option appears with a blue background indicating that it has been selected (G). To select a specific option the user taps on the deselected options. The reason that options respond to taps and buttons respond to dwell is to minimise user errors when manipulating the panel and provide a level of constancy with Focus. It is easier to recover from an accidental selection of options rather than accidentally launching the wrong application. These selections and buttons have a large selection zone so that they overcome the fat finger syndrome that is associated with touch based interfaces. The Application designer can specify icons for the filesets and options as shown in (F) to aid in recognition of visual element on table. In this particular application the user can only select one fileset as this application supports single fileset operations.

However, CAM can easily be configured to allow multiple filesets for applications that supports this, but the user can select more than one configuration option.

The application designer in setting up this application defines the defaults. So that when the user first opens a panel during a session the application defaults are already preselected to allow the quick launch of the application. To launch an application the user dwells on the start button which will blank the screen indicating that the application is loading.

In Application

Once an application has launched, CAM runs in Application mode. In this mode, only the application panel of the currently running application is available. When the application starts as shown in Figure 4.5, in this case the Travel Guide sample application, the panel appears in its icon form. This is the plane icon to the right of the screen. This is mixed in with the objects from the application, including the Focus viewer on the top left with a couple images taken out of it. To the top left there is the Black Hole used to remove objects from the table and the tabletop primitive Storage Bin next to the Black hole. The user can dwell on or rosize the application icon to show its panel.
In application mode the application panel is presented slightly differently from the Launcher configuration panel (Figure 4.6 A-F). The Start button is replaced by an Update (or Restart for legacy applications) and Exit button (D). The Filesets remain and Options are replaced by Tools as selections in the panel. The option selection from the Launcher is no longer available. This design choice was made for two reasons: 1) any modification to the options would require restarting the application; 2) the tools panel is more appropriate in the application panel to provide access tools in a centralised location removing the need for toolboxes. This adds an additional step in changing an application configuration but allows room for the tools. The Tools Section (F) enables the user to manipulate the application by adding primitives to the table such as a copier and storage bin. To add an object to the table, the user only taps on the tool option for Cruiser to add the object to the table at that instant.
CAM provides the user with the capability to update the fileset of the currently running application. This is done by changing the selection on the Launcher and dwelling on Update button to update the fileset.

For example in Figure 4.7 we illustrate how a user can update the Focus widget in the Travel Guide application. The user first opens the launcher by rosizing the plane icon from Figure 4.5 revealing the panel in Figure 4.7-A. They then select a new file set, in this case Florence and dwell on Update (Figure 4.7-B). The Focus widget animates indicating that its update function was called by CAM (top left of Figure 4.7-B). Finally the user is presented with the new fileset, Florence in the Focus widget (Figure 4.7-C).
For legacy applications that do not support the runtime update feature of CAM instead of displaying a Update button CAM displays a Restart button. The buttons on the panel behave in the same way as the Start button in the Start screen, where the user has to dwell on the button to prevent unnecessary restarts and exits when trying to move the panel.
Chapter 5

Implementation

This chapter describes the architecture and technical implementation of Cruiser’s Application Manager (CAM). It outlines the architecture that supports:

- application switching,
- plugin update infrastructure,
- the ability to add tools, tabletop primitives onto the table, and
- the user view of CAM.

5.1 High Level Architecture

CAM is an extension to the Cruiser Tabletop Framework to support rudimentary application management and configuration through an on table user interface. It extends Cruiser’s core architecture and also implements most of its functionality as plugins within Cruiser’s framework. As shown in Figure 5.1 the CAM framework architecture has four modules; the Cruiser Core; Plugin Update; Tools and CAM’s plugin. The CAM framework is designed to be modular, allowing developers to replace individual CAM modules, and still be able to utilise the remaining functionality of CAM and still work effectively within the framework.
5.2 Cruiser

Cruiser is a cross platform tabletop framework, written in C++ and utilising OpenGL (Apted, 2008). It is designed primarily around a plugin architecture where third party members can add additional functionality through the use of plugins. This architecture has four main layers:

- **The Core**: This is the core environment providing tabletop interaction functionality;
- **The Plugin Libraries**: These provide the application components;
- **The Plugins**: These provide the application logic including the front end viewable to the users;
- **The Utilities Libraries**: These provide the Core and Plugins access to utilities outside of core such as thread management.

The Cruiser framework was implemented to run in a single environment, with only one application. Prior to CAM the only way to access multiple Cruiser applications was to use multiple instances of the framework. To switch between applications the user had to exit the framework entirely and then restart the new application.

5.3 Technical Overview

CAM required modifications to Cruiser’s Core as well as creation of a series of plugins and plugin libraries. Figure 5.1 shows an overview of CAM’s architecture where only the new additions to Cruiser are shown. Each of these elements is described below.

5.3.1 Cruiser Core

The Cruiser Core modifications are the only part of CAM architecture that lie outside of the plugin space (green box in Figure 5.1). The Cruiser Core was modified to support multiple instances of the environment (the current screen) as a foundation for application switching. One environment can exist at a time. Each environment is loaded with a plugin configuration that defines the components and plugins to run, and their setup. The Core was also modified to allow multiple configurations to be appended to the previous configurations before the environment is loaded while holding information on the current application. When CAM loads an environment, it utilises these function calls from the Cruiser Core to determine what applications to load and to allow switching between applications.
In Cruiser, an application is defined as a configuration of the installed plugins. Each plugin has a series of variables that are defined by a configuration file. Prior to CAM, Cruiser would load a single default configuration file. This file contained settings for the framework such as graphics and sound that are specific to the install to the configuration for the individual plugins that makes up the application. This set up is fine for single applications but when introducing multiple configurations for applications and components it becomes difficult to manage and hinders portability of the application.

The Cruiser Core was modified so that it can chain load multiple configuration files allowing application designers to create and change configuration files on the fly, without the need to modify the default file. This allows application designer to produce portable configurations that do not need any editing to run on the client machine. This changes the way Cruiser loads the configurations for applications. For backwards compatibility with previous framework installations, the Cruiser Core can fall back to loading a single default file if no additional files are specified, either in start-up or in the framework.
When a plugin triggers a restart event, it embeds a list of configuration files for the Cruiser Configuration Loader in the restart event. At this point, the event handler triggers a restart in Cruiser's Main module and specifies the files to load. As illustrated in Figure 5.2, Cruiser's Main module first loads the default file, then chain loads the specified configuration files before starting a new environment, in this case the Application configuration (App) and selected File and Options configurations defined in the Application Definition file read by CAM. This process avoids the unnecessary reloading of the plugins in the framework as the plugins still remain in memory but are removed from the environment. This also allows for persistence between applications since only the objects created by the plugins are cleared from the environment, not the plugin itself, allowing plugin designers to create objects that remain persistent between application launches.

5.3.2 CAM

CAM (blue box in Figure 5.1) enables application designers to describe their application on the Application Panel, to set options and filesets for the application link to the Cruiser Configuration files for that application. CAM connects all the components of the framework including Cruiser Core, Plugin Update and Tools to provide the user with the capabilities to switch applications, update filesets and to add tabletop primitives to the environment. It functions in two modes: the Launcher and In Application. This mode is determined by the current application information held by the environment that is running the current application.

Application Definition XML File

CAM (light blue box in Figure 5.1) defines an application through an XML Application Definition file. This file defines the application appearance in CAM, the options that are available to the user, the configurations for Plugin Update and the Cruiser Configuration Files for loading the application.

Figure 5.3 shows that the XML Application Definition file is separated into three main sections; the Application, Fileset and Options definitions. Table 5.1 lists the XML specifications used by the Application Definition file in CAM. The application definition provides the name,
icon and the application description that is displayed on the Application Panel. This also links to the application’s Cruiser Configuration file that is used by the plugin loader.

Figure 5.3: Overview of the XML Application Definition file

The fileset grouping, contains part of the definitions for each fileset option available to the user. The definition comprises of its display name, button icon, a link to a Cruiser Configuration file as well as plugin update definitions. The plugin update definition has a list of pairs of variable names and values. These values are sent to the plugin to reload when a user updates the fileset through CAM. An example of a XML Application Definition file can be located in Appendix D.
### Table 5.1: List of XML Elements used in the Application Definition File

<table>
<thead>
<tr>
<th>XML Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
<td>Tag encapsulating the file, defining the start and end of the application definition</td>
</tr>
<tr>
<td>AppName</td>
<td>Name of the Application and the text that appears in the title bar</td>
</tr>
<tr>
<td>AppDescription</td>
<td>The text that appears in the descriptions box in the configuration panel in both modes</td>
</tr>
<tr>
<td>AppConfig</td>
<td>Cruiser configuration file to load when the application starts</td>
</tr>
<tr>
<td>AppConfig</td>
<td>Cruiser configuration file to load when the application starts</td>
</tr>
<tr>
<td>Config</td>
<td>Application icon for both modes</td>
</tr>
<tr>
<td>Configs</td>
<td>Starts the options grouping for the option panel</td>
</tr>
<tr>
<td>FileSets</td>
<td>Within the Configs group, defines a single option that appears on the panel</td>
</tr>
<tr>
<td>FileSet</td>
<td>Starts the fileset section for the files panel</td>
</tr>
<tr>
<td>Name</td>
<td>Within the FileSets group, the start of a single fileset option that is rendered on screen</td>
</tr>
<tr>
<td>Icon</td>
<td>In both the Fileset and Config groupings, the name of the specific option that appears on screen</td>
</tr>
<tr>
<td>Icon</td>
<td>In both the Fileset and Config groupings, the icon that gets renders in for the specific option</td>
</tr>
<tr>
<td>File</td>
<td>In both the Fileset and Config groupings, a link to the Cruiser Configuration file that is loaded when the option is selected</td>
</tr>
<tr>
<td>Selected</td>
<td>A Boolean in both the Fileset and Config groupings that defines whether or not that this option is selected by default.</td>
</tr>
<tr>
<td>Plugin</td>
<td>In FileSet only, states the plugin call-back used for changing filesets</td>
</tr>
<tr>
<td>PluginOptions</td>
<td>In FileSet only, defines the argument pair used in PluginUpdate</td>
</tr>
<tr>
<td>Variable</td>
<td>In PluginOptions, defines the variable used in the variable pair used in PluginUpdate</td>
</tr>
<tr>
<td>Value</td>
<td>In PluginOptions, defines the value used in the variable pair used in PluginUpdate</td>
</tr>
</tbody>
</table>

### Operation

CAM has two modes: In Application and Launcher. CAM presents itself and functions differently within these two modes.

In Launcher Mode; CAM displays an Application Container with a collection of Application Panels (see Chapter 4 Start Screen). To generate these panels, CAM transverses the application directory reading each application file. When a user dwells on the start button, CAM checks the selected options and generates a list of configuration files which are then embedded into the restart event as well as setting the application identifier in the environment (see Figure 5.2).
In Application Mode; CAM loads the current application XML file and retrieves the list of available tools from the tools registers. When a user updates a fileset selection, CAM generates a variable name and value pair and identifies the plugin to call from the XML definition file. It calls the update plugin module with the plugin name and the list of arguments.

When the user dwells on the exit button, CAM sends a restart event with no arguments, indicating to switch back to the start screen.

5.3.3 Tools

Each plugin primitive (orange box in Figure 5.1), such as the Copier, Storage Bin or Frame, are tools in the environment for users to add. CAM allows this by adding a tools section in its application mode Application Panel (see Chapter 4 In Application). This was created from the toolbox plugin built by Anthony Collins. This collects tabletop tools, embedded within a browser container, and users drag out copies of these primitives onto the table.

In CAM tools are added to the panel by registering a factory method with a constructor of the tool and some metadata, such as the name of the tool and its icon. This is then added to the list of tools which CAM displays on the Application Panel. When a user selects a tool, CAM calls the tool’s factory method which, in turn creates a new tool object on screen.

5.3.4 Plugin Update

The Plugin Update component (red box in Figure 5.1) is designed to allow CAM to communicate with new and existing plugins to update their filesets. This plugin update framework also allows other plugins to communicate with each other without needing to know specific function calls.

A plugin registers itself to this service by first registering an update call back to the plugin register. This call back consists of the name for the call back and a pointer to an update method that takes in a list of string pairs. These are a pair of variable name and their values (from the XML file). This design ensures that plugins can get both data and instructions sent to their update function. The values of the string pairs are determined by the plugin designer. In the update method the plugin designer must be able to parse the information passed through the list and take the correct actions based on it.

The Plugin register adds the plugin call back with its name to a list plugins in the Plugin Update module. When an external plugin needs to update another plugin i.e. changing filesets, it calls the update plugin function in the update plugin module, with the name of the plugin call back and the list of variable pairs. The Plugin Update module queries the list of registered plugins for the call back, then calls the update function with the variable list as the arguments.
Chapter 6

Evaluation

The goal of this thesis is to provide a way to move beyond appliance computing on tabletop systems. The Cruiser Application Manager (CAM) was designed to provide the user with a flexible way of switching applications, the ability to customise the environment and the ability to use different file sets within an application. This chapter describes the design of a qualitative user study to assess the usability of the features of CAM and the outcomes of this study.

6.1 Evaluation Goals

The primary goal of this evaluation is to assess the usability of CAM. Discount usability evaluations are useful at an early stage of design, assessing low and high fidelity prototypes (Nielsen, 2000) during the development cycle. However, a user study is needed to assess the learnability and usability of the features of CAM for people who are unfamiliar with CAM.

A qualitative user study was designed and conducted to assess the CAM interface, given that the users have some familiarity using the tabletop system. This user study was designed to assess how quickly users can learn and use CAM with no prior exposure to CAM and no explicit tuition about it. The experiment was designed to evaluate the hypotheses:

H1. Learnability, CAM is easy to learn;
H2. Ease of Use, CAM is easy to use;
H3. Error Recovery, it is easy to recover from any errors in using CAM.
H1 Learnability

Learnability was a key design goal for this project. We aimed to create an interface that users could explore to discover the functionality without the need of an explicit tutorial. Tabletops in general are designed to be used by people without prior knowledge. The best way to evaluate learnability is through a user study where the participants are introduced to tabletop with minimal tuition.

H2 Ease of Use

Ease of use was another design goal for this project. We aimed to create an interface that is easy to use by users unfamiliar to tabletops. Tabletops are designed to mirror a real life table creating an easy free flowing environment. The best way to evaluate ease of use is through a user study where the participants are introduced to tabletop and expected to perform tasks with minimum errors.

H3 Error Recovery

The goal of ease of use includes careful design to reduce user errors where possible. This design goal is another core aspect of usability. There are many cases where users make errors, for example, selecting the wrong element of the interface, starting the wrong application, accidentally exiting the current application. It is important that it is easy for the user to recover from such errors and get back to the intended system state. Table 6.1 lists examples of errors in CAM, aspects of the design intended to minimise these errors and what the user needs to do to recover from each. A user study is needed to evaluate the effectiveness of CAM error preventive measures.
Table 6.1: Examples of the main Errors and recovery from them. For each error, the table first states the design elements that reduce the risk of this error and then follows a description of the error recovery steps.

<table>
<thead>
<tr>
<th>Possible Errors</th>
<th>Solutions Implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accidental starting of an incorrect application;</td>
<td>• CAM minimises accidental starts because the user must dwell on the start button.</td>
</tr>
<tr>
<td></td>
<td>• For recovery, the user opens the “Application Panel” within the application and dwells on exit to return to the start screen.</td>
</tr>
<tr>
<td>Accidental exit of an application;</td>
<td>• CAM attempts to minimise accidental exit from an application because the user must dwell on the exit button.</td>
</tr>
<tr>
<td></td>
<td>• For recovery, the user navigates to the start screen and relaunches the application.</td>
</tr>
<tr>
<td>Loading an application with the wrong configuration;</td>
<td>• CAM minimises accidental loading of wrong configuration by proving the user with a default configuration preselected on “Launcher”</td>
</tr>
<tr>
<td></td>
<td>• For recovery, the user exits and reloads the application with the correct configuration.</td>
</tr>
<tr>
<td>Loading the wrong fileset of an application;</td>
<td>• CAM minimises the accidental loading of the wrong fileset by allowing the user to select the correct fileset and update it from the “Launcher” within the application.</td>
</tr>
<tr>
<td></td>
<td>• For recovery, filesets can be recovered by updating the application fileset.</td>
</tr>
<tr>
<td>Accidental addition of tools to a running application.</td>
<td>• For recovery, users can delete the tools through the use of the Black Hole.</td>
</tr>
</tbody>
</table>

6.2 Experimental Design

In order to analyse the evaluation goals stated in Section 6.1 a Think Aloud (Nielsen, 1993) user study was selected; this technique provides insight into the user’s reason for the action they performed. This methodology requests participants to think aloud to tell the evaluator what they are thinking at the time they are performing the task. This allows the evaluator to gain insight into how users react to CAM’s interface without prior knowledge of CAM but with an understanding of tabletop interaction and the thought processes behind the actions taken by the user when performing tasks. When choosing this methodology, we need to consider that the result is not quantifiable, and that this method slows the users down by adding additional cognitive load thus skewing timing results. Therefore time was not used in evaluating the outcome of these studies.

---

3 That persistence between application sessions are depended on the application developers and is not in the core scope of CAM
4 This depended on the application designer effectively setting the defaults when setting up an application.
5 This is depended on the application designer to provide the plugins that support “Plugin Update” feature and specify the “Plugin Update” settings in the application definition file.
The Think Aloud study was designed with 4 main sections:

1. Background questionnaire, determines details of participant population;
2. Tutorial of normal tabletop use, ensures the evaluation of CAM independently of the Cruiser elements;
3. A series of tasks assessing each functionality of CAM, assesses learnability ease of use and recoverability; and
4. A post experiment questionnaire regarding their experiences with completing the tasks, assesses participants perceptions of learnability, ease of use and to elicit general comments and responses to CAM.

6.2.1 Background Questionnaire

This was designed to capture relevant aspects of the participant’s background. The Background Questionnaire is in Appendix on page 118. The questionnaire asks about: the participant’s occupation; age; gender to gauge an understanding of their demographic. Questions were asked on the weekly average hours of computer usage and a description of how well they organise their file structure on their computers. This is to understand the technical background of the participants. One question asked if participants had used tabletops before. This was important for interpretation of the results since previous experience, particularly with other tabletops may have affected the participant’s expectations about the interface and their ability to learn to use it. One question asked about use of Focus on the tabletop. This too may be important in interpreting different users’ ability to complete the tasks and to learn how to use CAM.

6.2.2 Cruiser Tutorial

The tutorial was designed to provide the participants with a familiarity with Cruiser’s interface without CAM. The following features of Cruiser’s interface were demonstrated; Moving Objects on screen, Rosize (Rotate & Resize), Flipping, the Black Hole and using Smart Objects (Browser, Photocopier, Containers (Storage Bin), Frame). These are core aspects of Cruiser and are utilised throughout its user interface. In addition, the participants were introduced to Focus, Cruiser’s associated file system browser. This was important because Focus was one of the tools used in the tasks involving fileset switching and adding objects. The evaluator demonstrated the above mentioned features of Cruiser. After each feature was demonstrated the participant was allowed to practice using the feature until they felt comfortable. A copy of the tutorial sheet can be found in Appendix on page 119.
6.2.3 Study Task List

Each participant was requested to complete a series of tasks using the Think Aloud methodology involving CAM without any introduction to test learnability, ease of use and error recovery of CAM.

The tasks were separated into two scenarios (refer to Appendix on page 120); this was done to motivate participants to change applications. The tasks were designed so that participants could complete them all within 45 minutes. Each scenario comprised of 5 tasks that require the user to utilise multiple aspects of CAM’s functionality within an application. The design of the experimental tasks is summarised in Table 6.2, showing how the tasks called upon participants to perform five of the key functions at least 3 times. For example, the first row indicates that the task of starting a new application was required to complete Task 1 in Scenario 1 as well as Tasks 6 and 7 of Scenario 2. The task design called on participants to set the configuration just once, as part of Task 7 of Scenario 2. The task design, with multiple uses of most of the functionality, was particularly important to gain a better understanding of learnability and memorability of the interface.

So, for example, if a participant needed to explore the interface to correctly make use of functionality on their first attempt, the experimenter would note this. If they had similar difficulties on subsequent attempts, this would point to serious problems for learnability and memorability. However, if they completed the task quickly and easily on a second use, this is a strong indicator that the functionality is easily discovered and learnt.

<table>
<thead>
<tr>
<th>Function</th>
<th>User Tasks</th>
<th>Scenario 1</th>
<th>Scenario 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting Application</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Setting Fileset</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Setting Configuration</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Change Fileset</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Add Tools</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Exit Application</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

The first scenario involved selecting files for planning a fictional trip to Melbourne, London and Florence. This used a sample application named “Travel Guide” and filesets used consisting of a collection of images and documents relating to attractions and accommodation in three locations, Melbourne, London and Florence. This application has an option named “Widget Only Mode” which is enabled by default. This option sets Focus into its widget functionality as described in Section 4.3. When disabled, Focus is set to its table-only mode,
with the results scattered across the table. This is not used in the first scenario. Three tools were made available to the “Travel Guide” application, and so appeared on the CAM widget: the Storage Bin, Copier and Frame object. Only the Storage Bin was used in this scenario because it only was applicable for the tasks required by this scenario. Participants were requested to organise 3 separate collections of files into the storage bins. Using other tools would assess the same functionality in CAM, but would be difficult to introduce to the scenario. The other tools were available in the sample application to provide a sense of completeness to the sample application.

The second scenario involved reviewing pictures and documents from a trip to a museum, to create a collage of exhibits for a project. This uses “Museum Guide”. The filesets used consists of documents and images from an Ancient Greek History exhibit and Sydney Sidetracks exhibit. This application also has an option named “Widget Mode”, which this time was disabled by default. Three tools were made available as in the previous scenario. This time the Frame was used to capture an image of a user created collage and the storage bin was to store the files collected as in scenario 1. Again the other tool was available to provide a sense of completeness to the sample application.

The tasks were designed to ensure that the participants used CAM, with emphasis on Focus and a series of Cruiser primitives that include the Storage Bin and Frame object. Focus was used to display and interact with the filesets in the scenarios. The tasks used in the scenarios are as follows:

Scenario 1: The Travel Guide

0. Start State.
   - “The table is turned on and you are presented with the start screen.”

Cruiser (with CAM) was started up and the participants were presented with a table with only the application icon displayed.

1. Loading the Travel Guide.
   - “You have decided to use Travel Guide to organise your trip to Melbourne, and you also need a container to store these files. Now start the Travel Guide.”

As indicated in Table 6.2, this task requires the participant to open the launcher (which requires opening the Cruiser Guide Application container and pulling out the right application) and then setting the file set (from its default of Florence to Melbourne) and start the application (by dwelling on the Start button in the launcher).
   - “Now that Travel Guide has loaded with the Melbourne collection browse this collection to locate four files on Melbourne’s attractions and place them into your container.”
   - “You have now acquired a collection of items for Melbourne, and now are ready to plan the next part of your trip. You now need to get a collection of places to go from the London files. Now you need to plan the London part of your trip”

As indicated in Table 6.2, this task required the participant to change the fileset (which requires opening the launcher, selecting a new fileset, London and dwelling on update)

3. Using the Tools.
   - “Now that the Travel Guide has reloaded, again browse this collection to locate four files on London’s attractions and place them into your container.”
   - “You should notice that the container is getting crowded, so you decide to get another Storage Bin to place your files from London.”

As indicated in Table 6.2, this task required participant to add tools to the environment (which requires opening the Launcher, and tapping on the storage bin icon).

   - “Now you are ready to plan the final part of your trip Florence Italy. You now need to get 6 files of places to go in Florence. You may require an additional Storage Bin for Florence.”

As shown in Table 6.2, this task required the participants to change filesets and add an additional Storage Bin (this required the participant to open the launcher, select the new file set, dwell on update as well as tapping on Storage Bin to add it to the environment).

5. Finishing up.
   - “You have finished. Now get ready for the next task, which will use a different application”

As shown in Table 6.2, this task required participants to exit the application (this involves opening the launcher and dwell on exit).
Scenario 2: The Museum Exhibit

   - “You would like to review the exhibits you have explored using a tabletop application. You have decided to use the Museum Guide to look at the museum’s collections. You also want to make a collage of exhibits that you like. Go and start the Museum Guide Sydney Sidetracks files.”

As indicated in Table 6.2, this task requires the participant to open the launcher (which requires opening the Cruiser Guide Application container and pulling out the right application) and then setting the file set (from its default of Greek to Sidetracks) and start the application (by dwelling on the Start button in the launcher).

7. Reloading the Widget.
   - “You have now loaded the application and noticed that the files are scattered around the table. You decided to exit out of the Museum Guide, enable the widget only mode and rerun the Museum Guide.”

As shown in Table 6.2, this task requires the participants to exit the application (which required opening the Launcher and dwelling on exit), setting the configuration (first requires opening the Cruiser Guide Application container and pulling out the right application and selecting the configuration), setting file set (tapping on the selection) and starting the application (dwelling on start)

   - “Explore the file set, Locate three images of Eternity and two images of the Opera House (can be located through the black and white construction photo), and place them into the Storage Bin. “
   - “Once you are satisfied, have a look at the Greek History exhibit.”

As indicated in Table 6.2, this task required the participant to change the file set (which requires opening the launcher, selecting a new fileset, Greek History and dwell on update)

   - “Explore the file set, Locate 5 images of Greek vases, place them into the Storage Bin.”
   - “Now you are ready to make the collage. Take out all the images and place them on the table throwing away any unwanted images into the black hole.”
Chapter 6. Evaluation

6.2 Experimental Design

- “Add the frame to the table, arrange the images on the table and take a screen shot.”
  As indicated in Table 6.2, this task required participant to add a tool to the environment (which requires opening the Launcher, and tapping on the Frame icon).

10. Finishing up.
  - “You have finished. Please leave the table in the state that you started in”
  As shown in Table 6.2, this task required participants to exit the application (this involves opening the launcher and dwell on exit).

6.2.4 Post Experiment Questionnaire

The Post Experiment Questionnaire had a total of 14 questions and was divided into two parts. A copy of the exit questionnaire can be found in Appendix on page 122. The first 9 questions used a seven point Likert scale with descriptions designed to capture participant’s perceptions of learnability (H1) and ease of use (H2) of CAM. These included their assessment of task performance and understanding (Q1), key functional elements of CAM: starting (Q2, 3); setting and switching the filesets (Q7, 8) and using tools (Q9); and key user interface elements including the Resize on Dwell (Q5); dwelling on buttons (Q6) and tapping on options (Q7) and interface feedback (Q4). The rest of the questionnaire sought general comments about aspects that the participants liked best (Q10), disliked (Q11) suggestions for changes (Q12, 13) and any other comments (Q14).

The questionnaire was given to participants directly after the study component when their memories of CAM were fresh. The CAM interface was made available, so that they could reference it when completing the questionnaire.
6.3 Participants

Eight participants were recruited. This is well above the recommended for usability studies (Nielsen, 2000). The participants’ background consisted of a maths tutor, junior animator, and university students that were a combination of Information Technology, Electrical Engineering and Science backgrounds as shown in Table 6.3.

Table 6.3: Participants Demographics

<table>
<thead>
<tr>
<th>User</th>
<th>Age</th>
<th>Gender</th>
<th>Occupation</th>
<th>Hours of Computer use per week</th>
<th>Description of file organisation structure on computer</th>
<th>Previous use of tabletop interface</th>
<th>Previous usage of Focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21</td>
<td>Male</td>
<td>Student (Elect Eng)</td>
<td>31-40</td>
<td>Basic</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>Male</td>
<td>Math Tutor (Education)</td>
<td>41 or more</td>
<td>Extensive</td>
<td>Yes, NSW museum, Fish/eco system interactive to touch</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>Female</td>
<td>Student (Science)</td>
<td>21-30</td>
<td>Average</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>Female</td>
<td>Student (IT)</td>
<td>41 or more</td>
<td>Extensive</td>
<td>Yes, for demo that was recorded and a user study (both with Cruiser)</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>Male</td>
<td>Junior Animator</td>
<td>31-40</td>
<td>Average</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>22</td>
<td>Male</td>
<td>Student (Elect Eng)</td>
<td>31-40</td>
<td>Average</td>
<td>Yes, another user study</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>24</td>
<td>Male</td>
<td>Student (Elect Eng)</td>
<td>41 or more</td>
<td>Extensive</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>22</td>
<td>Female</td>
<td>Student (Science)</td>
<td>21-30</td>
<td>Average</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

All participants used computers form a minimum of 21 hours a week, most 31 hours or more. This indicates the population is highly computer literate. While this limits the generality of the result, it is a useful group for the first evaluation of an unfamiliar class of user interface on tabletops. In addition they provided a description of the their file system organisation, where most responded with an Average to Extensive organisation with the exception of one who stated that he had a rudimentary file organisation structure on his computer. From the questionnaire it is apparent all participants had an understanding of desktop computers. Finally three of the participants were exposed to the tabletop environment before, two of who used Cruiser and one who used an interactive museum tabletop. None of the participants had seen CAM or were familiar with its functionality prior to the study.
6.4 Results

This section presents the results of the qualitative user study conducted to assess the usability of CAM's user interface. It summarises the observations made by the experimenter during the study and feedback from participants' questionnaires about their experience with CAM. The user tasks observations were based on extensive note taking during the user study.

6.4.1 High level Results

This section first describes the results of each task in terms of the hypotheses. Table 6.4 provides a summary of the results for the first scenario, Tasks 1-5, and Table 6.5 for the second scenario, Task 6-10. For each task, the tables show the CAM functionality that participants needed to learn in the first use and remember for subsequent uses (refer to Table 6.2) for design of the tasks to demonstrate this.

As shown in the first row of Table 6.4, 2 participants out of 8 failed Task 1, Scenario 1, loading an application and setting fileset, and required assistance to complete task (✗), 5 participants were able to do this task after experimentation (●) and 1 participant was able to complete the task fluently (✔). Table 6.5 revealed that on the second and third run through (Tasks 6 - 7, Scenario 2) at loading an application, 7 out of 8 participants complete this task fluently and 1 had to explore to complete the task. This shows that in the subsequent tasks, participants completed the tasks with ease, indicating they had learnt that function.

Table 6.5 shows that 7 out of 8 participants successfully completed the task of setting the application options (Task 7, Scenario 2) and 1 was able to complete task after experimentation. This shows that the participants were able to complete this task with ease.

The switching fileset activity (Tasks 2 and 4, Scenario 1 and Task 8 Scenario 2) also demonstrated that it was easy to learn. 5 out of 8 participants were able to complete Task 2, Scenario 1 fluently, 2 participants were able to complete the task after experimentation and 1 participant failed the task and needed assistance as shown in Table 6.4. Task 4, Scenario 1 was evenly divided between 4 participants completing the task fluently and 4 requiring experimentation. Task 8, Scenario 2 (Table 6.5) had six participants completing the task with no assistance, 1 with exploration and 1 failed due to Focus. This shows that in subsequent tasks, participants completed the tasks with ease, indicating they had learnt that function.

The adding tools activity (Tasks 3 and 4, Scenario 1 and Task 9, Scenario 2) had 2 out of 8 participants were able to complete Task 3, Scenario 1 fluently, 6 participants were able to complete the task after experimentation as shown in Table 6.4. In Task 4, Scenario 1, this improved to 4 participants completing the tasks fluently and 4 with experimentation. In Task 9, Scenario 2, 3 participants needed experimentation to complete the task and 5 completed the
tasks fluently. This shows that in the subsequent tasks, participants completed the tasks with more ease, indicating they were learning that function.

The exiting the application (Tasks 5, Scenario 1 and Task 7 and 10, Scenario 2) all participants were able to complete Task 5, Scenario 1 fluently as shown in Table 6.4. Finally, Tasks 7 and 10 in Scenario 2 were completed with 7 out of the 8 participants completing the task fluently and 1 participant requiring experimentation.

<table>
<thead>
<tr>
<th>Table 6.4: Observation of Task Execution for Scenario 1</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td><strong>Task 1: Loading the Travel Guide</strong></td>
<td>×</td>
</tr>
<tr>
<td>Open Application Container</td>
<td>●</td>
</tr>
<tr>
<td>Open Application Panel</td>
<td>✓</td>
</tr>
<tr>
<td>Configure Application</td>
<td>✓</td>
</tr>
<tr>
<td>Start Application</td>
<td>×</td>
</tr>
<tr>
<td><strong>Task 2: Planning London Trip</strong></td>
<td>✓</td>
</tr>
<tr>
<td>Open Application Panel</td>
<td>✓</td>
</tr>
<tr>
<td>Select Options</td>
<td>✓</td>
</tr>
<tr>
<td>Update Application</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Task 3: Using the Tools</strong></td>
<td>●</td>
</tr>
<tr>
<td>Open Application Panel</td>
<td>✓</td>
</tr>
<tr>
<td>Add Object</td>
<td>●</td>
</tr>
<tr>
<td><strong>Task 4: Planning Florence Trip</strong></td>
<td>✓</td>
</tr>
<tr>
<td>Open Application Panel</td>
<td>✓</td>
</tr>
<tr>
<td>Select Options</td>
<td>✓</td>
</tr>
<tr>
<td>Update Application</td>
<td>✓</td>
</tr>
<tr>
<td>Add Object</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Task 5: Finishing up</strong></td>
<td>✓</td>
</tr>
<tr>
<td>Open Application Panel</td>
<td>✓</td>
</tr>
<tr>
<td>Exit Application</td>
<td>✓</td>
</tr>
</tbody>
</table>

✓ = Task successfully completed on first attempt
● = Task completed after experimentation
* = Task failed
<table>
<thead>
<tr>
<th>Task 6: Loading Museum Guide</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Application Container</td>
<td>✔ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
<tr>
<td>Open Application Panel</td>
<td>✔ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
<tr>
<td>Configure Application</td>
<td>✔ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
<tr>
<td>Start Application</td>
<td>✔ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task 7: Reloading the Widget</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Application Panel</td>
<td>✔ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
<tr>
<td>Exit Application</td>
<td>✔ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
<tr>
<td>Open Application Container</td>
<td>✔ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
<tr>
<td>Open Application Panel</td>
<td>✔ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
<tr>
<td>Configure Application</td>
<td>✔ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
<tr>
<td>Start Application</td>
<td>✔ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task 8: Switching to Greek History</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Application Panel</td>
<td>✗ ✔ ✗ ✔ ✔ ✔ ✔</td>
</tr>
<tr>
<td>Select Options</td>
<td>✔ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
<tr>
<td>Update Application</td>
<td>✔ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task 9: Using Tools (The Frame)</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Application Panel</td>
<td>✔ ✗ ✔ ✔ ✔ ✔ ✔</td>
</tr>
<tr>
<td>Add Object</td>
<td>✗ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task 10: Finishing up</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Application Panel</td>
<td>✔ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
<tr>
<td>Exit Application</td>
<td>✔ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
</tbody>
</table>

✔ = Task successfully completed on first attempt  
• = Task completed after experimentation  
✗ = Task failed  
* = Task intervention was related to Focus not CAM and therefore not covered
6.4 Results

6.4.2 User Tasks

The first scenario consisted of running the “Travel Guide” application, switching between the three filesets Melbourne, London and Florence, and adding multiple storage bins to the environment. This is the first task in which the participants were exposed to CAM for the first time.

When the participants were first shown the Application Container from CAM in its icon form, their first instinct was to tap and move the icon around. After a short period of non-responsiveness the participants would dwell on the icon to reveal its collection. Participants 1, 5 and 6 attempted to drag out components of the icon graphic, assuming these were part of a collection, after which they dwelled on the icon. Participant 4 dragged on the icon’s corners to incidentally resize it to reveal the collection.

From this the participants were presented with a container, containing a collection of Application Panels in their icon form. All participants but one, at this stage dwelled on the “Travel Guide” icon to reveal the panel. Participant 5 dragged the panel out of the container to reveal the Application Panel. All participants except Participant 7 tapped to select the Melbourne fileset, Participant 7 dwelled on the fileset option in an attempt to start the application with that fileset.

All participants except Participant 2 had difficulty with dwelling on the buttons. Participants 1 and 6 failed the task and required the evaluator to guide them through the task. Participants 3-5, 7, 8 tapped the button first then dwelled after no feedback. Participant 2 commented that “Dwell becomes double click, Dwell when unsure”.

At the point at which they need to switch to the London fileset, all but three participants successfully opened the Application Panel, selected London and dwelled on Update. Participant 3 and 7 after selecting the file set tapped the update icon a couple times before realising that it reacts to dwelling. Participant 5 tapped and attempted to dwell on update with no success. Before the evaluator could intervene, the participant dwelled on exit out of desperation exiting the application. After this the “Travel Guide” was reloaded and the evaluation continued.

After the participants had collected an additional 4 files on London they were required to get an additional storage bin from the Application Panel. Participants 5 and 7 added the storage bin to the table by tapping on the tools option. Participants 1 and 6 attempted to add the storage bin to the table by dwelling on it. Participants 2, 3, 4 and 8 attempted to drag out the tools options onto the table. It was observed that this may be linked to the behaviour of the browser container shown in the tutorial.
Next the participants had to change file sets to Florence, collect an additional four files on attractions and add them into a new storage bin. Participants 1, 4, 7, 8 successfully completed this task with no errors. Participant 5 attempted to dwell on a file set selection in an attempt to update Focus with the new file set, and then moved to dwelling on the update button. Participant 2 tapped on update button and after no response dwelled on update. Participant 3 attempted to drag out a storage bin from the panel, since the tap gesture precedes the drag, the button will always react to this. Participant 6 attempted to load the storage bin by dwelling on the button.

After the participants organised their collection, they needed to complete the last part of the scenario and exit out of the application. All participants completed this as if it was the most obvious action.

The tasks in the second scenario were designed to mimic those from the first. This was to assess the learnability of CAM’s interface. The majority of tasks in this scenario were successfully completed (as shown in Table 6.4 and Table 6.5) with the following exceptions. Participant 5 attempted to launch the “Museum Guide” by dwelling on the “Sydney Option”. Participant 8 when reloading the “Museum Guide” in widget mode forgot to specify the file set with which to load the application, loading the wrong file set. This action was corrected by updating the fileset through CAM. Participant 3 kept on tapping on the Update button when trying to change file sets to “Greek History” when loading the widget. Participant 2 and 8 attempted to drag out the Frame from the panel; both at this point realised that it was reacting to tap. Participant 6 still dwelled when adding a frame to the table. Participant 3 repeatedly tapped on the exit button instead of dwelling.
6.4 Results

6.4.3 Post Experiment Questionnaire

Table 6.6 provides an overview of the results from the post experiment questionnaires using a 7 point Likert scale, where 7 denotes strongly agreeing and 1 denotes strongly disagreeing to the statements in questions.

<table>
<thead>
<tr>
<th>Participants</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Perform Tasks with no Problems</td>
<td>6</td>
<td>3*</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>2) Find the application that was looking for</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>3) The Launcher is easy to understand</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>4) The Launcher provides adequate feedback to actions</td>
<td>7</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5) The resize on dwell function is easy to use and intuitive</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6) Understood need to dwell on the buttons</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>7) Understood need to tap to select File Sets and Options</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>8) I had no trouble changing the file sets using the Launcher</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>9) I understood how to select the tools I wanted</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>6.5</td>
</tr>
</tbody>
</table>

* This response is due to the participant exploiting a known bug in Cruiser

From Table 6.6 it is apparent that the majority of participants rated their experiences with CAM favourably.

The first row shows that the six of the eight participants agreed that they were able to perform the tasks without any problems (reflected in Likert scores of 5-7). With the median for all participants at 6, the overall response to this question can be assessed as positive. P3 gave a lower score, attributing this to a problem which is due to a bug in the underlying Cruiser system, which fails if a storage bin is placed within a storage bin. The low score from P6 was because he was having difficulty with dwelling on the buttons which required intervention.

The second row of Table 6.6 shows that all participants felt that they easily could find the application that they were looking for such as the “Travel Guide”. The lowest score was 6 on the Likert scale.

The results from the third row showed that all the participants felt that the Launcher is easy to understand with Likert scores of 5-7 and median of 6.

The results from the fourth row show that the Launcher provided adequate feedback to their actions (reflected in Likert scores of 3-7) with the median for all participants at 5, the overall response to this question can be assessed as positive. Participants’ comments for this question include: “Inconsistent audio feedback had somewhat frustrating experience trying to not accidently trigger behaviours. That said, all buttons provided feedback and worked consistently.”

The results from the fifth row, showed that all the participants felt that the resize on dwell function is easy to use and intuitive with Likert scores of 3-7 and median of 5, with the
exception of one, where P2’s confusion of dwelling and resizing of menu items resulted in negative ratings. “Dwelling on moving causes actions which then ends my resizing.”

The results from the sixth row, showed that all the participants felt that they understood need to dwell on the buttons with Likert scores of 2-7 and median of 6. There was a wide spread of responses about participants’ understanding of the need to dwell on buttons to enact a function. For example P6 stated “I think I used it without realising I was dwelling and resizing”.

The results from the seventh row showed that all participants understood need to tap to select File Sets and Options with Likert scores of 2-7 and median of 6. P 6 said “After assistance, I assumed that dwelling was necessary for all buttons,” and P 4 said “I was dwelling on them.”

The results from the eight row showed that all participants had no trouble changing the file sets using the Launcher with Likert scores of 4-7 and median of 7. This is a very favourable response.

The results from the ninth row showed that all participants understood how to select the tools they wanted with Likert scores of 4-7 and median of 6.5. This is a very favourable response.

6.5 Conclusions

This section presents the results obtained from the observation and the post experiment questionnaire. The analysis addresses the hypotheses outline at the start of the chapter.

6.5.1 H1 Learnability

The majority of participants were able to complete tasks without assistance as shown in Table 6.4 and Table 6.5. However, three participants needed assistance to complete the tasks in the first scenario. All three had the same problem: they forgot that they needed to dwell on the buttons. Table 6.4 shows that the participants were exploring the interactions available with CAM during the first scenario and Table 6.5 indicates that they learnt the tasks through experimentation as all the tasks were completed successfully in the second scenario. From this it can be concluded that CAM is a tabletop user interface that is relatively easy to learn through experimentation.

The study indicated that participants may be better able to learn to use CAM, without tuition, if the dwell time for the Start, Exit and icon enlargement actions were shorter. The value used in the study is the same as that used successfully over several user studies with Cruiser (Apted et al., 2005) and Focus. It is unclear why participants had more difficulty with this in CAM.
6.5 Conclusions

6.5.2 H2 Ease of Use

Through observations as tabulated in Table 6.4 and Table 6.5, it is apparent that all participants were able to complete the tasks in the tests scenarios, with the exception of the 3 participants who required intervention. Although these 3 participants required an initial intervention there were no repeat interventions related to CAM. From this it can be concluded that CAM is a tabletop user interface that is relatively easy to use.

6.5.3 H3 Error Recovery

The user study was set up so that one task would require the participant to run through an error event to be able to assess how well they recovered from it. The 8 users experienced a total of 33 errors, with 1 due to the failures in Cruiser or Focus, 29 occurring as they explored CAM to discover its functionality and 3 due to accidental actions later in their use of it. They were able to recover from 30 of the 33 errors they encountered with CAM without assistance. Based on this it can be concluded that it is easy to recover from errors using CAM.

6.5.4 Problems Identified

While the overall usability and learnability of CAM was high, the study did indicated the following aspects that could be improved

- reduce the dwell time, to reduced errors due to users failing to dwell for long enough;
- provide audio feedback on key changes, starting an application and changing filesets;
- refine the Application Menu icon to make its function more obvious.

Participants commented on how the interface could be improved upon including:

- allow direct launching of a new application from within a running application.
- provide a shortcut enabling users to dwell on an fileset in the launcher to have it update the fileset in the environment.
Chapter 7

Conclusions

The goal of this thesis is to move tabletops beyond single application appliance computing that has dominated the research area towards a multipurpose device. The aim is to move tabletops towards the generality offered by general purpose computing by providing the following:

- the ability to switch between arbitrary applications,
- allow defining the configuration of an application prior launching,
- setting filesets prior to load,
- switching filesets while running the application, and
- the ability to add tabletop primitives to the table while running an application.

7.1 Contributions

Exploration of several approaches to designing the interface for application management, based on prototyping and discount usability evaluations. The approach taken was to use a combination of parallel and iterative prototyping to develop a series of low fidelity prototypes. Discount usability methods were used to evaluate and determine which design was the most learnable and usable for implementation. Four designs were developed to explore the design space: Scatter, Drawable Menu, Zipper and Panel Box prototypes as described in Chapter 3. To evaluate these prototypes a combination of Heuristic Evaluations and Cognitive Walkthrough techniques were used. A total of 21 Heuristics was chosen, the classic Nielsen 10 Heuristics (Nielsen, 1992), a selection of Large Interactive Display Heuristics (Somervell et al., 2003) and Tabletop Heuristics (Apted et al., 2009). These prototypes were evaluated by 5 researchers with expertise in user interface design and evaluation. This evaluation identified
potential usability tradeoffs associated with each of the approaches explored in these prototypes. In light of these evaluations, the Panel Box was identified as the most promising approach. It was the basis for the first implemented CAM prototype.

**Design of a new tabletop user interface that allows users to load and switch between individual and sets of plugins (application) in Cruiser.** From the exploration of the design space two additional design goals were added; to support configuring of the application prior to launch and to have the ability to add tabletop primitives to the table. The user view of CAM is presented in Chapter 4. This design evolved from the Panel Box prototype with modifications implemented from the feedback from the Heuristics and Cognitive Walkthrough conducted for the Panel Box as described in Chapter 3.

CAM provides the user with the ability to select an application, configure it, select filesets to work off and launch it within a tabletop environment. CAM also allows the user to switch filesets within it as well as add tabletop primitives.

**An architecture that supports this functionality to operate within Cruiser by adding support to allow Cruiser to load and unload plugins on the fly.** The technical implementation and architecture overview can be found in Chapter 5. CAM was implemented as an extension off the Cruiser framework and consists of four modules: the Cruiser Core modifications; the Plugin Update; Tools and the CAM module. Cruiser’s core architecture was modified to allow for the ability to restart the environment without exiting the framework. CAM’s framework is implemented as plugins in Cruiser’s framework.

The Plugin Update library provides plugin designers the ability to send data or instructions to other plugins without the need to know its internal structure or interface. The tools allow plugin designers to create their own tool that is available through the CAM interface reducing table clutter. CAM interprets the XML Application Definition File to specify the available filesets and options, as well as its appearance in the Application Panel.

**Evaluation of the user interface with a user study.** A user study was designed and conducted to evaluate the learnability and usability of CAM. The design and results of this user study can be found in Chapter 6. The User study consisted of 4 parts: background questionnaire; tutorial (without CAM); tasks using CAM and exit questionnaire. A tutorial was given to introduce participants to Cruiser without CAM. Then participants were asked to complete a set of tasks which made use of CAM. This design meant that participants had to discover CAM’s functionality without any instruction. This design was to test the learnability of CAM. The exit questionnaire was designed to gauge participants’ impressions of the CAM
interface. This study demonstrated that the CAM user interface is easy to learn and is easy to use to perform the key tasks needed to move beyond the current state-of-the-art appliance model for tabletop interaction.

**Demonstration of the architecture in terms of its power, flexibility and effectiveness.** The usability study described in Chapter 6 demonstrates the effectiveness of CAM, both in terms of its interface and the underlying architecture and implementation. The user study demonstrated that this architecture allows:

- for switching between arbitrary applications,
- the ability to set the fileset prior to launch,
- switch the fileset while the application is running,
- the ability to configure an application prior to loading, and
- add tabletop primitives or tools on to the environment.

### 7.2 Future work

This thesis has resulted in the creation of CAM, Cruiser’s Application Manager for the Cruiser tabletop environment. This design has the potential to serve as a foundation for exploration of ways to provide additional facilities for tabletop interaction.

While the user study indicated that CAM was learnable without instruction and easy to use for the particular population represented by the participants, it will be important to conduct additional evaluations with participants from a broader range of technical expertise as well as a broader age range.

Another aspect of future work is to expand CAM’s display and architecture to allow the application designer using the XML file to customise the components shown on the Application Panel instead of only the filesets and options components. For example, it would be useful to be able to define a player panel for a game application, so that the panel can be used to establish the number of players.

Future work might also explore more of the design elements from the low fidelity prototypes. For example, the drawable menus seem promising. They offer promise to be very effective for use with applications for which the Application Panel is not appropriate. It may also be valuable to add support for external devices, such as mobile phones, to control some aspects of the tabletop.
7.3 Summary

This thesis explored interface design to provide application management capabilities onto the tabletop environment. Several low fidelity prototype designs within the design space were explored and evaluated (Chapter 3). Out of these one design was chosen for implementation based on Heuristics and Cognitive Walkthrough evaluations. The design was refined to remove some of the flaws identified and implemented (Chapter 4).

The architectural framework of CAM is designed to provide the ability to: switch applications; update plugins state without the need to explicitly know the plugin interface; allow current and future plugins to use this feature for fileset switching; add tools to the environment and allows application designers to define applications easily.

Finally to evaluate CAM’s potential, a user study was conducted evaluating usability aspect of CAM to determine how easy it is to learn, how easy to use and how readily users could recover when they made errors. From this study it was apparent that the users after a short time were able to learn and use the CAM user interface effectively. In summary, CAM provides the tabletop with the ability to switch between applications, change filesets and modify the environment that moves the tabletop one step beyond the appliance model, and to a more general purpose device.


conference on Human factors in computing systems, pages 587–592, New York, NY, USA. ACM.


Appendix A

Heuristics Definition

This appendix presents the heuristics used during the Heuristic Evaluation of the low fidelity prototypes in Chapter 3.

Nelsons Ten Heuristics

1 Visibility of system status
The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

2 Match between system and the real world
The system should speak the users’ language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

3 User control and freedom
Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

4 Consistency and standards
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

5 Error prevention
Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
6 Recognition rather than recall
Minimize the user’s memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

7 Flexibility and efficiency of use
Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

8 Aesthetic and minimalist design
Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

9 Help users recognize, diagnose, and recover from errors
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

10 Help and documentation
Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user’s task, list concrete steps to be carried out, and not be too large.

Large Displays Desktop Heuristics

11 Appropriate colour schemes can be used for supporting information understanding.
Try using cool colours such as blue or green for background or borders. Use warm colors like red and yellow for highlighting or emphasis.

12 Layout should reflect the information according to its intended use.
Time based information should use a sequential layout; topical information should use categorical, hierarchical, or grid layouts. Screen space should be delegated according to information importance.

13 Judicious use of animation is necessary for effective design.
Multiple, separate animations should be avoided. Indicate current and target locations if items are to be automatically moved around the display. Introduce new items with slower, smooth transitions. Highlighting related information is an effective technique for showing relationships among data.
14 **Show the presence of information, but not the details.**
Use icons to represent larger information structures, or to provide an overview of the information space, but not the detailed information; viewing information details is better suited to desktop interfaces. The magnitude or density of the information dictates representation mechanism (text vs icons for example).

**Tabletop**

15 **Design independently of table size,**
Design for different tabletop sizes and allow flexible resizing of all interface elements.

16 **Support reorientation,**
Allow all interface elements to be easily rotated to support users working at any position around the table, and consider users moving around the table while using it.

17 **Minimise human reach**
Consider that users may not be able to physically reach all interface elements. Elements must be moveable to all areas of the table.

18 **Use large selection points**
Design independently of tabletop input hardware, but support large input cursors (e.g. human fingers) where possible.

19 **Manage interface clutter**
Support quick removal or hiding of objects on the tabletop, while ensuring management of clutter by one user does not have unwanted side-effects on other users of the table.

20 **Use table space efficiently**
Avoid modal behaviour that limits the utilisation of table space. Allow arbitrary groupings of interface elements for personal and group spaces.

21 **Support private and group interaction,**
Support interaction by a single user or multiple users.
Panel Box Prototype Heuristics Results

Heuristic results of an study conducted by evaluators on the panel box low fidelity prototype. This is a representative example of the four heuristic evaluations on the low fidelity prototypes as described Chapter 3.

1 Visibility of system status

<table>
<thead>
<tr>
<th>User</th>
<th>Severity</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>There is no loading indicator after selecting an app or file-set - apps might take a few seconds to load and there should probably be an indicator or animation that something is happening.</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
<td>No indication of current fileset; could be placed inside or near the application icon</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>The application disappeared rather abruptly. Some kind of animation (i.e. user feedback) is quite important here. I was lost. One possible animation is shrink the application view into the blue square.</td>
</tr>
</tbody>
</table>

3 User control and freedom

<table>
<thead>
<tr>
<th>User</th>
<th>Severity</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>Much clearer than the previous prototypes on how to exit and go back.</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>Support both drag and touch for selection.</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>The application disappeared rather abruptly. Some kind of animation (i.e. user feedback) is quite important here. I was lost. One possible animation is shrink the application view into the blue square.</td>
</tr>
</tbody>
</table>

4 Consistency and standards

<table>
<thead>
<tr>
<th>User</th>
<th>Severity</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>3</td>
<td>The word &quot;exit&quot; needs careful consideration. Are we switching applications or exiting them? Likewise &quot;go&quot;.</td>
</tr>
</tbody>
</table>
The menu inside the application icon should be proportionally shown, otherwise the resizing action may be not clear.

The configuration panel is inconsistent with the other prototypes. (usability study design problem)

Scroll bar might not be intuitive.

Why not just click on the “GO”? It looks like a button.

“Exit” or “Home”? Maybe a better word needs to be used, as this potentially “shuts” the application.

<table>
<thead>
<tr>
<th>User</th>
<th>Severity</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
<td>Good - possibly a confirmation before exit could be useful</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>It is not obvious to user’s that they should shrink the application icon to cancel their changes</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>Can “GO” be checked without a fileset being selected? It should not be.</td>
</tr>
</tbody>
</table>

5 Error prevention

6 Recognition rather than recall

7 Flexibility and efficiency of use

8 Aesthetic and minimalist design

10 Help and documentation

11 Appropriate colour schemes can be used for supporting information understanding.

12 Layout should reflect the information according to its intended use.
### 14 Show the presence of information, but not the details.

<table>
<thead>
<tr>
<th>User</th>
<th>Severity</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>Unclear what (if anything) is optional to select on the screen (e.g. optional data-sets and/or configurations).</td>
</tr>
</tbody>
</table>

### 17 Minimise human reach

<table>
<thead>
<tr>
<th>User</th>
<th>Severity</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>Reachability is a problem with this design, because a user might not be able to reach the app menu. Consider the ability to pull the app menu out from the four corners of the tabletop.</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>If original box is in a far-off part of the table, this may be a problem. Also, does not work well for a single user</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>The application box may initially start out of reach of the user</td>
</tr>
</tbody>
</table>

### 19 Manage interface clutter

<table>
<thead>
<tr>
<th>User</th>
<th>Severity</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>2</td>
<td>When user has expanded the Application icon, the application box is left littering the table</td>
</tr>
</tbody>
</table>

### 21 Support private and group interaction,

<table>
<thead>
<tr>
<th>User</th>
<th>Severity</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>2</td>
<td>This interface is geared towards a single user, meaning that they will probably have control over what application is launched rather than it being democratic.</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>All icons are orientated to only one user when they are RO-Sized</td>
</tr>
</tbody>
</table>
Appendix C

Cognitive Walkthrough - Evaluator A

The cognitive walkthrough results from evaluator A for the study conducted on the low fidelity prototypes. This is a representative example of the four cognitive walkthrough results on the low fidelity prototypes as described Chapter 3.

C.1 Prototype 1 - Scatter

1. User dwells on application from menu
   0. will user understand this sub-task is needed? Yes.
   1. will correct action be obvious? e.g. button visible. No. If the user has never seen this before, dwelling it not immediately obvious - the user is more likely to just try clicking on objects instead of dwelling on them.
   2. user understand instructions? e.g. user understands the label on the button. Yes.
   3. will user know if progress has been made? Yes. The application icon animates to the new position.

2. User selects a file set and launches the application
   0. will user understand this sub-task is needed? Yes.
   1. will correct action be obvious? e.g. button visible. No. If the user has never seen this before, dwelling it not immediately obvious - the user is more likely to just try clicking on objects instead of dwelling on them.
   2. user understand instructions? e.g. user understands the label on the button. Yes.
   3. will user know if progress has been made? Yes. The application icon animates to the new position.

3. User accesses menu while in an existing application (dwells on wormhole)
0. **will user understand this sub-task is needed?** No. If the user has never touched the wormhole before, they would not know that they need to go to "control mode".

1. **will correct action be obvious? e.g. button visible.** No. The wormhole gives no indication on how to use it. A user is more likely to click on it rather than dwell on it if they have never used it before.

2. **will user understand instructions? e.g. user understands the label on the button.** Yes. Once in control mode, the interface is the same as what the user has used before (to launch the application). From here on, the user will understand how to use it.

3. **will user know if progress has been made?** Yes. Animations are used to show progress.

4. User selects a new application and fileset from menu
   0. **will user understand this sub-task is needed?** Yes.
   1. **will correct action be obvious? e.g. button visible** No. It is not initially obvious how to go back to the application menu (dwelling on the current application is not clear). After the user is back to the list of applications, it is then obvious how to proceed because the user has done this before.
   2. **will user understand instructions? e.g. user understands the label on the button** Yes. The interface is the same as what was used to initially launch the application.
   3. **will user know if progress has been made?** Yes. Animations are used to show progress.

5. User selects a new fileset from menu
   0. **will user understand this sub-task is needed?** Yes.
   1. **will correct action be obvious? e.g. button visible** Yes. The user does not need to go "back" to the application list.
   2. **will user understand instructions? e.g. user understands the label on the button** Yes. The interface is the same as what was used to initially launch the application.
   3. **will user know if progress has been made?** Yes. Animations are used to show progress.
C.2 Prototype 2 - Drawn Menu

1. User draws a path to access menu
   0. will user understand this sub-task is needed? No. If the user has never drawn a path before, then they would not know they need to do this to access the menu.
   1. will correct action be obvious? e.g. button visible No. It is not obvious that drawing a path will bring up a menu, unless the user has done it before.
   2. will user understand instructions? e.g. user understands the label on the button Yes. There are no instructions, but as the user draws the path then the menu will start gradually appearing.
   3. will user know if progress has been made? Yes. As the path is drawn, the menu is opened.

2. User selects application from menu
   0. will user understand this sub-task is needed? Yes.
   1. will correct action be obvious? e.g. button visible Yes. The buttons are clearly visible. The scrollbar is fairly obvious, but it would be desirable to have an indicator that there are more applications if the user scrolls left or right.
   2. will user understand instructions? e.g. user understands the label on the button Yes.
   3. will user know if progress has been made? Yes. The application icon animates to the new position.

3. User selects a file set and launches the application
   0. will user understand this sub-task is needed? Yes.
   1. will correct action be obvious? e.g. button visible Yes. The buttons are clearly visible. The scrollbar is fairly obvious, but it would be desirable to have an indicator that there are more applications if the user scrolls left or right.
   2. will user understand instructions? e.g. user understands the label on the button Yes.
   3. will user know if progress has been made? Yes. The application icon animates to the new position.
4. User accesses menu while in an existing application (dwells on wormhole)

0. **will user understand this sub-task is needed?** No. *If the user has never touched the wormhole before, they would not know that they need to go to "control mode".*

1. **will correct action be obvious? e.g. button visible** No. *The wormhole gives no indication on how to use it. A user is more likely to click on it rather than dwell on it if they have never used it before.*

2. **will user understand instructions? e.g. user understands the label on the button** Yes. *Once in control mode, the interface is the same as what the user has used before (to launch the application). From here on, the user will understand how to use it.*

3. **will user know if progress has been made?** Yes. *Animations are used to show progress.*

5. User selects a new application and fileset from menu

0. **will user understand this sub-task is needed?** Yes.

1. **will correct action be obvious? e.g. button visible** No. *It is not initially obvious how to go back to the application menu (dwelling on the name of the current application is not clear). After the user is back to the list of applications, it is then obvious how to proceed because the user has done this before.*

2. **will user understand instructions? e.g. user understands the label on the button** Yes. *The interface is the same as what was used to initially launch the application.*

3. **will user know if progress has been made?** Yes. *Animations are used to show progress.*

6. User selects a new fileset from menu

0. **will user understand this sub-task is needed?** Yes.

1. **will correct action be obvious? e.g. button visible** Yes. *The user does not need to go "back" to the application list.*

2. **will user understand instructions? e.g. user understands the label on the button** Yes. *The interface is the same as what was used to initially launch the application.*

3. **will user know if progress has been made?** Yes. *Animations are used to show progress.*
C.3 Prototype 3 - Zipper

1. User accesses menu with zipper
   0. **will user understand this sub-task is needed?** No. *If the user has never touched the zipper before, then there is no indication that the zipper is a menu.*
   
   1. **will correct action be obvious? e.g. button visible.** No. *As the zipper is animating (or pulsing), this will attract the user’s attention and indicate that the element can be manipulated. However, its function won’t be obvious to the user if they have not used it before.*
   
   2. **will user understand instructions? e.g. user understands the label on the button.**
      Yes. *There are no instructions, but it is modelled on a physical zipper, which should be familiar to the user. If the user attempts to move the zipper, then an animation will gradually show the contents of the menu, indicating that it is opening.*

   3. **will user know if progress has been made?** Yes. *As the zipper is pulled, the menu is opened.*

2. User selects application from menu
   0. **will user understand this sub-task is needed?** Yes.

   1. **will correct action be obvious? e.g. button visible.** Yes. *The buttons are clearly visible. The scrollbar is fairly obvious, but it would be desirable to have an indicator that there are more applications if the user scrolls down.*

   2. **will user understand instructions? e.g. user understands the label on the button.**
      Yes.

   3. **will user know if progress has been made?** Yes. *The application icon animates to the new position.*

3. User selects a file set and launches the application
   0. **will user understand this sub-task is needed?** Yes.

   1. **will correct action be obvious? e.g. button visible.** Yes. *The buttons are clearly visible. The scrollbar is fairly obvious, but it would be desirable to have an indicator that there are more applications if the user scrolls down.*

   2. **will user understand instructions? e.g. user understands the label on the button.**
      Yes.

   3. **will user know if progress has been made?** Yes. *The application icon animates to the new position.*
4. User accesses menu while in an existing application (dwells on wormhole)
   0. **will user understand this sub-task is needed?** No. *If the user has never touched the wormhole before, they would not know that they need to go to "control mode".*
   1. **will correct action be obvious? e.g. button visible.** No. *The wormhole gives no indication on how to use it. A user is more likely to click on it rather than dwell on it if they have never used it before.*
   2. **will user understand instructions? e.g. user understands the label on the button.**
      Yes. *Once in control mode, the interface is the same as what the user has used before (to launch the application). From here on, the user will understand how to use it.*
   3. **will user know if progress has been made?** Yes. *Animations are used to show progress.*

5. User selects a new application and fileset from menu
   0. **will user understand this sub-task is needed?** Yes.
   1. **will correct action be obvious? e.g. button visible.** No. *It is not initially obvious how to go back to the application menu (dwelling on the name of the current application is not clear). After the user is back to the list of applications, it is then obvious how to proceed because the user has done this before.*
   2. **will user understand instructions? e.g. user understands the label on the button.**
      Yes. *The interface is the same as what was used to initially launch the application.*
   3. **will user know if progress has been made?** Yes. *Animations are used to show progress.*

6. User selects a new fileset from menu
   0. **will user understand this sub-task is needed?** Yes.
   1. **will correct action be obvious? e.g. button visible.** Yes. *The user does not need to go "back" to the application list.*
   2. **will user understand instructions? e.g. user understands the label on the button.**
      Yes. *The interface is the same as what was used to initially launch the application.*
   3. **will user know if progress has been made?** Yes. *Animations are used to show progress.*
C.4 Prototype 4 – Panel Box

1. User accesses menu by selecting (enlarging) menu box
   0. will user understand this sub-task is needed? Maybe - it depends on the icon of the box which is not indicated in the diagrams.
   1. will correct action be obvious? e.g. button visible. No, but more obvious than the zipper and drawn menu designs. The user is somewhat likely to discover that they can enlarge it. An indicator on the box itself that it can be enlarged to show more information would be ideal.
   2. will user understand instructions? e.g. user understands the label on the button. Yes. There are no instructions, but it makes sense that more information is presented when the object is made larger.
   3. will user know if progress has been made? Yes. As the box is made larger, more information is shown.

2. User drags an application out from the menu
   0. will user understand this sub-task is needed? Yes.
   1. will correct action be obvious? e.g. button visible. No. This is not obvious - the more obvious action is to click or dwell on one of them, like in the previous prototypes. Some text or graphical instructions would be important here. Once the user has been shown once, then it will be obvious.
   2. will user understand instructions? e.g. user understands the label on the button. Yes.
   3. will user know if progress has been made? Yes. The icon stays underneath the users point of touch, and the new application box is opened once it is dragged off.

3. User selects a file set and launches the application
   0. will user understand this sub-task is needed? Yes.
   1. will correct action be obvious? e.g. button visible. No. It is unclear whether the user has to select a fileset and/or configuration before pressing the "Go" button - perhaps that should not appear until they have selected the required options.
   2. will user understand instructions? e.g. user understands the label on the button. Yes.
   3. will user know if progress has been made? Yes.
4. User accesses menu (enlarges it) while in an existing application
   0. will user understand this sub-task is needed? Yes.
   1. will correct action be obvious? e.g. button visible. Yes. The user enlarges just like when they launched the application.
   2. will user understand instructions? e.g. user understands the label on the button. Yes.
   3. will user know if progress has been made? Yes.

5. User exits the current application
   0. will user understand this sub-task is needed? Yes.
   1. will correct action be obvious? e.g. button visible. Yes. The button is clearly visible.
   2. will user understand instructions? e.g. user understands the label on the button. Yes.
   3. will user know if progress has been made? Yes.

6. User selects a new application and file set from the scatter box
   0. will user understand this sub-task is needed? Yes.
   1. will correct action be obvious? e.g. button visible. No. This is not obvious - the more obvious action is to click or dwell on one of them.
   2. will user understand instructions? e.g. user understands the label on the button. Yes.
   3. will user know if progress has been made? Yes. The icon stays underneath the user's point of touch, and the new application box is opened once it is dragged off.

7. User selects a new file set for the existing application
   0. will user understand this sub-task is needed? Yes.
   1. will correct action be obvious? e.g. button visible. Yes. The button is clearly visible.
   2. will user understand instructions? e.g. user understands the label on the button. Maybe - The "Go" button would probably be clearer as "Relaunch" or "Reload".
   3. will user know if progress has been made? Yes.
Appendix D

Sample Application Definition File

This the Application Definition File (TripPlanner.xml) for the sample application Trip Planner

```xml
<?xml version="1.0" encoding="utf-8"?>
<App>
   <!-- Defines the application name that appears in the menu -->
   <AppName> Trip Planner </AppName>
   <!-- Defines the information that appears in the menu -->
   <AppDiscription> This is Cruiser .Inc Tabletop Trip Planner </AppDiscription>
   <!-- Guide.rc is a common config that defines the settings for the Guide sample application -->
   <AppConfig>Guide.rc</AppConfig>
   <!-- Travel Guide Icon Image -->
   <AppIcon>Apps/Travel.png</AppIcon>
   <!-- Start of Options definition -->
   <Configs>
      <!-- The grouping that defines the widget only mode option -->
      <Config>
         <Selected>true</Selected> <!-- enabled by default -->
         <Name>Widget Mode</Name> <!-- The Name-->
         <File>FocusWidget.rc</File> <!-- The Cruiser config file that defines this option -->
      </Config>
      <!-- The grouping that defines the shared Data Wall mode option -->
      <Config>
         <Selected>false</Selected>
         <Name>Data Wall</Name>
         <File>Common/DataWall.rc</File> <!-- Shared Cruiser config for the Data Wall-->
      </Config>
   </Configs>
   <!-- Start of File Set definition -->
   <FileSets>
      <!-- The Grouping that defines the Florence option in filesets -->
      <FileSet>
         <Selected>true</Selected> <!-- Enabled by default -->
         <Name>Florence</Name>
         <!-- Cruiser config that defines the settings for Florence fileset -->
         <File>FSViewer/Florence.rc</File>
         <Image>Florence.png</Image> <!-- File set icon -->
         <!-- Defines which plugin to use for plugin update -->
         <Plugin>Focus</Plugin>
         <!-- The start of the first Plugin Update Variable Pair -->
         <PluginOptions>
```
<!-- The variable name for the variable pair This time the file server hostname--> 
<Variable>FSVIEWER_HOST</Variable>
<!-- The value of the previous variable, the fileserver host name--> 
<Value>fileserver</Value>
</PluginOptions>

<PluginOptions>
  <Variable>FSVIEWER_FILESET</Variable>
  <Value>/Florenc</Value>
</PluginOptions>
</FileSet>

<FileSet>
  <Selected>false</Selected>
  <Name>London</Name>
  <File>FSViewer/London.rc</File>
  <Image>London.png</Image>
  <Plugin>focus</Plugin>
  <PluginOptions>
    <Variable>FSVIEWER_HOST</Variable>
    <Value>fileserver</Value>
  </PluginOptions>
  <PluginOptions>
    <Variable>FSVIEWER_FILESET</Variable>
    <Value>/London</Value>
  </PluginOptions>
</FileSet>

<FileSet>
  <Selected>false</Selected>
  <Name>Melbourne</Name>
  <File>FSViewer/Melbourne.rc</File>
  <Image>Melbourne.png</Image>
  <Plugin>focus</Plugin>
  <PluginOptions>
    <Variable>FSVIEWER_HOST</Variable>
    <Value>fileserver</Value>
  </PluginOptions>
  <PluginOptions>
    <Variable>FSVIEWER_FILESET</Variable>
    <Value>/Melbourne</Value>
  </PluginOptions>
</FileSet>

</FileSets>
</App>
Appendix E

User Study Resources

This appendix contains the background questionnaire, Cruiser Tutorial, the CAM user study tasks and the post-experiment questionnaire.
E.1 Background Questionnaire

Background Questionnaire

Please complete the following questions about your background:

Age: _____    Occupation: ______________________________________________________

Gender:   □ Male   □ Female

1. How many hours do you typically use a computer each week? (please tick one)
   □ None   □ 1 to 10 □ 11 to 20 □ 21 to 30 □ 31 to 40 □ 41 or more

2. Have you ever used a tabletop interface before? (if yes, please give details)
   ______________________________________________________________________
   ______________________________________________________________________
   ______________________________________________________________________

3. How would you describe the organization structure of your computer file system (using folders/directories)? (please tick one)
   □ No Organization □ Basic □ Average □ Extensive □ Perfect

4. Have you ever used a tabletop file access interface before?
   □ Not sure   □ No   □ Yes, please explain _________________________________

Please tell the experimenter when you have finished completing the questionnaire.
Thank you for your participation in this study.
E.2 Tutorial Sheet

“The first part of this experiment will allow you to get a basic understanding of Cruiser, our tabletop framework that we are using for this experiment. The Cruiser interface is designed to allow a group of people to sit around a table to share and discuss digital images with each other. I will now go over some of the basic features of the Cruiser interface”

- Experimenter demonstrates the core aspect of Cruiser (Using a standard Cruiser setup without the application manager)
  - Moving Objects on screen
  - Rosize
  - Flipping
  - The Black Hole
  - Using Smart Objects
    - Browser
    - Photo copier
    - Containers
    - Frame
- Introduces the user to Focus and describe its functionality
  - Focus behaves as a search engine where it locates and lists files to a focus file
  - Note of the importance of Dwelling
E.3 Cruiser Guide Usability Study Main Task

Scenario 1 The Travel Guide

Task 1: Loading Travel Guide

The first task of this experiment is to assess whether the application manager interface is easy to use on the tabletop. Imagine that you are organising a Trip travelling through Melbourne to London and Florence. Your friends have provided you with a set of files on the attractions and accommodation from these three cities. Use the Interactive tabletop travel guide to organise your trip.

The table is turned on and you are presented with the start screen. You have decided to use Travel Guide to organise your trip to Melbourne, and you also need a container to store these files. Now start the Travel Guide.

Task 2: Planning London Trip in Travel Guide

Now that Travel Guide has loaded with the Melbourne collection browse this collection to locate four files on Melbourne’s attractions and place them into your container.

You have now acquired a collection of items for Melbourne, and now are ready to plan the next part of your trip. You now need to get a collection of places to go from the London files. Now you need to plan the London part of your trip. Please do that now.

Task 3: Using the Tools

Now that the Travel Guide has reloaded, again browse this collection to locate four files on London’s attractions and place them into your container.

You should notice that the container is getting crowded, so you decide to get another Storage Bin to place your files from London. Please do that now.

Task 4: Planning Florence Trip in Travel Guide

Now you are ready to plan the final part of your trip Florence Italy. You now need to get 6 files of places to go in Florence. You may require an additional Storage Bin for Florence.

Task 5: Finishing up

You have finished. Now get ready for the next task, which will use a different application.
Scenario 2 The Museum Exhibit

Task 1: Loading Museum Guide

Imagine that you returned from a trip to a local museum and took pictures. You would like to review the exhibits you have explored using a tabletop application. You have decided to use the Museum Guide to look at the museum’s collections. You also want to make a collage of exhibits that you like. Go and start the Museum Guide Sydney Sidetracks files.

Task 2: Reloading the Widget

You have now loaded the application and noticed that the files are scattered around the table. You decided to exit out of the Museum Guide, enable the widget only mode and rerun the Museum Guide.

Task 3: Switching Museum Guide to Greek History

Explore the file set, Locate three images of Eternity and two images of the Opera House (can be located through the black and white construction photo), and place them into the Storage Bin. Once you are satisfied, have a look at the Greek History exhibit.

Task 4: Using Tools

Explore the file set, Locate 5 images of Greek vases, place them into the Storage Bin. Now you are ready to make the collage. Take out all the images and place them on the table throwing away any unwanted images into the black hole. Add the frame to the table, arrange the images on the table and take a screen shot.

Task 5: Finishing up

You have finished. Please leave the table in the state that you started in.
E.4 Post-Experiment Questionnaire

Post-Experiment Questionnaire

Please complete the following questions about your experiences using the tabletop’s interfaces. For each of the questions, please mark one box in each line to indicate your response, and explain in further detail (give feedback) when indicated.

1) I was able to perform the requested tasks without any problems:

<table>
<thead>
<tr>
<th>strongly disagree</th>
<th>Disagree</th>
<th>somewhat disagree</th>
<th>neutral</th>
<th>somewhat agree</th>
<th>agree</th>
<th>strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

Please explain:

_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________

2) I was able to find the application that I was looking for, such as the Travel Guide.

<table>
<thead>
<tr>
<th>strongly disagree</th>
<th>Disagree</th>
<th>somewhat disagree</th>
<th>neutral</th>
<th>somewhat agree</th>
<th>agree</th>
<th>strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

Please explain:

_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________

3) The Launcher is easy to understand and I knew what to do, eg. the Travel Guide’s panel:

<table>
<thead>
<tr>
<th>strongly disagree</th>
<th>disagree</th>
<th>somewhat disagree</th>
<th>neutral</th>
<th>somewhat agree</th>
<th>agree</th>
<th>strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

Please explain:

_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
4) The Launcher provides adequate feedback to my actions, eg. buttons react as they should:

<table>
<thead>
<tr>
<th>strongly disagree</th>
<th>disagree</th>
<th>somewhat disagree</th>
<th>neutral</th>
<th>somewhat agree</th>
<th>agree</th>
<th>strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Please explain:

_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________

5) The resize on dwell function is easy to use and intuitive:

<table>
<thead>
<tr>
<th>strongly disagree</th>
<th>disagree</th>
<th>somewhat disagree</th>
<th>neutral</th>
<th>somewhat agree</th>
<th>agree</th>
<th>strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Please explain:

_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________

6) I understood that I need to dwell on the Start, Update and Exit buttons

<table>
<thead>
<tr>
<th>strongly disagree</th>
<th>disagree</th>
<th>somewhat disagree</th>
<th>neutral</th>
<th>somewhat agree</th>
<th>agree</th>
<th>strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Please explain:

_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
7) I understood that I need to tap to select File Sets and Options:

<table>
<thead>
<tr>
<th>strongly disagree</th>
<th>disagree</th>
<th>somewhat disagree</th>
<th>neutral</th>
<th>somewhat agree</th>
<th>agree</th>
<th>strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
</tbody>
</table>

Please explain:

_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________

8) I had no trouble changing the file sets using the Launcher:

<table>
<thead>
<tr>
<th>strongly disagree</th>
<th>disagree</th>
<th>somewhat disagree</th>
<th>neutral</th>
<th>somewhat agree</th>
<th>agree</th>
<th>strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
</tbody>
</table>

Please explain:

_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________

9) I understood how to select the tools I wanted eg. Coppyer or Frame:

<table>
<thead>
<tr>
<th>strongly disagree</th>
<th>disagree</th>
<th>somewhat disagree</th>
<th>neutral</th>
<th>somewhat agree</th>
<th>agree</th>
<th>strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
</tbody>
</table>

Please explain:

_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
10) Which aspects of the interface did you like best for the tasks you were asked to complete?
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________

11) Which aspects of the interface did you dislike for the tasks you were asked to complete?
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________

12) Which aspects of the interface would you change for loading applications?
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
13) Which aspects of the interface would you change for changing file sets?

_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________

14) Do you have any other general comments regarding the tabletop interface?

_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________

Please tell the experimenter when you have finished completing the questionnaire.

Thank you for your participation in this study.