A Fisheye Calendar Interface for PDAs: Providing Overviews for Small Displays

Benjamin B. Bederson
Human-Computer Interaction Laboratory
Computer Science Department,
Institute for Advanced Computer Studies
Univ. of Maryland, College Park, MD 20742
bederson@cs.umd.edu
+1 (301) 405-2764

Mary P. Czerwinski, George G. Robertson
Microsoft Research
One Microsoft Way, Redmond WA 98052
{marycz; ggr}@microsoft.com

ABSTRACT
Calendar applications for small handheld devices such as PDAs are growing in popularity. This led us to develop FishCal, a novel calendar interface for PDAs. It supports users in performing planning and analysis tasks by using a fisheye representation of dates coupled with compact overviews, user control over the visible time period, and integrated search. This enables users to see overviews and to easily navigate the calendar structure, and to discover patterns and outliers.

FishCal was evaluated in a benchmark usability study comparing it to Microsoft's Pocket PC 2002™ calendar. Eleven users performed complex tasks significantly faster and completed them more often with FishCal. Task by task user satisfaction data showed a significant advantage for FishCal as well. A number of usability issues were identified to aid in the iterative refinement of FishCal.

Keywords
Fisheye Views, Information Visualization, Calendar Interfaces, PDAs, Animation, Graphics.

INTRODUCTION
More and more people carry small Personal Digital Assistants (PDAs) with them to help manage day-to-day information. While these devices can be helpful for retrieving relevant information when it is needed, our informal polling of colleagues tells us that they are less helpful for planning and analysis tasks. In particular, we have heard many people complain about existing commercial calendar programs for PDAs.

This is not surprising since these devices have limited screen space, forcing users to jump around through multiple screens, making it harder to relate disparate pieces of information together.

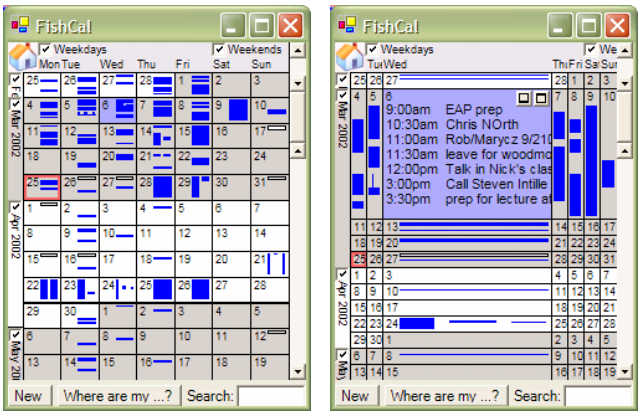

Figure 1: The FishCal interface with the view configured to show 12 weeks. The right view shows the result of tapping on March 6th which focuses on that day. All screenshots in this paper use the appointments that were used in the study described in this paper.

We designed a new calendar interface for PDAs that would support planning and analysis tasks such as "picking a good weekend this spring to go camping", "scheduling my next dentist appointment", or "finding all conflicting appointments in the next three months".

As a secondary goal, we hoped to design a calendar interface that would scale down to smaller devices such as mobile phones, and up to larger devices such as desktop displays. This second goal is important because individuals are likely to access their calendar information from these and other devices. Offering a single interface would give users a consistent user experience, and, eventually, the ability to more readily switch between devices using whichever one is readily accessible.

The FishCal design addresses these goals by using a fisheye distortion technique coupled with carefully designed visualizations and interactions appropriate for a pen-based device and small display (Figure 1). The fisheye visualization lets users see detail in context.

The basic approach starts with an overview of a large time period with a graphical representation of each day's activities. Tapping on any day expands the area representing that day, and reveals the list of appointments in context. Users may change focus days, zoom in further

for a full day view, search for appointments, and reconfigure the viewable space.

This interface shows varying time span displays within the same framework using animated transitions between view changes, and thus, may improve users' ability to maintain a sense of where they are. This paper describes the interface along with the results of a user study comparing FishCal to the traditional Pocket PC calendar interface. Evidence from this study supports our hypothesis.

Related Work

Fisheye distortion techniques, initially called bifocal displays, were introduced by Spence and Apperly 20 years ago [14]. At that time, the basic concept was to distort the information space so focus items were enlarged while peripheral items were shrunk. A few years later, Furnas generalized this approach by suggesting a "degree-of-interest" function [5]. This calculates the relevance of each item in the information space, which is then used to calculate the size and visibility of that item.

Fisheye distortion techniques have been applied to a number of domains, from graphs [12] to trees [8] to menus [3], among others. Their effectiveness has been mixed, but in at least some cases, such as for hierarchically clustered networks [13], fisheye interfaces have been shown to be beneficial to users. The common theme has been that fisheye views are appropriate when users need to see details of some specific items in the context of a large information space.

The idea of using fisheye distortion to view calendars is not new. It was first suggested over ten years ago by Furnas [6] where he described a textual Lisp-based calendar program. We followed the basic approach Furnas created at that time. A tabular display shows days in the calendar, and clicking on individual days causes the amount of space allocated to that day to be increased. Furnas' calendar used varying amounts of space to show different days, so that the focus day was largest, and other days were sized in inverse proportion to the distance from the focus day (although days in the past were always tiny because the assumption was that users were more interested in the future.) This program, while impressive for its time, did not support graphical representations of appointments, searching, or full screen views, and did not have widgets to control which and how many weeks to display. It was not designed with small displays in mind. In addition, it was not evaluated with users, and was not pursued past the publication of the above-mentioned technical report.

While fisheye approaches have not otherwise been used to display calendars, fisheye visualizations have been used successfully to view and interact with tabular information – which is quite relevant, since calendars are typically viewed with tables. The best known example of this is Table Lens, which presents an interface for numerical and categorical tabular data [11]. This visualization approach was designed for tables with many rows, but a modest number of columns. It represents each row with a horizontal bar whose length is proportional to the value of the cell for numerical data, and whose position represents categorical data. The height of each row is scaled to fit the available space. Users may then focus on individual or multiple cells (or rows or columns) by clicking and other interactions. In addition, users can sort rows to help see relationships within the data. While this approach is somewhat similar to the present work in that it uses a fisheye distortion to view tabular data, it is not directly useful for calendar information as it really is designed for spreadsheet style information that has one item per cell, rather than the multiple and possibly conflicting appointments of calendars. In addition, it does not support searching or navigation that calendar users require. Nevertheless, the acceptance of this technique (as demonstrated by its successful commercialization [1]) gives hope that users will be able to understand and navigate calendar information in a tabular format using a fisheye view.

Researchers have also developed other techniques to visualize and interact with calendar information. Plaisant et al. were among the first to develop small visual representations of calendar information [10]. Mackinlay et al. developed a 3D "spiral calendar" visualization [9]. This approach, while not suitable for small devices since it displays several visual representations simultaneously, does have a fisheye-like quality in that it displays detailed appointment information with visual links back to larger scale calendars. So, users can see what week an appointment comes from, what month that week is in, what year that month is in, etc.

Perhaps surprisingly, fisheye techniques have been rarely used for interfaces for PDAs and other devices with small displays. One use was by Staffan et al. who used "flip zooming" to display web pages on a PDA [15]. This consisted of presenting one medium size focus page and several tiny pages that could be used for navigation.

FISHCAL

FishCal is the fisheye-based calendar interface we designed for use on a PDA (Figure 1). It was designed and built at the University of Maryland, and Microsoft Research then joined the project to run the experiment described below.

As described in the related work section, much of the groundwork for this design was laid by a range of earlier work. So, while the individual features of FishCal represent only variations of existing approaches, the primary contribution here is in the integration of a host of techniques to create a novel application that is both usable and useful in an important domain. In addition, we are benchmarking the design against existing calendar software for small devices. We hope that if FishCal is successful, it will illustrate how existing techniques can often be applied in new ways to new domains, and in doing so, advance the state of the art.

FishCal was built to target currently available devices running the Microsoft Pocket PC operating system. These devices are small enough to fit comfortably in a hand, have

high quality 240 x 320 pixel screens, and fast enough processors to support modest animation.

Since FishCal was designed for a pen-based PDA, we have been careful to design the interaction so that it requires minimal text entry and simple interaction. The entire interface can be accessed with just single taps, although dragging offers some modest extra features – including access to tool-tips and fast scrollbar usage.

This rest of this section describes the FishCal interface in detail, including a description of its navigation capabilities, the visualizations that represent calendar information at different sizes, and how search capabilities are integrated into the interface.

Navigation

A fundamental characteristic of FishCal is its ability to support users in easily customizing their view of the calendar. Most commercial calendar applications provide mechanisms to directly switch between day, week, month, and year views, and to change which range of dates are visible with each view. However, the different views are disconnected. One goal of FishCal was to offer the same functionality in terms of a range of views, but to do so in an integrated fashion. Using animation and fisheye distortion, users can see the relationship between the range of dates they are viewing and the previous view. As such, users should not have to expend as much mental effort to manage context and figure out "where they are".

The basic organization of the display is tabular (Figure 1). Each row represents one week, with seven columns representing the days of the week. The number of visible rows can be changed from one (which represents a single week) to 52 (which represents an entire year).

The view can be changed through direct manipulation by interacting with the calendar itself, by manipulating widgets in the periphery of the display, or by using special hardware button shortcuts. One of the challenges was to make it extremely easy to configure the view. The final design only uses interaction mechanisms that most users are familiar with, including tapping on an item that they want more information about, and manipulating familiar buttons and widgets.

Direct Manipulation. FishCal was designed to take advantage of user familiarity with clicking on hyperlinks to find more detailed information about the thing they clicked on. It allows users to tap anywhere on a day to focus on that day, minimizing other days.

Within a focused day (Figure 1 right side), users can tap on the background, or tap on the maximize button to zoom in to a full day view. Or, users can tap on the minimize button to go back to original view with no days focused.

Within the full day view (Figure 2 left side), users can tap on the appointment background or the appointment's maximize button to view the appointment details. Tapping on the day's minimize button returns to the original view, and tapping on the overlapping-windows button returns to the focus day view.

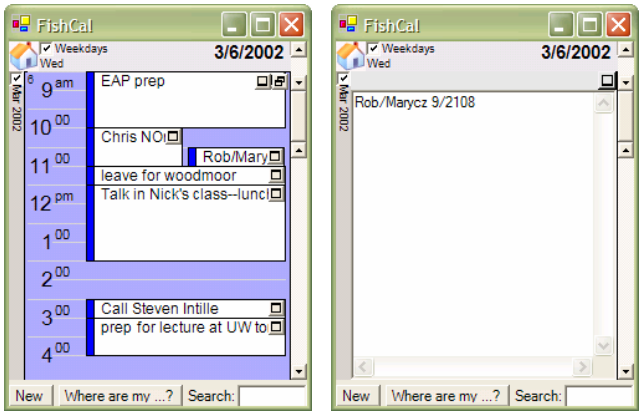

Figure 2: The FishCal interface zoomed in to a full day view (left) and then even further to see the details of a specific appointment (right).

Within the full appointment view (Figure 2 right side), scrolling shows the full contents of the appointment. Tapping on the minimize button returns to the full day view.

Peripheral Widgets. The custom double-headed scrollbar widget on the right side of the display controls how many weeks are visible at a time. It acts like a traditional scrollbar, but the thumb has two additional buttons that are used to manually set the low and high values of the current view. The view dynamically changes as the scrollbar is manipulated, but for efficiency, appointments within days are only shown when the scrollbar is released. Figure 3 shows a range of views as controlled by the scrollbar.

Another way to configure the space is to manipulate checkboxes on the top and left sides of the display. These checkboxes specify whether space gets allocated fully to the correlated set of items, or if those items are minimized. The left side of the display has one checkbox for each month. The top side of the display has one checkbox for weekdays and one checkbox for weekends. Figure 4 shows the result of two different configurations of checkboxes.

There is also a "home" button in the top-left corner of the display that resets all navigation settings to their default, so users can quickly return to the current day with a three month view.

Hardware Buttons. On desktop computers, graphical user interfaces typically offer keyboard shortcuts so that expert users can quickly access commonly used functions. On PDAs, there is no keyboard, but there are special hardware buttons that applications can use for a similar purpose.

When FishCal runs on actual Pocket PC device, the "calendar" button will be used to cycle between the preset views of one day, one week, one month, three months, and one year. The "joystick" (a small 4-way rocker switch) offers motion in four directions, and we plan on using that to move the "active" day which is indicated to the user by a dark blue highlight. Pressing the center of the joystick focuses on that day (or maximizes it if it was already focused). The joystick can be used even when a day is focused or maximized.

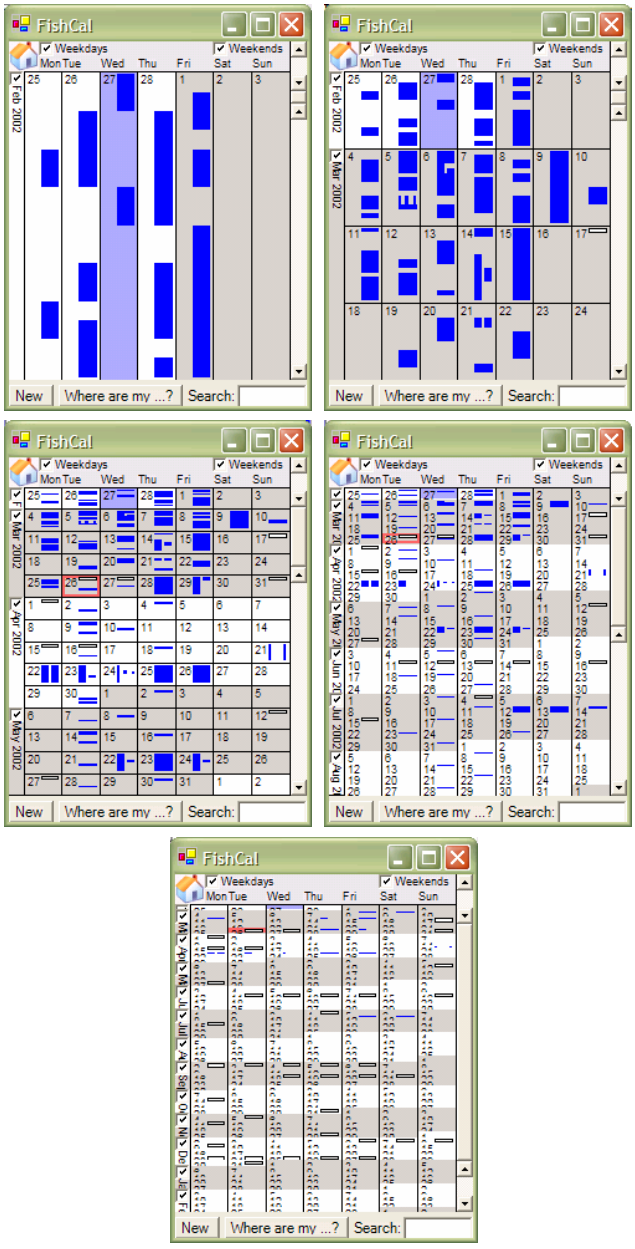

Figure 3: A series of views as the bottom of the scrollbar thumb is dragged downwards to shown in succession from left to right and top to bottom, 1 week, 1 month, 3 month, 6 month, and 1 year views.

For a desktop version of FishCal, we use the keyboard to offer these same shortcuts. The space bar changes between presets, the arrow and enter keys change the active day and zoom in. The escape key zooms out from focused and maximized viewpoints.

Visual Representations

A crucial aspect of the design of FishCal is the visual representation of the calendar for different configurations. We decided to use a "semantic zooming" approach that we developed from our prior work with Zoomable User Interfaces [4]. Semantic zooming means that objects are visually represented differently depending on how much space is available to display them. Using this technique, there are no explicit view modes. Rather, the fisheye

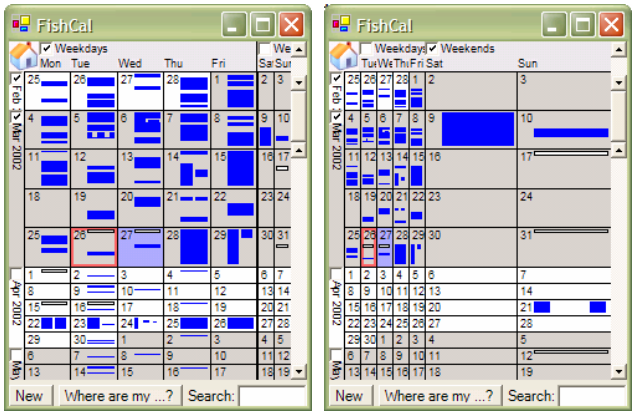

Figure 4: The views resulting from unchecking the April and May checkboxes, and on the left, the weekend checkbox is unchecked, and on the right, the weekday checkbox is unchecked.

distortion algorithms first allocate space, and then each cell renders itself using a view that is appropriate to the available space. The graphical views are scaled to fit the available space, while the textual views use a constant-sized font, and the text is clipped to fit in the available space.

The four views available are:

- *Tiny View*. This shows a graphical representation of the day's appointments. It includes depictions of all-day appointments with a white rectangle at the top of the rectangle. It uses color to represent different appointment types, and it depicts appointment conflicts using multiple columns. The pen can be dragged across appointments to show tool tips with textual information about the appointment under the pen. In large scale views, where each row is thinner than a threshold, the black lines separating rows are removed to make the display less "heavy" (Figure 3 bottom).
- *Agenda View*. This shows a textual list of appointments in order by time. There are actually two representations in this view. If there is a smaller amount of space available, a smaller font is used, and the appointment times are not listed. If there is more space available, a larger font is used, and the appointment times are listed (Figure 1 right).
- *Full Day View*. This shows a traditional full day view with a schedule of the entire day, and appointments positioned at the appropriate times. It shows all-day appointments and conflicting appointments, and uses color in the same way as the tiny view (Figure 2 left).
- *Appointment Detail*. A traditional textbox widget with scrollbars is used to show the detail of a particular appointment (Figure 2 right).

Search

The last primary component of FishCal is search. Search is important because it lets users identify patterns and outliers within a large time span. When users search in FishCal, the days that contain an appointment that match the search criteria are highlighted. The highlights are kept on while

users continue to operate FishCal normally so the space can be explored to understand the results of the search.

In addition to highlighting the visible days within the current view, "attribute mapped scrollbars" [7] show which days are highlighted in both the past and the future (Figure 5). The scrollbar shows indicators representing which days are highlighted within and outside of the current view.

While it is natural to support searching for arbitrary user-entered text strings, that is somewhat problematic because it is notoriously difficult and slow to enter text at all on PDAs. So, while we support free text search, we also support two search mechanisms that do not require text entry: pre-built searches and searches based on existing appointments.

Free Text Search. To search manually, users enter text in the text box in the lower right corner of the display. Days that contain matching appointments along with the scrollbar marks are highlighted incrementally as users enter text.

If all matching dates are outside the current view, FishCal automatically scrolls to show the nearer hits. In addition, the view is automatically expanded (to a maximum of 4 months) to show multiple hits if they are far apart.

A somewhat trickier issue is how to deal with search strings that consist of multiple words. Should the search consist of the conjunction or disjunction of the words, or the actual search string? None of those approaches worked for each of the experimental tasks. Instead, FishCal operates like many current Web search engines, using a simulated "vector" based search.[2; pp. 27-30].

Vector searches work by using a number of characteristics of the search to rank the order in which the results are shown. This results in an ordering that usually matches user expectations. Exact string matches are typically listed first, conjunctions (where all the words match) are listed next, and disjunctions (where not all the words match) are listed last.

FishCal is a little different since it does not present an ordered list of search results, but instead highlights whichever days match. Rather than ordering search results, FishCal, just presents highly ranked search results. It works by first performing an exact string match, and if there are any results, they alone are shown. If there are no results, then it searches for days with appointments that match all the words in the search string, and highlights those days. If there are still no matches, it then searches for days with appointments that match any of the words in the search string and highlights those. This combination of search strategies mimics the main effect of vector searches, and works well in practice.

Predefined Searches. Since it seems likely that many searches by a particular user will be for the same thing, we added support for predefined searches. The goal is to make it even easier to search for commonly sought events, such as travel, meetings, doctor appointments, or holidays.

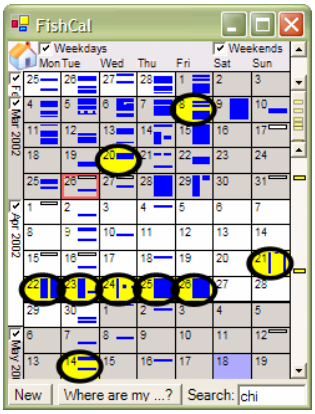

Figure 5: FishCal showing the results of searching for "CHI" (colored highlights are circled for black-and-white printing clarity). A few individual days with CHI-related meetings are highlighted, along with the week associated with the CHI conference. In addition, the scrollbar shows two days in the future that are highlighted, which have appointments for SIGCHI meetings.

A simple approach is to search on appointment metadata which is supported by Pocket PC as well as other calendar systems. The problem with this approach is that most users do not annotate each appointment with categories.

Rather than force users to do something they do not want to do, FishCal takes advantage of what information is already available – the appointment text itself. While there are no guarantees that a user will enter a similar event the same way every time, we have found through informal polling of our colleagues, that people often do represent similar events with similar textual descriptions – although they vary significantly from one user to another.

So we built support for predefined searches where each search would actually look for a match within any of a set of search strings. For example, searching "Doctor Appointments" actually searches for "doctor", "dr.", or "dr appt". While these predefined searches are currently hard-coded, our intention is for users to be able to modify them, or define their own.

This approach has been tested on the authors' calendar data, and it works quite well except for a few idiosyncrasies that we discovered. For instance, one of the authors uses textual graphics such as "->" to indicate travel. Some of these are searchable as a text string, but some are not because they span multiple lines.

Nevertheless, this approach still appears practical. Since having good quality predefined searches is so useful, some users are likely to adapt the way they write appointments to be more consistent. While the general idea of requiring users to adapt to system requirements is undesirable, this is better than the current solution which requires manual annotation of each appointment with categories.

Existing Appointment Search. Since it is quite common for people to create recurring appointments, where the same appointment happens at regular time intervals, it seems

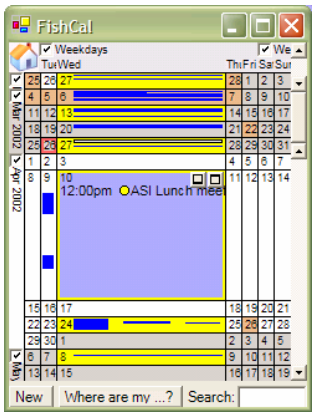

Figure 6: The results of clicking on the appointment "ASI Lunch Meeting". Every other Wednesday is highlighted in yellow which shows a recurring appointment. In addition, several days are highlighted in orange which match either "lunch" or "meeting".

natural to have a simple way to support finding and visualizing those recurrences.

We added one last search feature which is the ability to find all appointments matching an existing appointment. This works just by tapping on any appointment. All other days with exactly matching appointments are highlighted in yellow, just as if the text of the appointment subject had been typed into the search box.

We noticed, however, that sometimes users had similar appointments that were not exact matches. It would be natural to also support finding the relationships between those similar appointments. Based on the implementation of free text searching, we also search for days with appointments that partially match the specified appointment, and we highlight those in orange (Figure 6). This is a simple solution that can be readily ignored by users if they are only interested in the exact recurrence, but offers more information if desired.

Implementation

The implementation of FishCal consists of about 5,000 lines of C#. The most complex part of the implementation is the layout algorithm used to allocate space for each calendar day. The layout algorithm takes as input the number of days in a week, number of weeks displayed, the checkbox states, the focus day, and the size of the window.

The subtle part of the layout algorithm relates to the large set of configurations of the space for which it must work. Specifically, there must be a balance between the minimum size of unfocused cells and the maximum size of focused cells. That is, we have found it makes most sense for each day to stay within a range of sizes whenever possible. So, FishCal defines a preferred minimum and maximum size for unfocused and focused cells, and allocates space within those ranges whenever possible.

The other subtle part of the FishCal implementation is performance. To make FishCal respond to user interaction rapidly, and to animate transitions smoothly, the overall structure had to be carefully designed. The primary things taken into account which contribute to its performance are:

- Custom rendering loop. Rather than use a toolkit, which might have been easier in some respects, FishCal uses a custom data structure, rendering loop, and "picking" implementation. This was particularly appropriate since the basic data structure is a table, and is easily handled as a two dimensional array.
- Space vs. time tradeoff. Things were always precomputed and stored, rather than being computed on the fly. The most obvious place this occurs is in the layout of the days.
- Render only what is needed during transitions. However, some visual aspects, such as highlighted days have to be shown during scrolling since users sometimes look for that while scrolling.

All transitions in FishCal are animated with simple linear interpolation that occurs over 250 milliseconds. We picked such a short animation time because the visual changes are quite small (usually not changing by more than a few centimeters).

FishCal is implemented entirely in C#, and runs on whatever platforms the Microsoft Common Language Runtime (CLR) is available on. Currently, the CLR is available on all desktop versions of Windows except Windows 95. Microsoft has an early version of the CLR available for Pocket PC (called the "Compact Framework"), but at the time of this writing, it is too slow to run FishCal well. While we were able to get FishCal running on Pocket PC, the animations were so slow as to make it unusable. Microsoft has promised a version of the Compact Framework that will be substantially faster, and should be available by the time this paper is published. When FishCal does run reliably on Pocket PC, we plan on making it available for download at http://www.cs.umd.edu/hcil/fishcal. Until that time, we have made a short video of FishCal available at that site.

All features described in this paper are fully implemented. FishCal loads calendar data from a simple text file that is exported from Microsoft Outlook. We also have an experimental version of FishCal that is integrated with Outlook through the Office Add-in architecture. It is launched from a toolbar button, and loads appointment data directly from Outlook.

BENCHMARK STUDY

We performed a benchmark usability study of FishCal compared to the current shipping user interface of Microsoft's Pocket PC 2002™ calendar (Figure 7). The goals of the study were to examine the initial design ideas behind the fisheye calendar, in order to see if the user interface design could be improved, and to compare its overall usability against an existing product.

We gathered eleven knowledge workers (five females) who were all experienced MS Windows and Office users, as confirmed through an in-house validated recruiting

screener questionnaire. Participants were screened to be between 25-50 years of age (average age of 39.2). In addition, the participants fit some broad characteristics of being target end users of personal digital assistants (PDA), but were purposefully chosen to not own or use one at the current time. We thought this aspect of the user group would be especially interesting since for some reason these users had avoided buying a PDA, and perhaps the presentation of PDA information on a small screen was a primary issue for them.

Brief (approximately 5 minutes) tutorials were provided to participants prior to each set of tasks on each calendar. The tutorials consisted of a one page sheet of instructions on operating the interfaces, and the participants then tried each of the described mechanisms. The tutorial focused on the features and functions of each calendar that were necessary for completing the experimental tasks. However, 2 minutes were provided for the user to explore the calendar as he or she saw fit prior to starting. The participants performed an isomorphic set of 11 tasks using each calendar (example tasks are listed below). The order of calendar use and task set for the calendar were both counterbalanced in order to minimize the effects of training, or the possibility of one task set being slightly more difficult than the other. Participants completed a series of calendar viewing and planning tasks, introducing them to progressively more complex questions as they interacted with each calendar. The final task was the most complex, requiring the user to determine the number of conflicts in their calendars over a 3 month period. All tasks were given a deadline of two minutes to complete in order to keep the session under 1.5 hours (and because a two minute deadline seemed reasonable for being able to discover information from one's PDA calendar.) Task times and completion, verbal protocols, and user satisfaction and preference questionnaire data were collected throughout the session. Sessions lasted approximately one and one half hours.

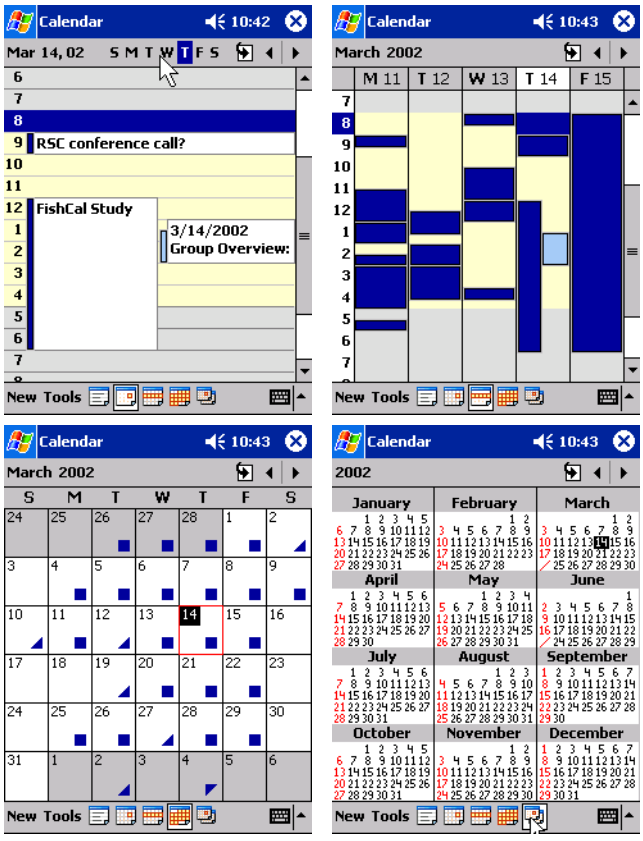

Figure 7: Screen shots of the Microsoft Pocket PC Calendar program that was used in the study showing day, week, month, and year views.

One of the co-author's calendars, seeded with several artificial calendar events for the study, was utilized as the target calendar. Ideally, we would have run both FishCal and Pocket PC Calendar on a Pocket PC device. However, as mentioned previously, the CLR is not yet fast enough on the Pocket PC to run FishCal well. In order to minimize extraneous differences in the study, we ran both calendars on a PC using a mouse and keyboard. The Pocket PC Calendar was run on a Pocket PC emulator and synchronized using Microsoft ActiveSync prior to the study, so that both calendars had the same content.

Participants were asked to carry out a variety of tasks, from finding the dates of specific calendar events (such as visits or trips), to determining how many Mondays a month contained, to viewing all birthdays for the next 3 months. Several tasks focused on finding free time on the calendar in order to schedule events.

The user's display was a LCD set to 1024 x768 resolution with 16-bit color, and each calendar occupied a 240 x 320 pixel window centered on the display (standard Pocket PC resolution). All participants were run singly in a usability lab, on a Dell Pentium 450 MHz computer running Windows XP. A MS Natural keyboard and an MS IntelliMouse were used as input devices, though the "wheel" was not functional with the calendars.

Study Results

Task Times. Task times for one participant were unavailable, as his session expired before he was able to get to the 4th task using FishCal. A tape jam prevented us from obtaining the task times for one other participant for the Pocket PC, and both participants' data had to be ignored for the task time analysis. A 2 (calendar type) x 11 (Task) repeated measures Analysis of Variance (RM-ANOVA) was carried out on the completion times for the tasks. Tasks were performed faster using FishCal (49 seconds versus 55.8 seconds for the Pocket PC, on average), a borderline significant result, $F(1,8)=3.5$, $p=.08$. There was also a significant main effect of task, $F(10,80)=12.9$, $p<.01$, and a significant calendar x task interaction, $F(10,80)=2.05$, $p=.04$. Of particular interest was the fact that, as the tasks became more complex (tasks 3, 5, 8 and 11), the FishCal task time advantage grew. This result was primarily due to the fact that FishCal allowed flexible views across time in a user-defined manner. In addition, the integrated search mechanism and its resultant views made finding particular sets of events via keyword matching quite effective. These results can be seen in Figure 8.

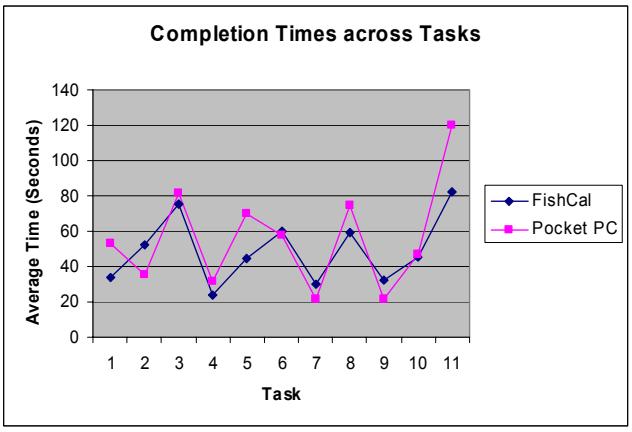

Figure 8: The time spent by study participants to complete each task using the FishCal and standard Pocket PC calendar interfaces.

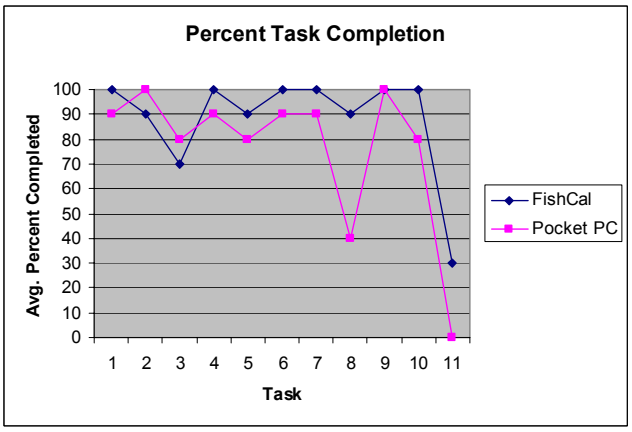

Figure 9: The percent of tasks completed by study participants for each task.

Task Success. The participant who did not complete all of the FishCal tasks was also removed from the Task Success analysis. Tasks were completed successfully significantly more often using FishCal (on average, a 88.2% success rate, versus 76.3% for the Pocket PC), $F(1,9)=37.1$, $p<.001$. In addition, there was a significant main effect of task, $F(10,90)=12.9$, $p<.001$. The interaction was not significant. These data are shown in Figure 9, where it becomes clear that the more difficult and ambiguous tasks (3, 5, 8 and 11) were successfully completed more often with FishCal. This was primarily because the user had the ability to get all the information across a particular time span into one view in order to answer the question. The Pocket PC user was confined to "pre-determined" views (day, week, month and year views), making some of the questions more difficult to answer. In addition, the "find" capability is not integrated into the Calendar application on the Pocket PC, so that if a retrieved calendar event needed to be scrutinized in context more closely, this required additional effort and short-term memory of the date to navigate to in the calendar itself. For the most difficult task (#11), no participant using the Pocket PC completed the task successfully.

Satisfaction and Preference. Users completed "ease of use" ratings on a scale of 1 to 5 (1=very difficult, 5=very easy) after every task. FishCal was rated higher across a majority of the tasks, especially the most difficult task (task 11—how many conflicts are there for the next 3 months?). FishCal was rated higher than the Pocket PC in terms of task by task satisfaction, on average, $F(1, 9)=4.37$, $p=.06$, a borderline significant result. The average task by task ratings are shown in Figure 10.

Usability Issues. Many usability issues were observed with this initial version of FishCal, as well as the Pocket PC calendar, and good design feedback was received from the participants about how best to move toward redesign. For the purposes of this paper, the focus will remain primarily on those issues pertaining to FishCal.

Many users disliked the view of the calendar when more than 6 months were shown at once, claiming that the individual days were simply too small at that point to be useful. In addition, users wanted to see all 24 hours of a day's full view, but the prototype was limited in functionality to simply show a 9-5 view for this iteration of user testing. More importantly, a visualization of search results tried to show as many "hits" in the calendar as possible without making the view so crowded as to be useless. If the result a user was looking for was scrolled out of view (into the future), there was no visual indicator as such (the attribute mapped scrollbar that shows search results was added after the study was run.) Users voiced strong concerns about the readability of text, and being able to set their own default views according to their individual eyesight needs. Users also wanted more control about how their weeks were viewed (e.g., should the week start with Sunday or Monday?). Finally, users wanted better visual indicators of conflicts for both calendars, e.g., red highlights and/or a "conflicts" filter.

Participants completed an overall user satisfaction questionnaire after completing each set of tasks, and again at the end of the session. No significant differences emerged in this satisfaction data, though the Pocket PC was slightly more preferred overall (6 out of 11 participants chose the Pocket PC Calendar; one participant abstained and stated that she wanted features of both calendars in the ideal calendar; 4 participants chose FishCal). Most

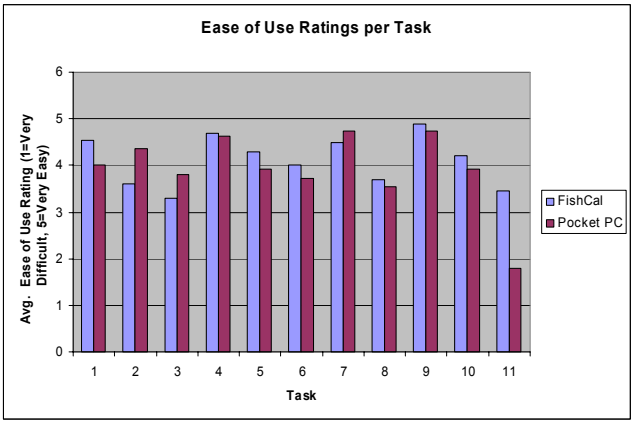

Figure 10: The ease of use rating for each task by study participants.

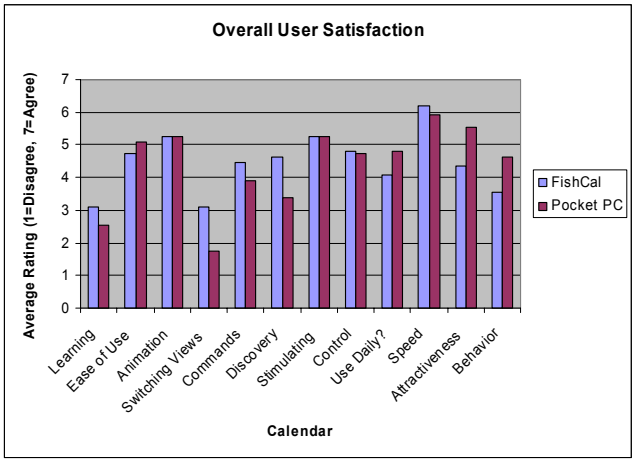

Figure 11: Overall user satisfaction.

participants said that they would prefer a combination of features from each of the two calendars during the post-session debriefing. The most often cited reason for choosing the Pocket PC calendar was the participants' familiarity with the Outlook XP calendar, which is similar in many ways. The overall satisfaction data is shown in Figure 11.

In summary, FishCal performed quite well despite its novelty and this being its first iteration of user testing. The responsivity to direct user manipulation, the ability to create custom views easily on the fly, its clear presentation of conflicts, and integrated search utility were all design innovations that participants thought would be valuable to any calendar used daily for planning and reviewing one's schedule. The Pocket PC calendar was seen by participants to be consistent with other MS calendar products, and this was seen as a plus. Several participants wanted to see a combination of the two calendars taking advantage of the good features of both in a final product.

MONDRIAN BACKGROUNDS

One last thing we did with FishCal was to experiment with making the display a little more fun. After visiting a modern art museum recently, the first author of this paper was inspired by several paintings he saw by Piet Mondrian (1872 – 1944).

We wrote a "Mondrian" mode for FishCal that takes a vectorized version of a Mondrian painting, maps it to the cells and edges of the current FishCal layout. The result is a fully functional Mondrian-style calendar interface. FishCal behaves normally, and the background painting moves and distorts as the user manipulates the calendar.

Admittedly, this is a distracting display that not many people are likely to use during most of their interactions. Nevertheless, we feel that as interface designers, it is important to move past pure function, and to also consider the form of our interfaces. While users are usually focused on efficiency and productivity, this is not always true. Sometimes, even the most serious of users have playful

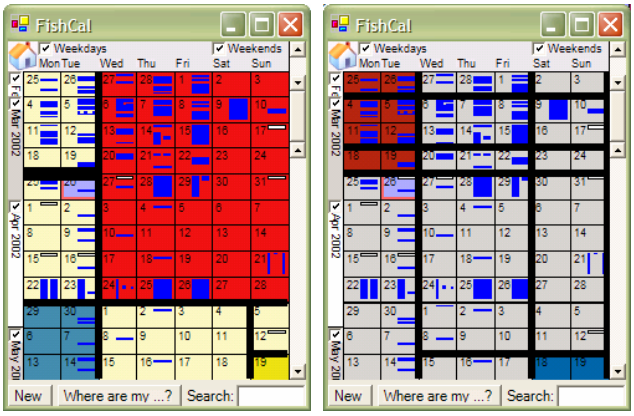

Figure 12: Two views of FishCal in "Mondrian" mode where paintings by Piet Mondrian are mapped to the FishCal display.

moments, and we want to encourage interface designers to support the full range of human activity and interests.

FUTURE WORK

There are several areas of future work for FishCal. From a technical standpoint, we have to integrate FishCal into the applications and devices that people are already using to make it easier for them to switch to an unusual tool for such an important task.

From a design standpoint, a number of usability issues that were found during the user study must be addressed. Naturally, there is also a long list of features that users have asked for that must be looked into as well, such as support for faster data entry. Understanding how these changes affect users, and keeping FishCal easy enough for novice users to feel comfortable with will be an ongoing and crucial challenge.

More studies must be run since it is likely that use of small hand-held devices with pens and touch-screens rather than mice and keyboards will affect usage patterns.

Finally, further design issues are likely to come up when the FishCal interface is applied to smaller devices (such as cell phones) and larger ones (such as desktop displays). While the basic paradigm scales nicely, there are likely to be specific details that need to be changed for different sized displays. Figure 13 shows what FishCal looks like on a large display. While FishCal currently runs as a standalone application on the desktop, we have started integrating it with Microsoft Outlook.

CONCLUSION

We are excited to have revived a useful application of fisheye technology. Given the encouraging results of our first user study, FishCal seems to be a viable competitor to traditional calendar interfaces. However, since managing one's calendar is so important, many users will be cautious about adopting non-traditional interfaces. Thus, one of the biggest remaining challenges is to refine FishCal so that it is appreciated by a broad spectrum of users.

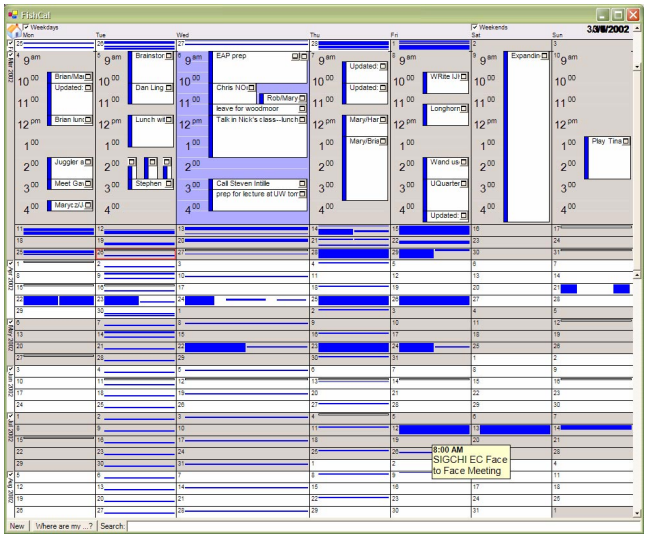

Figure 13: FishCal scaled up to 1024x768 pixels on a desktop computer. There is room to display an entire week with the full day representation, even while showing six months of the calendar.

ACKNOWLEDGEMENTS

We greatly appreciate the comments of our colleagues through the many revisions in the design of FishCal. We also appreciate the efforts of Neema Moraveji who has worked tirelessly to understand the intricacies of making FishCal work with Microsoft Outlook on the desktop. He has been among the first to create a complex .NET add-in for Outlook using MAPI and other undocumented APIs, running into a number of bugs along the way.

Finally, we thank Susan Wilhite for her help in running the user study, and her help along with Ben Shneiderman, Catherine Plaisant and Hilary Hutchinson for their comments on drafts of this paper.

The portion of this work performed at the University of Maryland was funded in part by a generous gift from Microsoft Research.

REFERENCES

[1] Inxight (2002). http://www.inxight.com.

[2] Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. New York: ACM Press.

[3] Bederson, B. B. (2000). Fisheye Menus. *UIST 2000, ACM Symposium on User Interface Software and Technology, CHI Letters, 2*(2), pp. 217-225.

[4] Bederson, B. B., Meyer, J., & Good, L. (2000). Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java. *UIST 2000, ACM Symposium on User Interface Software and Technology, CHI Letters, 2*(2), pp. 171-180.

[5] Furnas, G. W. (1986). Generalized Fisheye Views. *In Proceedings of Human Factors in Computing Systems (CHI 86)* ACM Press, pp. 16-23.

[6] Furnas, G. W. (1991). *The Fisheye Calendar System.* Bellcore, Morristown, NJ.

[7] Hill, W., & Hollan, J. (1994). History-Enriched Digital Objects: Prototypes and Policy Issues. *The Information Society, 10*(2), pp. 139-145.

[8] Lamping, J., Rao, R., & Pirolli, P. (1995). A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. *In Proceedings of Human Factors in Computing Systems (CHI 95)* ACM Press, pp. 401-408.

[9] Mackinlay, J. D., Robertson, G. G., & DeLine, R. (1994). Developing Calendar Visualizers for the Information Visualizer. *In Proceedings of User Interface and Software Technology (UIST 94)* ACM Press, pp. 109-118.

[10] Plaisant, C., & Shneiderman, B. (1992). Scheduling Home Conrol Devices: Design Issues and Usability Evaluation of Four Touchscreen Interfaces. *International Journal for Man-Machine Studies, 36*, pp. 375-393.

[11] Rao, R., & Card, S. K. (1994). The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information. *In Proceedings of Human Factors in Computing Systems (CHI 94)* ACM Press, pp. 318-322.

[12] Sarkar, M., & Brown, M. H. (1992). Graphical Fisheye Views of Graphs. *In Proceedings of Human Factors in Computing Systems (CHI 92)* ACM Press, pp. 83-91.

[13] Schaffer, D., Zuo, Z., Bartram, L., Dill, J., Dubs, S., Greenberg, S., & Roseman, M. (1997). Comparing Fisheye and Full-Zoom Techniques for Navigation of Hierarchically Clustered Networks. *In Proceedings of Graphics Interface (GI 97)* Canadian Information Processing Society, pp. 87-96.

[14] Spence, R., & Apperley, M. (1982). Data Base Navigation: an Office Environment for the Professional. *Behaviour & Information Technology, 1*(1), pp. 43-54.

[15] Staffan, B., Holmquist, L. E., Redström, J., Bretan, I., Danielsson, R., Karlgren, J., & Franzén, K. (1999). WEST: A Web Browser for Small Terminals. *UIST 99, ACM Symposium on User Interface Software and Technology, CHI Letters, 1*(1), pp. 187-196.