

# FOCUS+CONTEXT SKETCHING ON A PDA

A written creative work submitted to the faculty of

San Francisco State University

In partial fulfillment of the

requirements for

The degree

Master of Science

In

Computer Science

by

Son Phan

San Francisco, California

December 2003

# Abstract

Personal Digital Assistants (PDAs) have replaced notepads and paper organizers as the medium of choice for many common scheduling and organizational tasks. This is due, in no small part, to the pervasive nature of PDAs. However, one area where PDAs are only infrequently used is in note-taking situations, such as making notes on a talk or sketching a quick diagram. Undoubtedly one limitation of PDAs is their small screen size and poor resolution. To compensate for these limitations, we describe Fish-i-Pad, a note taking application for PDAs that supports sketching. Fish-i-Pad makes use of focus+context views based on display surface distortion to allow the creation of a note space that is significantly larger than a PDA's display. User testing of Fish-i-Pad against a traditional scrolling interface has supported its promise as a novel technique for improving user satisfaction in sketching applications on PDAs.

# Acknowledgments

I would like to thank Dr. Edward Lank for his insightful guidance through the project, and to S. Carpendale and her assistant, E. Pattison, for the tremendous support of the EPS library. I would also like to acknowledge the assistance of those Computer Science students in user trials.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation for PDA sketching . . . . .	1
1.2	Problems with sketching in PDA . . . . .	3
1.3	Terminology and definitions . . . . .	5
1.4	Structure of this Document . . . . .	6
<b>2</b>	<b>Background</b>	<b>7</b>
2.1	Information Capture on PDAs . . . . .	7
2.2	Introduction to Focus+Context Techniques . . . . .	9
2.3	Display Surface Distortion . . . . .	11
2.3.1	The Drop-off Function . . . . .	13
2.3.2	EPS Elements . . . . .	16
2.4	Summary . . . . .	19
<b>3</b>	<b>Approach &amp; Implementation</b>	<b>20</b>
3.1	Application Specifications . . . . .	21
3.1.1	Design Components . . . . .	21
3.1.2	The Screen Updating Algorithm . . . . .	22
3.1.3	Focus Translation . . . . .	24
3.2	EPS Integration . . . . .	25

3.2.1	Using the EPS Framework for Integrating Focus+Context . . .	25
3.2.2	Using the EPS library . . . . .	27
3.3	Performance issues and Solutions . . . . .	28
3.3.1	Performance issues in our Application . . . . .	28
3.3.2	The Translation Look-up Table . . . . .	29
3.3.3	The GapiDraw Library . . . . .	32
3.4	Test Applications . . . . .	33
3.4.1	Implementing Focus+Context drawing on a PDA . . . . .	33
3.4.2	Fish-i-Pad with Manual Readjustment of Focus . . . . .	34
3.4.3	Fish-i-Pad with Auto Readjustment of Focus . . . . .	36
3.5	Making PDA-based Note Taking a Reality . . . . .	37
<b>4</b>	<b>Test Results</b>	<b>39</b>
4.1	Test Setup . . . . .	39
4.2	Result Analysis . . . . .	40
<b>5</b>	<b>Conclusion</b>	<b>43</b>
	<b>Bibliography</b>	<b>47</b>
	<b>A Survey Tasks</b>	<b>52</b>
	<b>B Survey Questionnaire</b>	<b>54</b>
	<b>C Survey Results</b>	<b>58</b>

# List of Figures

2.1	Focus+Context screen. . . . .	10
2.2	Distortion in 3D space. . . . .	12
2.3	Solving occlusions. . . . .	13
2.4	RVP System and drop-off function. . . . .	14
2.5	Graphs of drop-off functions. . . . .	16
2.6	Types of focus shape. . . . .	18
2.7	Normal transformation vectors vs Viewer-aligned vectors. . . . .	19
3.1	Buffers Relation. . . . .	21
3.2	Focus Translation. . . . .	25
3.3	The symmetry of lens and Look-up table. . . . .	30
3.4	The matrix of pixels before and after EPS distortion. . . . .	31
3.5	Buttons assignment . . . . .	35
3.6	Panning effect in focus area . . . . .	36
3.7	Focus displacement in auto readjustment. . . . .	37
4.1	Responses for the manual Fish-i-Pad. . . . .	41
4.2	Responses for the auto Fish-i-Pad. . . . .	41
5.1	Drawing Tweety with the fisheye effect. . . . .	44
5.2	Dale's portrait and detail callouts . . . . .	45

# Chapter 1

## Introduction

Personal digital assistants, or PDAs, are popular, pervasive devices in the sense that they are highly portable. PDAs include a suite of excellent applications for exchanging information, for organizing and scheduling, and for viewing documents. PDAs still, however, struggle with the capture of information. Consider, for example, note taking. Note taking is a task for which pervasive technology should be well suited, but, beyond simple one-line notes, PDAs are not the medium of choice for note taking. When a user wishes to capture a large amount of information, the medium of choice remains pen and paper. This chapter describes the motivations, current challenges of PDA sketching, as well as some important terminologies.

### 1.1 Motivation for PDA sketching

Ubiquitous computing is a computing paradigm where technology merges into human behavior in such a way as to be unnoticed by its users. In using this term, Mark Weiser, the “father” of ubiquitous computing [30], called the “un-noticed” computer the “disappearing computer”. PDAs are pervasive devices that tie in well to this

vision. They have become an accepted component of daily activity, and thus are an instantiation of the ubiquitous nature of computing technology. Due to their physical attributes (specifically their weight and size), they can be taken anywhere and used anytime. While being extremely portable, they remain a window on the electronic world with all the benefits of electronic information capture such as archivability and sharability. After several generations of development, current PDAs come equipped with sufficient resources and processing power to tackle tasks such as multimedia and gaming that were only previously possible with personal computers.

Because of the pervasive nature of PDAs, developers have replicated important applications from desktop computers to PDAs. Applications implemented on PDAs include web browsing, scheduling, document viewing and editing, and others.

When we consider a typical PDA, we note that the input modality involves the use of a pen. It seems clear that the cognitive model of a PDA for a user is that of a paper notepad and pen which the user could carry with him or her. When we consider typical notepad tasks, such tasks might include scheduling, viewing and editing information, and note taking or sketching. Absent from the PDA platform, however, is an effective note taking or sketching application.

Sketching is a common human activity. Forbus, in his study of sKEA[13], suggests the effectiveness of sketching in knowledge acquisition and in support of communication. He argues that the human natural ability to employ visual symbols, linguistic labeling, spatial representations, arrows, layers, and analogical comparison in a sketch makes it much easier to quickly record information and communicate with others.

Sketching is also useful in unstructured meetings or informal brainstorming sessions [22]. In a recent research conducted to verify the functions of sketching in design, Lught discovers that sketches play a major role in stimulating new thinking directions. Earlier ideas are available in a sketch through its graphical and qualitative nature.

The combination of words and pictures in a sketch allows visualization, revision and reinterpretation of the earlier material.

Another setting for sketching activities with paper and pen is early stage design. Sketching allows individuals and groups to generate ideas faster than current design tools, because it assists human short-term memory by quickly representing ideas [26]. Because the thought process is being captured rapidly, an emerging design can provide feedback to enable designers to generate more concrete and useful information[25].

In the described situations, sketching activities often are serendipitous in nature as well as in final outcome. For instance, a short discussion over lunch of two engineers may result in many interesting ideas jotted down on a napkin. If there is a “digital napkin” to record these ideas, the information can be more valuable and better used. While stationary PCs are not practical in these unplanned circumstances, PDAs can be a convenient means to support these opportunistic encounters. There are, however, some drawbacks to sketching on PDAs. These include the small screen size, low resolution, and inefficient input methods.

## 1.2 Problems with sketching in PDA

Despite the benefit of mobility, the most undeniable disadvantage of a PDA is its small physical display size. Devices such as the Palm Pilot, HP IPAQ, and Sony Clie only provide touchable interfaces of approximately one-eighth the area of a regular paper notepad with a maximum resolution of 240 by 320 pixels. This worsens the problems of eyestrain as compared to a conventional computer screen, and also significantly reduces the available workspace for larger drawings. Even a simple task such as capturing two minutes worth of a lecture could be a daunting experience. Limited screen size also introduces the navigation problem. The visual cues of the extended

context of a drawing are eliminated since only a small amount of text or drawing can be presented at one time. Finding a certain figure in a drawing or a specific line of code within a long Java program are examples of this type of difficulty due to the loss of an awareness of context.

Writing proves even more problematic when performed with the awkward input methods currently available for PDAs. Touch screen devices offer the native tapping QWERTY keyboards, part of the Soft Input Panel (SIP), with which an experienced user achieves a “tapping speed” of approximately 20 wpm. The other major alternative is intelligent handwriting recognition software such as Palm OS’s Graffiti or the equivalent Windows CE block recognizer and letter recognizer. The maximum speed on these applications is up to 40wpm compared to an average of 60 wpm on regular 10-finger keyboard [29]. SIP, under later Windows CE devices, also includes an advanced word and phrase recognizer, the Transcriber, which translates cursive handwriting from anywhere on the screen. Though this is the most impressive advancement in hand recognition, accuracy still affects writing speed. In cases where rough note taking is sufficient, trading off recognition for faster speed and ease of writing is a more sensible choice.

Auditory input has also been extensively researched as an alternative for PDA-based note taking. The use of speech recognition in desktop computer environments and other stationary electronics shows promise. However, due to their portability, mobile devices such as PDAs are exposed to various acoustic environments and social settings that are sometimes inappropriate for this type of input. For instance, a noisy factory is not exactly the place for a QA engineer to issue audible commands to his PDA. There are also specific social situations, when users need to protect privacy of personal information. Finally, certain situations exist that do not allow audio interactions [18]. One would prefer not talking out loud to make personal notes in a

market place, or talking to make notes on a PDA in a quiet study room. Constantly “talking” to a computer is also a notion that is not yet appealing to the general public.

Beyond note taking, sketching is often a task where recognition is not initially desired. Gross notes that the informal nature of a sketch allows a user to delay commitment to an interpretation [25]. This delay is advantageous in discover and design tasks where the final outcome is often not known in advance.

### 1.3 Terminology and definitions

As mentioned before, the display size issue is the most challenging in PDAs since it associated with the very nature that make the device convenient. In fact, no matter what the size of a display is, it never seems large enough to contain the amount of digital information available [31]. Especially with hand held devices, this lack of presentation space, also known as the *screen real estate* problem, leads to navigational and visual difficulty.

*Focus-plus-context* and *detail-in-context* are the most frequently used terms to refer to a presentation technique designed to address the screen real estate problem. These techniques organize a given display into two groups of different detail level, *the focus* and *the context*, to enable a display to host more data than it normally could. The focus area has highly detailed information while the context only provides a global overview of things around the focus. This information overview can be achieved through information filtering, compression or other options[7].

*Fisheye view* [14], though sometimes used as a general term, is a specific variant of the detail-in-context concept that employs distortion. It comes from a photographic term describing a wide-angle lens, which radiantly increases the compression

of imaged objects the further they are from the lens center. *Bifocal view* [27] has a linear separation between regions of two magnification levels. Though not relevant to the research of this project, many other terms, i.e. filtered view, multi-scale view, distortion view, presentation emphasis view, etc., derived from different implementation methods of the original detail-in-context idea [7]. Some of these terms may be used interchangeably in specific situations. However, for the clarification of this writing, only the two original terms are used when referring to a general presentation technique.

## 1.4 Structure of this Document

This project describes a sketch-based application for use on PDAs. The sketch application uses focus-plus-context to visualize a display surface four times the size of a PDA display. By employing a fish-eye lens to distort the display, the application allows users to sketch in an area of high resolution while seeing the entire display surface. A modest user study, conducted at San Francisco State University, lends support to the hypothesis that display surface distortion is an effective method for overcoming the limited resolution of PDAs displays.

This material proceeds as follows. Chapter 2 describes related research in the capture of informal information using PDAs. Focus+context techniques are also introduced. Chapter 3 includes the description of relevant characteristics of our application, the Fish-i-Pad. Chapter 4 analyzes the user study and presents the significant results of the study. Finally, Chapter 5 concludes the material by outlining current contributions and on-going work.

# Chapter 2

## Background

### 2.1 Information Capture on PDAs

Several systems exist that explore the capture of informal, pen-based content on PDAs. Other systems support the visualization of information by providing virtual workspace. Here we outline some recent research in this area.

Though in need of other efficient input methods, PDAs are still appreciated for their natural touch-based interaction. In order to take advantage of this, assisted components have been combined either to enhance the computational power or provide an additional input choice. SmartPad [9] is a mobile pen-based system that falls into the first category. Supported by the processing power of a desktop computer, the SmartPad only acts as an interface to receive ink input from users and a display to present the result of diagram recognition. The system allows people such as field engineers to benefit from device mobility without completely giving up critical computational intensive applications such as CAD.

In PaperPDA [17], Hudson describes the design of a system that supports informal information capture. PaperPDA is a hybrid system that provides note paper as

an input medium. The information is then scanned and partially recognized on a desktop computer. After recognition and “sync-ing”, a new version of the paper-based environment can be printed. All advantageous properties of a paper notebook still exist in parallel with hyperlink and digital search capability. The focus of this project is on gestures recognition of a personal organizer, rather than on generic digital sketching. Both SmartPad and PaperPDA explore good input alternatives but struggle to integrate the capturing and displaying of information. SmartPad is limited in that it only provides a final view of user input. PaperPDA lacks the ability to display information in the same environment that it is captured in. Digital capability such as editability, hyperlinking, and searching is only available on the desktop computer.

NotePals [10], a system supporting capture of group meeting notes, improves on these issues by pairing pen-based input devices with a detail-in-context technique. The users are conceptually aware of their input location with respect to the workspace through a visual indicator in the form of a focus box. As the user selects a location for new information from a PDA screen, the system highlights this insertion location. Notes can then be scribbled at the bottom of the display. As well, shared note taking distributes the workload and allows an enhanced awareness of what others are perceiving. However, the physical separation of input area from resulting output display still causes discontinuity in information visualization. This separation, as noted by Ishii in his ClearBoard project [20], requires users to constantly shift focus from where they are working to a context area on which they are acting.

Extending the work of NotePals, two other systems have used virtual information space to address the screen real estate problem. Combining pen-based input with spatially aware palmtop computers [12], Yee created the Peephole editing system[28] to relate physical movement of a PDA to a virtual information space. The PDA’s small

screen provides a keyhole view of a larger virtual area. The device's physical location is monitored by tracking technology, and used to emulate the effect of scrolling through a fixed world and seeing a small part of the world through a view window. When used as a single unit, this system extends the work space, but the entire workspace cannot be seen. The users must instead keep the information not presented in their memory, thus increasing their cognitive load. Users can also couple the PDA with a separate display unit to assist in visualizing the global picture. Though the distance between displays still requires context switching [20]. Peephole does provide a virtually unlimited workspace in which users can store information.

PebblesDraw, a component of the Pebbles Project by Myers[24], focuses on computer supported cooperative work (CSCW) in situations where PDAs and personal computers are in the same environment. The physical touch screen on a PDA provides a user with access to a remote space (hence considered as virtual from the input area). In group meetings, this multi-machine user interface acts as a “common ground” for discussion through simultaneous input from different PDA terminals. PebblesDraw is similar to the Peephole display, and even NotePals, in that users, again, frequently switch focus between input area and context area to follow the interaction or to conceptually locate themselves in a work space. This is the problem that a focus-plus-context technique, when applied appropriately, can address.

## 2.2 Introduction to Focus+Context Techniques

Focus+context techniques maximize valuable display space to resolve the issue of limited physical display space. Among the earliest literatures in focus + context techniques are the Generalized Fisheye View [14] and the Bifocal Display[27] (see [21] for a review of other techniques). The following represents approaches that are

extensions of these original ideas.

Baudisch's work on Focus+Context Screens [2] merges a detailed LCD monitor, used as focus, into the image from an overhead projector, used as context, to create an amateur two-mode display. It is different from a traditional focus-plus-context technique in that there is no limitation on the size of the display space (except perhaps the limitation of how large an image the LCD display can project). The use of much larger media for context information makes this system suitable for editing large documents with non-distorted views (see figure 2.1 for a description of Baudisch's setup). It works especially well in stationary settings due to the large physical display size.

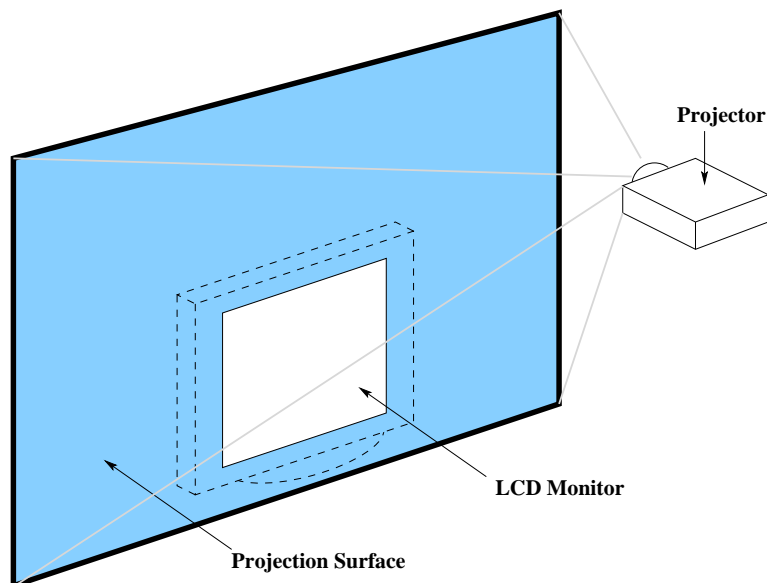


Figure 2.1: Focus+Context screen.

Another approach to focus-plus-context display is through information simplification. Cutrell, by organizing results by categories, page titles, and similar contextual cues on a web directory search engine, guides users closer to relevant results [11].

Without this categorization, the results of search on web directories is almost impossible to interpret. The *WEST* browser [5] uses a text reduction algorithm to enhance readability of the central focus page and surrounding items such as links, images, and other textual information. Users can flip-zoom to browse a simplified and magnified focus page before deciding to switch to a full view of text and graphics. Both systems summarize and cluster information to simplify browsing for the user.

Similarly, by compressing less relevant peripheral information in two dimension, as in DateLens [4] and Fisheye text editor [15], one can also achieve a central focus with much expanded detail on the material of interest. However, to comprehend a daily schedule in DateLens or the zoomed portion of text in the Fisheye text editor, a mental map knowledge of the overall environment is required. This is where the compression technique shows its weakness as the level of context compression quickly increases away from the focus location. If users are not familiar with the entire uncompressed content, it is impossible to extract the information from the highly compressed surroundings.

A fisheye display technique is used in this project to support a drawing application. Many fisheye techniques, summarized by Carpendale [8], were condensed into a unifying framework. With the distinction of data display (presentation) and data interpretation (representation), all fisheye implementations in this framework have the flexibility to revert back to an original condition. This unified framework supports variants of fisheye techniques that enhance the display of a 2D graphical data set.

## 2.3 Display Surface Distortion

To understand the approach of this project, an understanding of display surface distortion and Carpendale's framework is necessary. Consider, first, figure 2.2. In

this figure, we see a three dimensional view of a display space. In order to support detail in context, Carpendale's approach takes a portion of the context and raises it along the z-axis, thus bringing it closer to the viewer.

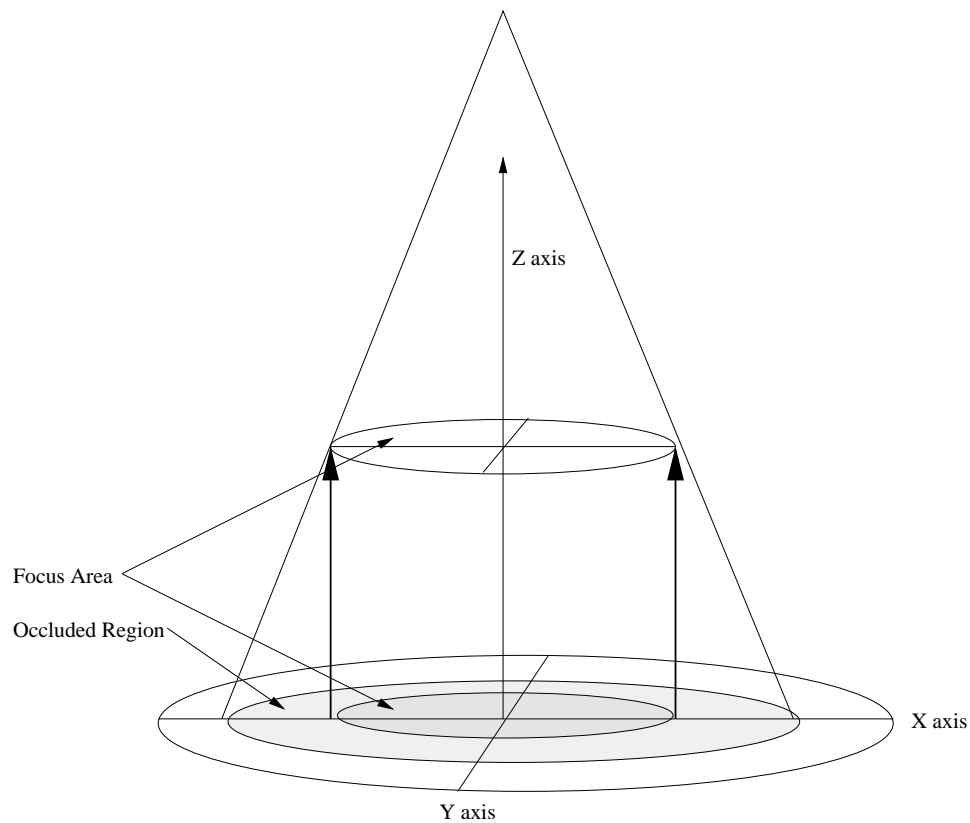


Figure 2.2: Distortion in 3D space.

One problem with raising a component of the display space is that this raised component occludes a portion of the context (see figure 2.2). To deal with this, four possible approaches exist.

1. Ignore the occlusion. This option eliminates local context but preserves more distant context. In sketching tasks, local context is at least as important as distant context.

2. Use partially occluded local context, for example by using a tilted 3D callout (see figure 2.3(a)). This still occludes some local context.
3. Provide linear connection from focus to local context. Diagonal slopes can be used to create small steps that abruptly drop from one height to another (see figure 2.3(b)).
4. Use a mechanism to gradually transition from focus to context. This gradual transition is an “embedding” of focus to context, and embeddings are created through the use of a continuous “drop-off” function.

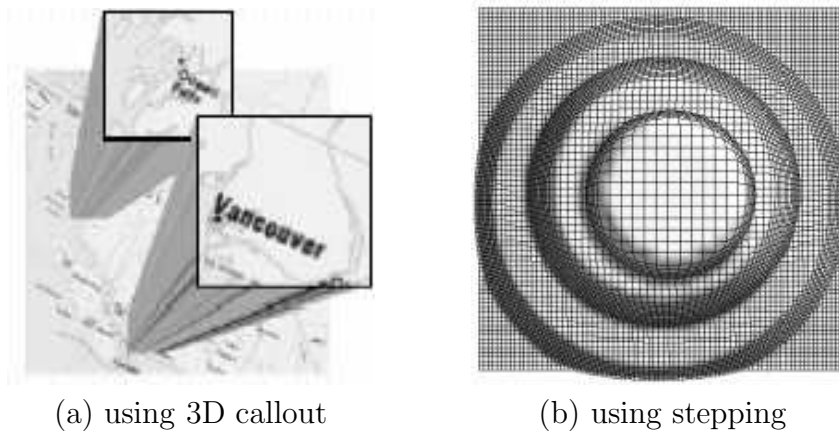


Figure 2.3: Solving occlusions.

### 2.3.1 The Drop-off Function

The EPS’ most important characteristic is its flexibility. In particular, it preserves the ability to always revert to a context view of the entire data set. As opposed to “value operations”, which alter the data of viewing objects, EPS uses “view operations” to manipulate image details to affect the reorganization of information without changing the data [7]. Given a 2D representation, the main tasks of EPS are to transform it

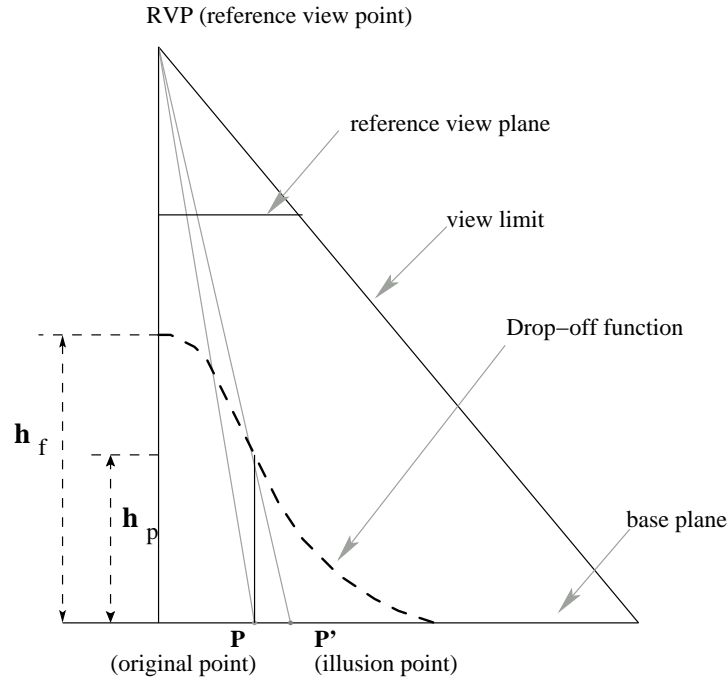


Figure 2.4: RVP System and drop-off function.

into 3D space and to reflect the new view using perspective projection. In EPS, this transformation is done using a set of polynomial curves called drop-off functions.

With a drop-off function, only a specified focus point will be raised to height  $h_f$  (see figure 2.4). Any other point in the base plane will use its distance from the focal point as x-input into the drop-off function to obtain the corresponding height. When the structure is perspectively projected under the reference view point (RVP), both the slope of the bell's surface and the  $z$ -distance, or the field depth of an object from the RVP, create the effect of radiant compression [7]. Since the base plane and view plane are parallel, all geometric relationships (angles, parallelism, and proximity) are reserved. The sidewall smoothly merges the detail focus to context edges at the bottom.

Figure 2.4 shows a point being translated using a drop-off function. Point  $P$  is part

of the information surface that is positioned flat on the base plane. First, it is raised upwards to the height  $h_p$ , and then projected onto the reference view plane. Through the RVP, point P now appears at location P'. This outward motion of points around the focus compresses space to make room for the focus view in EPS visualizations.

Unlike the magnification functions used in previous work by Spence and Apperley [27], and Furnas [14], et al., the drop-off function and numerous variables in the EPS framework support finer control over the magnification of a distorted image. A clever choice of a drop-off function will satisfy most desired features of a detail-in-context method. The followings are some major drop-off functions EPS covers:

- Step. Different region around the focus are lifted up to various heights, which may cause some occlusion of some immediate edge areas. There is little (non-occluding multiple step drop-off) or no transition at all (one step Manhattan drop-off) between the levels of magnification.
- Linear. The magnification drops linearly from full to one, creating the effect of a glass cone or pyramid (depending on the focal shape). Both edges, i.e. the border of the focus and the border of the lens base, clearly show discontinuity.
- Gaussian. The Gaussian function is continuous in both value and derivative adjacent to the focus. A Gaussian drop-off sometimes extends the information beyond its original borders to give the continuous effects at the outside. The inflection point of the curve has the highest level of compression.
- Hyperbolic. This function is a section of a hyperbolic curve. Its values decrease rapidly near the focus and it is only somewhat continuous with the outside of the lens.
- Cosine. Lenses based on this function have the context edge cut straight down

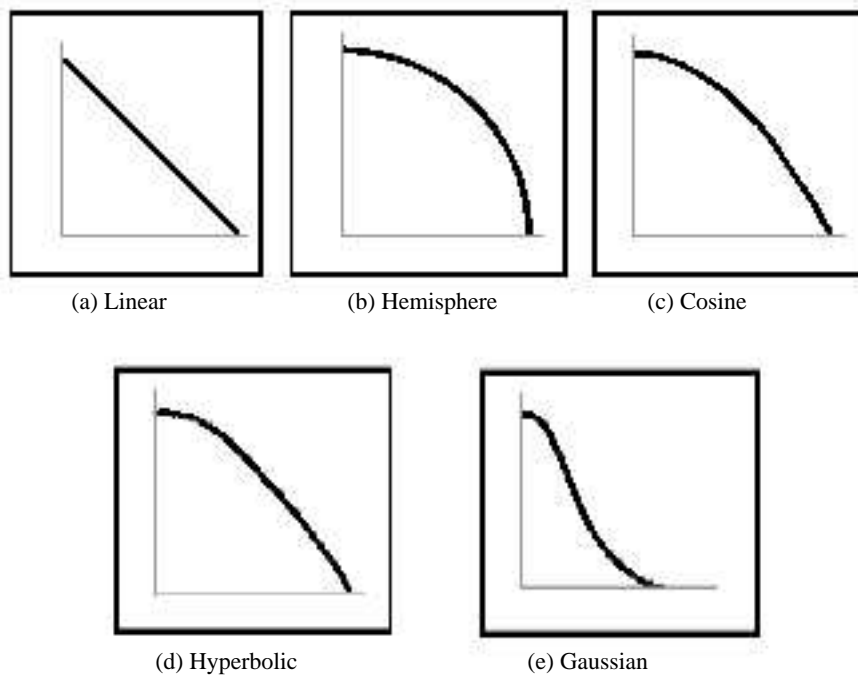


Figure 2.5: Graphs of drop-off functions.

to the lens surroundings. The focus is somewhat continuous.

- Drop Gaussian. Similar to Gaussian drop-off with a more slanted asymptotic edge to cover a larger width. This makes the lens appear bigger and contents on its edge are more noticeable. Connection between edges are smoother in exchange for more compression at both ends.

Figure 2.5 shows the graphs of these functions.

### 2.3.2 EPS Elements

In EPS, the drop-off function is used as a skeleton to allow functions such as zooming and compression. The framework, however, also requires implementation of focal shape, focal height, and focal location.

First, providing options for focal shapes is unquestionably necessary. Though point focus is effective, it is not always sufficient. For instance, when a tourist wants to examine a section of Market Street on a map of San Francisco to locate the nearest Bart station, he or she is particularly interested in the non-distorted portion of the street. Thus EPS provides various choices of focus to generalize the framework. Users can freely choose any focal shape from point, circular, line, polyline, as well as concave and convex polygons [7]. The rule for calculating z-distance still remains as for a point focus. However, for a user-defined region, not only the center but also the entire specified focus area will be lifted to the maximum height. All other locations will use their distance to the closest edge of the chosen area for drop-off function calculation. As a result, the selected area will be magnified without compression, but at the same time, distortion in the transition belt will increase with larger focus shapes (Figure 2.6).

In terms of magnification control, adjusting the focal height alone is increasingly hard as the reference plane approaches the RVP since this ratio of magnification over distance is non-linear. There are two methods to produce distinguishing effects for scaled-only focal regions: a chosen region can either be placed at the focal height or given a maximum height. Placing the focal region at focal height presents a softer edge because it preserves the intended continuity of a drop-off function. Limiting the output values of drop-off functions to a maximum height creates the effect of chopping off the lens' top. The first method was implemented in the EPS user library to produce a more natural appearance of the lens.

Many previous techniques are only applicable to focal regions that are centered at the y-axis. Choosing any other location might put some sections out of view, especially with magnification. This is because the conventional translation vectors, which show the direction of projection, are pointing straight up. However, if those

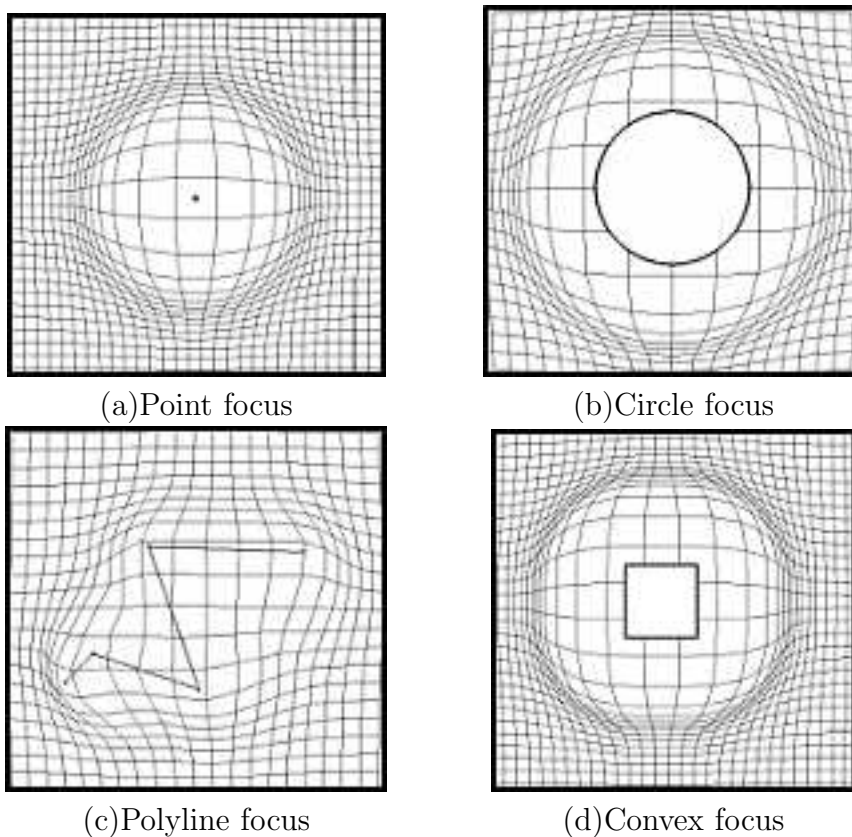


Figure 2.6: Types of focus shape.

vectors are redirected towards the RVP, all compression applied will be lost. Providing freedom of focal location while keeping the magnification effect involves retaining both relative distances from the RVP and surface parallelism [7]. In Figure 2.7, the curve on the right has all of its vectors adjusted to be parallel to the translation vector at the center, known as the viewer-aligned vector. As the focus center move closer to edges of the base plane, the magnified region directed towards the view point (representing human eyes).

There are many other important concepts behind the EPS, including distance metrics, multiple foci, and lens folding; however, the coverage of those concepts is

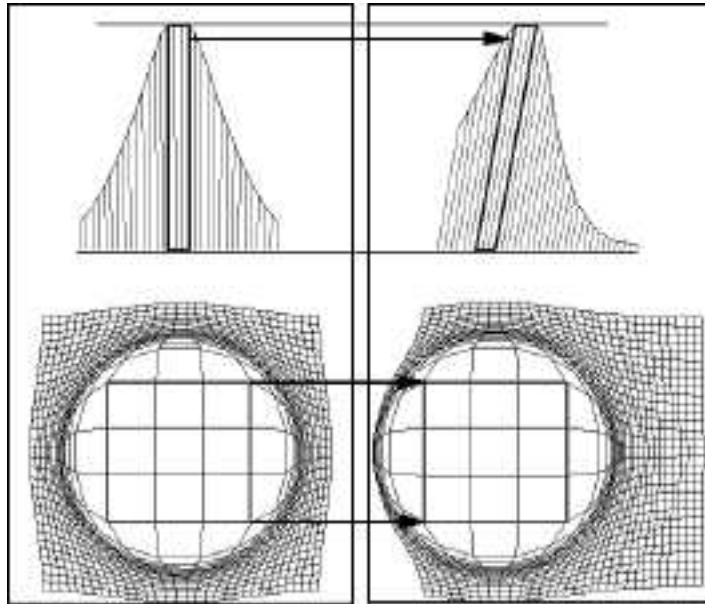


Figure 2.7: Normal transformation vectors vs Viewer-aligned vectors.

beyond the scope of this project. For a complete explanation, interested readers can refer to Carpendale's documents [7, 8].

Notes for figures in this section.<sup>1</sup>

## 2.4 Summary

This chapter has described some drawbacks in current PDA pen-based computing, specifically in capturing information and creating virtual space. Focus+context techniques were also discussed as a general tool in answering the need for screen space. The next chapter explains the details of our project, including the approach and implementation of a focus-plus-context sketch application called Fish-i-Pad.

---

<sup>1</sup>Some figures (2.3, 2.5, 2.6, and 2.7) in section 2.3 were extracted from Carpendale's literature [7], and may have been edited for clarity only.

# Chapter 3

## Approach & Implementation

This project integrates Carpendale's focus + context technique into a drawing application on Pocket PC devices. The application turns a Fisheye focus into a high resolution white board. The context information, computed using the EPS library, is still preserved around the focus area with some compression. By creating a continuous transition between the two levels of detail, the application attempts to make the drawing experience on small screens as natural as possible.

In this chapter, the construction of a focus-plus-context drawing application is described. The first section specifies the design components and logistics for implementing the focus-plus-context drawing surface. In the next section are descriptions of a series of performance enhancements that allow real-time drawing on WindowsCE-based devices. The final section covers the implementation of two prototype applications which are variants of focus-plus-context drawing.

## 3.1 Application Specifications

### 3.1.1 Design Components

As the terminology suggests, a focus-plus-context application has two main presentation components: the focus region and the surrounding context. In our application, the EPS library only creates the context by transforming the location of each individual pixel on the display surface. This transformation is based on expensive floating point calculations, which could make rendering in real-time browsing a time consuming process. To handle frequent updates of the context without making a slow PDA's display flicker, the *Off-screen* technique is utilized. This technique places an intermediate memory buffer, commonly known as back buffer, between a graphic program and the device's output screen. The purpose of this buffer is to temporarily hold a small amount of new display information such as the color change of one pixel. When enough data has been accumulated, the program can dump the entire buffer contents to the screen at once, thus avoiding the flickering problem.

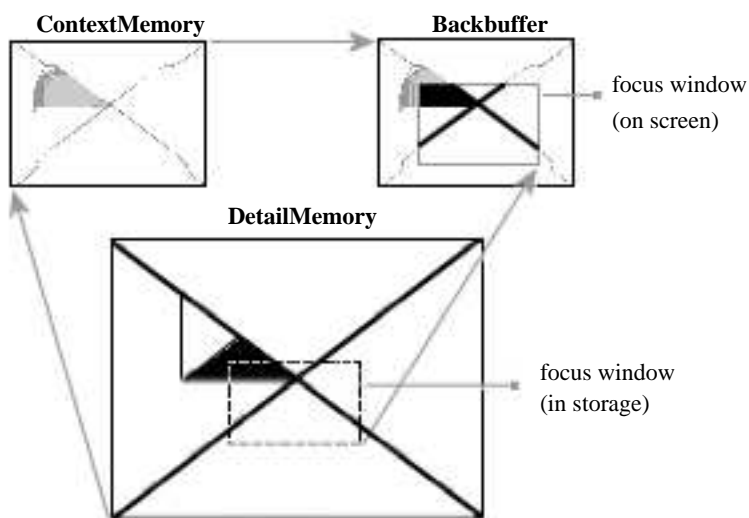


Figure 3.1: Buffers Relation.

Besides the back buffer, the graphic pipeline contains two other major components: the detail memory and the context memory. Working together, they integrate editing capability into the EPS-enabled application. The detail memory is a memory buffer used to keep the most updated snap shot of the complete work area in high resolution. The size of this buffer depends on the magnification value of the EPS lens. When a lens with the magnification value of two is applied on an IPaq screen, the detail memory must be 480-pixels wide and 640-pixels high. The focus window acts as an input peephole that allows a user to draw into the detail memory. Strokes made by users are written into the detail memory after being updated on the screen. On the other hand, the context memory can store the same amount of information as the device display. It contains information in the detail memory that has been reduced in size. This content will later be distorted to support the fisheye visualization. Figure 3.1 illustrates the relation between these buffers.

### 3.1.2 The Screen Updating Algorithm

The updating algorithm for the screen is initiated by some event criteria that requires screen repainting. The event could be movement of the focus window, the reading of a point from the stylus for drawing, or the signal of the stylus lifting up when completing a stroke. These events, while quite different in their effects on the content, are handled similarly by the application. If one is moving the focus window around the screen, there is the need to re-render context with each new lens center and update the focus area. If, however, a user is drawing on the screen, all strokes need to be drawn simultaneously to the back buffer as line segments while the stylus moves so that changes can be reflected immediately on the screen. When there is a pen-up event, or an action that causes the focus to move while a user is editing, the application

copies the entire focus area in the back buffer to the appropriate location in the detail memory (section 3.1.3 shows the details in finding this location). All new data will be saved since the focus is the only editable region.

The algorithm for updating information on the display is as follows:

On Update Events ( oldFocus, newFocus )

1.  $DetailMemory.oldFocus = \mathbf{TranslateFocus}( Backbuffer.oldFocus )$
2. Copy  $Backbuffer.oldScreenFocus \rightarrow DetailMemory.oldDetailFocus$
3.  $\mathbf{Scale}(DetailMemory, 0.5) \rightarrow ContextMemory$
4. For each  $pixel \subset Backbuffer$
5.      $pixel.color = \mathbf{WHITE}$
6. For each  $pixel \subset ContextMemory$  And  $pixel.color \neq \mathbf{WHITE}$
7.      $pixel' = \mathbf{Distort}(pixel)$
8.      $Backbuffer.pixel'.color = \mathbf{BLACK}$
9.  $DetailMemory.newDetailFocus = \mathbf{TranslateFocus}Backbuffer.newFocus$
10.  $DetailMemory.newDetailFocus \rightarrow Backbuffer.newFocus$

The algorithm handles these events based on two focus areas: the *old focus* (the focus before an event) and the *new focus* (the one after an event). The old focus is used in the process of saving users work while the new focus is needed to show the refreshed view. Once the new work has been saved using the old focus, the process of updating the context window begins. To do this, scaling is applied to the dimensions of the detail memory using the factor of magnification, and the result is copied into the context memory. Due to the visualizations used, all non-white pixels need to be distorted. While a number of techniques are possible to support this distortion, this project employs the display distortion technique developed by Carpendale (details of this technique can be found in section 2.3). After each non-white pixel in the context memory is translated into a new position, the point at the translated position in the back buffer is colored. The process of distorting the context into the back buffer is complete when all required pixels are calculated and colored.

The most important step of the algorithm is the embedding the detail area into the context. The premise is that traditional scrolling is ineffective as a drawing

or note taking paradigm on PDAs due to the loss of contextual information when more data is added by the users. To correct this, the application needs to embed the high resolution work area where a user is focused into the low resolution context surrounding the work area. This is done through the back buffer, which, as mentioned earlier, is used as an off-screen painting area to reduce screen flicker. After the back buffer is tiled with the context information, the next step is to translate the new focus in the back buffer to the corresponding “*focus area*” in the detail memory (described below) and load the correct detail information into the back buffer.

### 3.1.3 Focus Translation

In the process of updating focus information between the back buffer and the detail area, either storing or loading, the difference in resolution of the two environments requires an extra translation step. Given a focus bounding rectangle in the back buffer, the goal is to locate the exact rectangle in the detail area to exchange information with.

We notice that, as long as a point stays within the screen limit, translating its coordinates to the detail memory’s coordinate system only requires simple scaling by the factor of the magnification. There are cases where the lens’ center gets close to an edge such that some corners of the magnified focus region (rectangle R1 in Figure 3.2) extends beyond the screen boundary while the areas that it represents (rectangles R2 and R3) are still in bounds. In these situations, the coordinate-scaling method can no longer be used to translate all four corners of the magnified focus area.

We also observe that, while part of the focus area (rectangle R1) is out of bounds, the lens’ center is always within the boundary since a user can not tap beyond the screen. The lens’ center in the detail memory, therefore, can still be determined using coordinate-scaling. For instance, with a magnification of two, a lens center (x,y) in

the back buffer is located at  $(2x,2y)$  in the detail memory. Based on the given focus dimensions, which are specified as input parameters, the bounding rectangle around the focus center in the detail memory can be calculated. After this calculation, previously out-of-bound portions of the on-screen rectangle is eliminated from the bounding rectangle in detail memory to get the required data area.

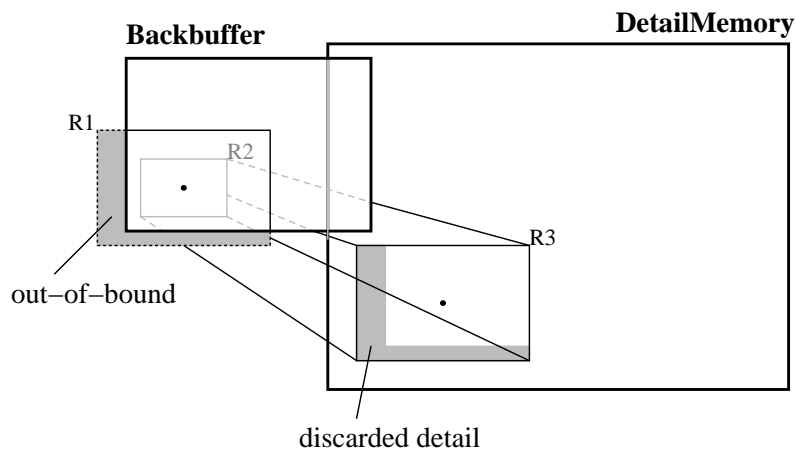


Figure 3.2: Focus Translation.

## 3.2 EPS Integration

### 3.2.1 Using the EPS Framework for Integrating Focus+Context

Carpendale's technique supports a number of different visualizations of focus-plus-context. These visualizations employ "drop-off" functions to embed the focus area seamlessly into the context area. Drop-off functions are polynomial curves that are used to raise points on a base image to various heights depending on their distance from the focal location. Carpendale's library provides eight different mathematical

drop-off functions: Manhattan, Linear, Gaussian, Power, Hyperbolic, Cosine, Cylindrical, and Drop Gaussian. These drop-off functions are described in some detail in Section 2.3.

For PDA-based note taking, several constraints motivate our selection of a drop-off function. First, one goal is to maintain a view of the entire context area. With this view, the users can reduce the required cognitive load by not having to remember where their current location is. Second, it is preferable to have a smooth transition from the work area to the context, making the sketching experience as continuous as possible. For our purposes, this indicates that Gaussian, Drop Gaussian, and Manhattan drop-offs are possibilities for the embedding of our focus into the context.

In Manhattan, there are two distinguish levels of detail. The focus is fully magnified while other areas remain unchanged to provide overview. One drawback is that some parts of the surrounding context are obscured to make room for the foreground focus. Despite the information overlap, the simplicity of this non-distortion based approach can lend speed advantage under the power constraint of a PDA. On the other hand, Drop Gaussian and the original Gaussian offer a better blend between distortion and magnification. Both are visually continuous while the Drop Gaussian has a more gradual asymptotic drop for better edge visualization. Since they create distorted views, 3D translation cost is expensive.

In our note-taking applications, the chosen method was the Gaussian drop-off. Though Drop Gaussian has a higher detail view at the edge, there is more loss of context information to make room for the extra resolution. Since the speed issue can be resolved (described in Section 3.3), Manhattan drop-off is also ruled out due to its missing context behind the lens. Therefore, the Gaussian drop-off is a sensible choice for our note-taking application. This method balances the magnification of detail and the compression of context to give an optimal effect.

### 3.2.2 Using the EPS library

Implementing the Gaussian drop-off function is not necessary. One motivation of Carpendale et al. at the University of Calgary is the design of a library that supports the use of focus-plus-context in applications. As a result, with some assistance from Carpendale and Eric Pattison, a graduate student at the University of Calgary, the library can be recompiled for and ported to various pocket PC architectures.

Using this library involves setting several parameters as follows:

- Lens location: the current screen location of lens center. This is used as the origin to determine the location of pixels in the lens coordinate system before computing their new, distorted positions.
- Focus shape: A rectangle with a longer width is our choice of focus shape to provide ease in writing.
- Focus dimensions: these are the width and height of the lens focus before magnification. The actual size of this focus window on the screen is archived by multiplying original dimensions with the factor of magnification.
- Magnification level: used to provide better detail view and give a larger virtual resolution. We use a magnification of two to increase the screen area by four times.
- Distance Metrics: define the concepts of distance used by drop-off functions. A distance metrics of L-two was used in this application.
- Blend Mode: defines how multiple lenses can be combined to achieve more complex effects. This mode is turned off for increased rendering speed.
- Drop-off function: Gaussian is set for optimized compression effects.

One potential drawback to the Gaussian drop-off is its computational requirements. Many floating point operations are required for the calculation of a Gaussian drop-off. Unfortunately, in many PDAs, floating point operations are implemented using a software-based emulator. The result is a need for some optimization of the application. The next section presents these performance issues in some detail and their respective solutions.

### **3.3 Performance issues and Solutions**

#### **3.3.1 Performance issues in our Application**

One important goal of this project is to provide users with an interactive and responsive interface. This is usually not considered an issue for a simple ink capturing application. However, the integration of a distortion lens requires many additional calculations. The transformation from 2D points of the base image into a 3D perspective system must be performed. Once the 3D view data is generated, 3D coordinates are projected back onto a 2D screen.

While modern Pentium-class computers can handle these translations with ease, Pocket PC processors struggle. There are two primary reasons for this. First, of course, the most powerful Pocket PC device currently offers a clock speed of 400Mhz. This is about three to four times slower than an average desktop processor. The second performance concern arises from the processor architecture itself. These translations are mathematically intensive. They focus on the use of floating point arithmetic to perform matrix multiplication, the most critical operation in 3D rendering. Without hardware support for floating point operations, the floating point simulation unit takes more processing time. It also wastes more of the limited memory resources of PocketPCs.

During the first few versions of the application, there was an extreme penalty for computational operations between frames. For a matrix of 100 by 100 points, each frame refresh took up to a full second for a 1 fps refresh rate. More than eighty percents of this time involved the EPS translation process.

To alleviate these concerns, the application implements two enhancements. The first is a translation look-up table, and the second involves the use of a freeware graphics library for game development.

### **3.3.2 The Translation Look-up Table**

The most powerful capability of the EPS library is its ability to calculate a new pixel position when viewed under a lens type. However, this 2D-3D-2D process involves many floating-point operations, which are only available in many embedded architectures through software emulation. Excessive use of the translation process would pose a tremendous delay in rendering time. Fortunately, there are two characteristic of the application design that allow performance enhancement: the constant magnification, and the symmetry of the lens.

Note, first, that in the application, users are not able to alter lens parameters such as lens size, drop-off function type or magnification level. This means all input variables for each frame calculation stay the same. Our optimization takes advantage of this constant nature of the magnifying lens. In particular, one can pre-calculate a magnification table to hold the translated value of each pixel(x,y). The x-coordinate of a pixel is used to index into the column while its y-coordinate references the row. Pixel coordinates take the lens center as their origin. Thus, to cover all possibilities, it takes a table four times the size of the screen. For instance, in a 320-by-240 screen, when the lens center is at the upper left corner, x-coordinates range from 0 to 320, but when the lens is at the lower right corner, x-coordinates range from -320 to 0.

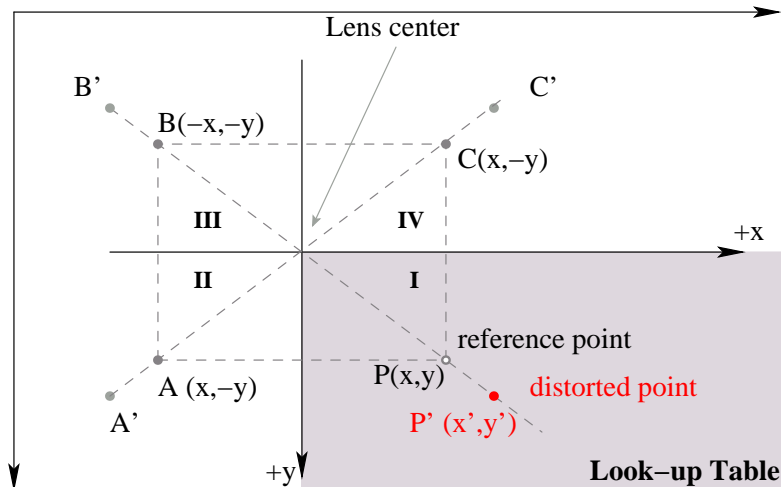


Figure 3.3: The symmetry of lens and Look-up table.

Similarly, the total number of columns for the translation table is 480.

Second, note that the lens effects of the Gaussian drop-off is symmetric if the focal shape itself is symmetric (see Figure 3.3). In that case, similar calculations can be performed in each quadrant. By applying a simple formula to convert coordinates between quadrants, one, in fact, only needs to use a look-up table big enough to hold the number of physical pixels on screen. The table for the above example is now 320-by-240.

In Figure 3.4, point P is extended to its new position  $P'(x', y')$  after being distorted by the EPS library. As mentioned before,  $x$  will be used as the column reference into the translation table and  $y$  will be the row reference. As a result, the entry at row  $y$  column  $x$  in the Translation table contains the location information  $(x', y')$  of point  $P'$ . When there is a need to distort a point that has a location  $(x, y)$ , the table will retrieve the location values from the correct entry.

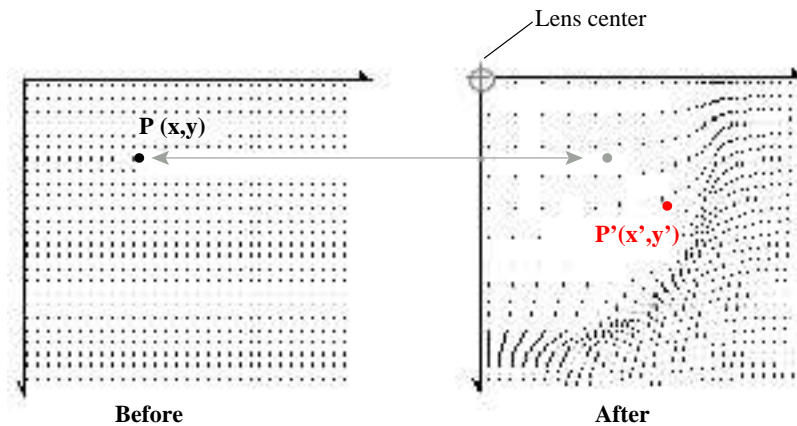


Figure 3.4: The matrix of pixels before and after EPS distortion.

The following algorithm summarizes the process of creating the translation table:

- |  |
|--|
| <ol style="list-style-type: none"> <li>1. TranslationTable = <b>CreateTable</b>(Height,Width)</li> <li>2. For i from 0 to Width</li> <li>3.     For j from 0 to Height</li> <li>4.         TranslationTable[j][i] = <b>EPS.Distort</b>(i,j)</li> </ol> |
|--|

At application loading time, a 320-by-240 table is created and the lens center is set at the upper left corner of the screen. This is because the design uses *Quadrant I* the lower right quarter of the lens' coordinate system as the reference quadrant (illustrated in Figure 3.3). The physical location of each pixel on the PDA screen is now also in lens' coordinates and corresponds to an entry in the table. Next, all pixels can be rendered through EPS and the results are put back into the translation table. At the end of this process, the table has the information to “distort” all possible points in *Quadrant I* regardless of the on-screen lens position.

Now that the entire *Quadrant I* can be supported, a simple translation between quadrants can extend the Fisheye effect over the whole lens. Based on the notion that an EPS lens is symmetric, if it distorts point P into point P', it also extends points A,B,and C further outward from the center to positions A', B', and C' respectively

(see Figure 3.3). For all points outside the reference quadrant, the *Distort* process of the Update Algorithm in Section ?? is as follows:

<p><b><u>Distort(pixel)</u></b></p> <ol style="list-style-type: none"> <li>1. If pixel <math>\subset</math> <i>Quadrant 1</i></li> <li>2.     pixel' = <i>TranslationTable</i>[pixel]</li> <li>3.     Return pixel'</li> <li>4. If pixel <math>\subset</math> QuadX</li> <li>5.     Translate pixel from QuadX to Quad1</li> <li>6.     pixel' = <i>TranslationTable</i>[pixel]</li> <li>7.     Translate pixel' from Quad1 to QuadX</li> <li>8.     Return pixel'</li> </ol>
---

These quadrant conversion of a pixel involves flipping either the x-coordinate (point A), the y-coordinate (point C), or both (point B). A pixel goes through the first translation to be positioned in the reference quadrant. Next, its distorted position is located in the Translation table and translated back to the original quadrant. Through this process, the entire pixel collection of the four quadrants can be distorted without the need to repeat similar calculations each time the lens moves.

With this solution, a matrix of 100 by 100 points in the same test application (mentioned above) achieved a frame rate of around 10 fps.

### 3.3.3 The GapiDraw Library

A drawing rate of 10 fps is a significant improvement over 1 fps. However, the performance of the application remains too slow for real-time motion of the context area.

The reason that drawing remains so slow is that the Windows CE GDI (Graphics Device Interface) routines include large overhead. These routines are optimized for

general Pocket PC applications rather than real-time gaming, or other rapid rendering. Obviously, some mechanism for an “immediate mode” graphics update is required.

While implementing immediate mode graphics rendering is an option, other alternatives exist. The one chosen here was the use of a share-ware library called GapiDraw. GapiDraw provides direct access to a device frame buffer through many optimized graphic tools. The analogy of this concept is similar to that of providing a programmer with lower level hardware command of Assembly language through a C-like set of powerful tools. GapiDraw enables the line and shape drawing functionality of the GDI at the speed of the Windows CE GameAPI.

The result of using GapiDraw is that the frame buffer can be written very rapidly, and the frame rate can be adjusted upwards. In the test application, frame rates now can be set around 80 fps for a seamlessly continuous focus movement.

## **3.4 Test Applications**

### **3.4.1 Implementing Focus+Context drawing on a PDA**

All applications were developed under the following environment:

- Host PC: Windows 2000 Professional, eMbedded Visual C++ 3.0 SDK, and Microsoft ActiveSync 3.5.
- Embedded Device: Microsoft PocketPC 2002 OS and eMbedded Visual C++ 3.0 on the IPAQ 3600/3800/3900.
- Utility packages: EPS framework library compiled for the Arm architecture and the GapiDraw 2.05 package.

All three test IPAQs carry the StrongARM 203Mhz microprocessor. Two were equipped with 64MB of RAM and another with 32MB.

We designed two different focus-plus-context drawing applications. These applications are based on the Gaussian drop-off function described above. The two applications enable users to write and draw in the work area while seeing all their previous work and remaining aware of current location within the document. We have named our sketch applications that incorporate Focus+Context drawing the **Fish-i-Pad**.

The first application involves a focus area with manual readjustment. The application allows a user to pan the current focus window horizontally and vertically by a preset distance. This explicit control of the work window is achieved through the rocker buttons on the bottom of the device.

The study of the first application notes two distinct behaviors. First, when writing, people wish to scroll from left to right on the screen. Second, when drawing, short strokes cause no problem in the focus area. However, drawing longer strokes that extend beyond the editable area will require repositioning. This need to reposition causes discontinuous gestures that make drawing large objects i.e. straight lines, and big rectangles more difficult. Because behavior is so regular, it seems possible to implement an application that would, at least locally, allow a user to avoid short scrolling. Our second application represents an initial attempt at automatically supporting local scrolling of the focus area.

### **3.4.2 Fish-i-Pad with Manual Readjustment of Focus**

In the first application, valuable screen real estate was maximized by using the available hardware buttons instead of directly placing control components on the screen. Users can activate one of three operating modes, Write, Draw, and Erase, by holding

down its designated button while moving the stylus (shown in Figure 3.5). The interface is also oriented 90 degrees counterclockwise to provide easy thumb access to the buttons at the bottom.

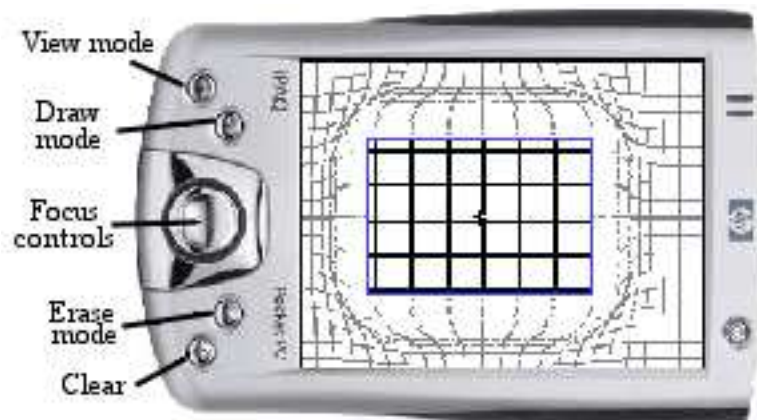


Figure 3.5: Buttons assignment

*Write* mode provides a regular 1-pixel pen trace to let users take notes in a very fine hand writing. The system is in *Write* mode when no buttons are pressed. *View* mode allows users to move the focus around the display with the stylus. *Draw* mode has a thicker inking of 2 pixels to make shapes and figures stand out in the context background. In *Erase* mode, the stylus functions as an eraser. The size of the eraser head is much larger than the pen, making erasing multiple words or large areas faster. There is also a *Clear* option to quickly wipe out the entire drawing surface. To distinguish the modes, the color of the focus frame can change from black (view mode) to blue (write and draw mode), or to red (erase mode).

To simplify the implementation, Fish-i-Pad only permits users to produce inking in the focus area. As the focus area is the only portion where data is presented in high resolution, this restriction is reasonable. The screen always displays a complete

view of a drawing four times larger in area than its physical size. Thus, everything in the context view appears as half of the actual size, with the exception of objects within the focus window. Though focal magnification is fixed, a user can change the focal center in two different ways. One way is to drag the stylus across the screen in view mode. While moving, focus window is always centered under the new stylus position. The second way is to use the rocker buttons. Each time the left, right, up, or down button is push, the focus will slide a short distance to review hidden contents in that direction (see Figure 3.6). Users can also use these controls to scroll to an empty area when running out of space in the work window.

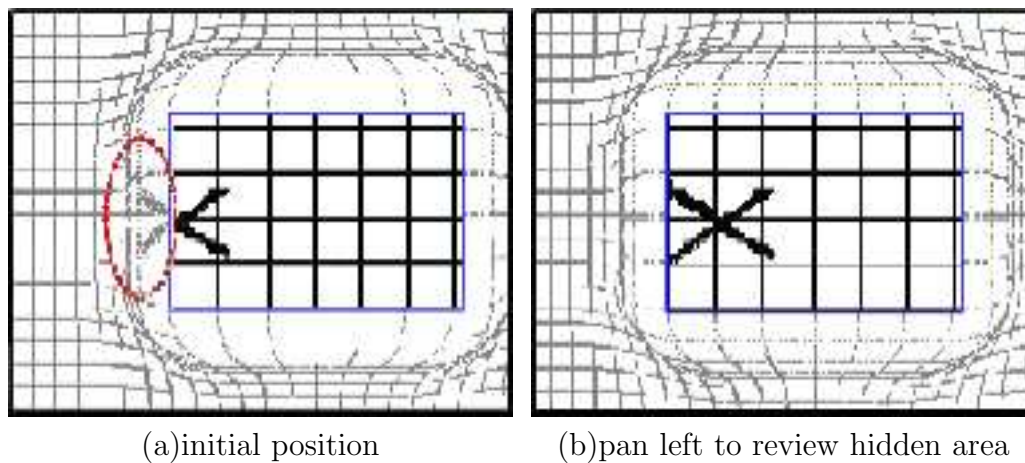


Figure 3.6: Panning effect in focus area

### 3.4.3 Fish-i-Pad with Auto Readjustment of Focus

Palm size PDAs are not extremely small but ergonomics concerns do arise. In particular, prolonged awkward handling, such as keeping the thumb at the rocker for a long period, can cause some discomfort for users. This is a potential problem, since the rocker button was the only way a user can slide the focus window without switching to view mode.

The second version of our Fish-i-Pad application attempts to address this issue by integrating an automatic focus adjustment. Each time the stylus is lifted, the application calculates and moves the focus window by a distance equal to the stylus displacement (see figure 3.7). This method provides a simple approximation that only depends on the pen-down and pen-up position as opposed to its travel distance. Moreover, the Fish-i-Pad Auto behaves differently depending on whether a user is writing or drawing. Naturally, in *Write* mode, the focus window should only move horizontally to the right when a word is completed. In *Draw* mode, the editable window can move freely in all directions, including up, down and backward, that correspond to pen displacement.

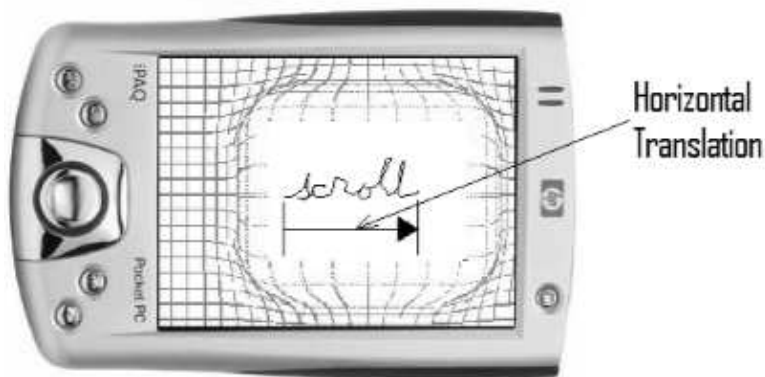


Figure 3.7: Focus displacement in auto readjustment.

### 3.5 Making PDA-based Note Taking a Reality

The applications developed in this project integrate an effective visualization technique. This technique allows PDA users to explore note taking from a brand new perspective. Enhancing a PDA's editing power with fisheye navigation can potentially enable more natural interaction in sketching applications.

In a recent study, Gutwin has tested the effectiveness of Fisheye view in large steering tasks in comparison with non-distorting approaches[16]. He argues that Fisheyes shows great advantages over conventional methods due to its ability to show the entire steering path in the view. The good accuracy of pen movements also contributes in faster finishing time from Fisheye. Most importantly, the paper concludes that Fisheye is a vital option in an interactive application.

To verify this observation for our focus-plus-context application, we obtain data from usability testing for statistical analysis. The fact that users prefer or dislike the application will also help us in shaping the strategy for our future research. Chapter 4 will present the results to validate the application with real users through simple note taking tasks.

# Chapter 4

## Test Results

### 4.1 Test Setup

We conducted a user trial of 35 users. The users were drawn from sophomore, junior and senior students in the Computer Science Department at San Francisco State University. While computer science students are often not good subjects for a general user trial, we were, in this case, looking for users who would be comfortable using PDAs. Unfortunately, we did not have easy access to a community of Pocket PC users. Our assumption is that the substitution of students comfortable with computers would be a good alternative. From the survey's demographic data, 30 of the users said they were familiar with the touch-sensitive interface of PDAs. Responses ranged from "somewhat familiar" to "expert" level.

We implemented a reference application that used scrollbars to navigate a display surface four times the size of the pocket PC screen. The scrolling version has similar functionality to our application.

We then divided our users into two groups. Of the 35 users, 17 users tested the version of Fish-i-Pad that required manual movement of the focus area against the

scrolling-based sketching surface. The other 18 users tested the automatic scrolling version of Fish-i-Pad against the scrollbars. Users were given two sketches to draw with each interface, and they were free to switch back and forth between interfaces as they wished. After completing the task, we administered a post-trial questionnaire (Appendix A and B includes the complete tasks description and questionnaire list).

In the questionnaire, we first asked users to respond to the following two statements:

1. I liked the scrolling interface.
2. I liked the fisheye interface.

An earlier preliminary trial indicated that many users liked both interfaces or liked neither. What we wanted to do was conduct a trial that allowed us to perform a paired t-test on a set of users to measure whether they liked one interface better than the other. Users were asked to indicate on a scale of 5 down to 1 how strongly they agreed or disagreed with the above statements.

## 4.2 Result Analysis

By testing the two versions of Fish-i-Pad separately, we want to minimize the possibility that users reject the novel idea due a specific implemented feature. In this case, the focus window movement is an important feature to consider.

The survey results from the first group (using manual control) are shown in the following histograms.

The histogram in Figure 4.1(a) summarizes the responses for Question 1 and 2. Users appear to show a marked preference for the fisheye over the scrolling interface ( $p = 0.024$  for a one-tail t-test with 16 degrees of freedom). The last question we asked our users was whether they preferred the Fisheye interface over the scrolling

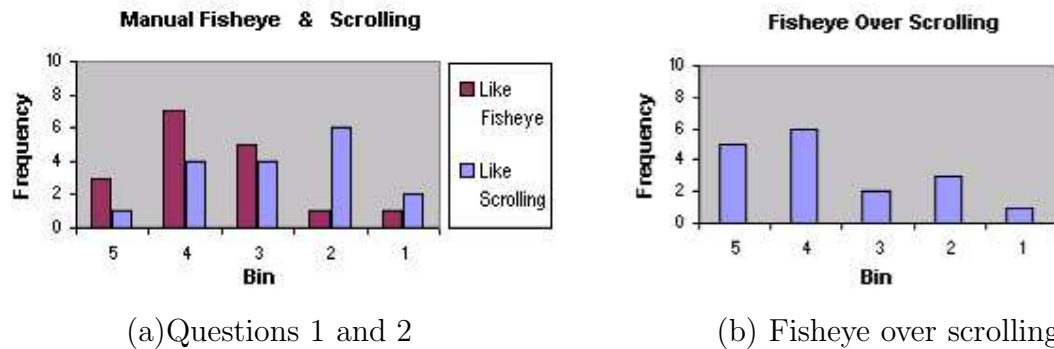


Figure 4.1: Responses for the manual Fish-i-Pad.

interface, and the histogram in Figure 4.1(b) summarizes those results and confirms the results of our t-test. When asked to freely comment on what improvements they would suggest to the interface, six of the seventeen users in our user trial stated that some mechanism for automatic scrolling would be a nice addition to the interface.

Our analysis of the fisheye with automatic movement presents, surprisingly, the opposite result. Figure 4.2(a) summarizes the responses to Question 1 and 2 in the auto Fish-i-Pad. Only sixteen of our eighteen respondents completed the survey, so we eliminated the two non-respondents.

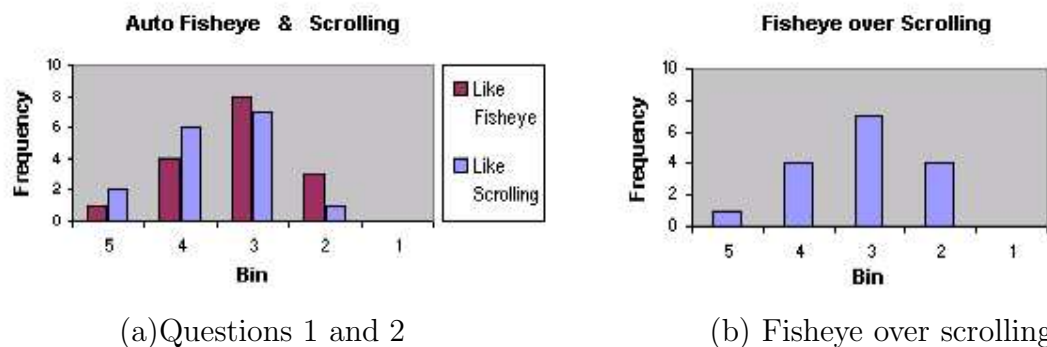


Figure 4.2: Responses for the auto Fish-i-Pad.

When we conducted our paired t-test on the above respondents, we conclude that

we do not have enough evidence to support a claim that users prefer the interface with automatic motion or the scrolling interface. Both distributions are remarkably similar, and the number of people who liked automatic versus scrolling was about equally split. The response to the last question on the survey supports the lukewarm response to the interface with automatic scrolling (Figure 4.2(b) ). The automatic movement of the fisheye is clearly at fault in the trials.

The movement on pen lift is annoying to users, and many commented on the difficulty involved in crossing a “t”, dotting an “i”, or drawing multiple strokes in the detail area. The automatic movement of the fisheye makes it difficult for users to target letters or locations ballistically. Users, instead, must stop and wait for the display to move between strokes. It seems clear that the pitfalls of Do What I Mean, or DWIM, are significant in this application. Unless automatic motion works perfectly, it does disrupt a user’s workflow in sketching applications.

# Chapter 5

## Conclusion

PDA's and other portable devices have become an important part of ubiquitous computing. Their small size allows their use in a wide variety of situations not suited to traditional desktop applications. One such situation is the use of a computer in support of spontaneous meetings. Those situations might required the use of sketching, a natural information capture method that has not yet been carefully considered for PDA's.

The small physical display size of a PDA is the main limitation that prevents the use of sketching on PDA's. To overcome this limitation, a prototype application, Fish-i-Pad, implements focus-plus-context sketching on PDA's. The focus serves as a high-resolution work area, and the context presents an overall view of a user's work. The focus-plus-context technique achieves the main goals of this project in that it virtually enlarges the sketch area, maintains a user's context awareness, and allows fine pen-based drawing and editing on a small PDA screen.

Feedback from our user trial indicates a significant user preference for focus-plus-context sketching. The users particularly prefer the ability to edit small detail and, at the same time, see the entire drawing. We have provided our prototype application

to a number of users, and it is freely available for download from the web. The application enables very detailed sketching on the HP IPaq display.

The fine detail enabled in sketching is clearly evident in the following images. Figure 5.1 demonstrate the effects of Fisheye as the user is sketching (note the finished image in the callout). Figure 5.2 displays some fine detail including a signature on the samurai's sword and an initial on his shirt.<sup>1</sup>

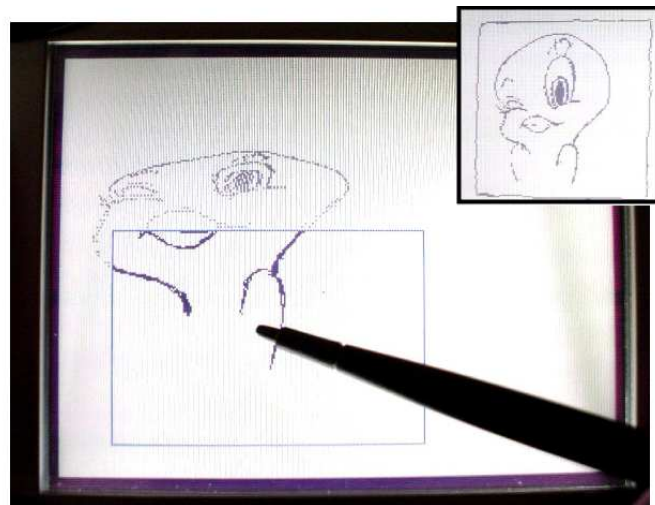


Figure 5.1: Drawing Tweety with the fisheye effect.

There were also some less successful, though promising, experiments with automatic repositioning of the focus region. On-going work is seeking to address some of the current problems with controlling the focus movement.

---

<sup>1</sup>Figure 5.1 and 5.2 were taken while the author sketches cartoon characters. Digital snapshots were not altered in way other than brightness adjustment and grouping for presentation purposes.



Figure 5.2: Dale's portrait and detail callouts

Our user trial indicated a desire on the part of users for automatic repositioning of the focus area during drawing and sketching. One variant of the applications represent an unsuccessful, though promising, attempt at realizing automatic repositioning.

One way to address the issue users had, i.e. difficulty in targeting, is to build a small delay into the focus motion. Another enhancement, suggested by some users, is to provide a modeless way in moving the focus area. Specifically, a small region at one corner of the focus area could be used to drag the entire focus window around. This eliminates the need of pressing down a button for view mode and gives users more freedom in writing and navigating.

An obvious next step in application development is the implementation of an application that uses the fisheye drawing technique developed here. One such application might be a note taking tool for Smart Classroom. Students can use this application to acquire notes and diagrams from others in the class, to annotate the content to their understanding and to store those information for personal reference. Fisheye drawing can also be useful as a UML sketching and recognition tool. One can concentrate on creating parts of a large diagram using the high-resolution focus. As

the focus window moves, the newly entered data is translated into predefined symbols by some recognition algorithms before being merged into the context. Beyond the educational settings, fisheye sketching applications suite well for artists whose work is often inspired spontaneously. With some enhanced drawing tips, a fisheye sketchpad can assist artists in capturing their creative thoughts anywhere they go.

This project contributes to the initial development of note taking applications based on the focus-plus-context techniques. It evaluates available options in software tools to make the initial application functional and practical. Some necessary performance enhancements are suggested in the process of building a prototype fisheye notepad. Finally, the results of this project indicate that focus-plus-context techniques improves user satisfaction in sketching applications on PDAs.

# Bibliography

- [1] Avrahami, D., Hudson, S., Moran, T., and Williams, B. Guided Gesture Support in the Paper PDA. Proceedings of the 14th annual ACM symposium on User interface software and technology, p.197-198. Orlando, FL, 2001.
- [2] Baudisch, P., Good, N., and Stewart, P. Focus Plus Context Screens: Combining Display Technology with Visualization Techniques. Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology, p31-40. Orlando, FL, 2001.
- [3] Baudisch, P., Good, N., Bellotti, V., and Schraedley, P. Keeping Things in Context: A Comparative Evaluation of Focus Plus Context Screens, Overviews, and Zooming. Proceedings of the SIGCHI conference on Human factors in computing systems, p259-266. Minneapolis, MN, 2002.
- [4] Bederson, B., Clamage, A., Czerwinski, M., and Robertson, G. A Fisheye Calendar Interface for PDAs: Providing Overviews for Small Displays. CHI '03 extended abstracts on Human factors in computer systems, p618-619. Ft. Lauderdale, FL, 2003.
- [5] Bjork, S., Bretan, I., Danielsson, R., and Karlgren, J. WEST: A Web Browser for Small Terminals. Proceedings of the 12th annual ACM symposium on User interface software and technology, p187-196. Asheville, North Carolina, 1999.

- [6] Bjork, S., Holmquist, L., Ljungstrand, P., and Redstrom, J. PowerView: Structured Access to Integrated Information on Small Screens. CHI '00 Extended Abstracts on Human Factors in Computer Systems, p265-266. The Hague, The Netherlands, 2000.
- [7] Carpendale, S. A Framework for Elastic Presentation Space, Ph.D. thesis. School of Computing Science at Simon Fraser University. March 1999.
- [8] Carpendale, S., and Montagnese, C. A Framework for Unifying Presentation Space. Symposium on User Interface Software and Technology, p61-70. Orlando, FL, 2001.
- [9] Citrin, W., and Gross, M. Distributed Architectures for Pen-Based Input and Diagram Recognition. Proceedings of the Workshop on Advanced Visual Interfaces, p.132-140. Gubbio, Italy, 1996.
- [10] Davis, R.C., Brotherton, J. A., Landay, J. A., Price, M. N., and Schilit, B. N. NotePals: Lightweight Note Taking by the Group, for the Group. CS Division, EECS Department, UC Berkeley, Berkeley, CA CSD-98-997, February 1998.
- [11] Dumais, S., Cutrell, E., and Chen, H. Optimizing Search by Showing Results in Context. Proceedings of the SIGCHI conference on Human factors in computing systems, p.277-284. Seattle, WA, 2001.
- [12] Fitzmaurice, G. Situated Information Spaces and Spatially Aware Palmtop Computers. Communications of the ACM, vol.36, no.7, p.38-49. July, 1993.
- [13] Forbus, K., and Usher, J. Sketching for Knowledge Capture: A Progress Report. Proceedings of the 7th International Conference on Intelligent User Interfaces, p.71-77. San Francisco, CA, 2002.

- [14] Furnas, G. Generalized Fisheye Views. In Proceedings of CHI 86, ACM. p16-23, NewYork, NY, 1986.
- [15] Greenberg, S. A Fisheye Text Editor for Relaxed-WYSIWIS Groupware. Conference companion on Human Factors in Computing Systems: Common Ground, p212-213. BC, Canada, 1996.
- [16] Gutwin, C., Skipik, A. Fisheye Views are Good for Large Steering Tasks. Proceedings of The Conference on Human Factors in Computing Systems. p201-208. Ft. Lauderdale, FL, 2003.
- [17] Heiner, J., Hudson, S, and Tanaka, K. Linking and Messaging from Real Paper in the Paper PDA. Proceedings of the 12th annual ACM symposium on User interface software and technology, p.179-186. Asheville, NC, 1999.
- [18] Hindus, D., Arons, B., Stifelman, L., Gaver, B., Mynatt, E., and Back, M. Designing Auditory Interactions for PDAs. Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology, p143-146, Pittsburgh, PA, 1995.
- [19] Holmquist, L., and Ahlberg, C. Flip Zooming: A Practical Focus + Context Approach to Visualizing Large Information Sets. In Proc. HCI International 97. p763-766, Elsevier, Amsterdam, 1997.
- [20] Ishii, H., and Kobayashi, M. ClearBoard: A Seamless Medium for Shared Drawing and Conversation with Eye Contact. Proceedings of the SIGCHI conference on Human factors in computing systems, p.525-532. Monterey, CA, 1992.
- [21] Leung, Y., and Apperley, M. A Review and Taxonomy of Distortion-Oriented Presentation Techniques. ACM Transactions on Computer-Human Interaction. Vol.1, No.2. p.126-160. June 1994.

- [22] Lugt, R. Functions of Sketching in Design Idea Generation Meetings. Proceedings of the fourth conference on Creativity and Cognition, p.72-79. Loughborough, UK, 2002.
- [23] MacKay, B. The Gateway: A Navigation Technique for Migrating to Small Screens. Doctoral Consortium Submission, CHI 03 Extended Abstracts on Human Factors in Computer Systems. p684-685, April 2003.
- [24] Myers, B. Using Handhelds and PCs Together. Communications of the ACM. Vol.44, No.11, p34-41. November 2001.
- [25] Nakakoji, K., Gross, M., Candy, L., and Edmonds, E. Tools, Conceptual Frameworks, and Empirical Studies for Early Stages of Design. CHI '01 extended abstracts on Human factors in computer systems, p.493-494. Seattle, WA, 2001.
- [26] Plimmer, B., and Apperley, M. Computer-Aided Sketching to Capture Preliminary Design. Australian Computer Science Communications , Third Australasian conference on User interfaces - Vol.24, Iss. 4, p.9-12. Melbourne, Australia, 2001.
- [27] Spence, R., Apperley, M., and Tzavaras, I. A bifocal display technique for data presentation. In Proceeding of Eurographics '82 p.27-43. 1982.
- [28] Yee, K. Peephole Displays: Pen Interaction on Spatially Aware Handheld Computers. Proceedings of the Conference on Human Factors in Computing Systems. p.1-8, 2003.
- [29] Ward, J. D., Balckwell, A. F, and MacKay, D.C. Dasher - a Data Entry Interface Using Continuous Gestures and Language Models. In Proceedings, ACM Symposium on User Interface Software and Technology, 2000.

- [30] Weiser, M. Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM*, Vol. 36, Iss. 7, p.75-84. Palo Alto, CA, 1993.
- [31] Zenda, A., Carpendale, S., and Rounding, M. On the Effects of Viewing Cues in Comprehending Distortions. *Proceedings of the Second Nordic Conference on Human-computer Interaction*,p.119-128. Aarhus, Denmark,2002.

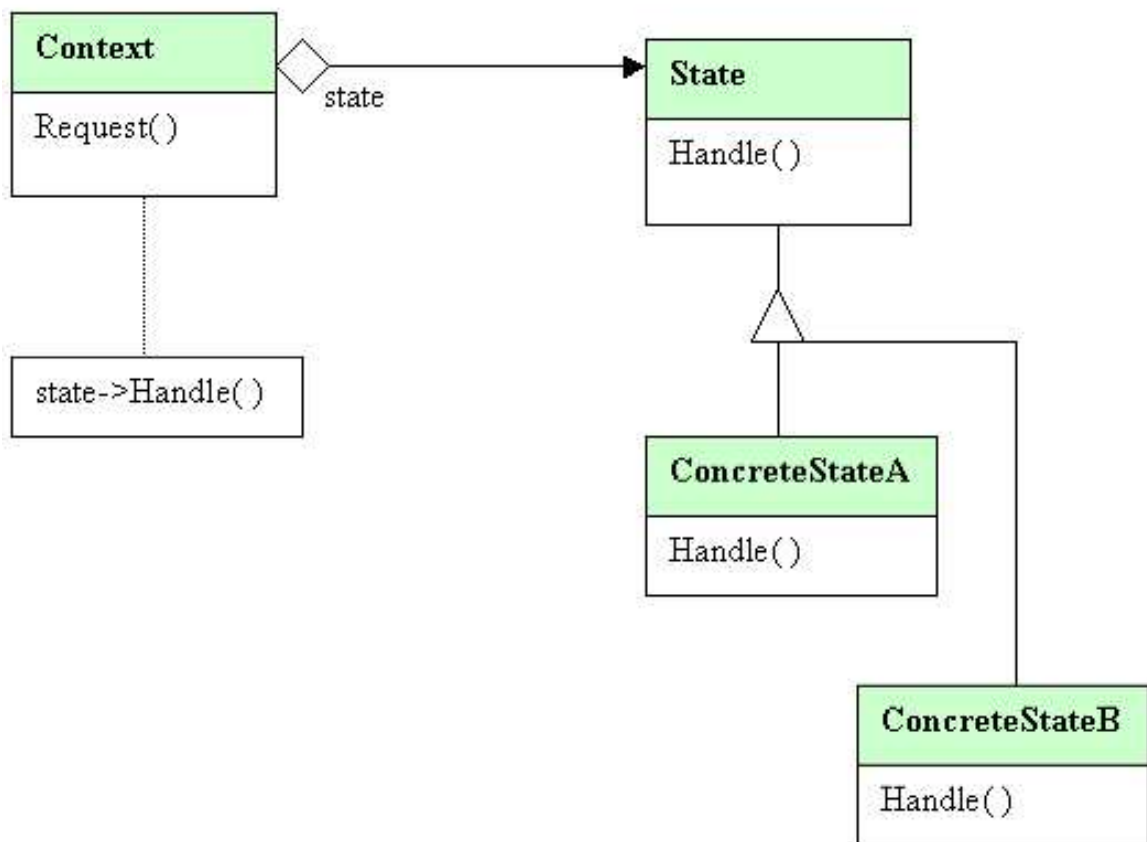
# Appendix A

## Survey Tasks

1. Enter the following paragraph into your PDA:

*“ At the beginning of the industrial age, transportation, and raw materials had been the two important factors that decided the location of a factory. However, as the revolution spreads heavy industries throughout most of the developed countries..”*

2. Reproduce the diagram on your PDA screen:



# Appendix B

## Survey Questionnaire

General questions: Please read and answer the following questions to your best knowledge.

What is your age?

- 18-25
- 26-35
- 36-45
- 46 up

What is your gender?

- male
- female

What is your year in the Computer Science program?

- freshman
- sophomore

- junior

- senior

- graduate

How often do you handwrite your notes?

- Never

- Sometimes

- Often

- Always

What kind of PDA do you own?

- Pocket PC (Compaq IPAQ, Dell Axim, etc.)

- Palm OS (PalmTungsten, Palm Zire, Sony Clie)

- Other (Sharp Zaurus, etc..)

- I don't own one.

How familiar are you with PDA touch screen, stylus interface?

- Never seen one

- Somewhat familiar

- Familiar

- Daily usage

- Expert with it

Have you ever take notes using a touch screen device?

- Yes
  
- No

Software Evaluation: For the following statements, use the provided scale:

1 = Strongly disagree; 2 = Disagree ; 3 = Neutral; 4 = Agree; 5 = Strongly agree

1. I like the scrolling interface.
2. I like the fisheye interface.
3. I found the auto-motion in Fisheye interface natural.
4. It is helpful to see the entire screen while drawing large objects.
5. It is helpful to see the entire screen while drawing small objects.
6. It is easier to draw large items with Fisheye than Scrolling.
7. It is easier to draw small items with Fisheye than Scrolling.
8. Navigating from top-left to bottom-right is faster in Fisheye.
9. It is easy to locate an item on the screen with Fisheye.
10. Consider having slower movement of focus window in the Fisheye, in my opinion, this would improve the interface.
11. I prefer the Fisheye over Scrolling

COMMENTS:

1. What do you like most about the Fisheye?

2. What do you like least about the Fisheye?
3. What do you like most about the Scrolling?
4. What do you like least about the Scrolling?
5. Would you be interested if the Fisheye Software is available for download?
6. What is ONE most important feature that should be added to the Fisheye?
7. Other comments.

# Appendix C

## Survey Results

Like Scrolling	Like Fisheye	Fish over Scroll	Download	Type
4	4	5	Yes	Manual
2	4	5	Yes	Manual
1	3	5	Yes	Manual
4	5	5	Yes	Manual
2	4	5	Yes	Manual
3	4	4	Yes	Manual
2	4	4	No	Manual
2	5	4	Yes	Manual
2	5	4	Yes	Manual
3	4	4	Undecide	Manual
2	4	4	Yes	Manual
4	3	3	Yes	Manual
3	3	3	No	Manual
5	2	2	No	Manual
4	3	2	Yes	Manual
3	3	2	Yes	Manual
1	1	1	No	Manual

Like Scrolling	Like Fisheye	Fish over Scroll	Download	Type
3	5	2	Yes	Auto
4	2	2	Undecide	Auto
4	2	2	No	Auto
3	4	2	Yes	Auto
4	3	3	Undecide	Auto
5	3	3	Yes	Auto
4	3	3	Yes	Auto
3	4	3	Yes	Auto
4	3	3	Undecide	Auto
3	2	3	No	Auto
5	3	3	No	Auto
3	3	5	Undecide	Auto
2	3	4	Undecide	Auto
4	4	4	Yes	Auto
3	3	4	Yes	Auto
3	4	4	Yes	Auto
n/a	4	n/a	Yes	Auto
n/a	3	4	Yes	Auto