Defence Research and Development Canada
Recherche et développement pour la défense Canada

Addressing usability deficiencies of large high-resolution displays

A. Bouchard
DRDC Valcartier

Defence R&D Canada – Valcartier
Technical Memorandum
DRDC Valcartier TM 2006-444
August 2007

Canada

Addressing usability deficiencies of large high-resolution displays

A. Bouchard
DRDC Valcartier

Defence R & D Canada - Valcartier

Technical Memorandum

DRDC Valcartier TM 2006-444

August 2007

Author

Alain Bouchard

Approved by

Yves van Chestein

C/GIC

Approved for release by

Christian Carrier

Chief Scientist

© Her Majesty the Queen as represented by the Minister of National Defence, 2007

© Sa majesté la reine, représentée par le ministre de la Défense nationale, 2007

Abstract

This technical memorandum addresses and offers a solution to some of the usability deficiencies of large and high-resolution displays. As the pixel density of displays increases, the size of the graphical user interface elements decreases proportionally with the consequence of reducing its usability. It may become difficult to operate the tools running on the display, the size of graphical user interface objects being too small to be activated easily. This document addresses these usability problems and presents a software prototype implementing a generic solution. The optimum size of graphical objects on touch screen display has been evaluated empirically and theoretically to 2.5 centimetres and a prototype integrating software lenses to zoom the graphical objects is presented.

Résumé

Ce mémorandum technique analyse un des problèmes de convivialité relié aux écrans de grande taille et à haute résolution. Au fur et à mesure que la densité de pixel d'un écran augmente, la taille des éléments graphiques diminue proportionnellement, ce qui a pour conséquence de réduire sa convivialité. Il peut alors s'avérer difficile d'utiliser les interfaces utilisateurs présentées sur de tels écrans, la taille des éléments graphiques étant trop petite pour une utilisation conviviale. Ce document fait l'analyse de ce problème de convivialité et présente un prototype logiciel offrant une solution générique. La taille optimum des éléments d'interface utilisateur y est évaluée empiriquement et théoriquement à 2,5 centimètres et un prototype intégrant une lentille logicielle qui permet de grossir les éléments graphiques y est présenté.

This page intentionally left blank.

Executive summary

The development of large and high-resolution displays is a step forward in replacing or complementing the traditional paper maps in Military Command Post. Various types of displays (e.g., front projection, retro-projection, flat panel display) are now being used by military personnel. Although these large displays offer a good alternative to paper maps, they are not as user-friendly as the paper version because of some usability deficiencies. As the size and the resolution of the displays increase, the user's behaviour toward the display changes, with a significant impact on usability. Any computer user has noticed that increasing the pixel density (pixel per inch) of a display necessarily decreases the size of the objects, including the menus, icons, etc. In the case of large high-resolution displays, increasing the resolution reduces its usability significantly considering that the user stays away from the screen to keep the overall context. It may become difficult to operate the tools running on the display, the size of graphical user interface (GUI) objects being too small to be activated easily. These usability deficiencies are reasons why some people still prefer the paper map, which constitute a more natural way of interacting with the data.

This technical memorandum addresses the usability problems described above and presents a software prototype implementing a generic solution. The first part of the document presents an estimation of the optimum size of GUI objects on the screen on a usability point of view. Empirical and theoretical results show that 2.5 centimetres represent the comfortable size of GUI object to be displayed on a touch screen display. The second part presents a prototype integrating software lenses, in particular inset and fisheye lenses, into the operating system to zoom the GUI objects on the display according to the optimum size.

Such a generic zoom capability avoids developing custom applications for large displays and has the advantages of improving the usability of the display by facilitating the access to the interface of any commercial or in-house applications on large displays.

Bouchard, A. (2007). Addressing usability deficiencies of large high-resolution displays. DRDC Valcartier TM 2006-444. Defence R&D Canada – Valcartier.

Sommaire

Le développement d'écrans de grande taille et à haute résolution constitue un pas vers un remplacement ou un complément des cartes papier traditionnelles utilisées dans les centres de commandement militaire. Différents types d'écrans (ex. projection frontale, rétro projection, matrices d'écrans plats) sont actuellement utilisés par le personnel militaire. Bien que ces grands écrans offrent une solution de rechange aux cartes papier, leur utilisation n'est pas aussi agréable en raison des problèmes de convivialité. Au fur et à mesure que la densité de pixel d'un écran augmente, la taille des éléments graphiques diminue proportionnellement, ce qui a pour conséquence de réduire sa convivialité. Tout adepte de l'informatique a remarqué qu'une augmentation de la résolution d'un écran engendre une diminution de la taille des éléments graphiques affichés, incluant les menus, icones, etc. Dans le cas d'écrans de grande taille et à haute résolution, la convivialité en est d'autant plus affectée, puisque l'utilisateur doit demeurer éloigné de l'écran pour conserver une vue d'ensemble. Il peut alors s'avérer difficile d'utiliser les interfaces utilisateurs présentées sur de tels écrans, la taille des éléments graphiques étant trop petite pour une utilisation conviviale. Cette déficience au niveau de la convivialité est des raisons pour lesquelles les militaires préfèrent la carte papier qui constitue encore une interface plus naturelle.

Ce mémorandum technique analyse ces déficiences et présente un prototype logiciel offrant une solution générique. La première partie du document présente une estimation de la taille confortable des éléments graphiques à présenter sur un écran qui en permet une utilisation conviviale. Les résultats empiriques et théoriques démontrent que 2,5 centimètres correspond à la taille optimum des éléments graphiques. En seconde partie, ce document présente un prototype utilisant des lentilles logicielles, en particulier des lentilles *insets* et *fisheye*, intégrées au système d'exploitation, qui permettent de zoomer les éléments graphiques d'une interface utilisateur.

Une telle capacité de zoom permet d'éviter le développement de logiciels sur mesure pour les grands écrans et a l'avantage d'augmenter la convivialité d'un système en facilitant l'accès aux interfaces de logiciels tant commerciaux que propriétaires.

Bouchard, A. (2007). Addressing usability deficiencies of large high-resolution displays. DRDC Valcartier TM 2006-444. R & D pour la défense Canada – Valcartier.

Table of contents

List of figures

List of tables

Acknowledgements

The author wishes to express his gratitude to Mr. Roger Fortin, Research Engineer from DRDC Valcartier, for his precious assistance and support from the development of the prototype to the review of this document. The author also wishes to thank the folks at IDELIX Software Inc for their technical support on the PDT technology as well as Matthew Phillips from DSTO Edinburgh for the multiple revisions of this document.

This page intentionally left blank.

1. Introduction

Ultra large displays are becoming ubiquitous in Military Command Post, increasing in size and resolution. Displays utilising front projection, retro-projection, or flat panel matrix are now commonly available to military personnel in order to complement or replace the traditional paper map. As with paper maps, a large display gives an overview of a scene while providing enough details to avoid the use of zoom and pan operations.

As the size and the resolution of the displays increase, the user's behaviour toward the display changes, with a significant impact on usability. Any computer user has noticed that increasing the pixel density of a display necessarily decreases the size of the objects, including the menus, icons, etc. In the case of large high-resolution displays, increasing the resolution reduces its usability significantly considering that the user stays away from the screen to keep the overall context. It may become difficult to operate the tools running on the display, the size of graphical user interface (GUI) objects being too small to be activated easily. The usability aspects of tools running on these displays need to be revisited.

In order to solve these usability problems, display providers develop custom applications with oversized GUI elements. Yet this solution only bypasses the problem and does not address it directly. A user is constrained to use the applications developed specifically for the display and is not allowed to use his own applications. A more robust and generic solution needs to be investigated in order to adjust the GUI elements on the screen, whatever the applications. Since the use of physical magnifying glasses is not that appropriate on electronic displays, software lenses, and in particular insets and fisheye lenses, are totally appropriate to provide the user with detailed and zoomed GUI objects.

This document presents a software prototype that integrates insets and focus+context (fisheye) lenses into high-resolution displays. Being part of the operating system display pipeline, the lens sits between the user and any potential applications. A lens that is placed over a region of the screen stretches that region to produce the zoom effect. The events sent to the application by the user are captured by the lens, transformed according to the zoom ratio, and forwarded to the underlying application transparently to the user.

Such a generic zoom capability avoids developing custom applications for large displays and has the advantages of improving the usability of the display by facilitating the access to the interface of any commercial or in-house applications on large displays.

This page intentionally left blank.

2. Usability Deficiencies of High-Resolution Displays

Very large and high-resolution displays (VLDs) are efficient hardware devices for multi-user collaboration and are becoming commonly used in command post. These VLDs are designed to offer as large a working area as the paper map and still highly-resoluted to offer accurate measurements and visualisation. These display devices have the advantage of showing an overview of a scene without needing to zoom and pan to obtain details.

Fortin [4] presented such a high-resolution tiled display, the Topographical Map Display (ToMaDi) (cf. Figure 1) developed by Defence R&D Canada as a research project to investigate better ways to present information to commanders by using a mosaic of LCDs screens.

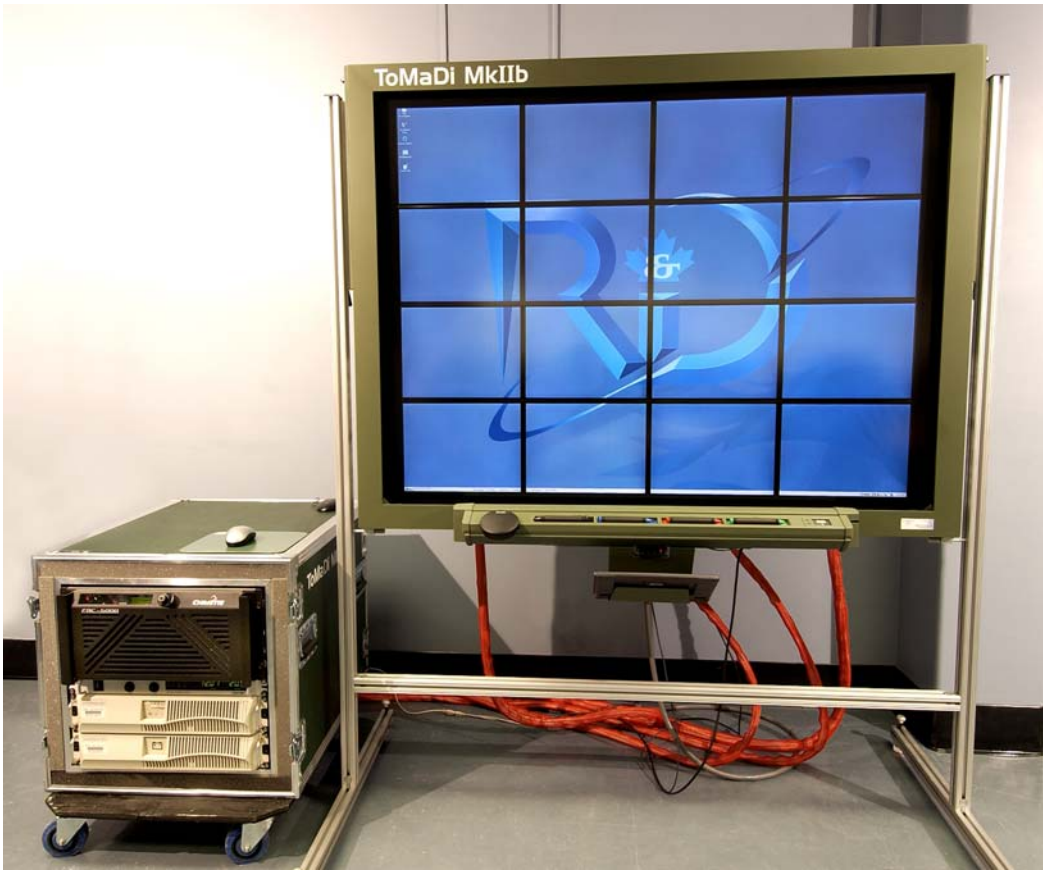

Figure 1. *ToMaDi MkIIb*

The ToMaDi display system is made of 16 LCD screens having a resolution of 1400x1050 pixels each and separated by 1.3 centimetres (half an inch) mullions[1]. A tremendous amount

[1] A slender vertical or horizontal member that forms a division between units of a window, door, or screen.

of information can be displayed on the 23 millions of pixels forming the entire screen with a pixel density of 117 PPI. The 152.4 centimetres diagonal working area allows a group of users to share the same view at high resolution while keeping the overall context, and after a while, the mullions are less perceptible. The display also has a touch screen capability allowing a cursor interaction with a finger or a pen. The use of a pen on the display provides better cursor accuracy, as mentioned later on, but using a finger is still more natural for the user.

However, as Fortin [4] mentions in his paper, there is an unfortunate characteristic inherent to these high-resolution displays: the higher the pixel density, the smaller are the GUI elements displayed on the screen. This undesirable property of highly-resoluted displays induces deficiencies that may significantly affect the usability of such systems:

1. The interaction with the GUI using the touch screen capability becomes harder. Point and click operations have to be achieved more accurately, which reduces usability;

2. The user needs to get closer to the display in order to discriminate the GUI elements properly, losing the overall context.

These usability deficiencies, amplified by the error induced by mullions, may have an important effect on the user-friendliness of the system and may lead to frustrating behaviours such as the activation of the wrong GUI control. Consequently, there is a need to address these usability deficiencies. An appropriate solution to this problem would be to identify a generic way of increasing the size of GUI controls on the screen, as a software lens would do, without affecting the actual working area where we wish to keep the resolution intact.

Prior to implementing such a generic GUI zooming lens, we need to evaluate the optimum size of GUI elements on large displays, especially on finger-driven touch screen, as explained in the next section.

2.1 Input device inaccuracy

Any pointing device used on a display system always has a level of inaccuracy reaching its maximum accuracy at the size of the display unit – the pixel size. The accuracy follows a Gaussian distribution along the area covered by the input pointer. The probability of hitting the desired display location is 0 outside the area covered by the pointer and increases up to a maximum approximately in the middle of this area. The shape of the Gaussian distribution is greatly dependant on the type of pointer input system (e.g., mouse, camera, pressure, electric detection) and on the position of the pointer on the display. The inaccuracy of pointing devices such as a standard mouse is usually considered insignificant. The area covered by these types of pointers is one pixel, which is the smallest unit of measure of the display matrix, and it makes no sense to have input device accuracy greater than the actual display resolution.

In the case of finger-driven displays, such as the ToMaDi MkIIb shown in Figure 1, the input device accuracy issue become significant. Because the input device covers more than a single pixel, the detected position is likely to be anywhere within the covered area. For example, Figure 2 from Beckie [1] shows the accuracy of camera-based touch screen display. Such display typically uses cameras in different positions around the display to detect marks on the

captured images and then triangulate the position of these marks. Although only two cameras are required to discriminate the pointer position, three to four cameras are usually employed to increase the detection accuracy. The figure indicates the presence of two pointers on the display, as detected by a camera situated on the top left corner of the display. The left pointer position detected appears in the center of the input device, while the right pointer position detected is a little shifted to the left because of the camera perspective.

Figure 2. *Two simultaneous pointer detected, as demonstrated by this binarized image camera data*

Because of the important variability of position detected using the different systems on different position on the display, it is assumed that the positional accuracy of the input device correspond to the area of the device.

Let's assume that the forefinger is the one driving the display and that this finger on the display is about 1.2 centimetres in size, both horizontally and vertically. The forefinger would then cover 55 pixels for a display at 117 PPI such as the ToMaDi MkIIb, which represent an accuracy of plus or minus 30 pixels in both axes. Considering that most of the GUI elements on the screen are 16 to 32 pixels wide, the inaccuracy induced by the finger size is significant.

2.2 Touch screen inaccuracy

In the case of the ToMaDi, the touch screen device system consists of a camera-based transparent touch sensitive digital board developed by the company Smart Technologies. The correspondence between the displayed desktop and the touch active surface is achieved through an *a priori* calibration, which results in a "zero position error" on a continuous surface. However, the active display surface of the ToMaDi is divided by 3 horizontal and 3 vertical mullions which cause a non-linear imprecision between the designation pointer (finger or pen) position and the actual localization of the cursor on the screen (cf. Fortin [4]).

Figure 3 presents the distribution of position error along a horizontal axis. The absolute error is null at the periphery of the display and reaches a maximum of 75% of the mullions size at L/4 and 3L/4. The same position error happens along the vertical axis. This leads to an error of plus or minus 45 pixels on both vertical and horizontal direction for the ToMaDi MkIIb. Although it is possible to fix that problem by changing the algorithm to determinate the cursor position, such a change has not been integrated into the digital board.

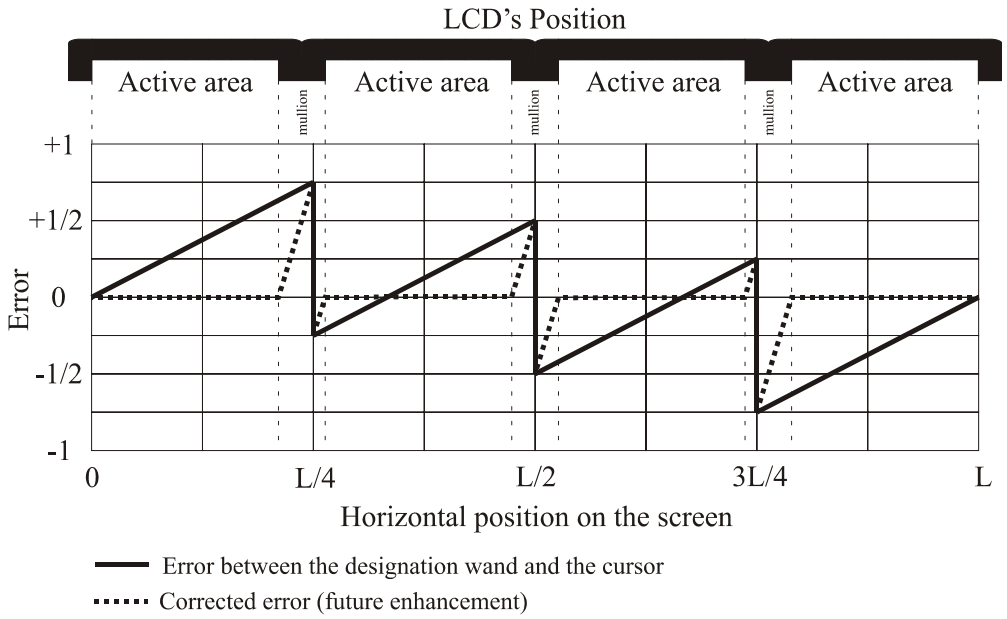

Figure 3. *Horizontal cursor position error*

2.3 Optimum GUI control size

As mentioned earlier, the size of GUI elements on large high-resolution displays is an important issue and is a trade-off between usability and space used. According to Fitts' Law (MacKenzie and Buxton [8]), the time to move and point to a target of width W at a distance A is a logarithmic function of the spatial relative error (A/W). The larger and nearer to the mouse pointer an on-screen object is, the easier it is to click on. However, the size of objects on screen is limited by design constraints and space available on the screen. On very large displays, the space available is sufficient to design oversized GUI objects. The size of GUI objects is even more important when the display has a touch screen capability; it has been clearly identified as a usability deficiency of the ToMaDi system (Fortin [4]).

After many hours of operation of the ToMaDi system and feedback from operational users, optimum GUI control size has been evaluated empirically to 2.5 centimetres (1 inch) in both horizontal and vertical directions. That dimension seems the most appropriate on a usability point of view, it is large enough to use with ease, and screen real estate is not wasted. A specialized geospatial viewer – ToMaDi Viewer – has been designed specifically for the ToMaDi system following this rule of 2.5 centimetres.

On a theoretical point of view, the optimum GUI control size can be evaluated based on usability as well as various characteristics inherent to the system: size of input pointer and touch screen inaccuracy. Table 1 indicates that both measures, empirical and theoretical, come with similar results in term of GUI elements size for the ToMaDi, considering the inaccuracy values described earlier.

Table 1. *Optimum GUI objects size*

PARAMETER	VALUE
ToMaDi MkIIb resolution	117 PPI or 46 pix/cm
Mullion size	1.3 cms
Inaccuracy induced by finger size (F)	± 30 pixels
Inaccuracy induced by touch screen (TS)	75% mullion size = 0.98 cm = ± 45 pixels
Maximum error (E = F + TS)	± 75 pixels
Optimum GUI object size (theoretical)	2 * E = 150 pixels
Practical GUI object size (empirical)	2.5 cms = 117 pixels

It is then possible to assert that an optimum GUI control on a highly-resoluted display should be approximately 2.5 to 3.5 centimetres in size. Nevertheless, the toolbar icons and buttons are usually represented by matrices of 16x16 or 32x32 pixels in most commercial applications. As Table 1 indicates, such icons are definitely too small to be activated using human fingers and even using a specialized input devices such as a pen the touch screen inaccuracy may cause serious usability flaws.

2.4 Potential solutions

Basically, three ways of solving this problem may be identified:

1. Hovering the GUI control underneath the input pointer;

2. Develop software components having oversized icons, menus, toolbars, and so forth, that are adapted solely to large displays;

3. Develop a generic module that allows increasing the size of any GUI element, whatever the origin of the application.

The first way of addressing this usability issue, which consists of hovering the GUI control underneath the input pointer, may be considered as a temporary solution. Whenever the mouse focus moves over a control, the borders of the control are drawn bolder, that is called the hovering effect. The control with the hovering effect will be the one activated by a mouse click at that mouse location. The only objective of hovering a control is to avoid activating surrounding controls in the interface by offering to the operator a visual cue indicating that the cursor is on the target. However, this solution does not actually increase the usability of the system, the operator still has to make sure the focused GUI control is the desired one, and it may be difficult to determine when his finger is partially hiding the hovering mark. Furthermore, this solution does not apply on systems having a touch screen capability because

it requires a pointer move event before any pointer click. On most touch screen systems, as soon as the operator touches the screen, a mouse-click event is generated, even before the hovering of any control. In that case, the hovering effect is null and void.

With the second option, the main problem is that only a limited number of software products would be compliant with large high resolution displays. It is very unlikely that commercial software producers would develop such specialized tools, unless someone pays for it. There is always the possibility of developing custom products in-house, a solution that is not affordable for most. In the case of the ToMaDi system, DRDC Valcartier has developed an in-house viewer, called ToMaDi viewer to run exclusively on the ToMaDi. It appeared to be a viable solution at the time, but the maintenance of the system ended up being time and resource-consuming.

Considering that operators usually prefer to use the products they know and are familiar with, a more appropriate solution is the third one. This solution consists of developing a generic module that allows increasing the size of any GUI element up to the optimum GUI control size, whatever the origin of the application. The following sections present a prototype implementing a software lens for any Microsoft Windows-based interface.

Robertson et al. [9] also present some usability deficiencies of large displays and suggest a whole set of software prototypes to address these issues. However none of the solution offered address the specific problem of the size of GUI elements on high-resolution displays.

3. WinLens prototype

The WinLens prototype is a software tool that allows zooming any portion of a computer screen using inset or fisheye visualizations. Although this prototype has been built specifically for the ToMaDi display being developed at DRDC Valcartier, it could potentially be used on any Microsoft Windows operating system.

The implementation of the WinLens is twofold: a lens controller and the lens itself. The lens controller allows creating a lens and managing the lenses currently displayed on the screen.

3.1 Lens Component

Being integrated into the operating system, a lens sits between the user and the applications. A lens that is placed over a region of the screen stretches that region to produce the zoom effect. Various zooming techniques may potentially be used to zoom a region of the screen, the prototype implements two of them: inset and fisheye lenses.

3.1.1 Inset magnification

Typically, a fixed or floating inset (e.g. Figure 4) captures a certain region of a screen, magnifies that region, and displays the result in a separate window. Some insets are configured to follow an input device as it moves and display the content of the region in a separate window. Such a tool, called Magnifier, is natively provided in Microsoft Windows operating system and many other similar products are freely available.

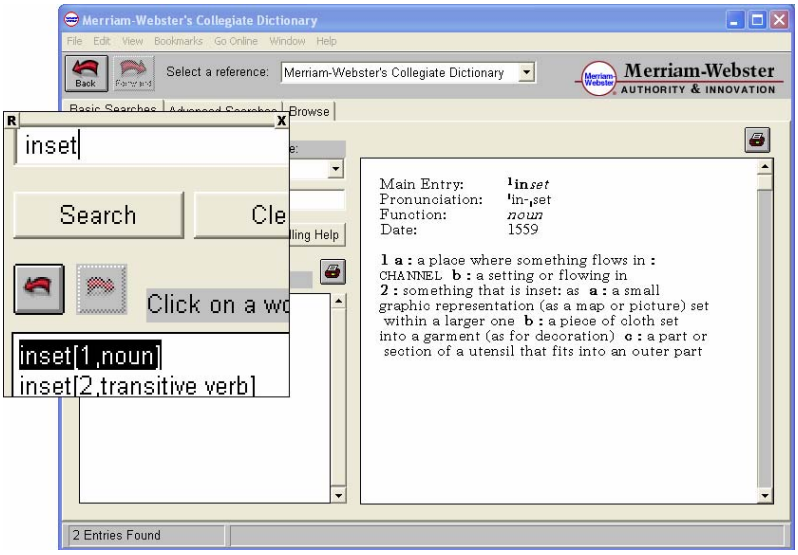

Figure 4. *Example of inset lens*

These inset windows may be used to address the deficiencies of large high-resolution displays mentioned earlier. An inset can be placed in such a way that a GUI control or a group of controls be magnified and displayed in the inset window. The WinLens prototype allows creating such insets. The inset then allows interacting with the underlying application directly, unlike most of the zooming tools currently available. The mouse and keyboard events captured by the inset are forwarded to the underlying application. Keyboard events are forwarded without modification while the mouse coordinates have to be previously displaced according to the zoom ratio and the offset vector. Therefore, simple mouse operations such as click and double-click can be forwarded while more complex operations as drag&drop and scroll are simply not possible. Nevertheless, these more complex operations are not frequent operations on GUI controls such as menus and toolbars and even less frequent on touch screen displays.

Figure 5 shows an example of coordinates translation where a point *Pr* on the screen has been displaced to its apparent position *Pa*. The offset vector corresponds to the positional difference between the base point of the inset (center) and its actual position on the screen, which is null in Figure 5.

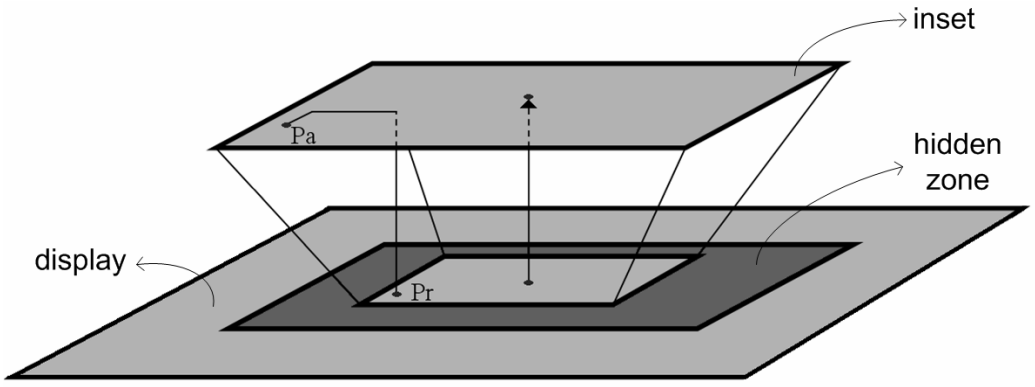

Figure 5. *Displacement of mouse coordinates within the inset lens*

Figure 5 also clearly presents an inherent characteristic of insets; some portion of the screen is hidden behind the inset. This hidden portion may include valuable information not displayed to the operator and represents a usability deficiency in itself. The inset should be placed on the screen in such a way that the hidden portion (dark grey zone) covers irrelevant information and does not affect the actual working area of the underlying application.

Figure 6 and Figure 7 show an example of how insets could be placed on GUI controls of Windows Paint without affecting the actual working area. When defining a zooming region, the operator has to identify the base point where the coordinates on the screen and in the inset will be identical, whatever the zoom ratio applied, and then identify the vector indicating the direction of the inset. This way of defining the position of the inset allows keeping the working area entirely visible, whatever the zoom ratio. The rule of thumb to define the direction vector is that the base point shall not be within the working area and the vector shall not be oriented toward the working area.

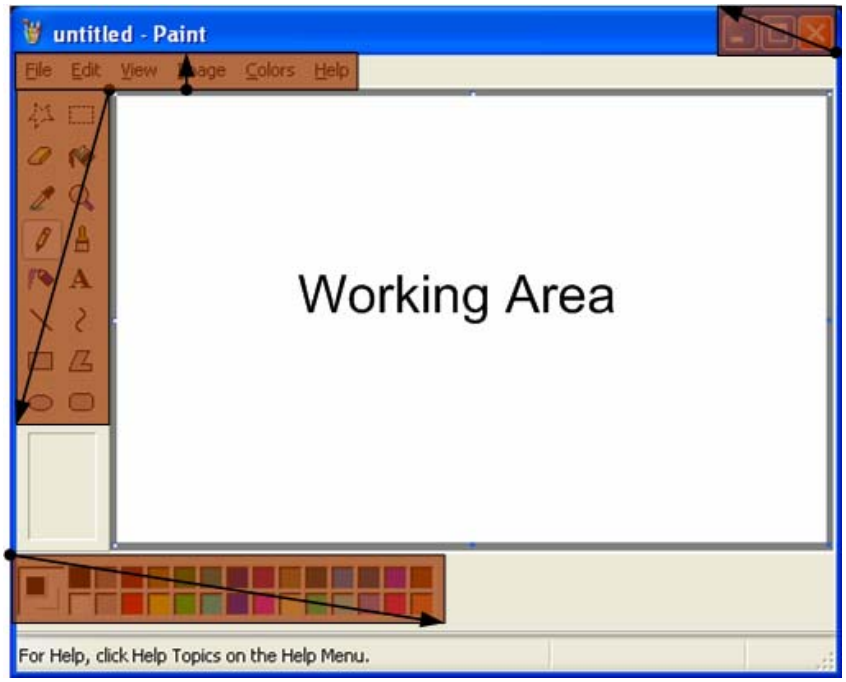

Figure 6. *Position of the insets (without zoom)*

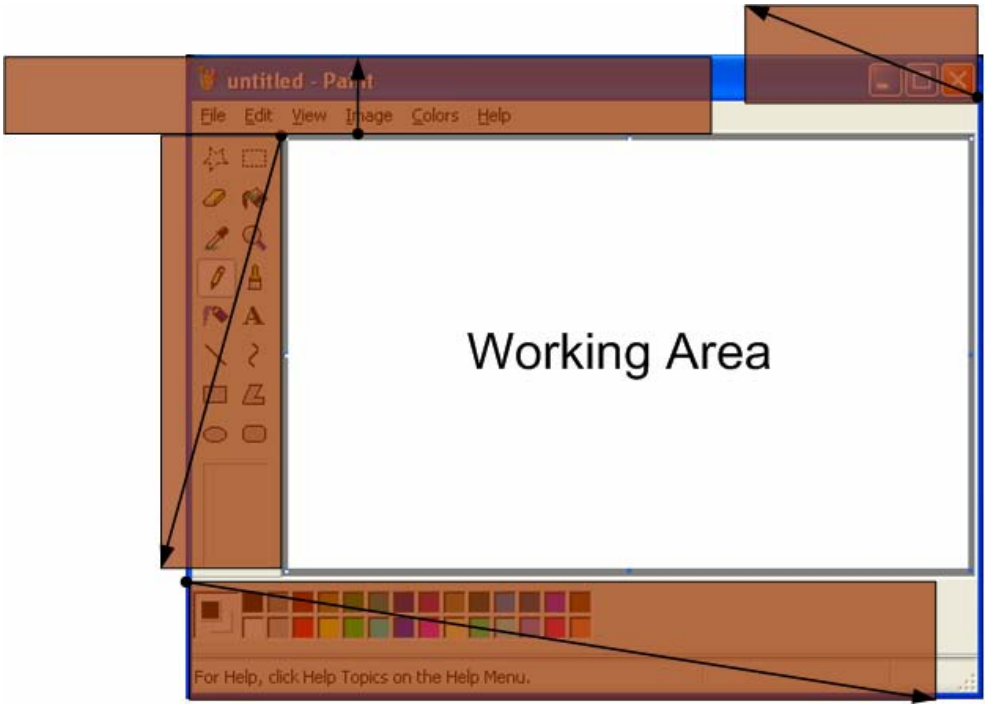

Figure 7. *Position of the insets (with zoom)*

As an example, insets have been positioned on Windows Paint running on the ToMaDi system, with the direction vectors shown in Figure 7, in order to evaluate the prototype efficiency. Figure 8 clearly shows how the GUI controls of the painting tool look tiny on a display such as ToMaDi. At a resolution of 117 dpi, every 32x32 pixels icon on the palette is about two thirds of a centimetre in size. Activating an operation through such an icon is very difficult using a pen and unlikely to work at all using a finger.

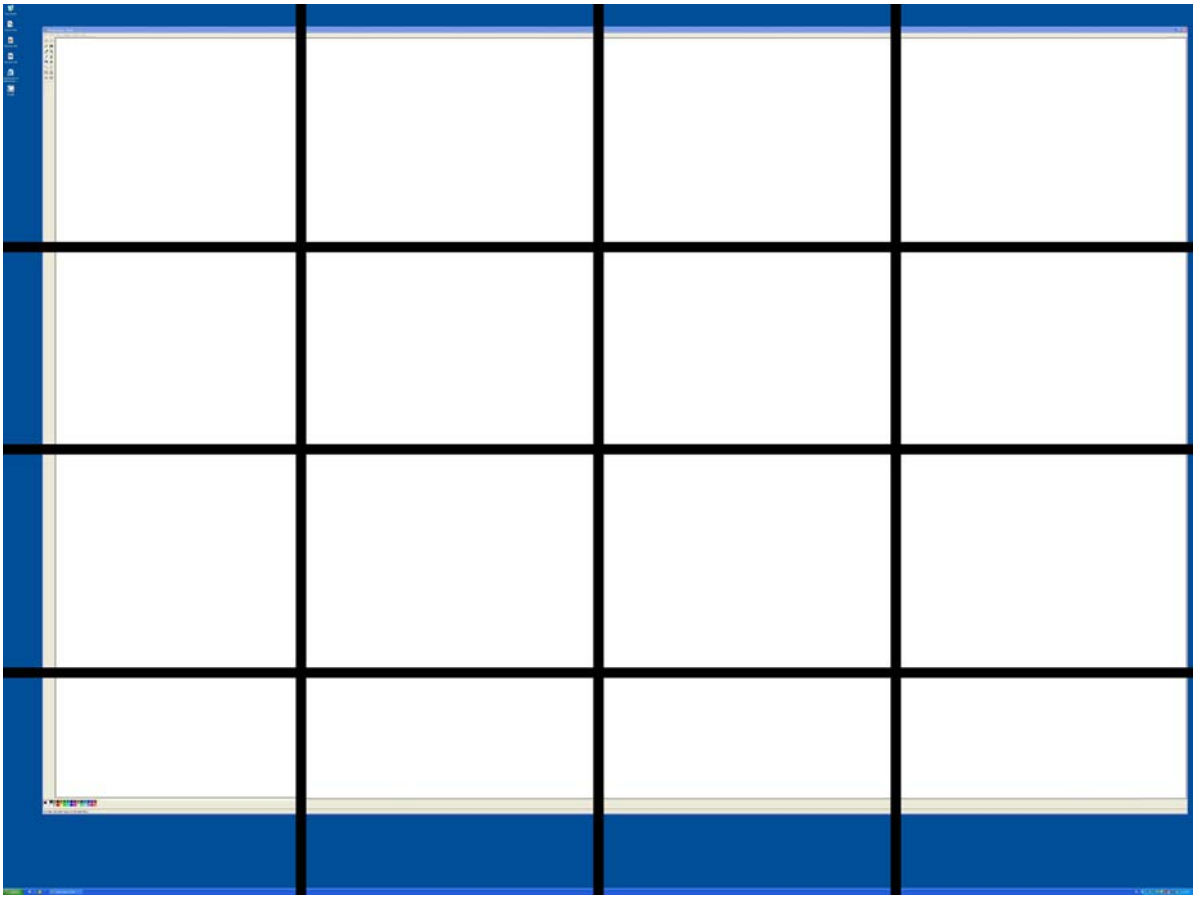

Figure 8. *ToMaDi without insets*

Figure 9 shows six insets with a zoom ratio of 4:1 have been applied to the interface above. At that magnification level the icons on the palette are about 2.5 centimetres in size, which is the optimum GUI control size evaluated earlier. Furthermore, the insets do not cover a single pixel of the working area of the software.

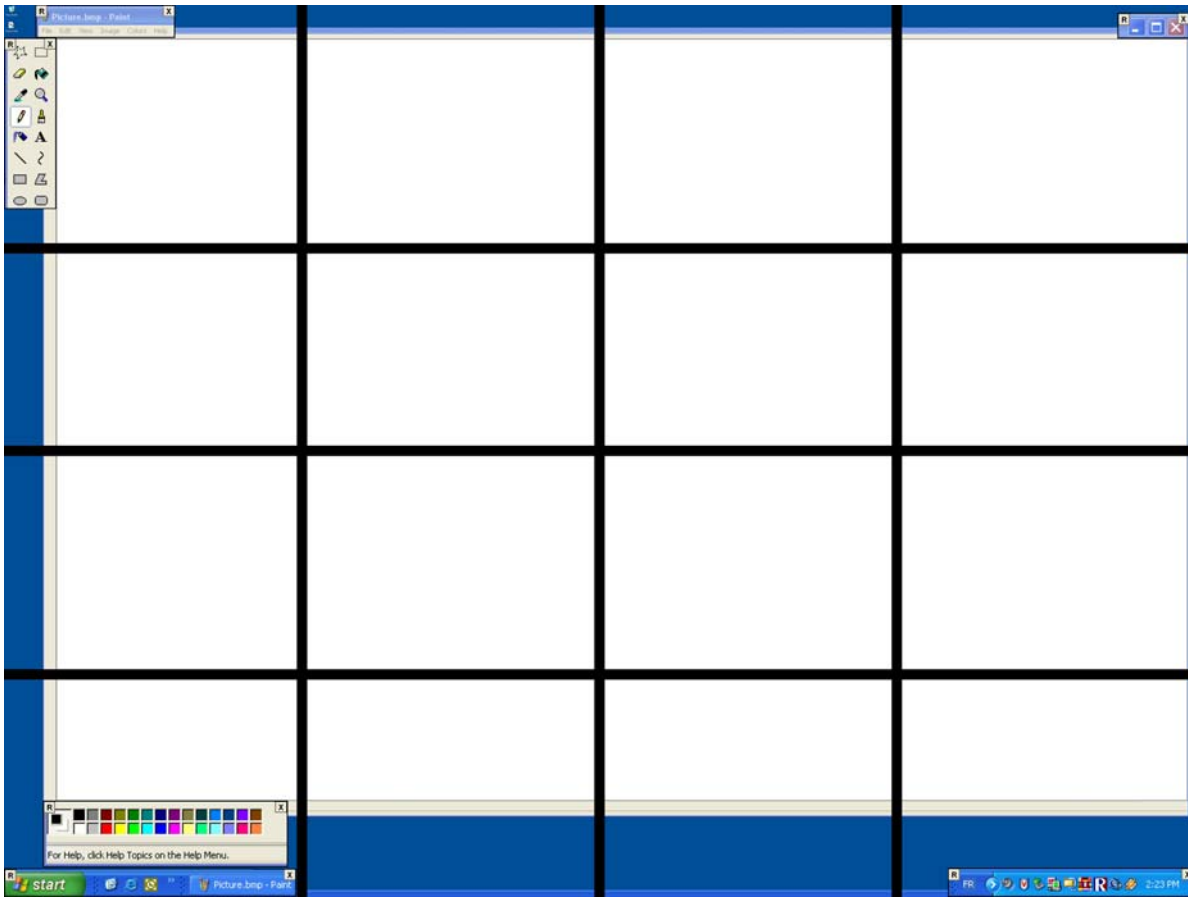

Figure 9. *ToMaDi with insets*

It is obvious that interacting with the insets activated in that particular case can be done much more easily. But it is not always possible to place the insets without any obstruction of valuable screen region, especially on cluttered software interface and on displays with limited screen real estate. In that case, the use of fisheye lens may be envisaged as an alternate solution.

3.1.2 Fisheye magnification

As mentioned earlier, the use of insets is efficient to zoom a lightly cluttered interface, but has the disadvantage of hiding portions of the screen and losing the context outside of the lens. Sometimes keeping the context around the lens does matter, on geospatial applications for example. The use of fisheye lenses is then appropriate. Focus+context techniques have been studied in various contexts and have been shown to perform well for navigation tasks (cf. Carpendale et al. [3], Gutwin and Fedak [5], Gutwin and Skopik [6]) as well as for GUI interactions (cf. Bederson [2]).

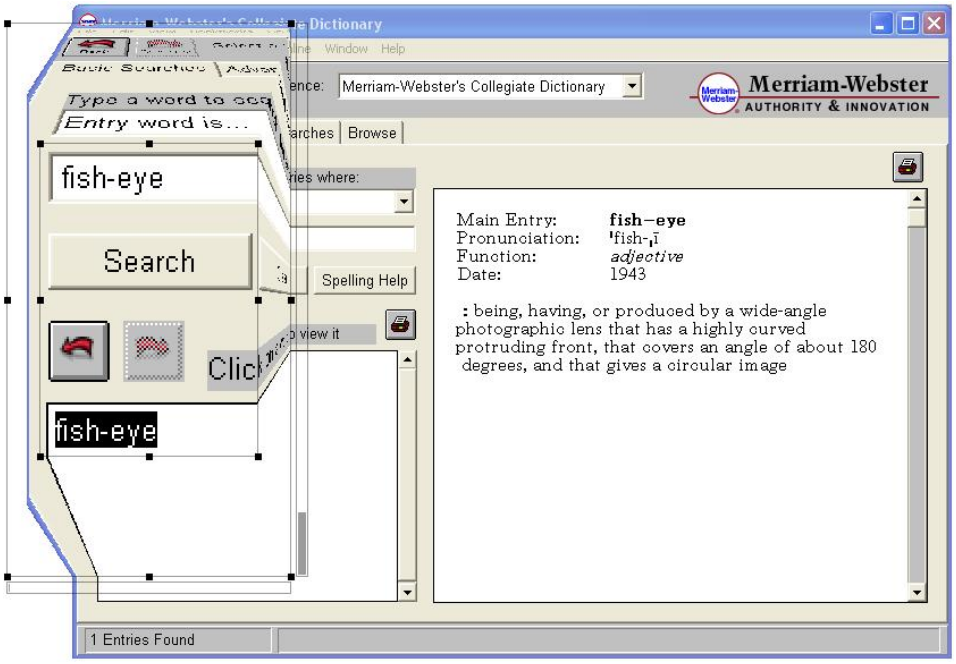

Figure 10. *Example of fisheye lens*

The solution adopted for the WinLens prototype consists of a fisheye lens drawn into the desktop display context (cf. Figure 10). The underlying applications are not aware that a zooming device is currently affecting their display context; the lens actually sits on top of all applications running on a desktop. As with the insets, mouse events sent by the user to underlying applications are captured by the lens, transformed according to the zoom ratio, and forwarded to the underlying application transparently to the user.

Figure 11 depicts schematically a situation where a pixel at position *p* is displaced to position *d3* with an apparent position at *d2* on the screen. As the user clicks on the pixel at the apparent position *d2*, the lens transforms the mouse coordinates and generates a click at the actual position *p*. The keyboard events are not modified.

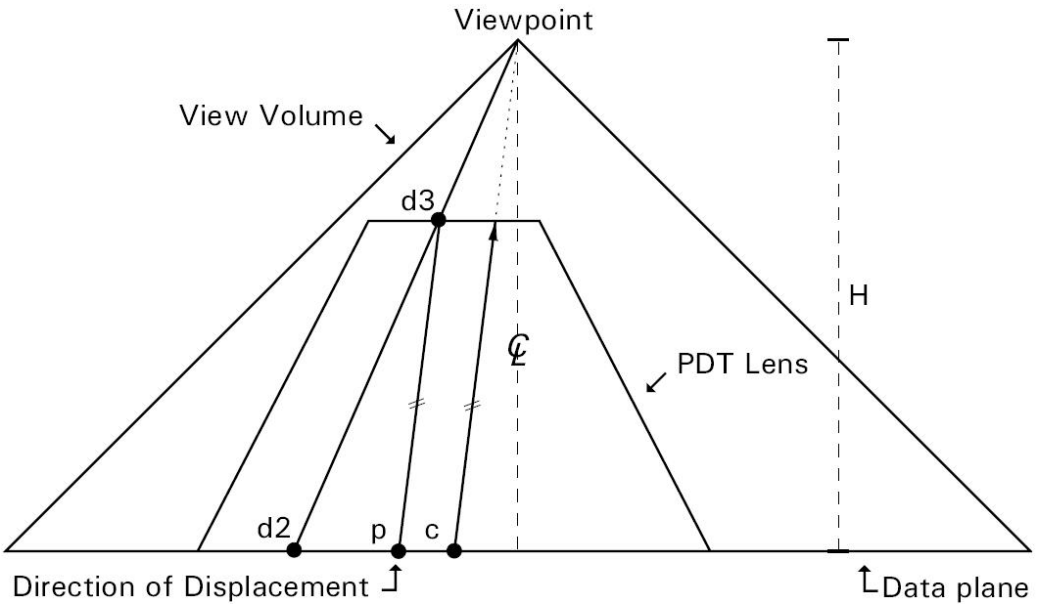

Figure 11. *Point displace and undisplace (cf. IDELIX Software Inc. [7] p.19)*

A commercial library, IDELIX Pliable Display Technology (PDT) (cf. Figure 12) provides the fisheye capability. The anchor points allow resizing the focal region and the whole lens to fit to a particular region on the screen. The focal region displays an undistorted zoomed view of the underlying region while the shoulder region fades out the pixels to create the fisheye effect. The scroll bars on the right and bottom side of the lens allows modifying the zoom factor within the focal region and the fading smoothness of the shoulder respectively.

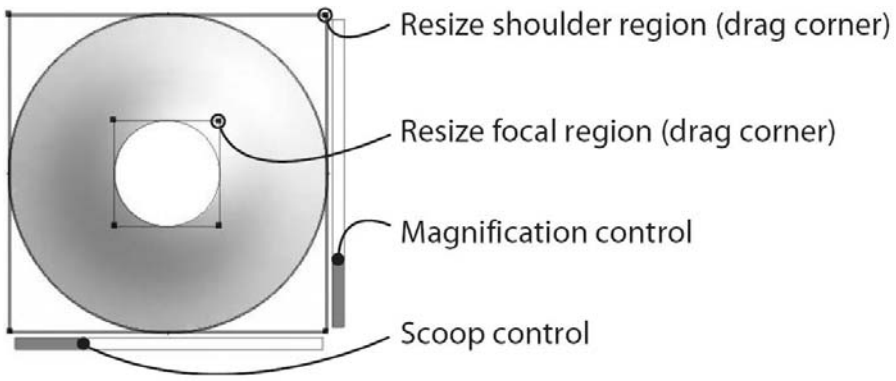

Figure 12. *Fisheye lens (courtesy of IDELIX Software Inc)*

A lens drawn on the screen operates in two modes: view and edit mode. While in view mode, a user has the full control on the lens displayed on the screen. The mouse clicks within the lens are not forwarded to the operating system so the user can move the lens without affecting the underlying applications. In edit mode, the drag corners disappear so the user can no longer change the position or properties of the lens, and every mouse action is captured by the lens, modified accordingly, and forwarded to the operating system.

The current version of the WinLens prototype only supports single lens on the screen. Nevertheless, it is possible to display multiple lenses on the screen by creating many instances of WinLens application. However, when using multiple lenses on the screen it may happen that a lens intersects other ones. In that case, the intersecting region will be magnified twice resulting of unpredictable results. The PDT provides a merging capability natively to deal with multiple lenses, although not supported by WinLens.

3.1.3 Fisheye Lens Controller Component

Basically, the fisheye lens controller (cf. Figure 13) allows creating, deleting, and refreshing a lens on the screen as well as modifying its display properties. Lenses may be created by clicking on the create button and similarly may be refreshed or removed from the screen by clicking on the delete and refresh button respectively. By default, a newly created lens appears in the middle of the screen with a square shape, the user then moves and resizes the lens to fit to a particular region of interest on the screen. The lens mode (view or edit) can be modified through the lens controller.

Figure 13. *Fisheye Lens Controller Component*

The lens preferences dialog box (Figure 14) can be activated using the button on the far right of the lens controller. This dialog box allows modifying the appearance of lens on the screen. Details about the various preferences are presented in Table 2.

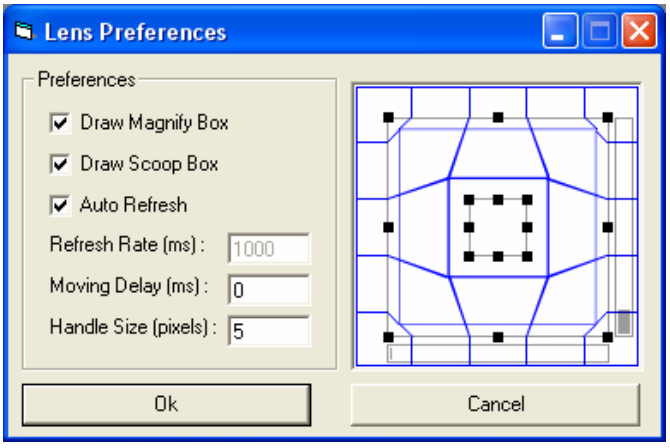

Figure 14. *Lens Preferences*

Table 2. *Lens Preferences*

PREFERENCE	DESCRIPTION
Draw Magnify Box	Indicate whether or not the magnify box on the right side of the lens is displayed. For fisheye lens only;
Draw Scoop Box	Indicate whether or not the scoop box on the bottom side of the lens is displayed. For fisheye lens only;
Auto Refresh	If activated, the auto refresh option allows refreshing the lens on every user action within the lens;
Refresh Rate	If the auto refresh is not activated, set the interval time for recurrent refresh (milliseconds);
Moving Delay	Introduce a delay in the movement of the lens (milliseconds);
Handle Size	Set the size of the drag corners (pixels). For fisheye lens only.

The moving delay option is particularly important when the touch screen system, such as ToMaDi, generates a large amount of mouse events for a single move and may slow down the system significantly. It is then possible for the lens to accumulate in memory the events during the specified delay and process them as a single event. Finally, the handle size parameter allows increasing the drag corners of the lens for improved usability, especially on high-resolution touch screen displays where the default handle size may appear too small.

This page intentionally left blank.

4. Conclusion

Large and high-resolution displays are becoming more and more used in a military context to complement or replace the traditional paper map. However, it has been showed that large high-resolution displays suffer of usability deficiencies being the small size of GUI objects on screen. In order to solve these usability problems, display providers develop custom applications with oversized GUI elements to run on the displays. But this solution only bypasses the problem and does not address it directly. A user is constrained to use the applications developed specifically for the display and is not allowed to use his own applications. A more robust and generic solution needed to be investigated.

This document presented an avenue of solution to this problem. A software prototype integrating insets and focus+context (fisheye) lenses into the display operating system have been presented. Being part of the operating system display pipeline, the lenses sit between the user and any potential applications. A lens that is placed over a region of the screen stretches that region to produce the magnification effect within an inset or a fisheye.

This solution has the advantages of improving the usability of the display by facilitating the access to the GUI objects while avoiding the development of custom applications specific for large displays. Although this prototype offers an appropriate workaround of the problem stated, it is not mature enough to be used in an operational context. The major display problems on some particular applications (e.g. Windows Explorer) may irritate the user. In order to solve these irregularities, the operating system developer, Microsoft for instance, has to integrate the lens tightly into the operating system to ensure that the lens works properly.

This page intentionally left blank.

References

1. Beckie, N., (2005). Touching Large Displays. *Information Display*, 21(1), pp 22-25.

2. Bederson, B. B. (2000). Fisheye Menus. *Proc. ACM UIST 2000, pp. 217-225.*

3. Carpendale, M. S. T., Cowperthwaite, D. J., and Fracchia, F. D. (1995). Three-Dimensional Pliable Surfaces: For Effective Presentation of Visual Information. *UIST'95, Proceedings of the ACM Symposium on User Interface Software and Technology.*: ACM Press.

4. Fortin, R. (2001). Novel display devices for command and control applications. *Proceedings SPIE Vol. 4362. Cockpit Displays VIII: Displays for Defense Applications.*

5. Gutwin, C. and Fedak, C. (2004). Interacting with Big Interfaces on Small Screens: a Comparison of Fisheye, Zoom, and Panning Techniques. Proceedings of GI.

6. Gutwin, C. and Skopik, A. (2003). Fisheye Views are Good for Large Steering Tasks. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'03)*. Fort Lauderdale.

7. IDELIX Software Inc. (2004). Pliable Display Technology (PDT). Technical Description. Vancouver, Canada. 40 pages.

8. MacKenzie, I. S. and Buxton, W. (1992). Extending Fitts' law to two dimensional tasks. *Proceedings of the CHI '92 Conference on Human Factors in Computing Systems.* New York: ACM.

9. Robertson, G., Czerwinski, M., Baudisch, P., Meyers, B., Robbins, D., Smith, G., and Desney, T., (2005). The Large-Display User Experience. *IEEE Computer Graphics and Applications*, 25(4), pp 44-51.

List of acronyms

DND	Department of National Defence
GUI	Graphical User Interface
LCD	Liquid Crystal Display
PDT	Pliable Display Technology
PPI	Pixel Per Inch
ToMaDi	Topographic Map Display
VLD	Very Large Display

Distribution list

INTERNAL:

1- DG

3- Document Library

1- Alain Bouchard

1- Denis Gouin

1- Roger Fortin

1- François Létourneau

1- Yves van Chestein

EXTERNAL:

1- DRDKIM (PDF File)

1- Rudi Vernik
Research Leader Command Decision Environments
Command and Control Division
Defence Science and Technology Organisation (DSTO)
PO Box 1500
Edinburgh SA 5111
Australia

1- Matthew Phillips
Command and Control Division
Defence Science and Technology Organisation (DSTO)
PO Box 1500
Edinburgh SA 5111
Australia

1- Peter Evdokiou
Command and Control Division
Defence Science and Technology Organisation (DSTO)
PO Box 1500
Edinburgh SA 5111
Australia

DOCUMENT CONTROL DATA

1. ORIGINATOR (name and address)	2. SECURITY CLASSIFICATION (Including special warning terms if applicable)
Defence R&D Canada Valcartier 2459 Pie-XI Blvd. North Québec, QC G3J 1X8	Unclassified

3. TITLE (Its classification should be indicated by the appropriate abbreviation (S, C, R or U)
Addressing usability deficiencies of large high-resolution displays (U)

4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.)
Bouchard, A

5. DATE OF PUBLICATION (month and year)	6a. NO. OF PAGES	6b .NO. OF REFERENCES
2007	33	9

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. Give the inclusive dates when a specific reporting period is covered.)
Technical memorandum

8. SPONSORING ACTIVITY (name and address)

9a. PROJECT OR GRANT NO. (Please specify whether project or grant)	9b. CONTRACT NO.
15al	

10a. ORIGINATOR'S DOCUMENT NUMBER	10b. OTHER DOCUMENT NOS
TM 2006-444	N/A

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

- ☒ Unlimited distribution
- ☐ Restricted to contractors in approved countries (specify)
- ☐ Restricted to Canadian contractors (with need-to-know)
- ☐ Restricted to Government (with need-to-know)
- ☐ Restricted to Defense departments
- ☐ Others

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

(U) This technical memorandum addresses and offers a solution to some of the usability deficiencies of large and high-resolution displays. As the pixel density of displays increases, the size of the graphical user interface elements decreases proportionally with the consequence of reducing its usability. It may become difficult to operate the tools running on the display, the size of graphical user interface objects being too small to be activated easily. This document addresses these usability problems and presents a software prototype implementing a generic solution. The optimum size of graphical objects on touch screen display has been evaluated empirically and theoretically to 2.5 centimetres and a prototype integrating software lenses to zoom the graphical objects is presented.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Information Visualisation, Usability, Large Displays, Fish-eye lenses, Detail-in-context

Defence R&D Canada

Canada's Leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

WWW.drdc-rddc.gc.ca