Information Visualization for Mobile Devices: A Novel Approach based on the MagicEyeView

Gheorghita Ghinea[1,2], Jorn Heigum[1,2], Anders Fongen[2]
Brunel University[1], Norwegian School Information Technology[2]
E-mail:{george.ghinea, jorn.heigum}@brunel.ac.uk;andfon@nith.no

Abstract

Visualization on mobile devices not only means accommodating to a small screen space, but also widely different aspect ratios. Improving on the MagicEyeView algorithm, this paper presents a visualization technique that is better suited to screens with skewed aspects ratios. The presented approach is a focus+context visualization effort which employs distortion of coordinate scales and a "fisheye" technique. The visualization algorithm is evaluated in the problem domain of business management and the presentation of "Key Performance Indicators".

1. Introduction

Mobile phones and PDAs are spreading throughout the business user community at a rapid pace. Theses devices are becoming more and more adept packing more processing power and storage space than ever before. Even though this might be the case, few users use these powerful devices for other tasks than contact databases or as an extension to their short term memory, e.g. via calendar applications.

One of the key reasons for this is the limited screen space available because of the constrained "form factor" that have been, and will continue to be a limitation of these devices. According to [10] [11], this places a special importance and a need for emphasis on the structure of the information to be presented. Information visualization (IV) might be a technology to help with this problem.

One of the main goals of IV is to enable processing of large amounts of data in an instant by delegating work to the visual processing of humans and their capability of visual pattern recognition. One of the main problems of visualizing information on small screen devices is to display enough information to give the user appropriate and adequate contextual information. Focus + context techniques have been used to solve this problem on classic hardware. In this paper we describe a novel approach which adapts the MagicEyeView algorithm to better handle the varying aspect ratios of small screen devices and thus utilises screen space more efficiently.

2. Focus + context techniques

Focus + context are used in visualizations to provide user details of a phenomenon as well as to relate the phenomenon to its context. In [6] Furnas proposes a Degree of interest (DOI) function which he generalizes in a follow up article describing generalized fisheye views [7]. His suggestions were made based upon studies of how humans perceive phenomena with local high detail that decreases according to the degree of interest.

The DOI function has two components, a so called A priori interest (API(x)) component, which is determined in accordance to the current task and a distance component D(x,y) which is a function of distance from focus point x. Furnas states that fisheye views can be realized for any structure supporting functions for API(x) and D(x,y). He uses a hierarchy to exemplify his ideas; by setting focus to a node in a hierarchy he calculates to DOI function for the sub nodes by using the distance from the focus (root), different algorithms can used to calculate the distance.

Even though research has been conducted into the application of focus + context techniques on portable devices, such work either focuses on general techniques or on relatively mundane applications. So far there has been a paucity of research which has targeted the use of focus + context techniques to display numeric information or status information on portable devices.

2.1 Focus+context techniques for mobile devices

One of the main problems encountered when attempting to use IV techniques on mobile devices is that the latter's screens usually have markedly different aspect ratios to the ones of classical, standalone devices and, as such, there is usually an inefficient use

978-1-4244-1653-0/08/$25.00 ©2008 IEEE

of space. To solve the problem of wide aspect ratios when working with information sets with spanning properties Mackinlay et al. [13] proposed the perspective wall. Utilizing the 3D hardware of the time, they displayed information items on a band with a central focus region with context regions to the left and right, where the band was set at an angle.

As an improvement to the generalized fisheye view and the perspective wall, Robertson et al. [16] introduced the lens metaphor for visualizing large documents. The lens they used had a constant magnification for an area in focus, with a gradual reduction of magnification towards the edges of the screen. The area of focus could be moved around the document and the lens could be varied in size by moving the lens in the Z plane. The document lens can be said to extend the perspective wall by extending the context regions.

The next development to the fisheye view, allowing for multiple foci points, uniform scaling in the focus area, and arbitrary focus size specification, was a rubber sheet metaphor introduced in [17]. This technique allowed a user to select a region of interest and stretch it into the desired size. This created a uniform focus region with horizontal and vertical bands having distorted aspect ratios. Building on the ideas of the rubber sheet and the DOI function, Rao and Card develop the table lens [14]. This is an application of the DOI function using the rubber-sheet metaphor, with discrete regions of focus delimited by rows and columns. This allows users to view information from huge tables in context.

After this initial flurry of activity in the area, research in the area resumed anew only with the advent and proliferation of mobile devices, with most of the research targeting technology adoption or specific enhancements to the field. Thus, the Halo technique [1] uses circles (halos) to introduce objects into the display region even if they are outside of it. In related work, three novel focus + context techniques targeted at mobile devices were introduced in [11]: the Large Focus display, Perspective Overview Display and the Transient Focus Display.

Further developments into mobile adaptation were performed with the DateLens application [2]. This is a calendar application for PDAs that utilizes a fisheye transformation to display a calendar application. Highly influenced by DateLens, Karlson et al [9] published an article comparing AppLens, a fisheye application launcher, and a pure zoom application launcher for one-thumbed operation on PDAs, with initial studies revealing that untrained users prefer AppLens. In related work, Engdahl et al. [5] applied treemaps to display discussion groups on PDAs. A user study confirmed that the benefits of using treemaps on traditional displays were retained even when ported to a display only 6% of the original size.

3. Problem Domain

Key performance indicators (KPI) are usually given by numbers confined to a limited range that indicate how well one or more aspects of a business are performing in relation to a strategic business goal. KPI are usually associated with a practice called Corporate Performance Management (CPM). The core of CPM is to define a set of KPI that align with the corporate strategy in a good way. These KPI are calculated from a set of measurements taken within the company, and can be, e.g.: the number of defects per thousand items produced, average customer satisfaction.

The task of measuring KPI can be quite hard. Moreover, the effort of defining good KPI that reflect the strategy of the business is considerable and represents an ongoing continuous process. According to Wade and Recardo [18] a corporation needs a set of around 5 to 10 KPI on a departmental level and possibly 20 or so refined ones at a corporate level. They also claim that one should realistically expect to spend approximately 2 years deriving adequate ones.

At present there exist a wide range of applications for presenting KPIs to end users, but few for handheld devices. Additionally, utilizing focus+context techniques in business related applications is a relatively unexplored area. Indeed, decision makers use KPIs to identify parts of their organizations that need extra attention, so making these conveniently available to them can change usage patterns. This would be one step further on the way to helping business leaders to cope with the rapid change and information overload in today's business environment.

In our work, we have addressed this issue and utilised a novel adaptation of the MagicEyeView to aid in the visualization of KPI on mobile devices.

4. The MagicEyeView

The MagicEyeView visualization [12] is specifically tailored for hierarchies. The MagicEyeView works by laying out a hierarchy on a hemisphere with the hierarchy root at the pole. The hierarchy is then projected down into the plane below. Initially the viewpoint is centered at the pole. Focus on a particular region can be achieved by shifting the viewpoint around on the plane. The space between successive levels in the hierarchy is colored using alternating colors to distinguish levels in the hierarchy. The geometry can be described more formally: each node in the hierarchy is mapped onto a two

dimensional Cartesian plane using Reingold and Tilford's [15] algorithm. These values are then normalized to 2π and described using two angles:

$\varphi = 2\pi * x/maxx$,

$\theta = 2\pi * y/maxy$.

These two angles together with the radius of the hemisphere are then used to calculate the Cartesian coordinates:

$x = r * sin(\varphi) * sin(\theta)$,

$y = r * cos(\varphi) * sin(\theta)$, $z = r * cos(\theta)$

Secondly, a projection is applied from the Cartesian coordinate P0(0,0,0), and the projection rays/vectors from P0 and the intersections with the hemisphere are calculated. The angles between these vectors are kept constant. To change focus, P0 is moved and the new intersection points between the vector set and the hemisphere are calculated and then projected down into the plane. In addition to moving P0, the MagicEyeView can be translated, rotated and zoomed. When moving P0 to change focus the new intersections can be calculated by solving the equation:

$|t*VP1 + VP0| - r = 0$

where VP0 is the vector from the original P0 to the new P0. This gives us the new intersection point from the vector:

$t*VP1 + VP0$.

4.1.Initial Prototypical Implementation

Our initial attempt was to prototype the MagicEyeView for KPI visualization on a Nokia 6680 using a 200MHz ARM9 process capable of 2.1 MIPS. The device had 5MB of memory and a screen size of 176 x 208 pixels, capable of displaying 262144 colours. The prototype implementation was a port of an earlier MagicEyeView implementation created by Petrik [14]. The port was made from J2SE to the J2ME CLDC1.1 plus MIDP2.0 compatible device using the games API with layered graphics support.

The developed prototype (Figure 1) supports focus change by pressing the navigation button on the mobile station. Because of the lack of a pointing device on some devices the focus movement accelerates when the navigation button is pressed continuously. Additionally, the focus point also follows the perimeter of the outer circle in the view if the user indicates navigation beyond this perimeter. These adaptations are implemented to assist the user and to avoid excessive button pressing. Focus is achieved by moving the focus point, as described in the previous section.

Figure 1. Initial Prototypical Implementation

One of the main concerns before implementing the prototype had to do with CPU usage. In practice, however, this turned out to not be an issue as it was possible to render around 30 frames per second (fps) even with a 10ms sleep inserted into the drawing loop to accommodate other threads in the program. As one can clearly see from Figure 1, the MagicEyeView visualization suffered two major problems when adapted to a portable small screen device.

- Because of the different aspect ratio, there is a lot of unused screen real estate.
- The node labels occlude parts of the presentation.

6. The MagicEyeView: An Improvement

To alleviate the problems of skewed aspect ratios identified above, we propose to introduce a half ellipsoid (specifically, half a spheroid) and apply the principles from the MagicEyeView to this geometry setup. A spheroid is expressed by the formula:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{b^2} = 1$$

Calculating coordinates and vectors in the spheroid is a bit more involved but parameterization might be expressed by:

$x = a * sin(\varphi) * sin(\theta)$,

$y = b * cos(\varphi) * sin(\theta)$,

$z = b * cos(\theta)$.

When moving the focus in this geometry the vector formula |t*VP1 + VP0| - r = 0 can no longer be used to find the intersection with the spheroid, because there is no longer a fixed radius. However, we know that the spheroid equation must still hold, so we can

decompose the vector formula and insert it into the equation. This then gives us the following:

$$\frac{(t \cdot x_i + \nabla x)^2}{a^2} + \frac{(t \cdot y_i + \nabla y)^2}{b^2} + \frac{(t \cdot z_i + \nabla z)^2}{b^2} = 1$$

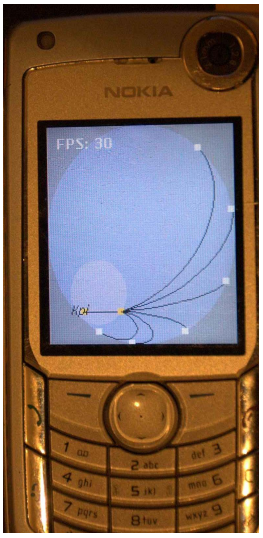

Figure 2. Improved MagicEyeView Algorithm (Nokia 6680)

However, here everything with the exception of t is a known constant. This means that solving the above equation entails the computationally simple task of solving a quadratic equation.

Another thing that must be adjusted to fit the new geometry is the focus-assisting functionality that steers focus to allowed points within the plain. For keypad navigation, the adjustment functions must solve the ellipse formula:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

with respect to y and x respectively :

$$y = \frac{b \cdot \sqrt{a^2 - x^2}}{b}, x = \frac{a \cdot \sqrt{b^2 - y^2}}{a}$$

When using a pointing device it will feel more natural to calculate the angle if the user attempts to focus outside the projected ellipse and adjust the x and y coordinates accordingly.

7. Discussion

We have implemented the modified MagicEyeView algorithm using the Java2 MIDP2.0 API with the CLDC 1.1 configuration on the original Nokia 6680 platform and simulated it on two prototypes with considerably different aspect ratios (Figures 3 and 4): Figure 3 shows an emulator for Nokia Series 60 device having a resolution of 208x176 pixels running the prototype implementation, while Figure 4 depicts the results of our approach on an emulator for the Nokia

Figure 3. Improved MagicEyeView Algorithm (Nokia Series 60)

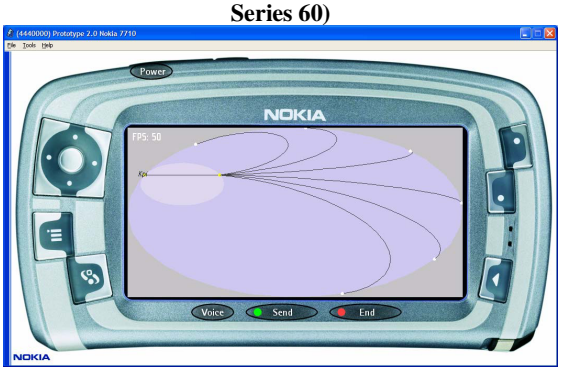

Figure 4. Improved MagicEyeView Algorithm (Nokia 7710)

7710 having a resolution of 320x640 pixels. The respective aspect ratio of these two devices is 1.15:1 and 1:2.

As can be observed, our approach handles the varying aspect ratios of the different devices efficiently and uses the screen real estate fully. Moreover, the issue of node occlusions is considerably alleviated. Lastly, we mention that the frame rate of 30 fps of the original prototype is also kept in our modified implementations.

Although one could achieve similar results using affine transformations (coordinate conversions), such transformations are not covered by the Java2 game API. Because there is no API support, this therefore excludes the use of hardware support when programming within the confines of that API. Moreover, implementing affine transformations will

incur a performance penalty, not to mention some programming complexity, since every pixel that is drawn on the screen must be remapped, a feat not easily done since it entails, for one, timely interruption of the painting calls. For instance, on a 200x300 pixel screen a minimum of an additional 60.000 floating point multiplications per frame is needed, albeit naively, without hardware acceleration or optimizations. Utilizing this approach will thus probably result in accuracy loss in the visualization.
Another disadvantage of affine transforms is that they will also distort the symbols/icons that will be used for pre-attentive recognition. This can be solved be either drawing the symbols in an unconverted layer, or by pre- converting the symbols to counter balance the effect of the coordinate conversion. The disadvantage is that it again complicates matters unnecessarily. Thus, although there is support for this in the jsr-184 specification [8], there is, however, no support for layering. Moreover, the specification is a 3D API, which again complicates matters, since simultaneously mixing content on the device screen from both APIs is a challenge not easily solved.

8. Conclusion

This paper has presented a novel adaptation of the MagicEyeView approach to such devices, by treating the problem as visualization on a spheroid. Our approach has general applicability, as it has been proven to be a resource friendly and complexity reducing alternative to affine transformations, irrespective of the specifications of the target device. Additionally, it has been shown to represent a performance enhancement for screen-limited devices of the MagicEyeView algorithm. Whilst we are encouraged by our results thus far, our future endeavors centre around the encoding of information into node icons and the utilization of preattentive effects for KPI status evaluation.

9. References

[1] Baudisch, P. and Rosenholtz, R. Halo: a technique for visualizing off-screen objects. In *Proc. CHI 2003*, ACM Press (2003), 481-488.

[2] Bederson, B.B., Clamage, A., Czerwinski, M.P. and Robertson, G.G. DateLens: A fisheye calendar interface for PDAs, ACM Trans. Computer-Human Interaction, 11, 1, (2004), 90-119.

[3] Björk, S., Holmquist, L.E., Redström, J., Bretan, I., Danielson, R., Karlgren,J., and Franzén, K. West: A Web browser for small terminals. In *Proc. ACM UIST 1999,* ACM Press (1999), 187-196.

[4] Carpendale, S., Ligh, J. and Pattison, E. Achieving higher magnification in context. In *Proc. ACM UIST 2004*, ACM Press (2004), 71-80.

[5] Engdahl, B. Koksal, M. and Marsden, G. Using treemaps to visualize threaded discussion forums on PDAs. *Ext. Abstracts CHI '05*, ACM Press (2005), 1355-1358.

[6] Furnas, G.W. *The fisheye view: A new look at structured files.* Technical report, Bell Laboratories, 1981.

[7] Furnas, G.W. Generalized fisheye views. In *Proc. CHI 1986*, ACM Press (1986), 16-23.

[8] JSR 184: Mobile 3D Graphics API for J2ME. http://www.jcp.org/en/jsr/detail?id=184

[9] Karlson, A.K., Bederson, B.B. and SanGiovanni, J. AppLens and launchTile: two designs for one-handed thumb use on small devices. In *Proc. CHI 2005*, ACM Press (2005), 201-210.

[10] Karstens, B, Kreuseler, M. and Schumann, H. Visualization of complex structures on mobile handhelds. In *Proc. IMC2003*, 2003.

[11] Karstens, B., Rosenbaum, R., and Schumann, H. Visual interfaces for mobile handhelds. In *Proc. HCI 2003,* Crete, 2003.

[12] Kreuseler, M., Lopez, N. and Schumann, H. A Scalable Framework for Information Visualization. In *Proc. IEEE INFOVIS 2000*, IEEE Computer Society Press (2000), 27.

[13] Mackinlay, J.D., Robertson, G.G. and Card, S.K. The perspective wall: detail and context smoothly integrated. In *Proc. CHI 1991*, ACM Press (1991), 173-176.

[14] Petrik, S. *Magic Eye Viewer - A tool for visualizing large hierarchies*. Technical report, Graz University of Technology, 2002.

[15] Reingold, E. and Tilford, J. Tidier Drawing of Trees. IEEE Transaction on Software Engineering, 7, 2 (1981), 223–228.

[16] Robertson, G.G. and Mackinlay, J.D. The document lens. In *Proc. ACM UIST 1993,* ACM Press (1993), 101-108.

[17] Sarkar, M., Snibbe, S.S., Tversky, O.J., and Reiss, S.P. Stretching the rubber sheet: a metaphor for viewing large layouts on small screens. In *Proc. ACM UIST 1993*, ACM Press (1993), 81-91.

[18] Wade, D. and Recardo, R. *Corporate Performance Management*, Butterworth Heinemann, 2001