ICS2002 / Workshop on Multimedia Technology

Animation with Flexible Flow Control and Dynamic Granularity to Aid Map Navigation on Mobile Devices

Chen-Ning Hsi

Philips Research East Asia -- Taipei

24FA, 66, Sec. 1, Chung Hsiao W Rd., Taipei 100, Taiwan, R.O.C.

Voice: +886 2 2382 3013 / Fax: +886 2 2382 4598

eric.hsi@philips.com

ABSTRACT

Among various mobile applications, map navigation has attracted major attention. However, due to the limited input and output capabilities of mobile devices, optimal usability in map-based applications is usually difficult to realize. This paper describes an animation mechanism to aid map navigation on mobile devices. This animation mechanism can be applied to close the mental gaps between consecutive but discrete views during map navigation to strengthen the navigation context. More significantly, through the partition of the animation generation and rendering process, the animation mechanism makes animation data-driven to enable flexible flow control and dynamic granularity adjustment. Support for fine-grained map navigation under minimal system I/O requirements can therefore be realized, with a good balance between the active system aids and necessary user interaction.

Keywords

Adaptive interaction, animation flow control, cartography, keypad configuration, map navigation

1 INTRODUCTION

In mobile computation, the use of map information in conjunction with other types of information has potentially major applications. Unfortunately, due to the limited display space and constrained input capabilities, it is usually very difficult to realize optimal usability within a map-based application. The application context further complicates this situation by placing additional limitations in one way or another. For instance, in the car navigation system scenario, it will be dangerous if the driver looks at the map for the instructions. It is also difficult, if not impossible, for the user to use her free hand to indicate a location on the map for further detail or information. As a result, it becomes critical to optimize the use of limited system resources, such as display space, computation power, and storage capacity, while respecting the constraints imposed by different application context when implementing map applications in the mobile setting.

The most significant characteristics differentiating mobile devices from desktop devices are the small display space, limited input support, less computing power, and constrained bandwidth and storage. Due to the mobile nature of such devices, application context is also much more dynamic than in the desktop environment. This in turn places extra restrictions and requirements on the mobile applications such as eye-free, single-handed operations, noise-resilient, or privacy-assured interactions. Accordingly, the challenge of a general mobile application lies in fulfilling the restrictions and requirements posted by the application context under the limitations of the device. Specifically, in mobile map applications the key stumbling blocks are the small display and the limited input support due to its visual nature and intensive navigation need. Restricted computing power and storage capacity are also major concerns.

This paper introduces an animation mechanism to aid map navigation to better utilize the display space with the limited input in mobile devices. The main strategy underlying the animation mechanism is based on the common tradeoff between time and space: when certain resources, including storage or screen space, are not sufficient to fulfill all the needs of a task at the same time, it is typical to partition the task into sub-tasks that can be carried out sequentially so that the resources needed for each sub-task will not exceed what is available. Marquee text is one example of displaying text information in a small and fixed space by scrolling animation. The animation mechanism in this paper is unique in 1) the way the created animation is used to aid the navigation task, and 2) the

framework the underlying animation is organized to support the required animation applications. To be more specific, animations are dynamically generated when the user wishes to move from one area to another or to up-scale or down-scale different areas while flexible control over the created animations are used so that navigation to any specific target area of the map in any particular scaling ratio is possible and easy to achieve. This arrangement transforms navigation activity from a purely manual operation into a semi-automatic operation. In addition, the intermediate changes between two originally discrete navigation steps are visualized using animations to orient users within the navigation context to help them avoid getting lost during the navigation. To support this type of animation, the underlying framework is composed of multiple threads synchronized on the shared data.

The rest of the paper is organized as follows. Section 2 describes related research in the areas of electronic map and computer animation. Section 3 describes the map data structure and navigation operations. The multi-threaded animation mechanism and different types of animation controls to support map navigation are introduced in Section 4. Section 5 describes a prototype system. Finally, Section 6 concludes with the future research directions.

2 RELATED RESEARCH

Electronic map research focuses on the underlying map structure, the storage and retrieval of map data, and the map rendering process[5,6,7]. The goal underlying this research is to find suitable data representations for efficient map data storage and retrieval and to search algorithms to render quality map presentation, both on screen and in print. Among other topics, the map labeling is an important and well-studied problem in the area of cartography, which is NP-hard in general. However, good approximate algorithms and heuristics in linear time and polynomial time have been developed for different constrained versions of map labeling[7,12]. These developments have greatly improved the quality, both in presentation and processing, of electronic maps. However, this enhancement mainly focuses on the static map presentation. The task of navigating around an electronic map on a small display, which is highly interactive in nature, is not well supported. In specific, the essential operations during map navigation, e.g. scaling and translation, are usually not refined to help to support navigation actions – locate the places of interest, find the paths to those places, and learn and mark related

locations along the route from the current location to the destination. As a result, map navigation usually requires intensive user interaction. Additionally, during the map navigation, the necessary scaling and translation operations usually introduces the danger that the users will lose navigation context and orientation, resulting in their getting lost within the map. This situation grows more serious in the mobile devices where displays are usually much smaller than those on desktop computers.

There are approaches to deal with the problem of losing context. The first approach is to enhance the visualization such that the viewing context is available by a) providing continuous context or b) making available the context in a map form. Fisheye lens[2,3], distorted views[14,16] and perspective walls[11] are notable examples in the first category; and cone tree[13] and metro map metaphor[15] the second. The second approach is to use animation techniques to bridge the gap between two consecutive but discrete views during the navigation. Graceful degradation techniques during object dragging[18] and animated interface[1] are sample technologies. Although these technologies reduce the chance of getting lost during navigation by providing continuous context or context visualization, they are difficult to implement in mobile devices due to the smaller displays and less available computing power. For any technology to be feasible in the mobile setting, it must take these two major constraints into account. Thus, animation is the chosen approach in this work to aid map navigation based on the following considerations. First, since the display is small and the device has limited computing power, it will be hard to realize any sophisticated visualization techniques to maintain continuous navigation context. Second, when the display is small, the presentation of each display item will unavoidably be simplified, e.g. thin lines oriented map without texture.

As a result, it will be hard to see the relevance, such as depth and enclosure, among different items shown in the display. With respect to the above considerations, animation technology stands out as a feasible approach because a) animation enables tradeoffs between display space and time and b) animation makes visible the relevance of simplified display items[1].

Besides video game and virtual reality applications, animation has been used and shown effective in providing on-line help[4] as well as in visualizing algorithm dynamics[9]. However, depending on the underlying animation

[1] It has shown that the depth and orientation of a simple 3D surface plotting can be visualized by rotating the plotting.

mechanisms, different types of control over the animation might be hard to realize. For example, in a process-driven animation, it is usually difficult to provide the play-backward control. It is also hard to tune the animation progress dynamically, or to pause an animation in any random position and use that as the starting point for another operation. The flexible control is essential when applying animation techniques to aid map navigation in mobile devices. The importance is based on the way the animation assists the map navigation. From the approach proposed in this work, animation is used to assist the map navigation process, especially in its scaling and translating operations. The intermediate changes (from the view of one scaling factor to another or from one anchor position to another) are visualized as animations to provide a continuous and consistent map view. More significantly, to allow users to scale or reposition freely, any intermediate view shown in the animation can be chosen for subsequent navigation operations, and the granularity can be adjusted dynamically during animation. This arrangement allows users to scale and translate a map any way he wants within a continuous navigation context. The underlying animation mechanism to provide these types of navigation aids is detailed in Section 4.

3 THE MAP STURCTURE AND NAVIGATION OPERATIONS

This section describes the underlying map structure and the navigation operations. It is assumed that each map items can optionally contains sub map items. There are two major navigation operations: scaling and translation. Both operations have pre-defined parameters, i.e. a *scaling factor* and a *translation vector*, with animation aids to support fine-grained scaling and translation.

3.1 The Map Structure

A typical digital map is composed of a number of features as its basic components. Our simple scalable map structure, following this convention, contains a number of features. To simplify the map structure for mobile application, however, each feature only consists of a type and type specific attributes, a number of points, a principal name with a number of alternative names, a span, and a number of sub-features. They represent the feature category, boundary (or route), the location labels, the bounding box, and the child features. The feature type, at least one anchor point and label are mandatory with the rest, optional. There are three major feature types: *point features*, *line features* and *range features*. A line feature can optionally contain a number of segments, and a range feature can optionally contain sub-features. As the sub-features of a range feature can

contain sub-features themselves, the top range-feature forms a tree structure termed a *range feature tree*. All top range features and line features with segments are stored in a point-range tree[5] for its efficiency of range index search, which is performed frequently during scaling and translation operations.

3.2 The Scaling Operation

The scaling operation is parameterized on the actual display size, the map dimension, the scaling factor, and an anchor point. The initial screen anchor point is computed so that the map center will be aligned with the screen center while the whole map is maximally displayed within the screen space. The subsequent screen anchor points are determined interactively through map navigation operations. Depending on the scaling factor, a feature might be too small to display. For such a feature, all its descendant features will not be displayed. A reasonable dimension to stop further display for an internal-node feature is 3x3 (pixel based), and 1x1 for a leaf-node feature, on the condition that no other features occupy the surrounding pixels.

The scaling operation is executed in two steps. First, all features are tested for their visibility in the adjusted screen viewing range. The visible features are then processed based on the new scaling factor. For point features and line features, position adjustment and line intersection are calculated. For range features, a typical pre-order traversal along the top range feature tree is performed, starting at the root node, with possible early back-tracks when the feature in the visited node is too small to display or when its bounding box is outside the current visible area.

3.3 The Translation Operation

The translation operation is parameterized on the actual display size, the map dimension, a translation vector, a scale ratio and an anchor point. This operation is performed by shifting the anchor point according to the translation vector. Both scaling and translation operations have pre-defined parameters: the scaling factor and translation vector respectively. To minimize the system input requirements for the translation and scaling operations, the scaling factor and translation vector are pre-defined with a typical keypad as the default input device. The keypad configuration for map navigation as well as the pre-defined parameters for the scaling and translating operation will be described next. This keypad assumption can be relaxed to allow more general interaction.

3.4 The Keypad Configuration For Natural Map Navigation

To navigate the map, i.e. to up scale, down scale or move to another area in the map, a pointing device is usually necessary. A pen with a touch sensitive display or a mouse-like device can be used for this purpose. However, these are not always available in all mobile devices, e.g. a mobile phone. In addition to applying speech commands, this work proposes to configure the keypad for natural map navigation. The central concept is to partition the map according to the keypad layout and to map each key to a map region respectively. The position correspondence between the key location and the map partition can then be employed to support natural navigation.

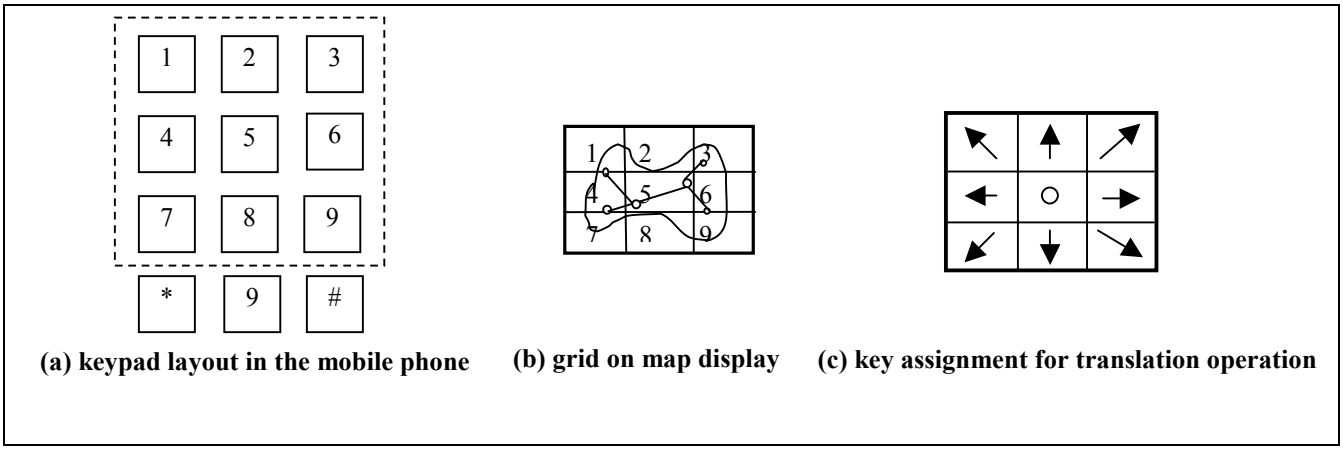


Figure 1 Mapping between the keypad and the map segmentation

A typical keypad arranges the keys in a grid fashion. Usually, it contains a 3x3 rectangular region (the number keys) with some additional keys around that region. Figure 1(a) illustrates the layout in a regular mobile phone and its 3x3 region. Based on the keypad layout, a 3x3 mesh can be placed on top of the map display to form a map grid. Accordingly, for each map region, a key in the respective position can be assigned to interact with that area. This arrangement is shown in Figure 1(b) and 1(c). The map navigation can therefore be realized by click the corresponding key. As shown in Figure 1(c), a key click can be configured to trigger a translation operation. The extra keys in the keypad can be configured to switch among different modes or to reverse operations.

According to a 3x3 keypad configuration, the scaling factors are pre-defined to 3 and 1/3 for upscale and downscale respectively. With the same configuration, the pre-defined translation vectors are [-r, -s], [-r, 0], [-r, s],

[0, s], [r, s], [r, 0], [r, -s], and [0, -r] in Figure 1(c), starting on the upper left corner counter-clockwise, where r and s is 1/3 of the current map width and height.

The keypad can also be configured as 1x2, 2x1, 2x2, 2x3, or 3x2 with the respect to the actual map with the pre-defined parameters adjusted properly. When a pointing device is available, the map grid configuration can be more flexible. For example, the scaling region can be drawn free-handed with the corresponding scaling factor computed based on the size of the bounding box of the drawing region relevant to the display size.

4. THE ANIMATION MECHANISM

As described earlier in this paper, the animation is employed to provide navigation aids in two ways: to maintain continuous navigation context, and to provide natural navigation control. From the perspective of map navigation task, the context is the surrounding areas of the current map region shown in the display. With the limited display space in mobile devices, animation can help to enhance the navigation context in two ways.

First, when a scaling or translation operation is issued, the displays of the map regions before and after the operation can be dramatically different. The perception gap introduced by the hard-to-relate changes between two consecutive map displays therefore introduces the risk of losing navigation context and orientation. Animation to visualize the changes by stepping through them incrementally is therefore helpful and critical.

Second, to enable map display in a small display it is often necessary to simplify map presentation with lines and no texture. Animation moving from one view to another, e.g. a different scaling factor or different anchor point, can therefore enhance the sense of relevance between different map items.

The other way to apply animation to aid navigation is to provide natural and fine-grained navigation control in situations with limited input support. As shown in Section 3.4, a keypad can be configured to navigate the map. However, to take advantage of the correspondence between the keypad and map display for a natural navigation control, the scaling factor and translation vector are restricted to a number of defaults. To alleviate this limitation and to provide finer control, the animation is designed and structured to allow it to be paused in any position in the animation sequence with the ability to start another scaling or translation operation from that paused state. Under this arrangement, any scaling factor and translation vector available in the animation sequence can be used.

Thus, finer control over navigation can be achieved with the degree of control depending on the animation granularity.

According to the above description, the underlying animation mechanisms must satisfy a number of requirements to provide continuous navigation context and fine-grained control under limited input and output supports. These requirements include:

a) flexible animation control,

b) dynamic animation generation,

c) multi-thread,

d) animation state recording, and

e) animation granularity control.

Requirements a and b are important for continuous navigation maintenance, and requirements c, d, and e for fine-grained control.

4.1 Animation With Flexible Flow Control For Continuous Navigation Context

To provide continuous navigation context for scale and translation operations, the intermediate changes between the starting and ending views are visualized with an animation sequence. With respect to the requirements of dynamic animation generation, the underlying process is partitioned into map rendering task and animation generation task with the map data in a shared area. For user interaction, user input is used to trigger the animation generation and to enable animation flow control and granularity control while the timer process drives the map rendering. Figure 2 illustrates the underlying animation framework.

In the framework, the active processes take the external events, user inputs and clock triggers, perform proper arrangement and management tasks, and drive the passive processes appropriately. The passive processes are synchronized on the shared map data and behave as a typical reader and writer model. The underlying logic of the animation generation process is based on the parameters for scaling and translation operations. These parameters include actual display size, the map dimensions, the scaling factor, and an anchor point for both scaling and translation operations, with an additional translation vector for translation. The granularity of the

generation process can be adjusted by configuring the *animation progress factor*. It determines the degree of animation smoothness. For instance, with a progress factor of 10, the generation process will create a 10-step animation from the original state to the end state using linear interpolation. Accordingly, a scaling operation with ratio of 4 and progress of 7, the generated animation steps are those with scaling factor of 1, 1.5, 2, 2.5, 3, 3.5, and 4.

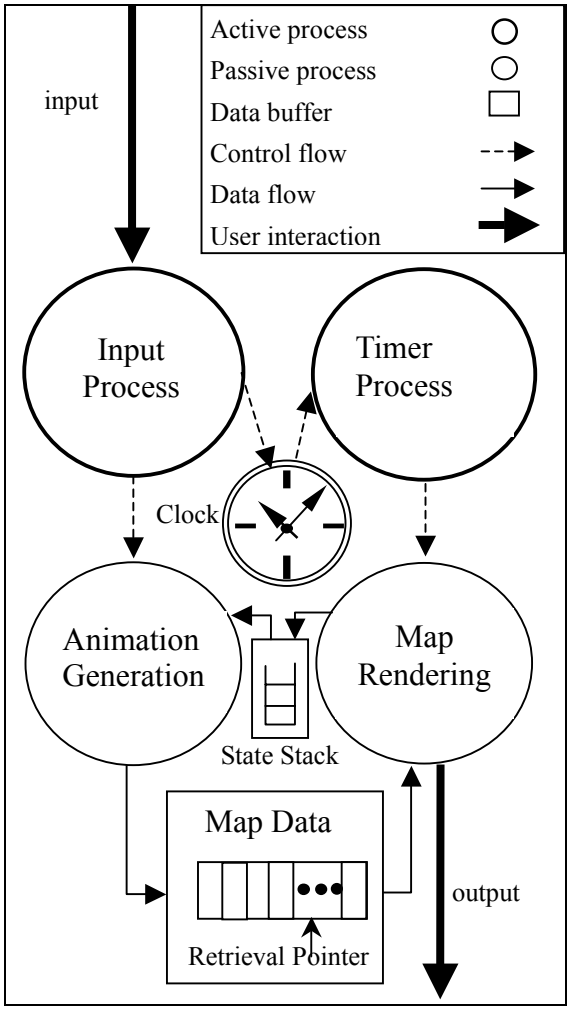


Figure 2 The underlying animation mechanism

There are different ways to provide flow control over the animation, i.e. play, pause/resume, abort, fast forward/backward, jump forward/backward and abort. Firstly, control can be accomplished by controlling the map rendering process, i.e. the output end of the animation. In this approach, typical control operations can be achieved by manipulating the timer, i.e. by adjusting the timer frequency and direction. Secondly, animation

control can be made available by controlling the animation generation process, i.e. the input end of the animation. In this approach, control operations can be achieved by manipulating the map data access, i.e. by locking and changing retrieval pointer of the map data. To support fine-grained map navigation, the second approach is taken as explained in the next section.

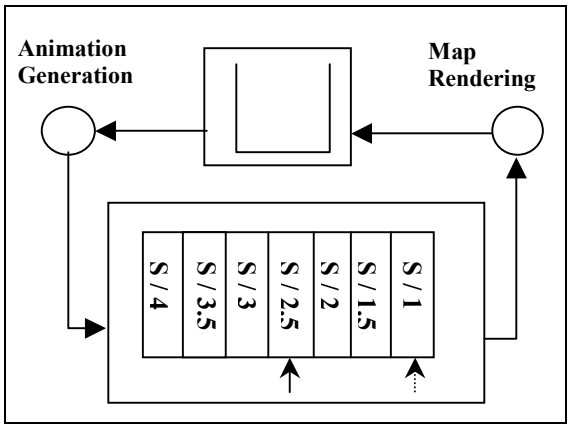
4.2 Animation With State Recording And Dynamic Granularity Adjustment For Fine-Grained Navigation Control

In Figure 2, the animation presentation is partitioned into two separate processes, namely animation generation and map rendering, with the animation base, i.e. the map data, separated to relate these two processes. This results in data-driven rather than process-driven animation. For a process-driven animation, these two processes are usually combined in one animation process with the animation data hidden within the process. While this design has the advantage of simple implementation, i.e. a single thread and single interface to control the overall animation, it suffers from the difficulty in supporting flexible animation flow control such as play-backward or animation pace control. As a result, the data-driven approach is used in order to make available the flexible control of animations to support continuous map navigation context as described in Section 4.1

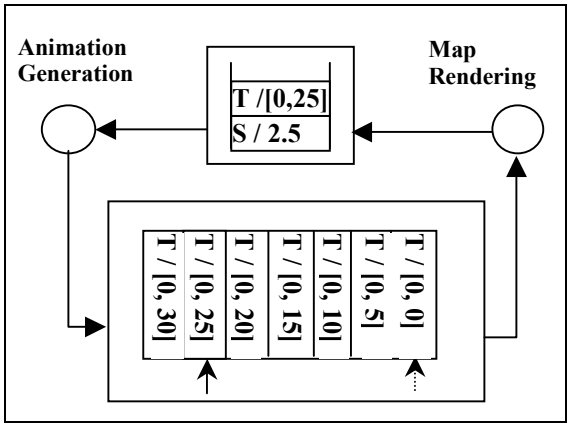
To support fine-grained navigation control, a data buffer, the state stack, is placed between the animation generation process and the map rendering process as shown in Figure 2. Based on the last-in-first-out nature of the map navigation activity, the state stack is designed to record necessary animation states during the map navigation such that finer scaling and translation operations can be supported, instead of simply applying the fixed and default scaling and translation factors. To realize this fine-grained navigation, the map rendering process is designed to record the animation state *after* the animation for a navigation operation is complete or paused and *before* a subsequent navigation operation is issued. The recording is done only when and right before the animation for the subsequent operation is performed. According to this arrangement, the animation flow-control across different navigation operations, e.g. play-backward all the way to the initial state or jump to any previous state, can be realized.

Figure 3(a) and 3(b) illustrate a fine-grained map navigation with default scaling factor of 4, translation vector of (0, 30) and progress factor of 7. As shown in the figure, the animation aids make available a finer set of scaling factor (1.5, 2, 2.5, 3, and 3.5) and translation vectors ([0,5], [0,10], [0,15], [0,20], [0,25]). During the navigation,

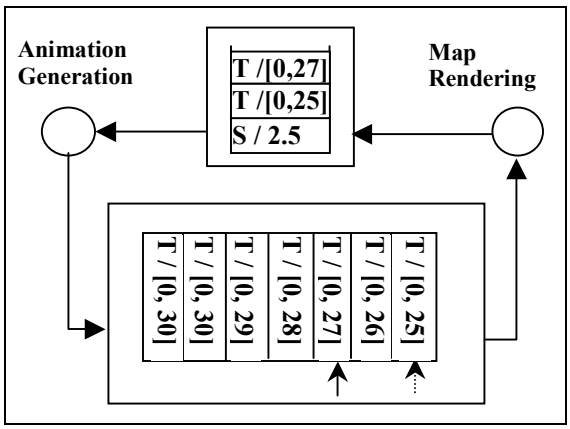
the user 1) issues a scaling operation, 2) pauses at scaling factor of 2.5, 3) issues a translation operation, 4) pauses at translation vector [0,25], and 5) issues other navigation operation.



(a) Scaling operation with ratio 4



(b) Translation operation with vector (0, 30)



(c) Translation granularity control

Figure 3 Fine-grained map navigation

To enhance the navigation granularity further, the progress factor is made adjustable dynamically during the animation. As shown in Figure 3(c), when the animation advances with translation vector of [0, 25], much finer translation vector ([0,25], [0,26], [0,27], [0,28], [0,29], [0,30]) become possible by changing the progress factor from 7 to 31. This can be accomplished by a) making available the animation progress control in the interface and b) properly devising the animation generation process to modify the map-data, synchronizing with the reading activity by the rendering process.

The necessary intermediate animation states are recorded in the state stack. The state information can therefore be used to support subsequent flow control operations such as the play-backward operation to backtrack to earlier animation stage for better orientation or further navigation.

5. The WORKING PROTOTYPE

The animation mechanism to provide map navigation aids is implemented in a prototype system. To enable various experiments, the simulated display can be configured in different size with the keypad as the input. Figure 4 illustrates an office layout map in displays with different sizes. To navigate the map, two operations, scaling and translation, are available. The default scaling factor is 3 and the default translation vectors are [-1/3w, -1/3h], [-1/3w, 0], [-1/3w, 1/3h], [0, 1/3h], [1/3w, 1/3h], [1/3w, 0], [1/3w, -1/3h], and [0, -1/3h], with respect to the keys 1, 4, 7, 8, 9, 6, 3, and 2 as shown in Figure 1, where w and h are the display width and height. As shown in Figure 4, the map is displayed with 9 regions, with each corresponding to a key in the keypad and the focus region highlighted. When scaling, a key press will enlarge the highlighted region by a factor of three. Figure 5 shows the scaling operation on the middle square by pressing the central key, i.e. key 5. Figure 6 displays the translation operation, shifting by [1/3w, 0], by pressing key 6. To provide continuous navigation context and fine-grained navigation control, intermediate states of the scaling and translation operations are visualized by animation, which can be paused at any position to provide a basis for subsequent navigation operations. Figure 7 and 8 shows a number of animation steps for a scaling and a translation operation. Figure 9 illustrates a scaling operation after pausing in the middle of a previous translation operation so that the names of the persons in the top row of office

layout can be browsed. Although user testing is planned to formally assess the degree of usability, through the prototype system the animation demonstrates its efficiency on aiding the map navigation task when flexible flow control and progress control are available.

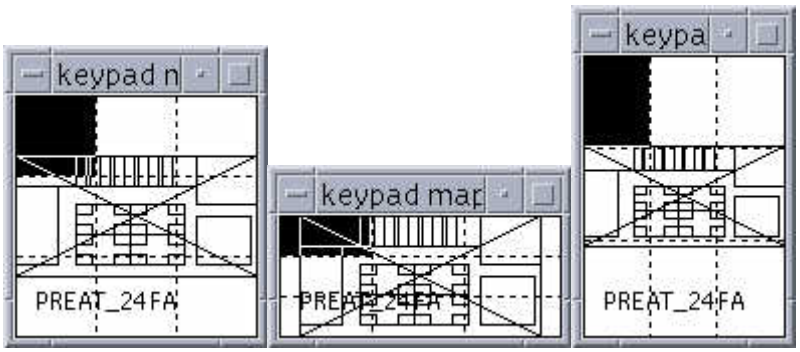


Figure 4 Map display in different screen sizes

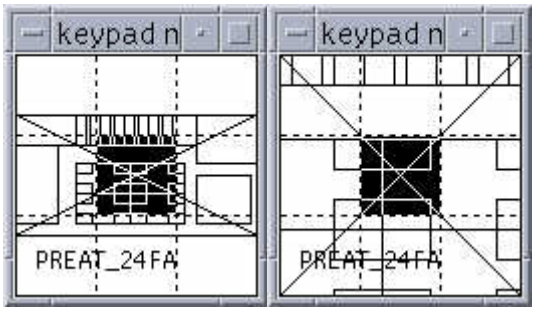


Figure 5 Scaling operation

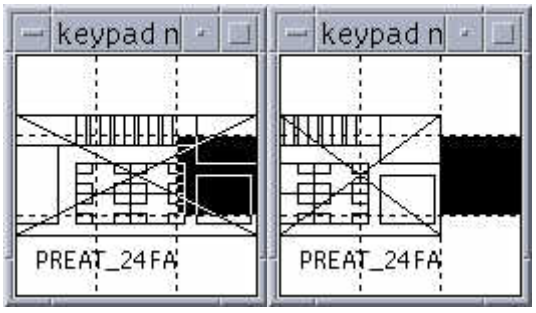


Figure 6 Translation operation

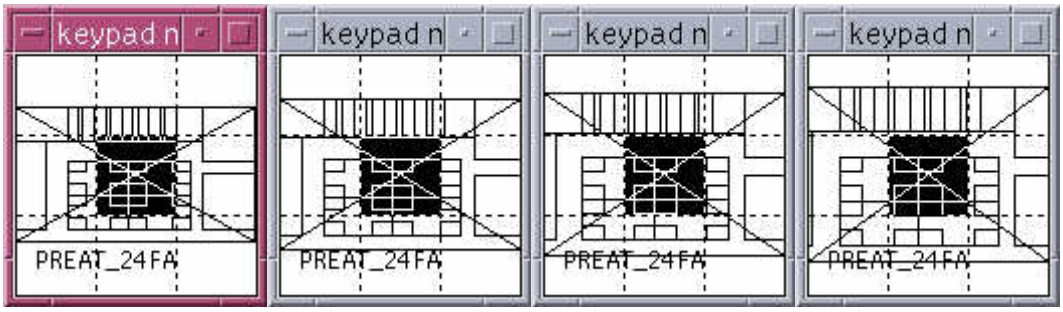


Figure 7 Animation sequence during scaling operation in Figure 5

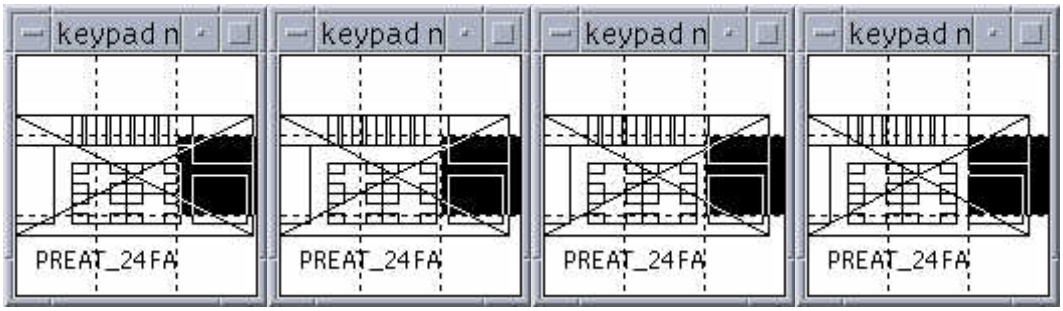


Figure 8 Animation sequence during translation operation in Figure 6

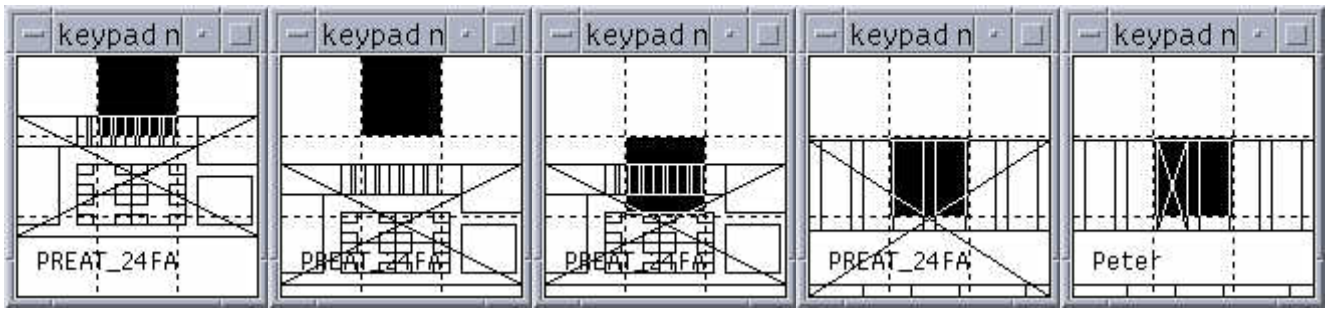


Figure 9 Fine-grained map navigation

6. SUMMARY

In this paper, we have described an animation mechanism to aid map navigation on mobile devices. The navigation operations and the underlying map structure designed to improve the flexibility of map presentation are also described. A keypad configuration utilizing the correspondence between the key position and map

segmentation is also devised for natural map navigation without the additional pointing facility. With the flexible flow control, dynamical progress control, and proper input configuration, this animation mechanism is demonstrated its efficiency in supporting fine-grained navigation task in a natural manner when display space and input support are limited. The overall strategy to derive this space efficient and user-friendly map navigation mechanism is based on the trade-off among time, space, and user interaction, with the goal in balancing between automatic system aids and minimal but still efficient and flexible navigation control.

For future work, user testing is planned to assess usability in a formal manner. The tradeoffs between time, space, and user interaction when employing this animation technique to aid map navigation task will be further studied to seek adaptation strategies to devise systems with variety of system resource availability and constraints in different type of mobile devises.

REFERENCES

1. Baecker, R., and Small, I. Animation at the interface, *The Art of Human-Computer Interface Design*, Addison-Wesley, New York, 1990.
2. Bartram, L., Ho, A., Dill, J., and Henigman, F., The continuous zoom: a constrained fisheye technique for viewing and navigating large information spaces, *ACM UIST'95*, November 15-17 Pittsburgh, PA USA, 207-215.
3. Bederson, B.B., Fisheye menus, *ACM UIST2000*, November 6-8, 2000, 217-225.
4. Bharat, K., and Sukaviriya, P.N., Animating user interfaces using animation servers, *ACM UIST'93*, November 1993, 69-79.
5. Chaabouni, M., and Chung, S.M., The point-range tree: a data structure for indexing intervals, *ACM Computer Science*, February 16-18, 1993, 453-460.
6. Chalmers, D., Sloman, M., and Dulay, N., Map adaptation for users of mobile systems, *The tenth international World Wide Web conference on World Wide Web*, 2001, 735-744.

7. Doerschler, J. S., and Freeman, H., A rule-based system for dense-map name placement, *Communications of the ACM, 35, 1* (1992), 68-79.

8. Horikawa, K., Arikawa, M., Takakura, H., and Kambayashi, Y., Dynamic map synthesis utilizing extended thesauruses and reuse of query generation process, *Proceedings of the 5th international workshop on Advances in geographic information systems*, November 10 - 14, 1997, Las Vegas, NV USA, 9-14.

9. Kehoe, C., Stasko, J., and Taylor, A., Rethinking the Evaluation of Algorithm Animations as Learning Aids: An observational Study, *International Journal of Human-Computer Studies, 54, 2*, February 2001, 265-284.

10. Kurtz, J. L., Damianos, L. E., Kozierok, R., and Hirschman, L., The MITRE map navigation experiment, *ACM Comput. Surv. 31, 2 Article 18*, Jun. 1999.

11. Mackinlay, J.D., Robertson, G.G., and Stuart K.C., The perspective wall: Detail and context smoothly integrated, *Proceedings of ACM SIGCHI'91*, April 1991, 173-179.

12. Poon, C. K., Zhu, B., and Chin, F., A polynomial time solution for labeling a rectilinear map, *Proceedings of the thirteenth annual symposium on Computational geometry*, June 4 - 6, 1997, Nice France, 451-453

13. Robertson, G.G., Mackinlay, J.D., and Card, S.K., Cone Trees: Animated 3D visualizations of hierarchical information. *ACM SIGCHI'91*, April 1991, 189-194.

14. Robertson, G.G., and Mackinlay, J.D., The document lens, *ACM UIST'93*, November 1993, 101-108.

15. Sandvad, E.S., Grønbæk, K., Sloth, L. and Knudsen, J. L., A metro map metaphor for guided tours on the Web: the Webvise guided tour system, *The tenth international World Wide Web conference on World Wide Web*, 2001, 326-333.

16. 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, *ACM UIST'93*, November 1993, 81-91.

17. Sorokine, A., and Merzliakova, I., Interactive map applet for illustrative purposes, *Proceedings of the 6th international symposium on Advances in geographic information systems*, 1998, 46-51.

18. Tang, S.H., and Linton, M.A., Pacers: time-elastic objects, *ACM UIST'93*, November 1993, 35-43.

19. Wagner, F., and Wolff, A., Map labeling heuristics: provably good and practically useful, *Proceedings of the eleventh annual symposium on Computational geometry*, 1995, Vancouver Canada, 109-118.