IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4966

Presenting Large Graphical Contents on Mobile Devices - Performance Issues

R. Rosenbaum
University of Rostock, Computer Science Department, Albert-Einstein-Str.21, D-18057 Rostock, rrosen@informatik.uni-rostock.de

H. Schumann
University of Rostock, Computer Science Department, Albert-Einstein-Str.21, D-18057 Rostock, schumann@informatik.uni-rostock.de

Ch. Tominski
University of Rostock, Computer Science Department, Albert-Einstein-Str.21, D-18057 Rostock, ct@informatik.uni-rostock.de

ABSTRACT

Mobile devices have become popular in recent years, and are more and more used to handle and display large graphics. Due to size and application, these devices suffer from limited resources. Especially if large graphical content must be displayed, the needed system resources often exceed the capabilities of a mobile device. In this paper, we want to give guidelines for the display of graphical content described by raster and vector graphics on mobile devices to allow appropriate and resource-saving implementations. Our considerations are based on the display pipeline of current mobile devices and requirements for appropriate applications. Taking this into account, different approaches, how to handle raster and vector data on mobile devices are discussed, evaluated, and substantiated by concrete tests. To allow a widespread application of the given guidelines, we also investigate the influence of the used development environment. Based on this, conclusions, when to use a certain approach to present large graphical contents on mobile devices, can be derived easily.

INTRODUCTION

The enthusiasm for mobile computing is still unbroken as a growth of 23% in sales of handheld devices and smart phones during the third quarter of 2002 has shown [Co02]. Nowadays, the opportunity to handle multimedia data opens new horizons in ubiquitous computing, and services like MMS (Multimedia-Messaging-Service) show that there is a demand for such offers. Nevertheless, handling graphical data is still expensive regarding resources of the mobile device. Especially if large graphics must be processed the system's limitations are quickly reached.

Although hardware of mobile devices is steadily improved, the main limitations are still the same and can be stated as lack of:

- screen size/resolution and
- processing power.

The relatively small *screen size* is one of the major drawbacks if large graphical content should be displayed. This problem is also related to the provided *screen resolution*, which is also lower than in stationary gadgets. Together this leads to less graphical data which can be displayed. The other bottleneck is lack of processing power. Due to the size of mobile devices, it is not possible to include hardware with similar performance as in stationary devices. This heavily affects the processing time, and therefore, usability of graphical data during presentation and interaction. Beside these, there are several minor limitations, which are beyond the focus of this paper.

Many publications describe the processing of graphical content in mobile environments, but they are rather limited to WWW-browsers [Jos96], interaction issues [Rek96] or other specific problems [ChS01, Ris01, KRS03]. Although there are some interesting approaches, these publications neither describe the actual efforts needed to process the used graphical data nor do they consider the nature of the used content description. Due to limitations of mobile devices this can be of crucial interest since every kind of data is differently processed, which might even render an accepted approach impossible if provided resources are exceeded. Thus, a comprehensive investigation of the processing and display pipeline is necessary to find out benefits and bottlenecks of current mobile hardware regarding the presentation of graphical contents.

To show in which circumstances the use of a certain approach leads to better results during the presentation of large graphical content on mobile devices is the aim of this publication. Our considerations are device-oriented, and problems based on the use of specific media or interaction techniques are only discussed as far as they influence the performance. To affiliate valid statements, in section 2 the display pipeline and the two main approaches to describe graphical content are reviewed. These statements together with derived demands for reasonable exploration of large contents form the basis for our tests and comparisons in section 3. Since there are big differences regarding content description and performance, we close our contribution by giving implementation guidelines for applications presenting large graphical content on mobile environments.

PRESENTING GRAPHICAL CONTENT ON MOBILE DEVICES

Working with mobile hardware causes a lot of problems due to limited capabilities of such devices. In this section we want to review basic steps of the display pipeline for graphical data together with important system and content properties, and requirements which should be fulfilled to allow a convenient and effortless exploration process. This gives us clues for later examinations and statements.

Due to the variety of possible descriptions for graphical content, e.g. by text, picture, video, or combinations of them, we limit our considerations to the most universal classes: raster and vector graphics. In either case before displaying graphics, data must pass a number of steps. The very first is loading the graphical content to memory. Here, properties like *file size* and *file format* play an important role. Since the content is often encoded, loading can be further split in pure file reading and the following decoding in memory. Thus, more detailed statements can be derived about affected properties of the device.

The decoding step transfers the content of a file to an internal memory representation (IMR), which form the basis for later display. There are different approaches for such representations, which can even coexist at same time. The IMR itself is mainly influenced by the properties *image size* and *precision*, but also by the actual image content.

The following display step shows either the whole or only portions of the IMR on display. Hardware-dependent and currently compulsory is the use of a discrete raster display with a certain *screen resolution*. It can be seen as an interface for drawing information. A transformation function is used to map graphical content from a logical coordinate

system to the physical display coordinate system. Kind and properties of the transformation function are determined by the graphical content, user interactions and a number of other points.

Due to the reason, that vector and raster data are based on completely different ways to describe graphical content, they also have different demands to several steps of the display pipeline. Raster graphics are used in areas where a content description by geometrical objects is difficult or even impossible, e.g. in digital imagery. When using raster graphics the image content is described by discrete image points (pixels), which are arranged on a regular 2D-grid of certain *image size* and *precision*. Every pixel is independent from each other regarding its color, which causes a quite large *file size* if using spatially large graphics. Thus, raster data is often stored in compressed representation (e.g. in GIF- or JPEG-format), only sometimes uncompressed (e.g. in BMP-format). The used *file format* heavily influences the time to load the content, whereby formats producing a small *file size* need generally more processing power for decoding, but are faster to read. However, the structure of the resulting IMR, mostly a bitmap, is the same, and no conversion is necessary to map the IMR to display since both are based on the raster approach. Additionally, modifications (e.g. for zooming operations) might be imposed by the transformation function, which influences the presentation quality.

Vector graphics use geometric primitives and their attributes to describe image content. Such primitives are for instance points, lines, rectangles, or circles, with attributes like fill- and stroke-color, or stroke-width. Due to this principle, it is necessary that the graphical content can be described with such primitives. Thus, typical applications for vector graphics are technical drawings or information graphics like charts. Macromedia Flash [WoU02] and SVG (Scalable Vector Graphics) [W3C01] are common *file formats* to store such graphics in mobile environments. Graphical content described by vector primitives is in general smaller than raster data. Thus, *file size* is also smaller, and demand on processing power while loading is less. This might not always be the case, and depends strongly on the *number* and *complexity of primitives*. In contrast to raster graphics, IMR can be rather different for vector data. The most obvious approach is to store a description of the vector primitives as IMR, and to render them directly to screen at display time (*direct drawing*). Again the *number of primitives* and *complexity* are the main properties to consider. To achieve different zoomed as well as panned views, a transformation matrix is preliminarily applied to primitives. Since no information gets lost during the transformation this delivers very good visual results. Unfortunately, it is costly in terms of processing power, especially if the *number of primitives* is large, but can be even worse if *primitive complexity* is high. To reduce the needs, it can be useful to render the whole vector graphic after decoding to an IMR-bitmap and to discard the primitives. Then, IMR, the display step, and thereby affected properties are the same as for raster data. We refer to this approach as *indirect drawing*.

While exploring large raster or vector graphics on mobile devices two main requirements should be fulfilled [RoT03]:

- High presentation quality
- Short presentation and update time

The degree of accomplishment of these requirements varies for raster and vector graphics, hardware capabilities and limitations. Therefore, they determine whether the system resources are used eligibly or not.

In order to explore large graphical content, interaction is essential, and thus, performance is influenced by the used interaction technique. We constrain our considerations to the elementary zoom and pan approach. This is eligible since more sophisticated techniques like fish-eye views are mostly a combination of zoom and pan. Interaction further requires the distinction in presentation and update time. While presentation time spread from loading until displaying the content, update time only considers the time to display the IMR.

In the next section, the appropriateness of vector and raster graphics for presentation of large graphical contents on mobile devices is verified by concrete tests using common development environments on current mobile hardware.

RESULTS

In this section, we present results derived from our experiments with large graphical contents described by raster and vector images in mobile environments. The results are taken by using a Toshiba e740 running with Pocket PC2002. We used different programming environments – MFC (Microsoft Foundation Classes), .Net, and Java to consider their rather varying performance, and to extent the applicability of our statements. Presentation quality, presentation time, and update time are the key points we are focusing on.

Presentation quality

Graphical content has to be presented at high quality in order to allow an easy comprehension. An optimal quality can be achieved if the graphical content can be rendered to the display without loss of information.

For raster graphics such a loss of information occurs for instance if the graphical content must be scaled to fit the screen. Additionally, users are often interested in looking at an image at different levels of detail. In order to create different views, the scaling operation again leads to a loss of information. We got the worst results in presentation quality if build-in system functions are used for image scaling. Especially, downscaling is rather simple and omits pixel rows and/or columns without regard of their color value. Hence, the results are poor. Better results can be achieved by using the more complex filtered scaling, which on the other hand strains the computational abilities of the mobile device. Since panning does not introduce any distortion, image quality is not decreased.

In case of vector graphics, panning and scaling operations (i.e. applying transformation matrices to primitives) are lossless, and therefore, lead to high presentation quality. However, since *screen resolution* is generally low on today's mobile devices there is a slight decrease in quality due to the rasterization necessary before mapping primitives to

Figure 1: *(a) Loading time of graphical content using raster images of different size and format, and (b) pure reading time of vector and raster images with varying content.*

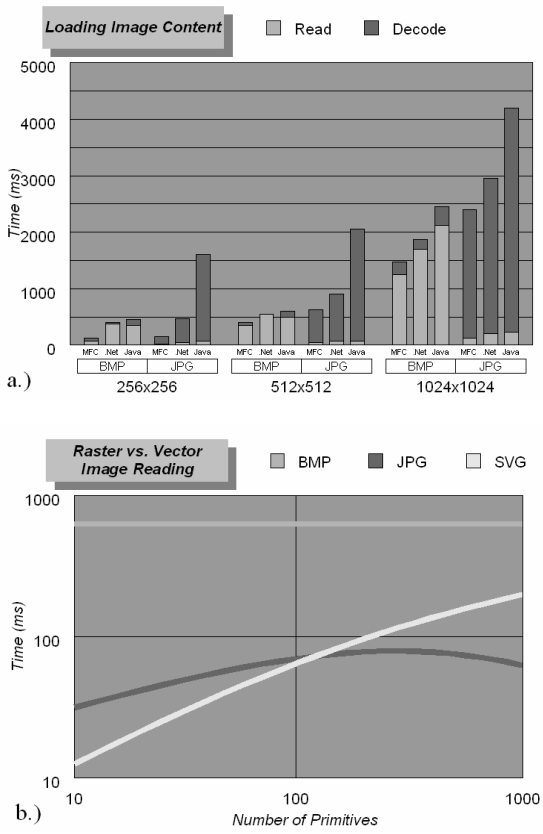
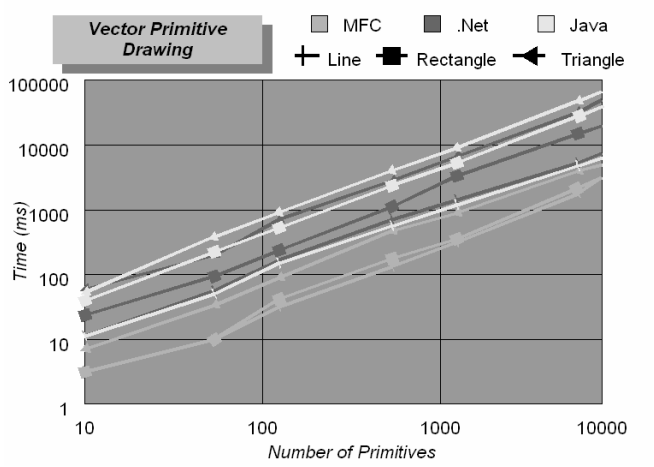


Figure 2: Direct drawing of vector primitives with different number and complexity.



screen. This decrease can be minimized by using anti-aliasing techniques and will not be a problem for future high resolution displays.

Presentation- and update time

Before presenting an image it has to be loaded and mapped to IMR. With regard to loading raster graphics, examples of different *image size* were selected. As shown in Figure 1a, loading time increases linearly with *image size*. Due to the reason that the used *image format* plays also an important role, we measured properties of BMP- and JPEG-images regarding loading time. JPEG-encoded images are much smaller than BMP images, and, as Figure 1a shows, can be read more than 10 times faster. While *file size* of BMP images depends only on *image size*, it may vary for JPEG images. As shown in Figure 1b, if image content changes, compression ratio and therefore *file size* are influenced; this also affects the time needed to read the image.

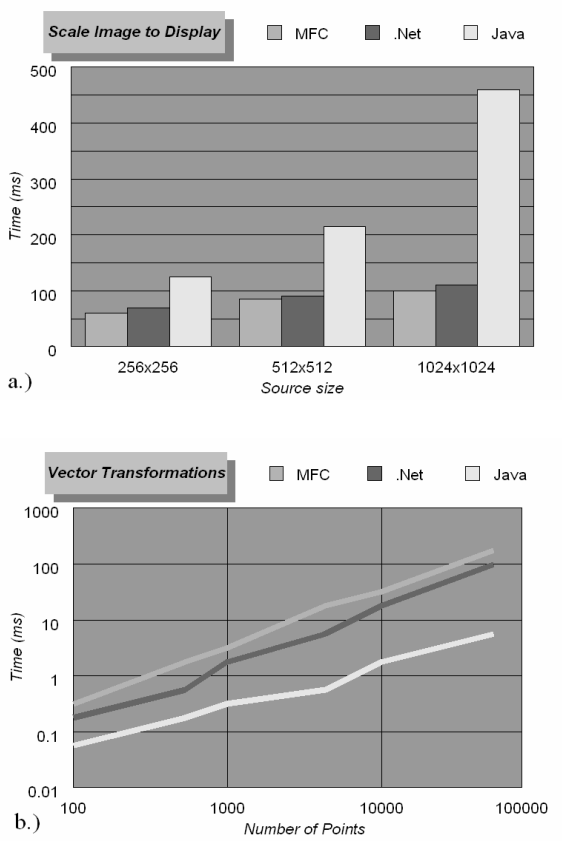
To evaluate loading time, decoding of image content must also be considered. Since image content in BMP files is stored uncompressed, no additional efforts are necessary. Not surprisingly, JPEG-images need significantly more time for decompression than for reading. Thus, loading time for JPEG images increases dramatically and is overall higher than for BMP images. As shown in Figure 1a the testbed implementation using MFC performs best followed by .Net and Java.

For vector images, it is more difficult to derive statements regarding loading time. While for raster images syntax and semantic is inherent in pixel representation, vector image syntax must be parsed and analyzed, before semantic can be added in order to create a valid IMR. Here, loading time depends even stronger on the used *file format*. If using Macromedia Flash, which is based on an optimized and even compressed binary description, files are fast to read. Contrary, the vector format SVG is based on a XML grammar. Here, *file size* is in general bigger, and the processing and interpretation of the file content is expensive. Thus, loading SVG files is innately slower than loading Flash. Due to the absence of a freely available SDK for accessing Flash-Files, we constricted our measures to the reading of SVG-files (cp. Figure 1b). As assumed, *file size* and therefore reading time correlates with the *number of primitives.*

As seen in Figure 1b, time to load raster graphics varies only slightly if graphical content is different. Contrary, loading vector graphics depends strongly on the *number of primitives*. Thus, if only a few primitives are needed, vector graphics are very fast to read. The break-even in our example is reached by using slightly more than 100 primitives. The concrete value also depends on *primitive complexity*. If more or complex primitives are needed to describe the content, e.g. to build a texture, better results are achieved by using raster data.

If the content is available in IMR, it can be drawn to screen. The IMR of raster data can be rendered directly to display. Thus, display pixels must only be set to a certain color and update time mainly depends on *screen resolution.* For the used handheld, processing takes between

Figure 3: Performance of scaling operations for (a) raster- and (b) vector graphics.



50 and 70ms which is rather fast. This update time keeps constant even if *image size* exceeds *screen resolution*. Obviously, if *image size* is lower than *screen resolution*, a faster update time can be achieved.

To simulate common applications for vector images we decided to measure the display time of several kinds of graphical primitives as they might be used in info graphics. For a different number of primitives update times for drawing and filling using *direct drawing* have been measured. As Figure 2 shows, processing time increases with *primitive number* and *complexity*, whereby processing triangles takes most time. There are also differences dependent on the used implementation. Here MFC performs best and is up to 3 times faster than .Net and 4 times faster than Java. .Net has slight advantage over Java. However, the first releases of .Net seam to have some potential for further optimizations especially the graphics part of the framework.

As mentioned, for *indirect drawing* of vector data only the IMR-bitmap must be transferred to screen, and same fast update times as by using raster data can be achieved. Since all primitives have already been drawn to IMR-bitmap (this operation is required only once and needs about the same efforts as required to render the vector graphic directly), update time is completely independent on *primitives complexity*.

Regarding an interactive exploration using zoom and pan, we examined how panning and different scaling operations affect update times. Since panning simply draws a different image part, for raster data there is no difference in update time. Nevertheless, zooming is different and time to transform the content depends heavily on *image size* and *screen resolution.* To show this, we measured the time to scale images of different *image size* to *screen resolution* (Figure 3a). Regarding the programming environments, MFC and .Net achieve similar results, while Java needs more than 4 times longer. Furthermore, complex scaling techniques which offer high quality are of course slower than straight-forward approaches.

Panning and zooming for vector primitives is realized by using a transformation matrix. In order to compare transformation times, we

measured how long such matrix multiplications take. This depends especially on *primitive number*, respectively points. If more point must be transformed, e.g. by high *primitive complexity*, it takes more time to process them. Due to the use of RISC processors, we found that integer vector transformations can be computed rather fast in mobile devices (see Figure 3b). As expected, calculations in double precision are much slower. Again we could get the fastest transformations by using the MFC-implementation.

CONCLUSIONS

Summarizing the paper, we reviewed the display pipeline in mobile devices together with important properties of vector and raster data. Furthermore, we stated demands for an appropriate presentation and exploration of graphical content. Using these requirements, we examined the appropriateness of using raster and vector data to describe graphical content on mobile devices by concrete tests and comparisons. Dependent on situation, statements when to use raster or vector data can be summarized to the following guidelines:

- Graphics loading
- Loading time depends strongly on the used file format.
- Loading raster data is generally fast and requires additional computational efforts if content must be decompressed.
- Loading vector data is fast for up to about 100 primitives.
- Graphics rendering
- Drawing raster data is fast on mobile devices.
- Rendering vector primitives directly to display is generally slow and depends on primitive number and complexity.
- Rendering vector primitives indirectly using an IMR-bitmap achieves faster update times.
- Quality
- A simple scaling of raster data is fast but leads to low quality presentations.
- Scaling vector graphics by integer matrix multiplications is very fast and achieves high quality.
- Development environment
- Implementations based on MFC are fastest.
- .Net has as slight performance advantage over Java on the used mobile device.

Our claim was also to answer in which circumstances raster or vector data are more suitable for presentation of large graphical contents on small mobile devices. We found that both classes have their eligibility depended on the content and external demands, like quality vs. response time. In case of simple graphics, which can be described by a few vector primitives (around 100 in our case), vector graphics are best to reduce constrains caused by limitations of current mobile hardware. Vector graphics offer better quality than raster graphics, especially for scaling operations, and, up to a certain threshold, they are faster to load. As computational power of mobile devices increases in the future, this threshold will surely become higher, which also broadens the application area of vector graphics. On the other hand, our measures show that raster graphics are more eligible if large and complex graphics must be presented at interactive response rates, because their system requirements are content independent. They are even easier to handle, since their nature is not as complex as for vector graphics, and fits to properties of current mobile hardware.

Regarding the used programming environments, we found that applications developed with MFC performed best in all tests. Nevertheless, with the development of faster mobile hardware and even more optimized runtime environments for .Net and Java the advantage will decrease.

REFERENCES

[ChS01]I. Chisalita and N. Shahmehri: “Issues in Image Utilization within Mobile E-Services”, In: P*roceedings of 10th International Workshops on Enabling Technologies*, June, Massachusetts, 2001.

[Co02]CONTEXT – IT Information services, “Newsletter”, London, October, 2002.

[Jos96]A. Joshi, R. Weerasinghe, S.P. McDermott, B.K. Tan, G. Benhardt and S. Weerawarna: “Mowser: Mobile platforms and web browsers”, Bulletin of the IEEE Technical Committee on Operating Systems and Application Environments 8, No.1, 1996.

[KRS03]B. Karstens, R.Rosenbaum, and H. Schumann: “Information Presentation on Mobile Handhelds”, In: *Proceedings of 15th IRMA International Conference*, Philadelphia, 2003.

[Rek96]J. Rekimoto: “Tilting Operations for Small Screen Interfaces”, In: *Proceedings of ACM Symposium on User Interface Software and Technology ‘96*, (Tech Note), 1996.

[Ris01]T. Rist: “Customizing Graphics for Tiny Displays of Mobile Devices”, In: *Proceedings of IPNMD01*, Italy, 2001.

[RoT03]R. Rosenbaum and Ch. Tominski: Pixels vs. Vectors: “Presentation of large images on mobile devices”, In: *Proceedings of IMC’03*, Germany, 2003.

[W3C01]World Wide Web Consortium W3C: “Scalable Vector Graphics (SVG) 1.0 Specification”, www.w3c.org/TR/SVG/, 2001.

[WoU02]S. Wolter and S. Ünlü: “Flash MX - Grundlagen und Praxiswissen”, Galileo Press, Bonn, 2002.