Graphical Content on Mobile Devices

R. Rosenbaum, C. Tominski, and H. Schumann
University of Rostock, Institute for Computer Science
Albert-Einstein-Str.21, D-18057 Rostock, Germany
Tel: +49 381 4987490, Fax: +49 381 4987482
Email: {rrosen, ct, schumann}@informatik.uni-rostock.de

1 Introduction

The enthusiasm for mobile computing is still unbroken as a year on year increase of 51% in the overall global shipments of mobile devices in the fourth quarter of 2004 has shown (Canalys.com, 2005). With 300 million new subscribers in 2004 alone, 27% of the world's population now has access to mobile communications (Svanberg, 2005). A mobile device is a natural multi-functional device and the opportunity to handle multimedia data is only the first step on a long way. Although the development of mobile devices has already made great progress, handling graphical data is still expensive due to the limited resources in mobile environments. Especially if large graphics must be processed limits are quickly reached. However, it is necessary to provide effective and appealing graphical representations for successful m-commerce.

The aim of this publication is to derive major limitations of current mobile hardware and to show how large graphical content can be appropriately processed on such devices. Since visual content can be described either by vector (SVG, Flash) or raster data (Bmp, Gif), both approaches are explained and particular properties are shown. Based on experimental results, this enables us to give guideline for the appropriate handling of large graphical contents in m-commerce applications.

This contribution is structured as follows: in section 2 properties of current mobile devices are reviewed and the displaying pipeline together with basic principles of vector and raster images is explained. These statements form the basis for our tests and comparisons in section 3 and statements for future work in section 4. Since there are huge differences in the performance in the handling of vector and raster images, we close our contribution by giving implementation guidelines for applications presenting large graphical content on mobile devices (section 5). Appended section 6 and 7 serve to provide sources for cited literature and definitions for related terms.

2 Background

2.1 Recent work

Effectively representing information by graphical means is a key issue in m-commerce applications. Many publications describe the processing of graphical content in mobile environments, but they are rather limited to WWW-browsers (Joshi et al., 1996), interaction issues (Rekimoto, 1996) or other specific problems (Rist, 2001, Want et al., 2002, Karstens et al., 2003). Nevertheless, these publications neither describe the actual efforts needed to process the used graphical data, nor do they consider the nature of the content description actually used. Due to limitations of mobile devices, this can be of crucial interest since each kind of data is processed differently, which might even render an accepted approach impossible if provided resources are exceeded. This has been shown in (Rosenbaum & Tominski, 2003) by a comprehensive investigation of the processing and display pipeline of mobile hardware.

2.2 Related Limitations of Mobile Devices

Due to fast progression in this research field, properties of mobile devices change quickly. Thus, current limitations regarding the handling of large images are not the same as a few years ago (cf. (Rosenbaum & Tominski, 2003)). We found, that some constrains still exist (screen size/resolution, processing power), whereas some have strongly decreased (storage space) or have been overcome (lack of colour). To give the reader an impression of current hardware, we compiled a list of important properties of current mobile devices (Table 1). Since classic tablet-PCs or Laptops are more aligned to stationary systems than to light-weight mobile assistants, they have not been considered.

Devices	Display	Resolution	Colours	Processing power	RAM
Mobile:					
Siemens SXG75	2.2”	240 × 320	18bit	-	128MB
Hagenuk S200	2.2”	160x220	16bit	Texas Instruments OMAP 310	32MB
Samsung i600	- -	176x220 128 x 32	16bit -	Intel® PXA250 200MHz	32MB
Palm Treo 650	2.5”	320x320	16bit	Intel® PXA270 312MHz	32MB
Asus MyPal P505	2.8”	240x320	16bit	Intel® PXA272 520MHz	64MB
Qtek 9090	3.5”	240x320	16bit	Intel® PXA263 400MHz	128MB
Palmsize:					
Palm Zire 72	2.5”	320x320	16bit	Intel® PXA270 312MHz	32MB
BlackBerry 7750	3.0”	240x240	16bit	-	16MB
Palm Tungsten-T5 Premium	3.7”	320x480	16bit	Intel® PXA272 416MHz	256MB
Sony Clié PEG-UX50	4.0”	320x480	16bit	Sony CXD2230GA 123MHz	64MB
Handheld:					
Gizmondo Force	2.8”	240x320	-	Samsung ARM9 S3C2440 400MHz + GPU Nvidia Goforce 3D 4500	64MB
Fujitsu Siemens LOOX 720	3.6”	480x640	16bit	Intel® PXA272 520MHz	128MB
Sharp Zaurus SL-6000	4.0”	480x640	16bit	Intel® PXA255 400MHz	128MB
Toshiba Pocket PC e830	4.0”	480x640	16bit	Intel® PXA272 520MHz	128MB
Dell Axim X50v	3.7”	480x640	16bit	Intel® PXA270 624MHz + GPU Intel® 2700G - 16MB	196MB
HP iPAQ HX4700	4.0”	480x640	16bit	Intel® PXA270 624MHz	128MB
Sony VAIO U71	5”	800x600	24bit	Intel® Pentium® M733 1100MHz + GPU Intel® 855GME – 64MB	512MB
Stationary PC:					
generic	21”	2048x1536	32bit	Intel® P4-570J 3800MHz + GPU Nvidia GeForce 6800 Ultra	1 GB

Table 1: Specifications of different mobile devices (03/2005).

Screen dimension:

The relatively small screen dimension is one of the major drawbacks of mobile devices if large graphical content is to be presented. The display size of current devices varies dependent on the respective device class and spreads from 2 to 5 inches screen diagonal, which is by far less than what is offered by common stationary devices. Thus, the available space for presenting images is very limited and most parts of the content might be hidden. Interestingly, some mobiles offer a second, smaller display to show additional data.

Screen resolution:
The pixel density of mobile displays is rather high. Some devices offer a resolution of 800x600 by a screen dimension of only 5 inches. This leads to a very detailed presentation of the content. Nevertheless, this property is limited by the human visual system, which can resolve visual content only up to a certain extent (Hubel, 1988, Wandell, 1995). Thus, the provided space to display data is and will be by far lower compared to stationary gadgets.

Processing power:
The appropriate handling of graphical content is heavily affected by the available processing power. Due to the dimensions of mobile devices, it is not possible to include hardware with performance similar to stationary devices. This is mainly due to the limited energy supply. Although the speed of current systems has increased a lot, the performance regarding data processing is innately much slower. An interesting trend is the use of additional GPUs to improve the processing of graphical data. Currently, they are only provided by two devices specialized on gaming (Gizmondo) or highly detailed output (Sony VAIO), but are expected to be supported by other devices too. The advent of such additional peripherals will strongly enhance the handling of visual content in the near future (Rasmusson et al., 2004).

Other limitations:
As predicted in (Rosenbaum & Tominski, 2003), nowadays mobile devices offer by far more storage space than two years ago (cf. table 1). Thus, limitations in image handling are mostly overcome and might only occur if many images must be handled or stored at same time. Minor limitations, as user interaction and data transmission, do not influence content presentation and are considered to be out of scope for this publication.

2.3 Presenting Graphical Content on Mobile Devices

Working with mobile hardware causes a variety 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 properties and requirements which should be fulfilled to allow a convenient and effortless exploration process. This gives us clues for later examinations and statements.

Before graphical content can be shown on screen, it must pass the display pipeline. First it must be loaded 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. Based on this, more detailed statements can be derived about affected properties of the device. After this, the content is stored in an internal memory representation (IMR) forming the basis for the later display. There are different approaches for such representations, which can even coexist at the same time. The IMR itself is mainly influenced by the properties *image dimension* and *precision*, but also by image content. The final pipeline step shows the whole or partial IMR on screen. Here, all current devices make use of a discrete raster display with a certain *screen resolution.* Since there are rather different approaches for the IMR, their demands on the display step vary heavily and are worth to be examined.

Arbitrary graphical content can be described by either raster or vector graphics. Due to the fact that each approach uses completely different ways to describe graphical content, they require different resources at the respective pipeline steps.

Raster graphics are used in areas where content is rather complex, e.g. in digital photography. When using raster graphics, image content is described by pixels arranged on a regular 2D-grid of certain *image dimension* and *precision*. Each pixel is independent from others regarding its color, which causes a quite large *file size* if using spatially extensive graphics at high precision. Thus, raster data is often stored in compressed representation (e.g. in PNG- 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. Nevertheless, additional modifications (e.g. for zooming operations) might influence the presentation quality.

Vector graphics use simple geometric primitives and their attributes to describe image content. Due to this principle, it is necessary that the graphical content can be appropriately described by such primitives, e.g. in technical drawings. Content described by vector primitives is often smaller than raster data. Thus, *file size* is also smaller, and the demand for processing power while loading is little. This might not always be the case, and depends strongly on the *number* and *complexity of primitives*. In contrast to raster graphics, the IMR can be rather different for vector data. The most obvious approach is to store a description of the vector primitives and to render them directly to screen at display time (*direct drawing*). Here, the *number of primitives* and *complexity* are the main properties to consider. To achieve differently zoomed as well as panned views, a transformation matrix is preliminarily applied to primitives. By doing so, no information is lost and the visual appearance is very good. Unfortunately, this is costly in terms of processing power, especially if the *number of primitives* is large, and can be even worse if *primitive complexity* is high. To reduce these needs, it might be useful to render the whole content after decoding to an IMR-bitmap. Thus, similar results as for raster data can be achieved for display time. We refer to this approach as *indirect drawing*. Common *file formats* to store vector graphics in mobile environments are Macromedia Flash (Besley & Bhangal, 2003) and SVG (Scalable Vector Graphics) (W3C, 2001).

It has been proposed that while exploring large raster or vector graphics on mobile devices, two main requirements should be fulfilled (Rosenbaum & Tominski, 2003):

- High presentation quality
- Short presentation and update time

The degree of accomplishment of these requirements varies for raster and vector graphics, hardware capabilities, and user interaction. Regarding interaction, we constrain our statements to the elementary zoom and pan approach since more sophisticated techniques are mostly a combination of the different options offered by this method. Interaction further requires the distinction in *presentation* and *update time*. While presentation time spreads from loading until displaying, update time only considers the time to display the IMR.

3 Main Discussions

In this section, we present results derived from our experiments with large graphical contents described by raster and vector images in m-commerce environments. The results are taken by using a modern LOOX 720running with Pocket PC2003 and providing average performance compared to the devices listed in table 1. To extent the applicability of the statements, we used different programming environments – MFC (Microsoft Foundation Classes), .Net Compact Framework, and Java to also consider their rather varying performance. *Presentation quality*, *presentation time*, and *update time* are the key points we are focusing on.

3.1 Presentation quality

A high presentation quality is mandatory for creating successful m-commerce applications. Such high quality representations can be achieved if the graphical content can be rendered to the display without loss of information.

For raster graphics some loss of information occurs if content must be scaled, e.g. for zooming. If so, we got the worst results if build-in system functions were used. Especially when downscaling, pixels are simply omitted without regarding their color. 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.

In case of vector graphics, panning and scaling operations are lossless, i.e. do not cause loss of information. However, there is a slight decrease in quality due to the rasterization necessary to map primitives to IMR/screen. This might be reduced by using anti-aliasing techniques, which again come at a computational cost.

3.2 Presentation- and update time

Fast access to graphical information is another issue in m-commerce applications. Before presenting an image it must be loaded and mapped to IMR. To meaningfully compare raster graphics, examples of different *image dimension* were selected. As shown in Figure 1a, loading time increases linearly with *image dimension*. Since the used *image format* does strongly influence processing time, we measured properties of BMP- and JPEG-images. JPEG-encoded images are much smaller than BMP-images, and can be read more than 10 times faster. While *file size* of BMP-images depends only on *image dimension*, it might vary for JPEG-images. If image content changes, compression ratio and *file size* are influenced, which also affects the time to read the image (cf. figure 1b).

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, in overall higher, than for BMP-images. As shown in Figure 1a the testbed implementations based on MFC and .Net achieve similar results and outperform Java by a factor of about 2.

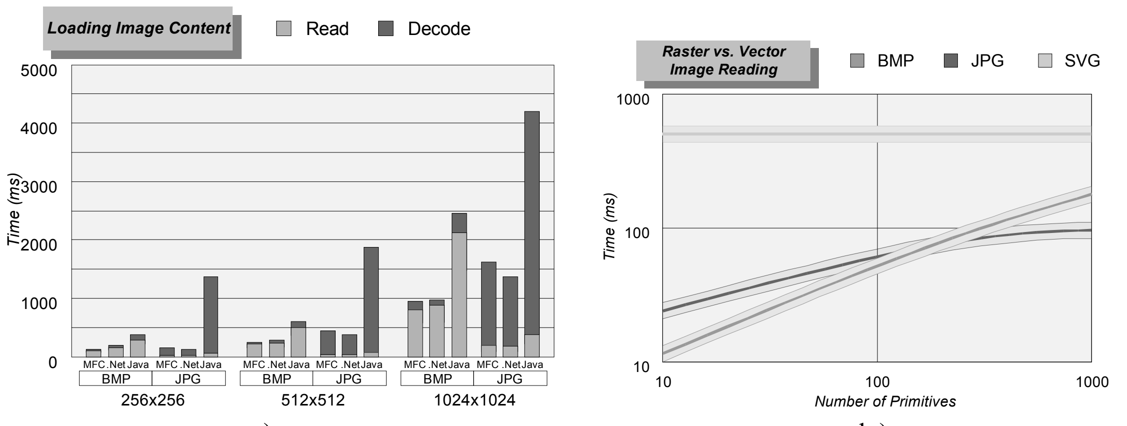


Figure 1: (a) Loading time of graphical content using raster images of different dimension and format, and (b) pure reading time of vector and raster images with varying content.

For vector images, it is more difficult to derive statements regarding loading time. While for raster images creating an IMR is straightforward, vector images must be parsed and analyzed. 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, SVG is based on XML grammar, and thus *file size* is bigger and processing the content is more expensive. Hence, loading SVG-files is innately slower than loading Flash. Due to the absence of a freely available SDK for accessing Flash-files, we restricted our measurements to the reading of SVG-files (cf. Figure 1b). As assumed, reading time is highly correlated with the *number of primitives*.

As shown in Figure 1b, the time to load raster graphics is only loosely coupled with the content. Contrary, loading vector graphics depends strongly on the *number of primitives*. Thus, if only a few primitives are processed, vector graphics are faster to read than raster graphics. The break-even in our example is reached by using slightly more than 150 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.

When the content is available in IMR, it can be shown on screen. As described above, the IMR of raster data can be rendered directly to display. Thus, update time mainly depends on *screen resolution*. For the handheld used, processing takes approximately 130ms in all tested environments. This update time stays constant even if *image dimension* exceeds *screen resolution*. Obviously, if *image dimension* is lower than *screen resolution*, a faster update time can be achieved. This also applies for *indirect drawing* of vector data, where only the IMR-bitmap must be transferred to screen (cf. figure 2). Thus, the same fast update times can be achieved. Since all primitives have already been drawn to the IMR-bitmap and been discarded, update time is completely independent from *primitives complexity*.

To simulate common applications for *direct drawing* of vector images, we measured the display time of several types of graphical primitives. As shown in figure 2, update time increases depending on *primitive number* and *complexity*, whereby drawing triangles takes the most time. There are also differences depending on the used programming environment. Here, MFC performs best in most of the tests, but the advantage over .Net is often marginal. Contrary, Java is up to twice as slow.

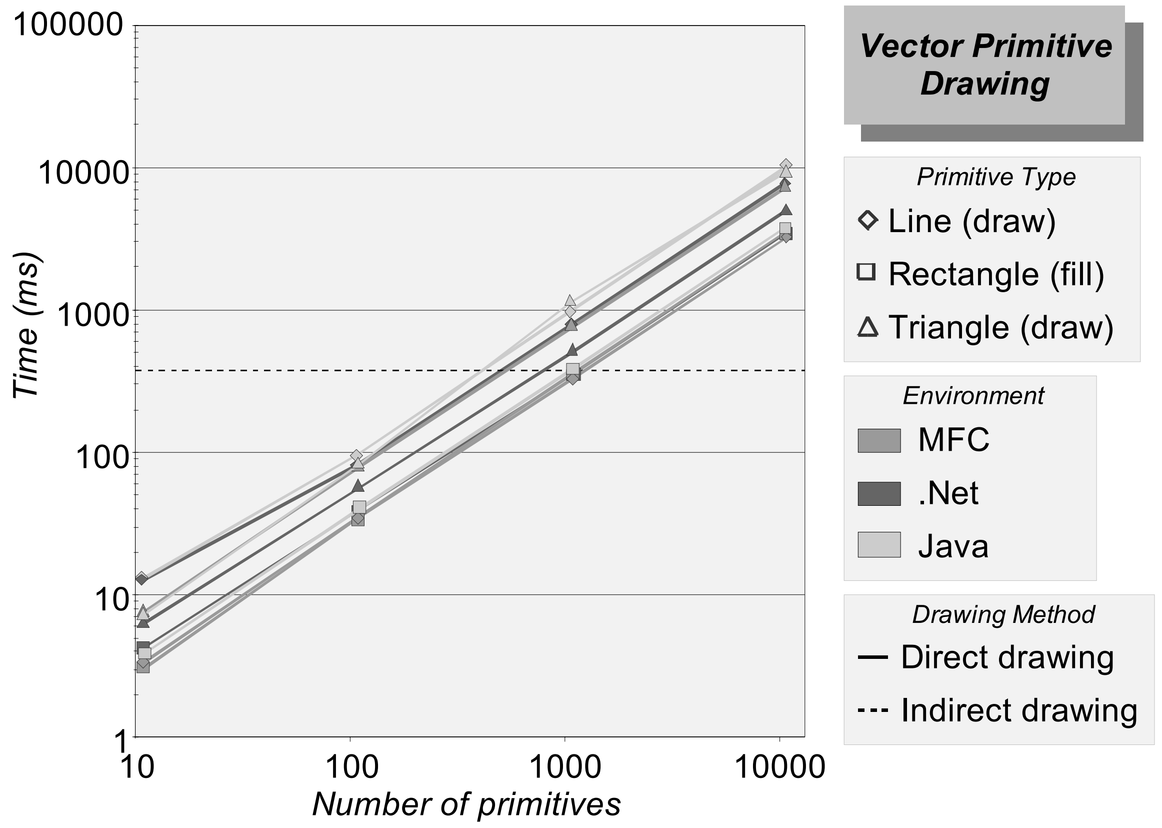
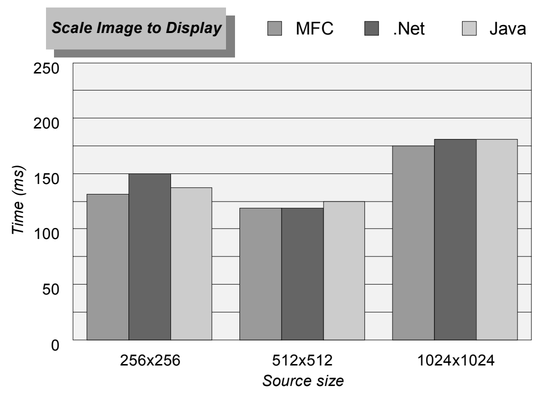


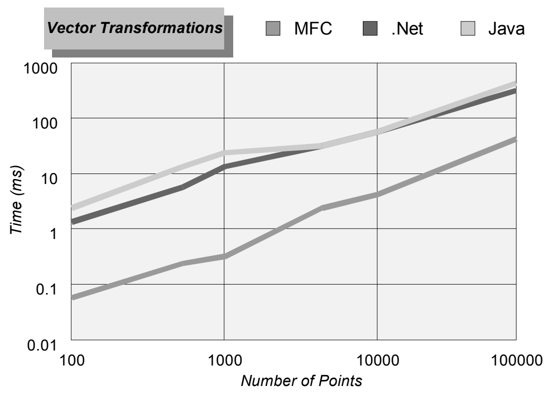
Figure 2: *Direct drawing* of vector primitives with different complexity.

Regarding an interactive exploration using zoom and pan, we examined how panning and different scaling operations affect update times. Since for panning simply a different image part is shown, there is no difference regarding update time for raster data. However, zooming is different and time to transform the content depends heavily on *image dimension* and *screen resolution.* To show this, we measured the time required to scale images of different *image dimension* to *screen resolution* (Figure 3a). Surprisingly, all programming environments achieve similar results. Nevertheless, more complex scaling techniques are slower than straightforward approaches.

Panning and zooming for vector primitives is realized by using a transformation matrix. In order to compare scaling, we measured how long such matrix multiplications take. This depends especially on *primitive number*, respectively points. If more points must be transformed, it takes more time to process them. As shown in figure 3b, integer vector transformations can be computed rather fast on mobile devices. As expected, calculations in double precision are much slower. The fastest transformations are achieved by the MFC-implementation.



a.)



b.)

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

4 Future Trends

As already shown for the last 3 years the mobile sector is subject to rapid development of the belonging hardware. Thus, computational power of mobile devices will also increase and the application area of vector graphics will surely expand. The inclusion of additional GPUs in current devices seams to back this outlook. Furthermore, more sophisticated compression approaches for raster images, as JPEG2000, require more computing power to be processed. Nevertheless, they provide better performance, and even more interesting, many additional features for interactive imaging. Thus, our conclusions given in the next section might still be valid for the near future.

5 Conclusion

Summarizing, we stated the properties of current mobile hardware to show still existing limitations of such devices. This gave us the motivation to review the displaying pipeline together with important properties of vector and raster data in order to derive statements for an appropriate handling of graphical content in such environments. Based on this, concrete tests and comparisons have been conducted and lead to the following statements when to use raster or vector data in m-commerce applications:

- *Graphics loading*
    - Loading time depends strongly on the used file format.
    - Loading raster data is generally fast but requires additional computational efforts if content must be decompressed.
    - Loading vector data is fast for up to about 150 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 fast update times.
- *Quality*
    - 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 and Java implementations often achieve a similar performance

Our claim was also to answer in which circumstances raster or vector data are more suitable for presentation of large graphical contents on mobile devices. We found that both classes have their eligibility depending on the content and external demands, like quality vs. response time. In case of simple graphics, which can be described by less than 150 primitives, vector graphics performed best. Vector graphics also offer better quality than raster graphics. On the other hand, our measures show that raster graphics are better suited if large and complex graphics must be presented, since their system requirements are content independent. They are simply not as complex as vector graphics, and thus, easier to handle. Due to this, they fit to current mobile hardware, and we favor raster graphics to describe graphical content in such environments.

6 References

Canalys.com (2005). Global mobile device shipments hit new peak in Q4 2004. Canalys research release 2005/012, Reading, UK, http://www.canalys.com/pr/2005/r2005012.pdf

Hubel, D. (1988). Eye, Brain and Vision. Scientific American Library.

Joshi, A., Weerasinghe, R., McDermott, S.P., Tan, B.K., Benhardt, G. & Weerawarna, S. (1996). Mowser: Mobile platforms and web browsers. Bulletin of the IEEE Technical Committee on Operating Systems and Application Environments. 8(1).

Karstens, B., Rosenbaum, R. & Schumann, H. (2003). Information Presentation on Mobile Handhelds. Proceedings of 15th IRMA International Conference. Philadelphia.

Rasmusson, J., Dahlgren, F., Gustafsson, H. & Nilsson, T. (2004). Multimedia in mobile phones - the ongoing revolution. Ericsson Reviews 2/2004.

Rekimoto, J. (1996). Tilting Operations for Small Screen Interfaces. Proceedings of ACM Symposium on User Interface Software and Technology. Seattle.

Rist, T. (2001). Customizing Graphics for Tiny Displays of Mobile Devices. Proceedings of International Workshop on Information Presentation and Natural Multimodal Dialogue. Verona, Italy.

Rosenbaum, R. & Tominski, C. (2003). Pixels vs. Vectors: Presentation of large images on mobile devices. Proceedings of International Workshop on Mobile Computing. Rostock, Germany.

Svanberg, C.H. (2005). Ericsson fourth quarter report 2004. Ericsson Financial Reports.

W3C, World Wide Web Consortium. (2001). Scalable Vector Graphics (SVG) 1.0 Specification. http://www.w3c.org/TR/SVG/

Wandell, B.A. (1995). Foundations of Vision. Sinauer Associates, Inc.

Want, R., Pering, T., Danneels, G., Kumar, M., Sundar, M. & Light, J. (2002). The Personal Server: Changing the Way We Think about Ubiquitous Computing. Proceedings of 4th International Conference on Ubiquitous Computing. Göteborg, Sweden.

Besley, K. & Bhangal, S. (2003). Foundation Macromedia Flash MX 2004. Friends of ED. Berkely.

7 Terms and Their Definition:

Mobile devices: gadgets not needed to be attached for a certain place, mostly subject to certain →*limitations*

Graphical content: information represented by an image, perceived via the human visual system

Raster graphics: approach for the description of →*graphical content* by colour points arranged on a regular pixel grid

Vector graphics: approach for the description of →*graphical content* by →*vector primitives*

Vector primitive: geometrical object with certain properties, mostly belonging to one of the basic classes: point, line or triangle.

Displaying pipeline: collective term for the different stages which must be passed to show →*graphical content* on screen

Update time: time necessary to draw →*graphical content* to display, the update step is part of the →*displaying pipeline*

Image format: used to permanently store →*graphical content*, based on the use of →*raster graphics* or →*vector graphics*

Limitations (of →*mobile devices*): hardware restrictions mostly imposed by the application area, main limitations are processing power, screen dimensions and transmission bandwidth.