STREAMING MPEG ANIMATIONS ON MOBILE DEVICES

A MAJOR QUALIFYING PROJECT REPORT:
SUBMITTED TO THE FACULTY
OF THE
WORCESTER POLYTECHNIC INSTITUTE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF BACHELOR OF SCIENCE
BY

ZHUO YAO CHEN
ARAM ARMEN DUYLAN

DATE: JANUARY 17, 2005

APPROVED:

PROFESSOR EMMANUEL O. AGU, MAJOR ADVISOR

PROFESSOR MARK L. CLAYPOOL, CO-ADVISOR

Abstract

This report outlines our progress of porting an existing MPEG-4 player written in C++ for personal computers to a mobile phone coded in a Java programming environment. We focused on porting animation rather than audio and video due to analyzed bottlenecks. We also researched MPEG-4 and existing software. The end results are two programs. The first program parses MPEG-4 graphic files and returns a list of command calls. The second program reads the calls and uses them to render graphics image on a mobile phone.

Acknowledgements

We wish to express our utmost appreciation to the following individuals and agencies:

Professor Emmanuel Agu and Professor Mark Claypool for their comments and contributions.

The members of IBM alphaWorks for their quick reply to our questions about MPEG-4 and their MPEG-4 Toolkit.

The community at Java's Developers Forum for their help and suggestions about J2ME programming.

GPAC for their open source MPEG-4 application, without which, none of this would be possible.

Last, but not least, we would like to thank our parents for their ever growing support in our journey through life.

Table of Contents

List of Tables

List of Figures

List of Acronyms

3GPP – 3rd Generation Partnership Project
API – Application Programming Interface
BBC – British Broadcasting Corporation
BIFS – Binary Format for Scenes
CDMA – Code Division Multiple Access
CDC – Connected Device Configuration
CLDC – Connected Limited Device Configuration
DCT – Discrete Cosine Transform
DVD – Digital Video Disc
GIF – Graphics Interchange Format
GPRS – General Packet Radio Service
GUI – Graphical User Interface
HTML – HyperText Markup Language
HTTP – HyperText Transfer Protocol
HTTPS – HyperText Transfer Protocol Secure
IO – Input / Output
ISMA – Internet Streaming Media Alliance
ISO – International Organization for Standardization
JAXP – Java API for XML Processing
J2ME – Java 2 Platform Micro Edition
JPEG – Joint Photographic Experts Group
JVM – Java Virtual Machine
MIDI – Musical Instrument Digital Interface
MMS – Multimedia Messaging Service
MP3 – MPEG-1 layer 3
MPEG – Motion Picture Experts Group
OS – Operating System
PDA – Personal Digital Assistant
PDC – Pacific Digital Cellular
RTSP – Real Time Streaming Protocol
SMS – Short Messaging Service
VCD – Video Compact Disc
VRML – Virtual Reality Modeling Language
WAP – Wireless Application Protocol
WML – Wireless Markup Language
XML – eXtensible Markup Language
XMT – eXtensible MPEG-4 Textual Format

Executive Summary

The introduction of mobile devices have changed the way people live their lives, communicate, and do business. Mobile phones in particular have had the ability to reach all demographics across the world. Not limited by any boundaries, mobile phone users can contact other mobile phone users at the touch of a button. As the consumer base grows larger there is the demand for new phones with more features.

Manufacturers tried to meet these demands with a mix of technologies including, text messaging, multimedia messaging, and web access. These services did not attract customers because they were cumbersome to access, limited in its functionality, and was overpriced. With the introduction of more powerful phones there needs to be a standard that can provide users with the content that they want. MPEG-4 could be the media format that can solve manufacturer's problems across the board.

MPEG-4 was developed by the Motion Picture Experts Group to address the need for reliable transmission and delivery of content rich media. MPEG-4 provides many features that make it a viable format for years to come. A few of its features include support for audio, video, streaming, 2D and 3D animation, and interactivity. This extensive list of features covers most conceivable applications for the near future.

Java 2 Micro Edition is a reduced Java Virtual Machine (JVM) developed by Sun Microsystems. It is a programming environment with Application Programming Interfaces (APIs) that are designed specifically for small embedded devices. Currently, the manufacturers of mobile phones do not support MPEG-4 but they do support Java 2 Micro Edition.

Our objective was to create a MPEG-4 2D animation player on the J2ME platform for mobile phones. To do this we identified existing MPEG-4 applications that we could use as guidelines. The application that we modeled our player after was GPAC. GPAC is an open source MPEG-4 player that is currently being developed by Jean Le Feuvre. GPAC offers robust features, and in particular 2D animation.

Work began on the conversion of GPAC's C++ code to J2ME. The process proved to be difficult due to lack of technical documentation of the MPEG-4 standard. The final implementation was a combination of tools. It consisted of a heavily modified version of GPAC, a client program, and partially converted code. The modified version of GPAC was used to parse and read MPEG-4 files and output command calls that were used to define the MPEG scene. The command calls were then compressed and stored on a web server in a raw binary format. The client downloads the command calls from the web server, decompresses it, and renders the output on a mobile phone. The partially converted code, once complete, will replace the modified version of GPAC and combine with the client to form a fully functional MPEG-4 application.

1 Introduction

The introduction of wireless technology has changed the daily lives of millions of people. The cell phone market in particular, is one of the world’s fastest growing markets, with over 170 million units sold worldwide in the third quarter of 2004 alone (ITFacts.biz, 2004). Cell phones have become a common accessory spanning across geographical, financial, age, gender, and other boundaries. As the popularity of such mobile devices grows, so do the expectations of what such a device should offer. No longer satisfied with the ability to make and receive calls, mobile phone users expect their phones to allow them to access various information and entertainment sources, as well as to provide extensive multimedia functionality, including the ability to play games, take pictures, or play music, among others (Batista, 2004).

Such demands are additionally fuelled by the blurring of the lines between the capabilities of mobile phones and more specialized devices, such as PDAs, digital phones and MP3 players. As support for features provided by such devices is included in mobile phones, the users expect the phones to become increasingly versatile and proficient in their capabilities.

Streaming video appears to be one of the next technological steps in multimedia and information services for mobile phones and developments in the area are well under way, led by the 3rd Generation Partnership Project (3GPP) group. The 3GPP group is comprised of telecommunication companies from around the world to ensure compatibility of formats across different networks. Most new mobile phone releases already incorporate a video compression format suitable for streaming, developed by the 3GPP.

However, the majority of the recent success stories in mobile phone technologies have been those that did not require a phone upgrade or additional purchase of any kind from the user. Thus, the potential for success is likely to be much higher for a video streaming implementation that utilizes technologies supported by the devices that the users already own, rather than the newer devices available. To develop such an implementation, the mobile phone market needs to be analyzed to identify the applicable technologies, and a plausible implementation model should be built on these technologies, taking into account the potential bottlenecks.

Once the set of applicable standards has been identified, including preliminary testing to ensure none of them creates a significant bottleneck, a set of requirements for the final implementation can be compiled. Following the requirements an implementation will be created. With the implementation we can test the capabilities and versatility of it as well as identifying areas for optimization.

2 Background

2.1 Overview of Current Mobile Phone Trends

Mobile phones today are used for an enormously broad range of applications, the breadth of which however fades in comparison to the applications that are potentially possible. Data rates for mobile phones are now comparable to those that were used by the majority of PC users to connect to the Internet a few years ago.

Besides minor recreational use, the main purpose of the connectivity is usually stated as providing current information to those "on the go". But the small and cumbersome WAP network often offers little to the user who wants information on an event he/she is missing because he/she is "on the go". With the increases in transfer rates that allowed for multiplayer gaming, it could also be possible to follow a sports match, or to have live access to a news ticker with current developments on the event.

With data streaming, such applications seem more than plausible. An accessible and working implementation of data streaming for mobile phones is however only recently starting to appear on SymbianOS-based phones, unaffordable by many, and centered around pre-recorded video clips (Alvear, 2001). The breadth of the MPEG-4 format specification can allow for actual live streaming of useful content, where useful content includes more than video, to a wider range of phones.

2.2 The Networks

The Third Generation (3G) mobile networks are composed of three different networks (3Gtoday, 2004). The networks themselves: CDMA, used in Korea and the United States; GSM, used throughout the world, and PDC, used in Japan, differ mainly in the radio frequencies used and their characteristic user base. Those differences are in fact gradually diminishing; as all three are working towards conforming to the IMT-2000 specifications, but currently still remain incompatible. Their original vision of a 3G mobile network that would be universally accepted has disappointingly collapsed and the original networks remain separated. A preferred network should be selected. This will greatly simplify the process of finding generalizations and patterns needed for the research, while having little effect on the final implementation, as the specific technology standards such as J2ME, SMS, MMS and WAP are identical across the different networks.

In examining the three networks their strengths and weakness were exposed. The PDC network is only supported in Japan. Despite its handicapped access to the worldwide market it is widely regarded as most technologically advanced market (DeZoysa, 2002). This is because 62.7 percent of Japan owns a cell phone and competition is fierce for customers (Gross, 2004). The competition for a limited number of customers, forces manufacturers and carriers to offer more features and reduce subscription fees. Although the PDC carriers offer the latest technology available, it is not accessible to the rest of the world.

CDMA, unlike PDC, is a global network and is available in many countries worldwide. Currently, CDMA has a small percentage of the global market. In order to rectify this problem, CDMA has introduced a new technology named CDMA 450 (Clark, 2004). The CDMA 450 gets it name from its ability to transmit at 450 MHz. This allows for a greater range of service and requires less transmission towers. Unfortunately, CDMA faces two major problems that prevent it from capturing a larger global market. First there are not enough transmission towers built and CDMA subscribers often find themselves without a signal. The other problem is mobile phone manufacturers are not creating CDMA mobile phones. Subscribers are limited to a much smaller selection of mobile phones than other competing networks. Currently, CDMA does not have a secure global market presence.

GSM network is the largest global mobile phone network. Many mobile phone manufacturers and carriers support GSM. GSM currently has over 1 billion subscribers and is continually growing (Pearson, 2004). It is estimated that 81% of the phones sold globally are GSM phones (Parbat, 2003). Mobile phone manufacturers have produced a variety of phones, not just in different styles, but also with different capabilities. While some mobile phones might be a simple black and white phone others can be as complex as color phone with a built in camera and Java support.

The PDC is clearly not a good choice, due to striking differences in common user behavior from the rest of the world and the very limited user base. A limited user base, unique phone use traditions, and strong corporate involvement with the standards, indicate that CDMA may also be a bad candidate. Easily emerging as the victor, GSM, offering the largest and most diverse user base, the most open and independent standard, rapid new technology implementation and a large amount of available research, will be the presumed network under which our final implementation must function. While likely to function just as well under the other networks, the implementation may not be as readily accepted by their average user.

2.3 The Market Needs

2.3.1 The Complex Consumer

With the ongoing trend of technology miniaturization, mobile phones have served as a perfect example, showing consistent increases in density and reduction of size throughout the last few years. However in contrast to the market for expensive laptops and PDAs featuring latest technology developments that have been mostly restricted to those in need of office mobility and high-end gadget lovers, mobile phones have become a commonplace accessory.

The use of those newly implemented technologies to provide various services to the consumer has thus been a significant area of research and development, mainly because of the large potential user base. However, numerous technologies receiving significant investment and promotion by corporations have failed miserably to bring in the expected dividends, including WAP and MMS. While the exact reasons for the failure of the average user to embrace these technologies remain unclear, one deduction is generally agreed upon: the mobile phone market is much more complex and diverse than it appears.

No longer divided into the simple groups of the simple-minded, the trendy, and the business user, current mobile phone users have demands and expectations that vary more or less individually, with technophobia, brand loyalty, aesthetics and social status, among others, becoming important factors (Alcatel, 2003). Broad generalizations can however be observed when comparing the failure of WAP and MMS to the unexpected success of SMS and Ringtone/Logo/Game downloads.

2.3.2 Wireless Application Protocol

Wireless Application Protocol (WAP) is a protocol designed to allow access to the Internet from mobile phones, providing access to services equivalent to a web browser. Only allowing pages written in the newly designed, XML-based WML (Wireless Mark-up Language) to be displayed, and using its own transmission layer protocol, it provides limited features to the Internet. With significant incompatibility, it was found inadequate and not worthwhile by most users and was publicly recognized as a failure (Banan, 2000).

The failure of WAP may have been a combination of numerous factors, including the lack of content, the lack of need for the service, complexity and lack of widespread availability. The new mark-up language (WML) meant that web content providers had to write additional, mobile-friendly code, which not many did. The service itself was inferior to the Internet, and was introduced at a time when many users already had

Internet in their homes. WAP was based from, the Wireless Transaction Protocol (WTP), and one of the main reasons it failed (Banan, 2000). WTP recreated new protocols that were similar to those already used but modified them slightly to be incompatible. For example, WAP replaces UDP with WDP, TLS with WTLS, HTTP with WTP, and HTML with WML. This make it unnecessarily complicated just to connect to the Internet with a mobile phone. Those that required connectivity on the road had been long using their phones as a modem for their laptops, thus the nature of demand for the service was neither well-defined nor apparent.

The WAP billing system was data-based which was unfamiliar to the average user, and lack of interconnectivity between WML websites made the service overly complex. The service was also not a standard option in most user contracts, and even when the majority of phones had WAP capability, the owners often did not have data services enabled, lacking a compelling reason to pay the extra fees.

2.3.3 Short Message Service

Short Message Service is a protocol allowing 140 byte messages, primarily consisting of text to be transferred between mobile phone users. Its use has reached staggering heights, with 20 billion messages being been sent worldwide in 2003 (BBC News, 2004).

Amidst these failures, usage of SMS had rocketed beyond all expectations. While a major factor in its success may be psychological, rather than technology-related, it is notable that SMS was available to almost all mobile phone owners and was very easy to use (Jenson, 2004). In fact, its popularity and broad support led to additional functionality being built on top of the existing standard, which initially was very basic, and offered little more capability than to transfer 140 byte blocks of data at a time.

However, it was adapted to support non-ASCII characters (further reducing the possible message length due to 8-bit character encoding), splitting of long messages, and, perhaps most unintuitively, the transfer of data such as images and melodies. While the implementations of these features suffer from a fair amount of awkwardness due to the restrictions of the original standard, they remain in use to this day, illustrating the increased importance of availability and usability over robustness.

2.3.4 Multimedia Messaging System

Multimedia Messaging System is a communication protocol designed to be the logical extension to the Short Message System (SMS), extending its text transfer capabilities to allow transfer of complex presentations including audio and images. An MMS message is composed of “slides”, each with a definable duration containing a combination of text, audio data and/or images. The system, advertised as a replacement for SMS, failed to live up to its intended purpose, and is mostly used for specific, trivial tasks, if at all.

Introduced at a time when data services were a relatively common standard feature, the failure of MMS is perhaps best attributed to its complexity and unintuitiveness, as well as, to an extent, the lack of camera phones on the market. Billing per kilobyte of data transferred, rather than the familiar charges per messages sent and complex configuration options alienated numerous users.

More importantly, however, was the newly designed “slide” structure, completely novel to users expecting a familiar E-Mail-like “text with attachment” structure or images embedded within the text. In fact, EMS (Extended Messaging System) technology, implementing the latter, was more commonly used, even though severe restrictions on the type and quality of the pictures were imposed. The lack of camera phones on the market, and thus lack of data a user might want to send via MMS was also a contributing factor.

2.3.5 Ringtone, Logo & Game downloads

Especially popular with teenage users is the ability to customize their mobile phones to suite their own personal tastes. Common features include ringtone downloads, allowing customized melodies to be played on incoming calls; logo downloads, allowing the operator's name on the phone screen to be replaced with a small graphic; and game downloads, evolved from being defined by proprietary standards transferred via SMS into established universal standards transferred via data services. Heavily advertised and used by teenagers, they have become a major revenue source for the mobile industry.

The currently flourishing market for downloading ringtones, logos and games has arisen from very humble beginnings. The option to replace the operator logo with one received via SMS had been an undocumented feature in almost all Nokia phones since the starting days of the GSM mobile phone market. Unexploited and unknown for a long time, it started gaining popularity when numerous websites sprung up, offering to have a new operator logo of your choice sent to you. As popularity grew, fuelled mostly by word of mouth and users showing off the logos to their peers, both independent companies and network operators began offering the logos for a fee, and soon the notion of receiving free logos was eliminated.

The concept of receiving ringtones, and eventually games, were soon introduced, however interestingly, these were all proprietary standards only available on Nokia phones. Other manufacturers followed suit with their own proprietary standards in their new phone releases, however the universal acceptance of JPEG and GIF operator logos, MIDI ringtones, and Java games was introduced before any manufacturer could compete with Nokia in terms of users with new logo/ringtone-enabled phones or choice of available logos/ringtones.

The adoption of the standardized formats, as one can easily guess, propelled the market into new heights, aided by the introduction of color screens and polyphonic melodies. It should also be noted that users happily paid the small fees charged for the downloads, even though most of these were also available free, but required some technical knowledge to transfer to the phone. Yet again, this demonstrated the emphasis users place on simplicity.

2.3.6 Recovery from failure

While WAP and MMS have been widely labeled failures, it should be noted that their failures primarily arose from of their inability to meet up to expectations and serve the purpose they were intended for. Phone operators have since adjusted their marketing of the technologies and attempted to make improvements, with some results. Allowing non-WML content through their gateways and providing the aforementioned ringtones, logos and games via WAP has significantly increased WAP usage. MMS advertising has also been changed to portray the service as a messaging service exclusively for pictures, a purpose it has fulfilled well and increasingly more often with the growing popularity of camera phones.

2.3.7 Discussion of Market Needs

The pattern that emerges quite clearly from the data above is that the average mobile phone user prefers technologies that are easy to use and available, with little concern for how advanced and robust a technology is (Jenson, 2004). A solution that will work on as many phones as possible, focused on making the most of the available technology and overcoming its shortcomings is thus the most likely to succeed.

Since users have shown themselves to be comfortable with downloading and launching Java games, it can be assumed that a standalone downloadable program offering specific streaming information will be well received. Such an implementation is preferable to a general data streaming front end, where numerous types and sources of information can be accessed, due to the greater ease of use offered by the former. Thus providing a backbone to enable data streaming for a range of applications will be considered to be the desired format of an MPEG-4 implementation. An analysis of the current trends in mobile phone technology is also required to identify the technologies that are available to the largest percentage of users.

2.4 Mobile Phone Technology Trends

2.4.1 Market representation

The majority of GSM phone users are within the region often referred to as EMEA, short for Europe, Middle East and Africa. The area, especially Europe, is quite representative of the overall GSM phone market, having the highest numbers of mobile phones per capita. Europe also leads the market trends, being the first to implement new phone technologies and offer new phone releases. The device market of the area is led by three companies: Nokia, Sony Ericsson and Siemens, together accounting for 89.8% of the phones sold in the area (GSMArena, 2004). A database of all mobile phones released in the past 2 years by those companies for the EMEA market has been compiled to serve as a representation of the entire GSM phone market (See Appendix B). We will base most our subsequent forecasts and analysis of phone ownership on this database.

The database contains the release date of each phone and the features relevant to data streaming, including Internet connectivity, audio playback capabilities, screen properties and the availability and version of Java implementation among others. A more compact version has also been produced with a layout that facilitates the identification of trends (See Appendix C). The phones have been sorted by release date and grouped in 6-month periods, representing a release date in either the first or second half of a given year. The general time of introduction of a given technology is readily apparent on the table, as is its growth towards dominance over preceding technologies among new releases.

2.4.2 Observed Trends

After an overview of the database, the technologies can be sorted according to the strength of their establishment in the market. The availability of GPRS connectivity has been a standard feature for a very long time, and its replacement by EDGE is in the early stages. Color screens have the second most prominent presence as a standard feature, with 16-bit color availability showing rapid growth over 12-bit screens. Implementation of the new version of WAP 2.0, even though introduced a while ago, is showing relatively slow growth in popularity and has yet to become a de facto standard.

With Java already being a standard feature, its new Mobile Information Device Profile (MIDP) version 2.0 shows much more rapid growth, its relatively recent introduction being the reason for its limited availability. The support of sound playback besides MIDI ringtones is another feature well on its way to becoming a standard. The popularity of smartphones, using the Symbian OS, is however lacking, experiencing little growth since their introduction in 2002.

Mobile phone ownership statistics are needed to determine the availability of a given technology in the general market, where users retain their old phones for lengthy periods of time. The information provided by the trends should however be used to ensure that the data streaming solution can take advantage of the new technologies as they become available.

2.4.3 Ownership prediction

In this section we shall try to predict the ownership of mobile phones after the device's initial release date. The concept of the shelf life and lifespan of a given device can be employed together with its release date as a means of predicting the number of such devices owned by users at a given time relative to devices released at other times. The shelf life of a device refers to the amount of time since its release when its market availability becomes negligible (i.e. it is no longer marketed, has been replaced by a superior device at the same price level, or is taken off store shelves). The lifespan of a device refers to the amount of time since its purchase by a user until it is no longer usable, otherwise damaged or misplaced, or replaced by another device. A mathematical formulation has been derived for the concept using simplified system dynamics theory $[q+6q(x^3/(3(2L+s)^3)-x^2/(2(2L+s)^2))]$ (See Appendix A). The variables to be defined in the formulae are the shelf life *s*, lifespan *l*, and quantity sold *q*. The formula gives a rough estimate for the number of devices that are in the possession of users at a given time after the device's release date.

The derived formulae can now be used to predict mobile phone ownership in 2005. By grouping the phones into 5 groups by release date, each group spanning half a year, with the "announced" phones counted as 2nd half of 2004, the calculations can be simplified by assuming that the phones in each group were simultaneously released at the midpoint of the 6 month period, which does not affect accuracy significantly. The additional phones released before June of 2002 help to compensate for leaving out older phones from the calculations and March of 2005 will be used as the date for which the estimation is made and months will be used as units of time.

The average shelf life and lifespan of a mobile phone, as given by some sources and confirmed by observations, are 18 months and 15 months respectively (Rogerson, 2003). The final variable that is to be defined, *q*, is arbitrary, since only percentage values are sought. However, *q* should account for the average growth of device sales in the 2002-2004 period of 28% (Gartner, 2004). Hence, after defining it arbitrarily as 100 for 2H 2002, the values of 114, 128, 146 and 164 are assigned to *q* for the subsequent groups respectively. Months are the most appropriate units of time for the calculations.

With all the variables defined, the calculations can be performed, yielding the following results: in May of 2005, 10.66% of phones owned by users will be from the 2H 2002 group, 19.2% will be from 1H 2003, 29.47% from 2H 2003, 28.74% from 1H 2004 and 11.95% from 2H 2004. This information can be combined with the percentages of phones in each group that have a given feature, to obtain the percentage of phone users that will have a given feature on their phone in the first quarter of 2005 as shown below (See Appendix C for more detailed figures and calculations used):

Table 1: Technology Access

% of mobile phones with access to a given technology in the 1st half of 2005												
data transfer			sound playback			MIDP version			colours			
CSD/none	GPRS	EDGE+	no	just MIDI	yes	no Java	1.0	2.0	grey	256-4096	65536+	Symbian
10.34	72.46	17.21	16.95	17.46	65.60	23.29	54.52	22.18	14.14	47.52	38.34	18.03

These values can now be used to forecast the technology available to the average user at the time of this project's planned completion and choose the appropriate technologies for the implementation of a practical data streaming solution.

2.4.4 Discussion of Mobile Phone Trends

With the estimates indicating 80% of mobile phone owners having some sort of Internet connectivity and 75% having Java capability, a non-proprietary data streaming solution is certainly feasible and marketable to the vast majority of phone users. With smartphones holding an estimated 20% share of the market, it is best to avoid a solution that requires Symbian OS features, thus Java is the preferable choice to C++, which is only supported in Symbian OS, as the development language.

While the majority of phones will have color screens, most of those will only support 8 or 12 bit colors, thus the cost of implementation of a feature to determine screen limitations and compress data accordingly would certainly be justified. Additionally, audio streaming support should also be considered, as sound playback is likely to be supported by the majority of phones.

Only one third of Java-enabled phones is predicted to have a MIDP 2.0 implementation, hence using the older MIDP 1.0 should be favored, its compatibility with MIDP 2.0 ensuring a significantly larger number of users.

The presence of the GPRS standard’s bandwidth restrictions will have to be assumed, as the faster EDGE technology will only be supported by 20% of the phones, and it is unclear whether EDGE support by network operators will be widespread at the time, decreasing the percentage estimate even further.

Additionally, a switch to a less available technology may be necessary if functionality limitations of the selected technologies pose a significant threat to successful completion of development.

2.5 Java 2 Micro Edition

Java 2 Micro Edition (J2ME), one of the technologies to be used by the implementation as suggested by the trend analysis, was introduced by Sun Microsystems primarily for use in embedded systems on consumer devices (Sun Microsystems, 2004). Designed to be flexible and robust enough to allow a wide range of applications, while compact enough to run in resource constrained environments, J2ME is a stripped down version of its enterprise and desktop counterparts, with a vast number of classes and methods found standard in other versions of Java, such as floating number support, simply omitted.

J2ME has several configurations, profiles and optional packages available to meet a wide range of devices with varying capabilities (Sun Microsystems, 2004). Currently there are two configurations available, Connected Limited Device Configuration (CLDC) and Connected Device Configuration (CDC). CLDC is designed for devices that have limited processing power, memory and network connections. This includes most mobile phones and older PDAs. CDC is designed to take advantage of higher end devices such as new PDAs, set-top boxes and GPS systems.

Each configuration has a set of profiles and optional packages to make the most of each device. For CLDC there are two versions of Mobile Information Device Profile (MIDP) available. In MIDP 1.0 a standard set of features was introduced for basic application development. Some of its features include HTTP network connectivity, user interface, and persistent storage. MIDP 1.0 can be found in the first few generations of phones marketed as “Java Enabled”.

With the development of MIDP 2.0, many new features were included to enhance the language’s capabilities (Sun Microsystems, 2004). MIDP 2.0 offers better security through secure connection (HTTPS) and certifications for applications. MIDP 2.0 also offers new multimedia function. There are new functions for audio tone generation. This allows developers to create their own custom songs for their applications. There is also a new class called Item. The Item class allows developers to customize their own drawing functions and objects. There is also a new Layer class for game developers. This allows for easier drawing methods of tiles and sprites.

In addition to the functionality already provided, there are optional packages that may be supported by developers and device manufacturers. Some of these include XML parsing, Bluetooth, and 3D graphics. J2ME allows for expandability beyond its current limitations. In the future when small handheld devices become more powerful, there will be greater support and availability of new and better applications.

J2ME is one of the best programming language environments for small handheld devices. Built upon the familiar framework of Java and already widely available on cell phones, J2ME is the preferred language for cell phones. Sun Microsystems is making an effort to incorporate feedback from developers, customers, and manufacturers in each iteration of its software. By supporting J2ME it ensures availability and continued support in the future for developers from devices manufacturers.

2.6 Motion Picture Experts Group

Motion Picture Experts Group (MPEG) is part of the International Organization for Standardization (ISO), and is in charge of developing standards for audio and video media. Their work is widely known, as they are the developers of the technology behind Digital Video Discs (DVD) and MPEG Audio Layer 3 (MP3).

2.6.1 MPEG-1

MPEG-1 was created in 1993 to address the newly emerging need for transferring audio and video in a more efficient fashion. Years after its introduction, the usage of MPEG-1 remains widespread to this day. It is still used as the format for encoding Video Compact Discs (VCD), the predecessor format to DVDs, and also maintains popularity as a means for distribution of short movie clips over the Internet.

MPEG-1 success lies in its ability to compress audio and video with very little detail loss noticeable to the human eye. This is done through a series of compression techniques. Each frame is divided into smaller parts for compression and compact transmission. A single image frame is broken down into “Blocks”. From those Blocks, much larger structures are built, including Macroblocks, Pictures, Group of Pictures and finally a Sequence, thus forming the basic representation of video in MPEG-1.

Another compression technique that is used takes advantage of human perception. Instead of saving every frame, the changes between each frame are saved as motion vectors. When watching a video, such as an actor talking, there are usually only a few changes occurring during the scene, such as movement of the actor's mouth and facial expression changes. By recording and transmitting only those changes rather than the entire scene, a much more efficient usage of memory is achieved.

To allow smooth playback between frames of video, another type of frame is needed. Special frames that contain the information about the previous frames (P-Frames) and bidirectional frames (B-Frames) that contain the information about both the previous and future frame are used to predict the current frame. These frames ensure the perceptual accuracy of the video. These special frames are sequenced throughout the video and allow for removal of temporal redundancy.

The vectors, along with other prediction error checking, are further compressed using Discrete Cosine Transform (DCT). The DCT transforms the image into a set of frequencies that are better suited for transmission. The least significant bits are reduced to zero to allow for further compression but introduces "lossy" compression. The bits that are dropped however usually represent small amounts of motion, thus the detail lost after those final stages of compression is near negligible to the human eye.

2.6.2 MPEG-2

MPEG-2 is a more recent standard developed to deliver rich audio and video content over a much broader range of mediums (MPEG, 2004). MPEG-2 has been already been adopted and used in many devices including High Definition Television, Video on Demand, and DVDs, taking advantage of the increases in network bandwidth through the introduction of new technologies.

The transversal of multiple mediums is due to the introduction of a Transport Stream. The Transport Stream allows for error free transmission of audio and video packets. It does this by separating the audio, video and timing information into different streams, where each individual stream is divided into packets. Each packet along with its associate timestamp is sent to the receiving end. Once the packets are received, the audio and video are rebuilt according to the timing information. This feature can be demonstrated by switching angles while watching a DVD or using Video on Demand where audio and video synchronization is maintained.

Along with the synchronization, MPEG-2 offers multiple simultaneous streams, not previously offered in MPEG-1. Multiple streams allow for separate audio and video to be encoded together. Applications of this can be observed when watching a DVD and changing the angle or the audio channel. MPEG-2 also offers higher video resolution and bitrates, and better compression for both audio and video.

2.6.3 MPEG-4

MPEG-4 is a new object-based format that supports more than just audio and video compression and transmission. It offers a variety of new features to make the standard more robust and versatile, as well as improving its existing audio and video compression through the introduction of sprites and layers.

For example, a scene in a movie where an actor is talking and the background remains still. The extra bits that have to be encoded for the background are wasted since the image is still. This also lowers the image quality because valuable memory is spent capturing motion. MPEG-4 resolves this by enabling layers to represent the background and sprites to represent the actor. Through the separation of layers, more memory is dedicated towards the image quality and motion of the actor. The background remains completely stagnant as it is only an image and not video. There is a great range of possible applications for this technology such as news broadcast and sporting events.

Besides just audio and video representation, MPEG-4 is capable of displaying 2D vector animation and 3D facial and body animation. For simple 2D animation, rather that storing it as video block-based motion, it is more efficient to store the shape and the beginning and ending state and have the renderer calculate the steps in-between. This reduces the amount of data that needs to be stored in the MPEG-4 file and usually takes less processing power than displaying video files.

3D facial and body animation was introduced to being able to produce synthetic human characteristics. In addition to being able to talk to someone it would be possible to be able to see a 3D representation of that person’s face.

MPEG-4 is also interactive. It allows interaction between the content and the user. With this technology it is possible not only to have complex menu system but it also possible to have complex interactive games. Sample applications have already been developed such as Othello and checkers.

With all these new features of MPEG-4 a new framework had to be created. BInary Format for Scenes (BIFS), an extension of Virtual Reality Modeling Language (VRML), was created just for that purpose. BIFS allow for different types of media to be added, deleted and manipulated over the time of the scene. The media drawn on the scene is not stagnant; it can be controlled by various BIFS commands such as those for rotation, translating and scaling.

MPEG-4 combines various types of media and functionality into a single standard. Based on the successes of their previous technologies, MPEG-4's more advanced features will be more widely adopted in the near future.

2.7 Current MPEG-4 Software

MPEG-4 software tools are currently being developed by several different organizations. Well known software include DivX and Xvid which are coder-decoder (codecs) based on of MPEG-4 audio and video compression. Well-known media players that support MPEG-4 include Apple's Quicktime and Real Media Player. MPEG-4 has a variety of different profiles that can be selected depending on the capabilities of the device. MPEG-4 software that is currently available usually only feature audio and video profiles and not the more advanced profile such as 2D animation, 3D animation, and synthetic speech and audio. The few projects that are currently under development are from the work of IBM and computer enthusiasts. Each program has their set of strength and discrepancies.

2.7.1 IBM Toolkit for MPEG-4

IBM Toolkit for MPEG-4 is a set of tools to help aid in the creation and playback of MPEG-4 files (IBM alphaWorks, 2004). The toolkit is a high level API designed for content authors and programmers alike to be able to use MPEG-4 now. The toolkit is designed for Java compliant devices and built on the premise of cross platform compatibility. The toolkit also extends to the Internet, allowing for MPEG-4 files to be viewed on any device that is online and Java-compliant. Included with the API are several sample applications and tools.

Avgen, is a tool included in the IBM Toolkit for MPEG-4. It is an audio and video program for creating MPEG-4 files. Built for simplicity and ease of use it employs a GUI for creating MPEG-4 and Internet Streaming Media Alliance (ISMA) compatible files. Briefly, ISMA is trying to provide standards to ensure streaming across multiple platforms. Different media files can simply be dragged and dropped together over a timeline and output as an MPEG-4 file. Although the interface is simple it allows users who do not want to use command line tools to create MPEG-4 compliant media. By providing a simple tool, it allows more users to create and spread MPEG-4 files quickly and efficiently.

XMTBatch is command line authoring tool that convertx eXtensible MPEG-4 Textual (XMT) to MPEG-4. Based on XML and Synchronized Multimedia Integration Language (SMIL), it is a textual scene description language. XMT allows content authors to compose and edit complex scenes by simply writing text. Since XMT only directs the scene layout and presentation; it does not edit or change the media files therefore allowing editing to be done on the fly rather than recompression of large media files. XMT also allows authors to share their scene with other authors without having to send all the content. XMTBatch enables programmers to experiment and create MPEG-4 files without having to learn all the complexities of the file format.

The toolkit also provides three sample programs built upon their API. The API allows for MPEG-4 compliant player to customized based upon the needs of the end-user and any device limitations. This allows for a customized player that can be used on something as resource limited as a cell phone or something as powerful as a high-end graphical workstation. To demonstrate the potential of their API, they included a stand-alone application, a Java applet, and a Java applet that plays files from the Internet.

IBM has designed an easy-to-use, well document, and customizable API that allows for content authoring and playback on multiple devices. Being based on Java it is supported on multiple platforms and has many potential uses.

2.7.2 MPEG4IP

MPEG4IP is an open source audio and video end-to-end streaming solution (MPEG4IP, 2004). The project is led by several employees of Cisco Systems. The goal of MPEG4IP is to provide open standard and open source streaming solution. MPEG4IP combines other open source projects and utilizes them together. Originally written in C++ for Unix based systems it has since been ported to several different operating systems including Windows, Solaris, FreeBSD. BSD/OS and Mac OS X.

MPEG4IP relies heavily on other open source projects especially for its audio and video codecs. This has both its benefits and downsides. Using existing well-known and supported software allows developers to focus on their own code and not have to rewrite software that already exists. This also poses problems as the software is not seamless and must be patched together in order for it to function properly. End-users must compile each individual project from separate sources to see if the software works as advertised. If an error is received there is often no explanation and hours must be spent browsing forums searching for a solution.

MPEG4IP is poorly documented and cumbersome to use. Being a patchwork of existing software requires reading through each individual software guidelines before

MPEG4IP will work. Also, as is common with open source driven projects, there is a lack of useful documentation and comments. There is often no explanation of what the sections of the code do. MPEG4IP clearly warns that its software is intended for developers only and not the end-user, as executables are not provided.

Despite its difficult nature, MPEG4IP does offer useful technology. MPEG4IP provides MP4Live, a Linux-only audio and video capturing and streaming program. With MP4Live it is possible to capture and stream in real-time from any capturing device, such as a web cam. The possible applications for this technology are live video broadcasts for both Internet and television audiences. If utilized with more advanced features of MPEG-4 it would be possible to have interactive live broadcasts.

MPEG4IP provides a standard set of tools for converting existing audio and video files into the proper MPEG-4 format. MPEG4IP also provides a process called hinting, which prepares MPEG-4 files for streaming by adding an additional header and timing information throughout the file. Received on the client side, the file is then reconstructed correctly.

MPEG4IP comes with a MPEG-4 audio and video player. In addition to MPEG-4 it also supports other popular formats such as Xvid and Mp3. The media player also has the ability to receive streaming MPEG-4 files locally and from the Internet. Files are streamed through the Real Time Streaming Protocol (RTSP, 2004). By using RTSP, it is possible to have the server and client to communicate and offer content better suited to the clients needs.

MPEG4IP is a good tool but lacks the documentation and organization to present a solid software package. It supports several great features such as real-time encoding and streaming of MPEG-4 files but is limited to a select audience. Greater support for cross platform compatibility and documentation would go a long way in helping MPEG4IP's progress.

2.7.3 GPAC

GPAC Project on Advanced Content (GPAC) is an advanced MPEG-4 player that supports several functions including both 2D and 3D animation (GPAC, 2004). Being developed by independent programmers it is an open source project written in C++ whose ultimate goal is to provide a MPEG-4 player that is as intuitive as it is functional. GPAC started with the bulky reference software provided by MPEG and have since modified it use less resources. To do this, it relies on other open source programs for some of its functionality.

Osmo4 is a 2D MPEG-4 player that was created by the group ENST (ENST, 2003). They have since been disbanded. GPAC has taken the reins and have used the code provided in Osmo4 to improve upon their own program. This merging of technology allows GPAC to be able to render 2D MPEG-4 files correctly. 2D animation is important when considering the limitations of small-embedded devices. Embedded devices usually do not have the processing power or memory capacity to run full-length movies or complex sequence of animations. 2D animation however would be ideal for embedded devices.

GPAC is currently developing the 3D profile specified by MPEG. It currently supports 3D through the use of VRML and OpenGL. It currently can display primitive shapes, texture mapping and materials. Much work has to be done before full integration of 3D with other aspects of MPEG-4 such as user interactivity with objects.

GPAC, like the previous tools mentioned, has a set of programs to generate MPEG-4 files (GPAC, 2004). MP4Box is a command line tool that can produce MPEG-4 files from other audio and video formats. It can also use scene descriptions written in XMT and convert it into valid MPEG-4 files. In addition to creating MPEG-4 files, it can dump information from the files such as XMT and other various statistics.

GPAC had support for PocketPC/Windows CE platform until version 0.1.4. Older versions of GPAC could be run on handheld devices with only minor limitations compared to its Windows counterpart. The PocketPC version of GPAC demonstrated the possibilities of MPEG-4. GPAC was the only program that could test the claims of the MPEG-4 standard with real world testing. With its discontinuation, it will now be more difficult to test the possibilities of embedded devices with MPEG-4.

GPAC, like most open source projects, suffers from lack of documentation and complexities making it unusable for everyday computer users. GPAC is a great tool for developers and enthusiasts who want to experiment with some of the more advanced features of MPEG-4.

2.7.4 Discussion of MPEG-4

MPEG-4 is the latest rich media technology standard. Still under development it will most likely become the new standard for rich media content in the coming years. It supports advanced features such as animation, interactivity, and 3D modeling. MPEG-4's biggest problem currently is the lack of developmental tools and documentation. MPEG-4, being designed to support all types of devices, will eventually be supported on embedded and handheld devices. Currently phones are not powerful enough to support anything more advanced that simple 2D animation and vectors but development in this area will help MPEG-4 progress in the right direction. In chapter 3, we shall use simple experiments to establish the resource limits of cell phones, and thus what types of rich content they can handle.

3 Methodology

Having chosen J2ME as the development environment for the implementation, benchmarking programs were created and ran to ensure that bottlenecks would not be encountered in areas that were identified as potentially problematic. This work included establishing the graphics processing and networking capabilities of a mobile device, which were both tested on a mobile phone and a PDA. Portions of the code used for benchmarking would also be of use in the final implementation.

3.1 Benchmarks

3.1.1 Graphical and Processing Capabilities

One of the bottlenecks that is likely to be encountered when streaming visual data to mobile phones is the limited display frame rate caused by insufficient processing power and display speed. An investigation was thus made into the various graphic display methods offered by Java MIDP 1.0 on mobile phones, and the attainable frame rates using the different methods.

The phone used for testing was the Siemens model S55, a high-end phone at the time of its release in 2003, with a 256-colour 101x80 screen and Java MIDP 1.0. According to results from a popular benchmark database (JBenchmark, 2004) (which have been included in Appendix B and Appendix C), it is projected to be about 35% slower than the average phone in the first half of 2005 (see Appendix D for calculations) in terms of graphics computation and display speed, thus the thresholds of acceptability should be lowered slightly when evaluating the test results.

The MIDP 1.0 specification allows very few programming library calls of drawing, with the “Graphics” object being used for drawing on the screen in combination with the higher level “Canvas” and “Display” objects (Java Community Process, 2004). Methods for drawing provided by the “Graphics” object include rectangle, arc, string and line drawing, with no specific pixel or polygon drawing methods. To evaluate which one of the methods is fastest for drawing a single pixel, as well as to assess achievable frame rates for video display, a series of testing MIDlets were written. The MIDlets fill a specified percentage of the screen with pixels with alternating colors. The percentage of the screen to be drawn were set to increments of ten, and based from the total screen size of the Siemens S55. Each pixel was drawn using a rectangle, arc or line command starting from the upper left hand corner then down until it reached the edge where it would draw the next column. The frame rate was calculated by counting the total number of screen updates performed in one minute. The frame rate was then displayed upon completion.

The following is a table of frame rates that were obtained by running the program for 60 seconds with various screen fill percentage amounts:

Table 2: Graphic Fill Results

	Frames per second		
% of screen	**Rectangles**	**Lines**	**Arcs**
100	0.233	0.217	0.117
90	0.250	0.250	0.133
80	0.283	0.283	0.150
70	0.333	0.317	0.167
60	0.383	0.367	0.183
50	0.450	0.433	0.233
40	0.567	0.550	0.283
30	0.750	0.733	0.367
20	1.133	1.100	0.550
10	2.233	2.167	0.097

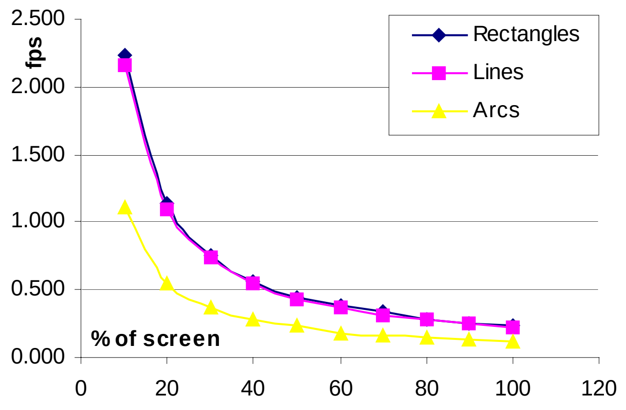


Figure 1: Graphic Fill Chart

Even with the exponential rate growth with decreasing screen percentage, not even the lowest frame rates come close to what may be deemed as acceptable, indicating graphics capabilities to indeed be a significant bottleneck in video streaming. Based on the low frames rate received from the benchmarks, we believe fast raster video playback would be implausible on an average mobile phone.

With the possibility of streaming and displaying vector animations encoded in MPEG-4 still being a possibility, a frame rate estimate for animation display is required. While running the JBenchmark tool, reasonable frame rates were observed for most of the animations that filled up the screen, sparking a hypothesis that the graphics speeds were limited by the number of objects, rather than the screen area.

To test the hypothesis, another program was written, where the exact same number of rectangles would be drawn as in the previous program given the number representing screen percentage in the previous program. However, the rectangles would be drawn to fill as much of the screen area as possible. Two screenshots comparing the pattern displayed in the 1st and 2nd programs given an input of 20, and the set of results are shown below.

Table 3: Rectangle Size Test Results

% of screen	Fps.
100	0.233
90	0.250
80	0.283
70	0.317
60	0.367
50	0.450
40	0.550
30	0.733
20	1.100
10	2.167

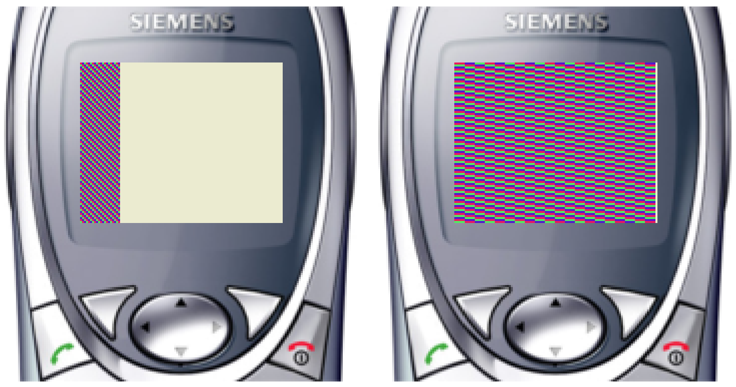


Figure 2: Rectangle Size Test Screenshot

The frame rate column is almost identical to those for rectangles and lines in the first program, indicating that the speed of graphic display in fact depends on the number of objects on the screen. This also suggests that finding algorithms to minimize such objects would be an important requirement for the project.

Two additional tests were run using the written programs to verify some assumptions. First, the 2nd program was run 10 times with the same arguments to ensure there was no random frame rate variation. All 10 runs produced the exact same frame rate, indicating that the graphic and processing power available to a program on a phone remains constant. The second test, run on the 2nd program used various arrangements for a constant number of rectangles, with varying numbers of rectangle rows and columns. The horizontal to vertical combinations 1x60, 2x30, 3x20, 4x15, 5x12 and 6x10, all of which require 60 rectangles to draw, were drawn and frame rates recorded. Again, there was no frame rate variation, further validating the above conclusions.

3.1.2 Network Transfer Size and Delay

In addition to graphical bottlenecks, conjecture indicates that mobile phone transfer speed and available bandwidth may be a source of additional problems. To investigate this hypothesis, we would have to perform tests on currently available software and hardware. Observing first the limitations of the J2ME software, we found it has limited support for network protocols. J2ME, as of the time of testing, only supported HTTP network connections. MPEG-4 defines streaming through RTP/RTSP but J2ME does not have support for that protocol. This meant that we could not stream data with J2ME. With this in mind, we decided to test the capabilities of establishing a network connection and transferring data over HTTP.

Two tests were performed to test potential network bottlenecks. Our test bed was a PDA, the iPAQ h4100. The iPAQ was chosen because it could connect to the Internet in two different ways. In the first test, the iPAQ would connect to the Internet via USB 2.0. In the second test, the iPAQ would connect to the Internet via Wi-Fi (801.11b). The application that was programmed connects to a web server and downloads a file, copying it into a byte array. A web server was setup with files of various file sizes ranging from 1000 bytes to 1 MB. Measurements that were taken during the process included the amount of time to establish a connection with the web server, the memory usage of the PDA, and total time it took to transfer the downloaded data into a byte array.

Connection time (Ping) is the time it took for the client application to connect to the server and request a file. The connection time was calculated by subtracting the system time after the connection had been established and the system time just before the connection was requested. The run time (Run) is the time it took to copy the file into a byte array. Run time was calculated by measuring the system time between the commands to start copying the file to a byte array. This was done as a representation of the time it would take to parse a file from beginning to end. The perceived time (Perceived) is the time the user would have to wait before being able to play the file. This was the total time beginning with requesting a connection to finish copying the file to a byte array. Additionally, various memory measurements were made on the PDA. J2ME supports memory commands that return the amount of memory available and used. Memory usage is likely to vary across different devices; nonetheless it provides an interesting perspective of handheld devices. Measurements included the total memory available on the device, amount used, and the amount available.

Table 4: USB Network Performance Test

File Size (bytes)	PDA Perceived (ms)	PDA Ping (ms)	PDA Run (ms)	Memory Usage (bytes)
1000	934	871	8	156928
2000	2486	1471	990	157992
4000	2480	1529	925	160192
10000	2170	1214	930	173248
20000	2430	2230	172	190660
40000	1848	1475	344	225476
100000	3907	2226	1654	295112
200000	4997	1996	2968	434380
400000	4915	1984	2890	712908
1000000	9257	1710	7519	1269964

Table 5: 802.llb Network Performance Test

File Size (bytes)	PDA Wireless Perceived (ms)	PDA Wireless Ping (ms)	PDA Wireless Run (ms)	Memory Usage (bytes)
1000	1663	1625	7	152124
2000	978	936	14	158004
4000	1292	1170	97	160172
10000	1336	1124	187	173240
20000	1306	946	333	190660
40000	1508	982	498	225468
100000	3420	913	2478	295104
200000	3978	764	3180	434372
400000	7517	1135	6342	712900
1000000	16556	1066	15465	1269992

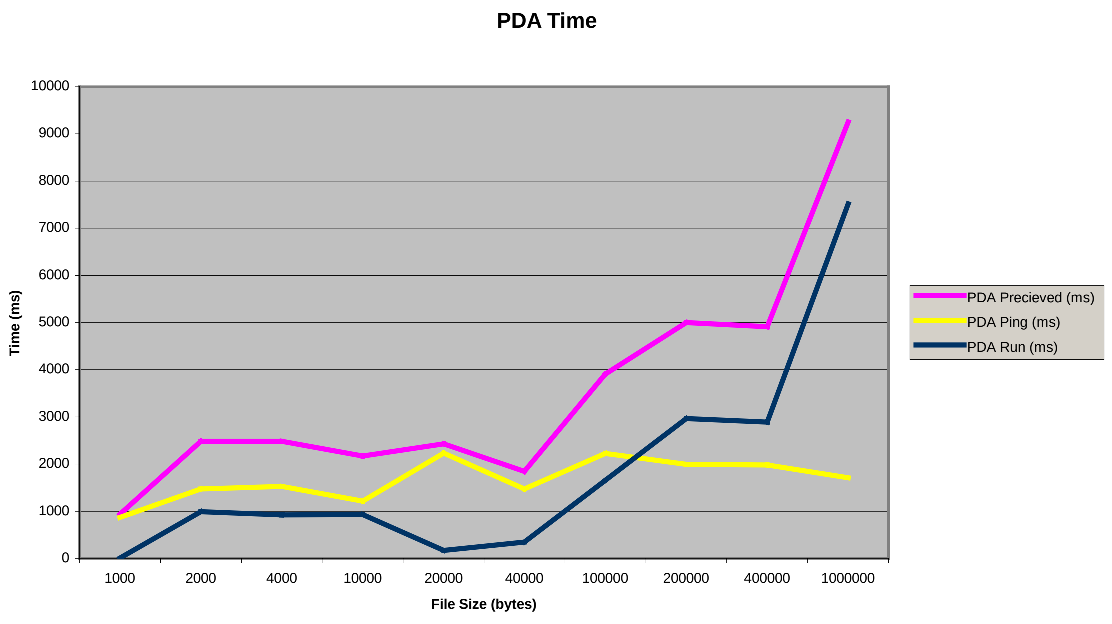


Figure 3: USB Network Delay

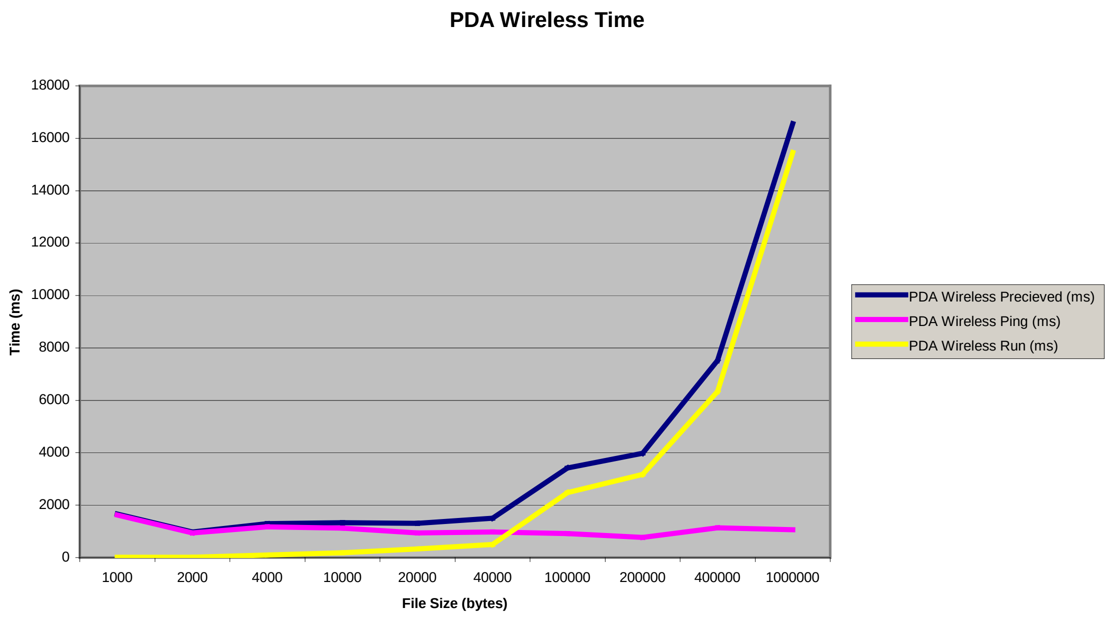


Figure 4: 802.11b Network Delay

From the results obtained, it is clearly observable that as files size increases, so does the processing time and therefore so does the delay on the user's end. The time that the application takes to connect to the web server does not fluctuate significantly, however it was observed that the USB 2.0 connection was slower than the wireless connection. Memory usage between the two tests remained consistent. Since these tests were only performed on the iPAQ, we cannot make recommendations for all handheld devices, but only suggest a guideline for optimizing network performance. In order to provide a comfortable user experience, we concluded that it is best to keep file sizes small to reduce processing time. For the constraints of the iPAQ, keeping file sizes below 40 kilobytes would be ideal. Smaller devices with limited resources, such as mobile phones, would probably require file sizes to be even smaller than 40 kilobytes to run well.

Additionally, some basic testing was done to ensure that the bandwidth available to mobile phones is sufficient for streaming, by simply using the Siemens S55 phone as a modem, and viewing various video streams on a computer connected to the Internet through the phone. The acceptable playback speed observed in viewing various streams well exceeding those expected to be streamed to mobile phones in complexity, indicated that network bottlenecks are not a major point of concern.

3.2 Implementation

With the benchmarking complete, a prototype could now be designed and the implementation requirements could now be established. Due to the graphics processing limitations imposed by mobile phones, an implementation would need to support the 2D animation capabilities of the MPEG standard. Additionally, in conformance with the initial intentions to provide a solution that is applicable to user base that is as large as possible, utilizing technologies that are already supported by devices that users own, the implementation would need to be written in J2ME.

Our final implementation consisted of three separate parts. The first is modified version of GPAC that returns the internal calls made by GPAC. The next is a client program, and main implementation of MPEG-4, which has the ability to read the calls made by GPAC. Lastly, a partial implementation of the MPEG-4 systems that will eventually replace the modified GPAC version and consolidate both it and the client program into a stand-alone MPEG-4 animation program.

In the search for existing implementations supporting MPEG-4 animation, GPAC was found to be the only such open source implementation. It was thus chosen as a starting point for a solution conforming to the above requirements. This however introduced several obstacles that needed to be addressed, the most notable of which being the fact that GPAC was written in the C language. Converting the code to Java was thus necessary, with the resulting code then to be adapted for J2ME and mobile device limitation requirements.

3.2.1 GPAC Code Simplification

Before the conversion could commence, simplification of the GPAC code was necessary. Since GPAC was designed to be a robust, versatile tool, the code was quite large, incorporating support for modules allowing functionality such as MPEG video playback, authoring support, and others not explicitly required by the project requirements. GPAC is approximately 1,200,000 lines of code including the main systems and plug-in modules, over 40 megabytes in size, and well over 1,200 files.

25 GPAC Modules	Required by GPAC	Our Implementation
amr_dec	×	
codec_pack	×	
dx_hw	✓	not necessary
ffmpeg_in	×	
file_dnload	×	
ft_font	×	
gdip_rend	×	
m4_rend	✓	rewrote for J2ME
M4Systems	✓	partial conversion to J2ME
mp3_in	×	
MP42AVI	×	
mp4_io	✓	rewrote for J2ME
MP4Box	×	
MP4Client	×	
OpenDivx	×	
Osmo4	✓	not necessary
raw_out	×	
render2d	×	
render3d	×	
rtp_in	×	
SDL_out	×	
SGGen	×	
V4Studio	×	
wav_audio	×	
wxOsmo4	×	

Table 6: GPAC Module Structure

Fortunately, the code structure incorporated heavy use of removable modules, thus the most advanced functionality, such as the ffmpeg video decoding module, could be easily excluded from the project. Once stripped of the module-based extensions, the remaining code, the minimum required for playback of MPEG animations, occurred in 5 modules, named M4Systems, mp4_io, mp4_rend, dx_hw and Osmo. Table 6 summerizes the main GPAC modules, highlighting which ones were converted, rewritten, or not necessary for our implementation.

The Osmo module, as previously mentioned, consists of the player interface code. Since the implementation was specified to be in the format of a backbone that could be adapted for specific applications, with a variety of interfaces to the MPEG-4 data, the Osmo code did not need to be converted. The dx_hw, also essential for playback on a computer, consists of code enabling the use of Microsoft DirectX API for displaying the animation, and also did not need to be converted. These two modules were however needed during the code stripping process to ensure that the code still compiles and functions correctly.

The mp4_io and mp4_rend modules provide running environment-specific functionality for reading the MPEG-4 file and displaying the graphics respectively. The nature of these modules required a re-write of both in J2ME, rather than a simple conversion, to ensure compatibility with a mobile phone running environment. The module written to replace mp4_io would need to provide functionality to retrieve MPEG-4 data from a server or a file, storing it into a byte stream, while mp4_rend would need to output simple graphical shapes to the display.

The M4Systems module is the core module of GPAC, providing all of the code necessary to process MPEG-4 data, and was designed to be completely platform independent, a feature that was especially beneficial to this project. However, it is approximately 800,000 lines of code in length, and needed to be stripped of unnecessary functionality to simplify the major undertaking of converting the C code into Java, as well as to ensure that it is as compact as possible, in an attempt to avoid any potential issues with memory limitations on mobile devices.

While a proportion of the functionality that was not essential for playback of MPEG-4 animation, such as MPEG-4 authoring support, was easy to identify and remove, the majority of the non-essential code required significantly more effort. The techniques used to identify non-essential code included searching for code that was no longer executed after the removal of the additional modules. Anything not explicitly specified in the implementation requirements were also removed. The latter includes audio support, interactivity support, support for corrupted and non-standard MPEG-4 files, support for standards such as JPEG2000 and QuickTime, and various minor MPEG-4 components such as meta data.

The task was made immensely more difficult due to lack of comments in the code. With no explanations of data structures, the functions performed by methods, and the purpose of various properties and components, the process required guesswork and a time consuming trial-and-error approach to identify whether code was essential.

Throughout the process of code removal, the program was repeatedly tested, ensuring that it compiled and played back MPEG-4 animation files. Despite such extensive code stripping, approximately 500,000 lines of C code still remained in the M4Sytems module. This was mainly due to the design of the MPEG-4 format, intended to allow it to be highly versatile, with a number of different data structures that need to work together to allow even a simple file to be processed.

3.2.2 Conversion to J2ME Process

The remaining code of the M4Systems module was then ready to be converted into J2ME. As the module was also further divided into numerous smaller modules, they were tackled individually. Since the conversion was expected to be a lengthy process, an assessment was first done on whether it was plausible to complete it within the time constraints of the project by converting a single module first. The module that was chosen for this was the code to handle BIFS, presumed to be a representative module, fundamental in rendering 2D animation.

BIFS in an MPEG-4 file are used for positioning various objects on the screen and transforming them, as well as animating such transformations. While conversion time for the BIFS code indicated that a complete conversion of the GPAC code was plausible, later it became apparent that the module, consisting primarily of mathematic calculations, especially matrix manipulations, was not representative of the majority of the code to be converted.

This was however revealed in later stages, as the code, the purpose of which was better understood was converted first. The next module that was tackled was thus chosen to be the atom parsing code, as the concept of atoms was outlined by numerous sources. Atoms are used to define the hierarchal structure of MPEG-4 files, where a particular type of atom contains a particular type of data. Atoms are generally nested within each other, with the “moov” atom being at the top of the hierarchy, and atoms such as “hint”, “mdat” or “trak”, containing hinting, video data or track information respectively, contained within it.

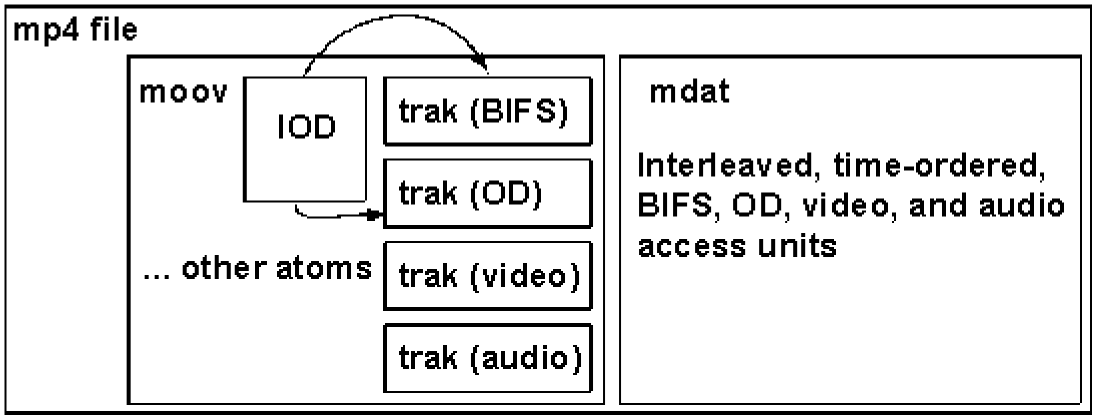


Figure 5: MPEG-4 File Format

However, since GPAC included support for over 50 different types of atoms, with no coherent explanation of their purpose given, the task was rendered much more difficult. Some helpful information was extracted from the documentation of Apple's QuickTime format, on which the MPEG-4 file structure is based, with additional help from an open-source Java application that could read and display the atom hierarchy of QuickTime and MPEG-4 files. The latter helped identify the atoms encountered in sample MPEG-4 files containing 2D animation, and the code to handle those was fully converted. Due to time restraints, the code for the other atom types was replaced with generic code to skip over such atoms as a temporary measure.

The rendering code, which parsed various types of objects contained in the atoms, such as polygonal objects, text, areas of interactivity and many others, was then attempted to be converted, as its workings appeared to be quite straightforward. The basic functionality for 2D object support was converted, with the rest of the code planned to be attempted at a later stage, once overall functionality was achieved.

The remaining M4Systems code, however, proved to be a significantly greater ordeal to convert. While the code of the modules outlined above did not make use of extensive abstract data structures, and was of a primarily mathematical nature, the remaining modules incorporated not only complex abstract data structures coded in a somewhat awkward manner, but also dealt with concepts that were themselves abstract and vaguely defined. The near complete absence of documentation in the GPAC code was also one of the primary factors complicating the task.

Due to these obstacles, only partial work was done on the remaining modules, with focus shifted to the mp4_io and mp4_rend interface modules that needed to be re-written. The modules where only partial conversion attempts were made include the Scene Graph, Stream Management and Object Descriptor modules. The purpose of the modules was partially obtained by conjecture, due to the ambiguity of the code and lack of documentation. However, Scene Graph appears to be responsible for constructing a graph of the object hierarchy within a scene, additionally defining objects or groups of objects that BIFS transformations apply to. The Stream Management module is mainly concerned with synchronization of audio, video and other tracks, providing timing information, as well as helping manage tracks from different sources, such as streamed media. The Object Descriptor module provides numerous data structures allowing the manipulation of abstract objects, such as atoms and tracks among others. One of the major obstacles in performing the conversion caused by those modules was their awkward use of direct memory access for data structure conversion, overcoming the shortcomings of the C language, which does not provide object-oriented programming support.

Due to the aforementioned difficulties encountered in the conversion process and the fact that the various modules are required to work as a whole to be able to parse even the simplest MPEG-4 files, it became apparent that a complete conversion was implausible within the time constraints. Converting the interface modules allowed a method of confirmation that a working implementation would be functional on a mobile device if the M4Systems code was eventually converted. Figure 6 illustrates the progress made on converting the various GPAC modules, with darker shades used to fill the module boxes indicating more progress made towards full completion. If the unfinished portions of the MPEG-4 parsing code are completed, the interface implementations guarantee correct and efficient functionality of the player as a whole on a mobile device.

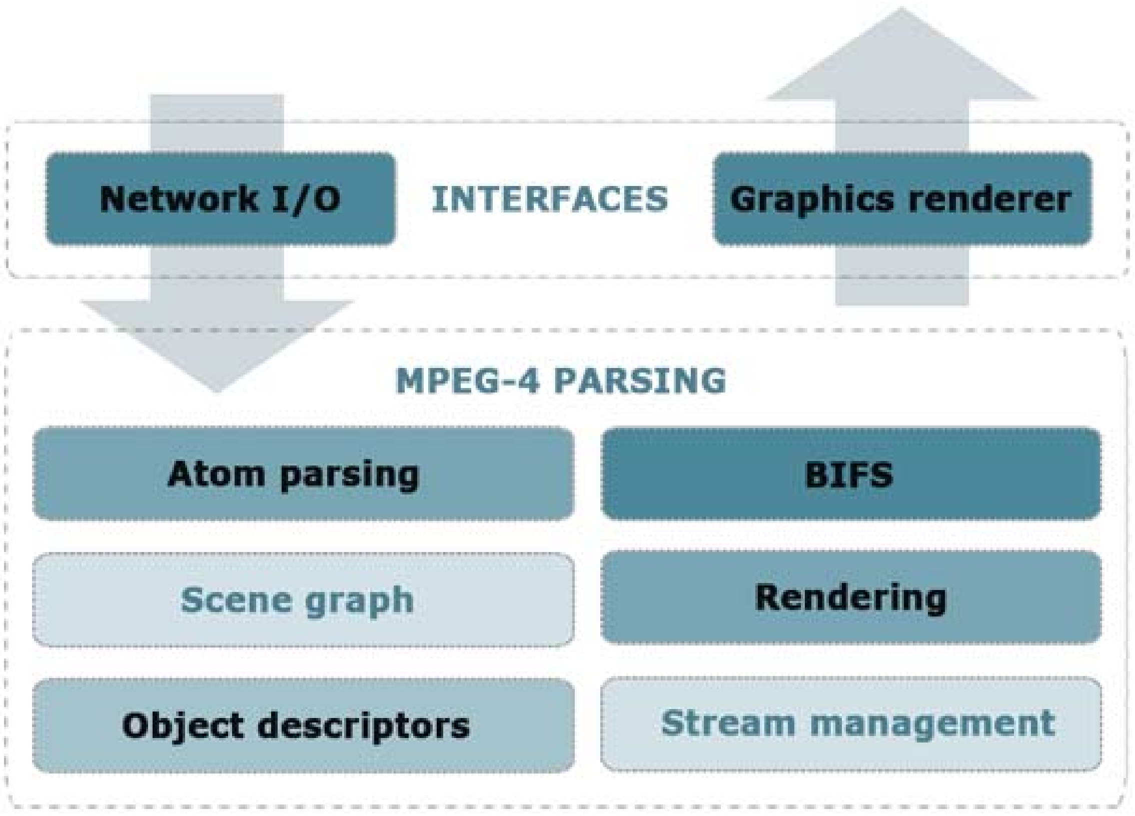


Figure 6: Module Conversion Progress

3.2.3 MPEG-4 Network Protocol

MPEG-4 defines streaming media through RTP/RTSP. MPEG-4 also has various error-correction algorithms to ensure delivery and playback. Unfortunately, J2ME currently does not support RTP/RTSP and only supports HTTP. J2ME does offer the Generic Connection Framework. The Generic Connection Framework provides an outline to establish such a protocol in the future.

Since J2ME cannot support streaming, HTTP had to be used. The client program via HTTP accesses the compressed calls on the web server. The URL is simply inputted and the client then downloads the file. Once downloaded the file is stored in a byte array to be processed by the renderer.

Ideally, the RTP/RTSP protocols should be used, but under the constraints of available technology, HTTP was used instead. By using J2ME's native support for HTTP, the client can download any MPEG-4 file available on the Internet. With future support and additions, migration to RTP/RTSP could be implemented for true streaming.

3.2.4 MPEG-4 Renderer

Our MPEG-4 player takes compressed GPAC calls and outputs them in the J2ME environment. Data is stored and downloaded from a web server and decompressed by the client. Each set of bytes represent the calls and their parameters necessary for output. The calls are then interpreted and displayed on the screen of the mobile device. The calls represent simple figures such as squares, rectangles, triangles, lines and circles.

The drawing functions are coded in the J2ME environment. The drawing functions are optimized to minimize the number of objects stored in the memory. Rather than using the defined objects provided by J2ME, we defined our own drawing objects that paint directly onto the canvas. So instead of each pixel, line, rectangle or arc being represented as an object, the group is defined within the canvas as a singular object. This reduces part of the memory bottleneck and allows for customization further down the line.

Figure 7 is a snippet of code that defines how to draw a triangle in J2ME. The calls, although simple, were transferred and displayed correctly on our client. Due to hardware problems, our MPEG-4 player could only be tested on an emulator. Being built on J2ME, the program should work with mobile device that supports the standard and has Internet access.

```
public void drawTriangle
       (Graphics g, int x0, int y0, int x1, int y1,  int x2, int y2, int color)
       /*****************************************************
       *  Draws triangle using DDA Algorithm
       *****************************************************/
       {
       int dxdy1 = 0, dxdy2 = 0;
       int edge1 = 0, edge2 = 0;
       int start = 0, end = 0;
       int y;
       int tempx,tempy;

       // sort the triangle vertices by y

       if (y0 > y1) {
        tempx = x0;
        tempy = y0;
        x0 = x1;
        y0 = y1;
        x1 = tempx;
        y1 = tempy;
       }

       ...

       // being rendering with selected color
       g.setColor(color);

       // calculate top half of triangle
       // initialize edge
       edge1 = edge2 = (x0 << FIXPOINT_SHIFT) + FIXPOINT_ROUNDUP;

       // perform slope calculation
       if (y0 != y1) {
        dxdy1 = ((x0 - x1) << FIXPOINT_SHIFT) / (y0 - y1);
        dxdy2 = ((x0 - x2) << FIXPOINT_SHIFT) / (y0 - y2);

        if (y0 < 0) {
           start = 0;
          if (y1 < 0) {
             edge1 += (y1 - y0) * dxdy1;
             edge2 += (y1 - y0) * dxdy2;
          }
           else {
             edge1 += (-y0) * dxdy1;
             edge2 += (-y0) * dxdy2;
          }
        }
        else {    start = y0; }

        if (y1 > SCREEN_HEIGHT - 1) {    end = SCREEN_HEIGHT - 1; }
        else {    end = y1; }

       // draw lines
        for (y = start; y < end; y++) {
           g.drawLine(edge1 >> FIXPOINT_SHIFT,y,edge2 >> FIXPOINT_SHIFT,y);
           edge1 += dxdy1;
           edge2 += dxdy2;
        }
       }
       // calculate bottom half of triangle
       ...
```

Figure 7: Draw Triangle J2ME Code

4 Results and Analysis

This chapter presents the results gathered from the conversion of GPAC to a mobile phone environment. Presented first is the original vision of the implementation, the problems that were encountered, their solutions, and the final version of the implementation. Our final results were a modified version of GPAC and a client program that was capable of rendering graphics on mobile phones. In addition to the programs, several of the core systems of MPEG-4 were converted.

The original objective of this project was to create a MPEG-4 player on a mobile phone. After researching available software, documentation, and limitations of both hardware and software, our objective changed slightly. Instead of a full implementation of a MPEG-4 player, it would be capable of MPEG-4 animation for mobile phones. GPAC was chosen to be stripped down to just its animation core and converted into a J2ME application. This was our objective, but unforeseen problems make the task difficult.

We encountered problems from the limitation of current hardware, software, and documentation. The problems revealed the infancy for support of the MPEG-4 format. Although MPEG-4 will most likely be adopted by the audio and video industry in the future, progress outside of large companies is slow due to lack of resources. Creation of a viable MPEG-4 player for mobile devices proved difficult under these circumstances.

4.1 Constraints of Technology

During the porting of GPAC to J2ME, we encountered bottlenecks due to technological limitations of both hardware and software. The greatest limitation for mobile devices is CPU processing power and available memory. These bottlenecks were exposed during the preliminary testing of graphics capabilities on the Siemens S55. This was also observed in the final implementation of our MPEG-4 player. When trying to load a complex vector image, such as that of a tiger or clown, the player returned a memory overload exception. This constraint limits MPEG-4 graphics to relatively simple shapes.

The development platform we chose, J2ME, also had constraints. J2ME MIDP 1.0 was chosen because of its availability on a wide range of mobile phones. MIDP 1.0 is the first version of the platform, and like most initial versions of software there is a limited number of features and plenty of room for improvement. In particular it did not have support for floating-point numbers or complex graphics capabilities. Although this does not affect older less powerful mobile phones, it does hinder newer phones that have the capability to execute complex applications.

J2ME also had other limitations. Currently J2ME only supports HTTP network connections. With only HTTP support it is impossible to have real time streaming as defined by MPEG-4 standard. Without streaming capabilities, our client application was limited to the amount of memory available on the mobile phone. The client was forced to download the entire graphic before it could be displayed. With real time streaming, smaller chunks of data would be needed at any given time to display an animation, rather then the entire animation.

The remedies for these constraints are to improve the hardware and software on which MPEG-4 is used. As phones continue to increase in functionality, so will CPU speed and available memory. This will lead to applications that can take advantage of the increases, such as J2ME. With each future release of J2ME and better support from manufacturers, mobile devices will become more capable of performing the functions of complex applications.

4.2 Lack of Documentation

The greatest obstacle in the development of a MPEG-4 player is the lack of freely available technical documentation. MPEG is not an open standard and requires a fee to view documentation. The documents that are freely available are general descriptions of MPEG-4 features. In addition, the majority of the documents focus only on audio and video profiles. Without a detailed description of the MPEG-4 standard it is difficult to complete a full implementation.

An alternative to achieve a technical understanding MPEG-4 was to reverse engineer the standard. There exist several open source projects developing MPEG-4 applications. It was our intention to go through these bodies of code to learn about MPEG-4 and to create an implementation. The complexity of the MPEG-4 standard combined with poor documentation and lack of useful comments made the task much more difficult. This was true for GPAC as many sections of code were uncommented and remained a mystery of its true functionality. It became a trial-and-error search for understanding MPEG-4 and the GPAC modules.

MPEG-4 has been praised and labeled as the technology of the coming future. It is unlikely that MPEG will release its documents for free, but there is a chance that other open source projects will become available or the current open source projects will provide better comments. The growth of interest will push for better documentation and technical journals to be released. As for now, MPEG-4 documentation remains in the confines of large companies and organizations.

4.3 Implementation of Interface Layer

Our MPEG-4 player is a partial implementation of the standard. It is built upon the core of GPAC with J2ME interfacing code written upon it with the porting at various completed stages. The interface layers replace the functionality of mp4_io and mp4_rend modules of GPAC but are coded in J2ME. Through the programs implemented, it is possible to view simple MPEG-4 graphics.

As the program functions now, it requires MPEG-4 media to be passed through a modified version GPAC to output its command calls. The list of command calls is then compressed and available for the client program to use. The client contacts the web server, downloads the calls, and renders the output on a mobile device.

If technical documentation becomes available, future work can remove the modified GPAC that is required for parsing and reading MPEG-4 files, and in its place a fully ported version of GPAC. Through our work it has been shown that an MPEG-4 player is a viable option for mobile devices for the near future. For a complete implementation of an MPEG-4 standard documentation, hardware, and software issues need to be resolved.

4.4 Overview of Our Reduced Prototype

To overcome the obstacle of converting GPAC into a J2ME program, we modified GPAC to work with a J2ME interface. We separated our original goal of a MPEG-4 player into two programs. The modified version of GPAC was used to open and parse MPEG-4 files that would then be passed to our client J2ME interface via the

Internet. This was the only feasible approach to be able to parse MPEG-4 files since we do not have the information to write our own parser.

GPAC was modified to output its own internal calls and BIFS code into a text file. The textual calls made were then compressed using bytes to represent the calls and parameters. This was done to reduce the amount of memory required to store textual calls. In turn, mobile phone memory for both transferring and storing the data is reduced. To get a better understanding of the calls, we parsed simple MPEG-4 files with shapes through the modified GPAC. In viewing the calls made, most of them represented drawing functions for lines, connections, colors, and gradients. Once compressed, the calls are then stored on a web server to be downloaded by the client. Figure 8 shows our MPEG-4 prototype.

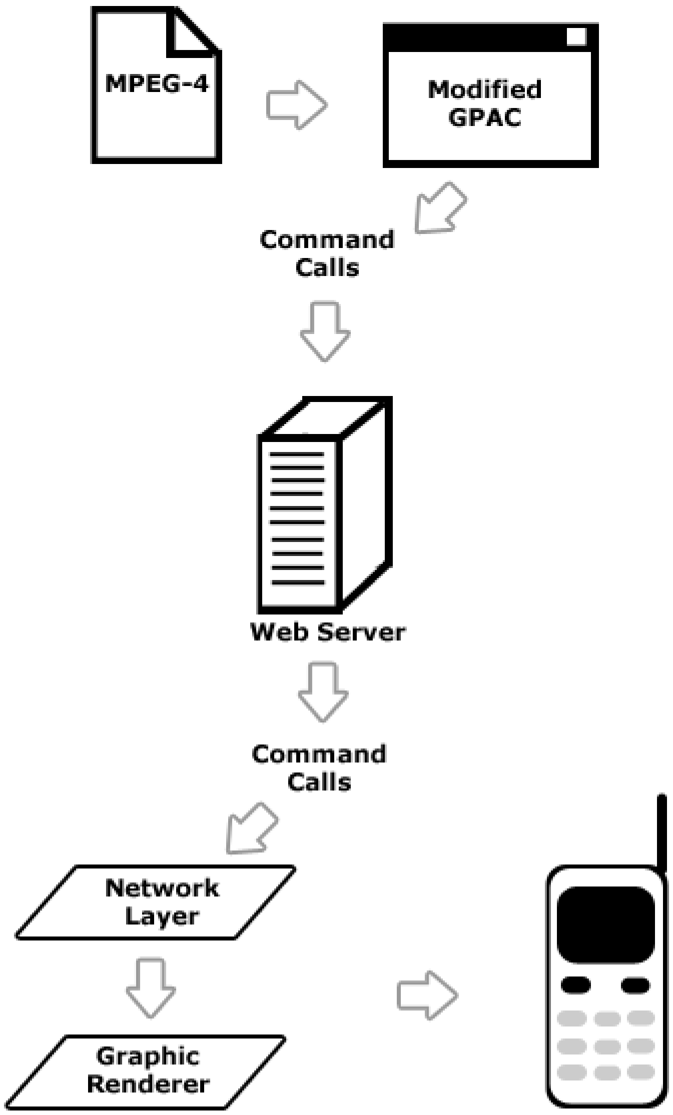


Figure 8: Our MPEG-4 Prototype

The overall conversion process is a small fraction of the GPAC, and even smaller fraction of the whole MPEG-4 standard. From GPAC we rewrote network input, the graphics renderer, and partially converted the MPEG-4 main systems. From the MPEG-4 standard we tried to implement the 2D profile for animation, but could not get all the systems required for animation to work together.

5 Conclusion and Future Work

MPEG-4 will become the standard media format for rich content delivery in the near future. The MPEG-4 standard was created to meet the needs of growing consumer and technological needs. MPEG-4 was designed for a large range of devices. These devices include everything from small-embedded devices such as mobile phones, PDAs, set top boxes, to high-end systems such as High Definition TV. Through its support of multiple platforms and wide range of features, MPEG-4 has a lot of potentially useful applications.

5.1 Recommendations for MPEG-4 Adoption

Although MPEG-4 is a feature rich media format, it still has a limited amount of support outside of large corporations. Only large companies such as Apple and IBM have been developing applications to support the varied features of MPEG-4. Other organizations are still only supporting the lowest profile of MPEG-4, namely the audio and video profiles. In order to push for complete support of the MPEG-4 standard it has to be made publicly aware on both industry and consumer levels.

For MPEG-4 to be supported on mobile phones, both hardware and software have to improve. Currently, mobile phones are not powerful enough to support the complete standard of MPEG-4. The mobile phones that are being released with MPEG-4 only support poor quality audio and video streaming. The MPEG-4 standard addressed the limited capabilities of small-embedded devices by defining multiple profiles. Ideally for devices such as mobile phones, layers, sprites, vectors, and 2D animation would be used to overcome the limitations.

To gain support for MPEG-4, more organizations have to work on applications for the standard. The largest community of talented technical individuals exists on the Internet on programming forums. To gain the support of programming enthusiasts, MPEG has to release better technical documentation of its format. Better documentation allows more individuals to understand the format and begin work on projects. By gaining the support of individual programming enthusiasts and the programming communities available online, there will be a growth of open source MPEG-4 projects. Thus increasing the support for MPEG-4.

5.2 MQP Recommendations

There are many MQPs that could be created directly from this project or involve MPEG-4. Most notably is the continuation of this project to achieve a working implementation of the MPEG-4 standard for mobile phones. This would mean completing the conversion of GPAC code to Java. Another project involving MPEG-4 could test MPEG-4 audio and video streaming performance over a selection of varied networks. Since MPEG-4 has robust error correction features it would be instructive to test it under different circumstances.

Other branches could involve the technologies that MPEG-4 is based upon; technologies such as Apple’s Quicktime, VRML, or SMIL. This could be creating a player for mobile phone or researching the similarities and differences of the formats. A better understanding of the technologies that MPEG-4 is based on might help to understand the new format. The benefits of this approach are the availability of technical documentation for the existing formats.

It is also possible to continue the work on a PDA or desktop computer. This could be supporting the other profiles of MPEG-4. Regardless of the project, continuation or new, it is suggested that the technical documentations be bought, which will make it considerably easier for any sort of implementation of MPEG-4.

5.3 Potential Applications

MPEG-4 has many potential applications because of its broad range of features. MPEG-4 could be used for improving the quality and detail for television programs or making a broadcast interactive. Both the entertainment and education industry could benefit from MPEG-4. With MPEG-4, games such as a ‘choose-your-own-adventure’ type game could help to educate and entertain young children. With MPEG-4’s stream management it would be possible to start watching a movie on television and finish watching a lower resolution version on a mobile phone. The potential of MPEG-4 covers most conceivable applications for the near future.

References

3Gtoday (2004). 3G is Here Today. Retrieved September 9, 2004, from World Wide Web: http://www.3gtoday.com/index.html

Alcatel (January 2003). Understanding the Mobile Phone Market Drivers. Retrieved September 9, 2004, from World Wide Web: http://www.privateline.com/archive/alcaatel.pdf

Alvear, Jose (June 29, 2001). RealNetworks Partners with Symbian. Retrieved September 9, 2004, from World Wide Web: http://www.streamingmedia.com/article.asp?id=7609

Apple Computer Inc (2004). MPEG-4: The Next Generation. Retrieved September 9, 2004, from World Wide Web: http://www.apple.com/mpeg4/

Banan, Mohsen (May 26, 2000). The WAP Trap: An Expose of the Wireless Application Protocol. Retrieved January 13, 2005, from World Wide Web: http://www.freeprotocols.org/wapTrap/split/main.html

Baron, Stanley N (1996). Digital Image and Audio Cmmunications : Toward a Global Information Infrastructure. New York : Van Nostrand Reinhold

Batista, Elisa (March 19, 2004). New Cell Phones Smarter, More Fun. Retrieved September 9, 2004, from World Wide Web: http://www.wired.com/news/wireless/0,1382,58085,00.html

BBC News (January 2004). Text record smashed for New Year. Retrieved September 9, 2004, from World Wide Web: http://news.bbc.co.uk/1/hi/technology/3368815.stm

Clark, Robert (January 2004). CDMA 450 – All You Need to Know. Retrieved January 12, 2005, from World Wide Web: http://www.telecomasia.net/telecomasia/article/articleDetail.jsp?id=94929

Datta, Kanika (June 2004). Nokia Makes a Connection. Retrieved September 9, 2004, from World Wide Web: http://www.business-standard.com/strategist/storypage.php?chklogin=&autono=158076&lselect=3&leftnm=lmnu7&leftindx=7

DeZoysa, Sanjima (March 2002). Japan and Europe – Worlds Apart? – Profitable Carrier Strategies. Retrieved January 13, 2004, from World Wide Web: http://www.findarticles.com/p/articles/mi_m0IUL/is_3_36/ai_84211212

ENST (2003). Osmose @ ENST. Retrieved September 24, 2004, from World Wide Web: http://www.comelec.enst.fr/osmo4/

Gartner (2004). Press Releases. Retrieved September 9, 2004, from World Wide Web: http://www4.gartner.com/press_releases/pr2004.html

Gartner (2004). Quick Statistics. Retrieved September 9, 2004, from World Wide Web: http://www4.gartner.com/media_relations/asset_61934_1595.jsp

GPAC (2004). GPAC Project on Advanced Content. Retrieved August 16, 2004, from World Wide Web: http://gpac.sourceforge.net/

Gross, Daniel (2004). Buy Cell – How Many Mobile Phones Does the World Need? Retrieved January 13, 2005, from World Wide Web: http://slate.msn.com/id/2101625/

GSMArena (April 2004). EMEA Mobile device market Q1 2004. Retrieved September 9, 2004, from World Wide Web: http://www.gsmarena.com/newsdetail.php3?idNews=37

IBM alphaWorks (December 2004). IBM alphaWorks – Emerging Technologies. . Retrieved Decemeber 13, 2004, from World Wide Web: http://www.alphaworks.ibm.com/

ITFacts.biz. (2004). 170 mln mobile phones sold in Q3 2004. Retrieved December 12, 2004, from World Wide Web: http://www.itfacts.biz/index.php?id=P2039

Java Community Process (2004). JSR-000037 Mobile Information Device Profile (MIDP). Retrieved September 9, 2004, from World Wide Web: http://jcp.org/aboutJava/communityprocess/final/jsr037/index.html

JBenchmark (2004). JBenchmark 2.0 Results. Retrieved September 9, 2004, from World Wide Web: http://www.jbenchmark.com/index.html?F=2

Jenson, Scott (2004). 16 Default Thinking: Why consumer products fail. Retrieved September 9, 2004, from World Wide Web: http://www.jensondesign.com/DefaultThinking.pdf

Knudsen, Jonathan. (November 2002). What's New in MIDP 2.0. Retrieved December 18, 2004 on the World Wide Web: http://developers.sun.com/techtopics/mobility/midp/articles/midp20/

Kosch, Harald (2004) Distributed Multimedia Database Technologies : Supported by MPEG-7 and MPEG-21. Boca Raton, FL : CRC Press.

Mobil.cz (2004) Mobil.cz. Retrieved September 9, 2004, from World Wide Web: http://www.mobil.cz

MPEG4IP (2004). MPEG4IP Open Streaming Video and Audio. Retrieved August 10, 2004, from World Wide Web: http://www.mpeg4ip.net/

Nokia (2004). Nokia Forum. Retrieved September 9, 2004, from World Wide Web: http://forum.nokia.com

Parbat (August 5, 2003). GSM Overtakes CDMA in Q1, Surges 81% Globally. Retrieved January 13, 2005, from World Wide Web: http://economictimes.indiatimes.com/cms.dll/html/uncomp/articleshow?msid=112991

Pearson, Chris (February 24, 2004). 1 Billion for GSM Wireless. Retrieved January 13, 2005, from World Wide Web: http://www.3g.co.uk/PR/Feb2004/6631.htm

Rogerson, Simon (October 2003). What is wrong with mobile phones? Retrieved September 9, 2004, from World Wide Web: http://www.ccsr.cse.dmu.ac.uk/resources/general/ethicol/Ecv13no5.html

RTSP (2004). rtsp.org Real Time Streaming Protocol Information and Updates. Retrieved August 22, 2004, from World Wide Web: http://www.rtsp.org/

Siemens (2004). Siemens Mobile Phone Overview. Retrieved September 9, 2004, from World Wide Web: http://www.siemens-mobile.com/developer

Sony Ericsson (2004). Sony Ericsson Phone Specification Overview. Retrieved September 9, 2004, from World Wide Web: http://developer.sonyericsson.com/

Sun Microsystems, Inc. (November 2002). J2ME Technologies Overview. Retrieved December 18, 2004 on the World Wide Web: http://java.sun.com/j2me/docs/j2me-ds.pdf

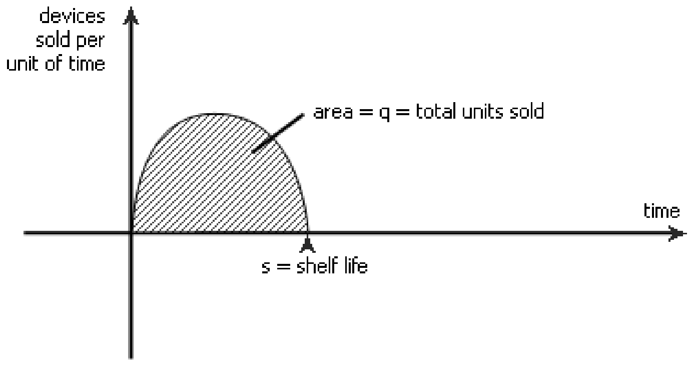
Sun Microsystems, Inc. (May 2004). J2ME Web Services (JSR-172) White Paper. Retrieved December 18, 2004 on the World Wide Web: http://java.sun.com/j2me/docs/webserv/Webservices172.pdf

Sun Ming-Ting (2000). Compressed Video Over Networks. New York : Marcel Dekker.

Appendix A – Mobile Phone Lifespan

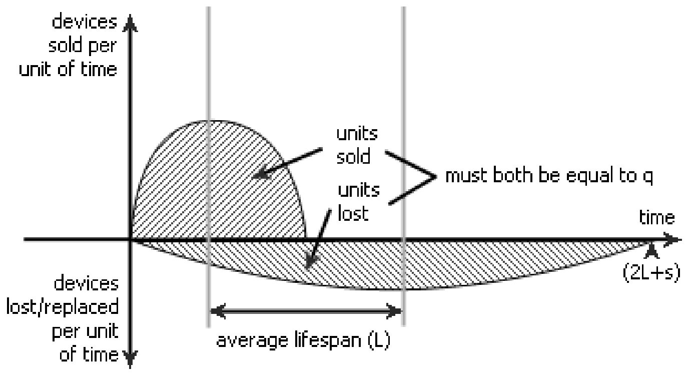
Assume the sales of a mobile phone follow a pattern as shown on the graph:

(A new phone release generally tends to have low demand, gradually increasing due to price decreases, special offers and acceptance of newly implemented technologies if any. Reaching a peak in the middle of the phone's shelf life, the demand withers as the phone becomes obsolete.)



According to the definition of a phone's lifespan, the devices misplaced, disposed of, damaged or replaced can thus be represented as shown:

(If the lifespan of a phone represents the average amount of time a user keeps the phone, the highest number of phones misplaced/replaced/etc will occur at one average lifespan after the peak in demand, and the number of phones remaining after two lifespans since the last phone sold can be considered insignificant.)



The equation of the first curve in terms of *s* and *q* can now be found:

The general form of the equation: Ax^2+Bx+C

- since $Ax^2+Bx+C=0$ when $x=0$ => $\mathbf{[C=0]}$
- since the peak (inflection point) occurs at $(1/2)s$ => $d/dx(Ax^2+Bx+C)=0$ when $x=(1/2)s$
  $2Ax+B$ => $2A*(1/2)s+B=0$ => $\mathbf{[B=-As]}$
- since the area from *0* to *s* under the graph is *q* => $\int(Ax^2+Bx+C)dx=q$ when $x=s$
  $(1/3)Ax^3+(1/2)Bx^2+Cx$ => $\mathbf{[(1/3)As^3+(1/2)Bs^2+Cs=q]}$
- solving the simultaneous equations yields: $(1/3)As^3-(1/2)As^3=q$ => $A(-s^3/6)=q$
  => $\mathbf{[A=-6q/s^3]}$
- and: $\mathbf{[B=6q/s^2]}$

The first curve can thus be written as $\mathbf{[(-6q/s^3)x^2+(6q/s^2)x]}$ *(Eq.1)*

The equation of the second curve can be derived by substituting $s=(2L+s)$ and $q=-q$
yielding: $\mathbf{[(6q/(2L+s)^3)x^2-(6q/(2L+s)^2)x]}$ *(Eq.2)*

The total number of units sold at any given point *x* if *(x<s)* is the sum of the areas under both curves in the region from *0* to *x*, thus is equal to: $\int(Eq.1)dx+\int(Eq.2)dx$
Integrating yields: $\mathbf{[6q(-x^3/(3s^3)+x^2/(2s^2))+6q(x^3/(3(2L+s)^3)-x^2/(2(2L+s)^2))]}$

When *(x=>s)*, the equation is simplified to: $\mathbf{[q+6q(x^3/(3(2L+s)^3)-x^2/(2(2L+s)^2))]}$

Also, if *(x>2L+s)*, the number of units by definition is *0*.

Appendix B – Mobile Phone Database

A database of all mobile phones released in the past 2 years by those companies for the EMEA market has been compiled to serve as a representation of the entire GSM phone market. The data was gathered from the respective companies website and from Jbenchmark.

SIEMENS	Release Date (YYYY.MM):	Data Transfer:	WAP Version:	Sound Formats: *square brackets indicate formats that cannot be played by the loudspeaker*
M50	2002.05	**GPRS**	1.2	MIDI (mono)
C55	2002.09	**GPRS**	1.2.1	MIDI, WAV
A50	2002.12	**CSD**	1.2.1	mono
S55	2003.01	**GPRS**	1.2.1	MIDI, SMAF
A55	2003.04	**CSD**	1.2.1	poly
SL55	2003.04	**GPRS**	1.2.1	MIDI, SMAF
M55	2003.07	**GPRS**	1.2.1 parts 2.0	MIDI, WAV
ST55	2003.08	**GPRS**	2.0	MIDI, SMAF
A52	2003.09	**CSD**	1.2.1	poly
MC60	2003.11	**GPRS**	1.2.1 parts 2.0	MIDI, WAV
C60	2003.11	**GPRS**	1.2.1	MIDI
A60	2003.12	**GPRS**	1.2.1	poly
C62	2003.12	**GPRS**	1.2.1	MIDI
SX1	2004.02	**GPRS**	2.0	MIDI, WAV
ST60	2004.03	**GPRS**	2.0	MIDI (and probably more)
CF62	2004.05	**GPRS**	1.2.1 parts 2.0	MIDI, WAV
CX65	2004.05	**GPRS**	2.0	MIDI, WAV
C65	announced	**GPRS**	?	?
M65	announced	**GPRS**	?	?
S65	announced	**GPRS**	?	?
U10	?	**UMTS**	1.2.1, 2.0	MIDI, WAV, MP3, AAC
U15	?	**UMTS**	1.2.1, 2.0	MIDI, WAV, MP3, AAC

SIEMENS	Screen Width:	Screen Height:	Colours:	MIDP Version:	Additional Java APIs: *all phones with Java have CLDC 1.0 unless specified otherwise*
M50	101	64	**2**	**1.0**	Game API
C55	101	64	**2**	**1.0**	Game API
A50	101	64	**2**	**N/A**	
S55	101	80	**256**	**1.0**	Game API
A55	101	64	**2**	**N/A**	
SL55	101	80	**4096**	**1.0**	Game API
M55	101	80	**4096**	**1.0**	Game API
ST55	120	160	**65536**	**N/A**	
A52	101	64	**2**	**N/A**	
MC60	101	80	**4096**	**1.0**	Game API, Bitmap API
C60	101	80	**4096**	**1.0**	Game API
A60	101	80	**4096**	**N/A**	
C62	128	128	**4096**	**N/A**	
SX1	176	220	**65536**	**1.0**	Game API, JSR-(120,135,82)
ST60	120	160	**65536**	**2.0**	?
CF62	130	130	**65536**	**1.0**	(probably Game API)
CX65	132	176	**65536**	**2.0**	(probably JSR-184 and more)
C65	?	?	**65536**	**?**	
M65	132	176	**65536**	**2.0**	
S65	132	176	**65536**	**?**	
U10	176	220	**4096**	**yes**	
U15	176	220	**65536**	**yes**	

	SymbianOS Version:	Additional Info:	Jbenchmark Rating:
SIEMENS		*the 3GPP video standard uses the H.263 codec*	
M50			**552**
C55			**489**
A50			N/A
S55			**763**
A55			N/A
SL55			**543**
M55			**651**
ST55			N/A
A52			N/A
MC60			**549**
C60			**330**
A60			N/A
C62			N/A
SX1	**6.1**	Series60 1.2.2, 130Mhz TI OMAP 310	**1971**
ST60			**729**
CF62			**224**
CX65			**1426**
C65			**800**
M65		3GPP	**1213**
S65		MPEG-4	**1376**
U10		MPEG-4	?
U15		MPEG-4	?

NOKIA	Release Date (YYYY.MM):	Data Transfer:	WAP Version:	Sound Formats:
3410	2002.05	**CSD**	1.1	poly
7650	2002.06	**GPRS**	1.2.1	MIDI, AMR
7210	2002.11	**GPRS**	1.2.1	MIDI
5100	2002.12	**GRPS**	1.2.1, TCP/IP	MIDI
6100	2002.12	**GPRS**	1.2.1	MIDI
6610	2002.12	**GPRS**	1.2.1	MIDI
3510i	2003.02	**GPRS**	1.2.1	MIDI
6800	2003.03	**GPRS**	1.2.1	MIDI
3650	2003.04	**GPRS**	1.2.1	MIDI, AMR
7250	2003.05	**GPRS**	1.2.1	MIDI
8910i	2003.06	**GPRS**	1.2.1	mono
2100	2003.06	**N/A**	0	mono
3300	2003.07	**GPRS**	1.2.1	MIDI, WB-AMR, [MP3, AAC]
N-GAGE	2003.10	**GPRS**	1.2.1	MIDI, AMR, WAV, MP3, AAC
3100	2003.11	**GPRS**	1.2.1	MIDI
6220	2003.12	**EDGE**	1.2.1, TCP/IP	MIDI
3200	2004.01	**EDGE**	1.2.1	MIDI, AMR, [MP3, AAC]
7600	2004.01	**UMTS**	2.0, TCP/IP	MIDI, AMR, MP3, AAC
6600	2004.02	**GPRS**	2.0, TCP/IP	MIDI, WB-AMR
6820	2004.02	**EDGE**	2.0, TCP/IP	MIDI, AMR
1100	2004.02	**N/A**	0	mono
7200	2004.05	**EDGE**	2.0, TCP/IP	MIDI, AMR
6230	2004.05	**EDGE**	2.0, TCP/IP	MIDI, AMR, MP3, AAC
7610	2004.06	**GPRS**	2.0, TCP/IP	MIDI, WB-AMR, WAV, MP3, AAC, Real
6650	?	**UMTS**	2.0, TCP/IP	MIDI
N-GAGE QD	announced	**GPRS**	1.2.1, TCP/IP	MIDI, AMR, WAV
3220	announced	**EDGE**	2.0, TCP/IP	MIDI
3660	announced	**GPRS**	1.2.1	MIDI, AMR
5140	announced	**EDGE**	2.0, TCP/IP	MIDI, AMR
7700	announced	**EDGE**	TCP/IP	MIDI, AMR, WAV, MP3, AAC, Real
9500	announced	**EDGE**	TCP/IP	MIDI, AMR, WAV, MP3, AAC

NOKIA	Screen Width:	Screen Height:	Colours:	MIDP Version:	Additional Java APIs:
3410	96	65	**2**	**1.0**	UI API, JSR-120
7650	176	208	**4096**	**1.0**	UI API
7210	128	128	**4096**	**1.0**	UI API
5100	128	128	**4096**	**1.0**	UI API
6100	128	128	**4096**	**1.0**	UI API
6610	128	128	**4096**	**1.0**	UI API
3510i	96	65	**4096**	**1.0**	UI API
6800	128	128	**4096**	**1.0**	UI API, JSR-120
3650	176	208	**4096**	**1.0**	UI API, JSR-(120,135)
7250	128	128	**4096**	**1.0**	UI API
8910i	96	65	**4096**	**1.0**	UI API
2100	96	65	**2**	**N/A**	
3300	128	128	**4096**	**1.0**	UI API, JSR-(120,135)
N-GAGE	176	208	**4096**	**1.0**	UI API, JSR-(120,135)
3100	128	128	**4096**	**1.0**	UI API, JSR-120
6220	128	128	**4096**	**1.0**	UI API, JSR-120
3200	128	128	**4096**	**1.0**	UI API, JSR-120
7600	128	160	**65536**	**1.0**	UI API, JSR-(120,135)
6600	176	208	**65536**	**2.0**	UI API, JSR-(120,135,82)
6820	128	128	**4096**	**1.0**	UI API, JSR-120
1100	96	65	**2**	**N/A**	
7200	128	128	**65536**	**1.0**	UI API, JSR-120
6230	128	128	**65536**	**2.0**	UI API, JSR-(120,135,185,82), CLDC 1.1
7610	176	208	**65536**	**2.0**	UI API, JSR-(120,135,82)
6650	128	160	**4096**	**1.0**	UI API
N-GAGE QD	176	208	**4096**	**1.0**	UI API, JSR-(120,135)
3220	128	128	**65536**	**2.0**	UI API, JSR-(120,135), CLDC 1.1
3660	176	208	**65536**	**1.0**	UI API, JSR-(120,135)
5140	128	128	**4096**	**2.0**	UI API, JSR-(120,135,185), CLDC 1.1
7700	640	320	**65536**	**2.0**	UI API, JSR-(120,135,82)
9500	640	200	**65536**	**2.0**	UI API, JSR-(120,135,82,185,36,46,75), File Connections, Personal Profile

NOKIA	SymbianOS Version:	Additional Info: *note: the full specifications for the Siemens 6X series haven't been released yet.*	Jbenchmark Rating:
3410			**541**
7650	**6.1**	Series60 1.0	**2524**
7210		Series40 1.0	**692**
5100		Series40 1.0	**679**
6100		Series40 1.0	**760**
6610		Series40 1.0	**705**
3510i		Series40 1.0	**941**
6800		Series40 1.0	**954**
3650	**6.1**	Series60 1.0, 3GPP	**2229**
7250		Series40 1.0	**752**
8910i		Series40 1.0	**838**
2100			N/A
3300		Series40 1.0	**700**
N-GAGE	**6.1**	Series60 1.0, 3GPP	**2169**
3100		Series40 1.0	**794**
6220		Series40 1.0	**768**
3200		Series40 1.0	**869**
7600		Series40 1.0, 3GPP, GPRS	**1913**
6600	**7.0s**	Series60 2.0, 3GPP, MPEG-4, Real	**2411**
6820		Series40 1.0, 3GPP	**943**
1100			N/A
7200		Series40 1.0, 3GPP, MPEG-4	**614**
6230		Series40 2.0, 3GPP	**1953**
7610	**7.0s**	Series60 2.0, 3GPP, MPEG-4, Real	**2704**
6650		Series40 1.0	**2415**
N-GAGE QD	**6.1**	Series60 1.0, 3GPP	?
3220		Series40 2.0, 3GPP	?
3660	**6.1**	Series60 1.0, 3GPP	?
5140		Series40 2.0, 3GPP	**768**
7700	**7.0s**	Series90 2.0, 3GPP, MPEG-4, Real, touchscreen	**954**
9500	**7.0s**	Series80 2.0, 3GPP, MPEG-4, Real	**861**

SONY-ERICSSON	Release Date (YYYY.MM):	Data Transfer:	WAP Version:	Sound Formats:
T68i	2002.04	**GPRS**	2.0	mono
R600	2002.09	**GPRS**	1.2.1	mono
T200	2002.10	**GPRS**	1.2.1	mono
T300	2002.10	**GPRS**	2.0	MIDI, AMR
T600	2002.11	**GPRS**	1.2.1	mono
T100	2003.01	**CSD**	1.2.1	mono
P800	2003.02	**GPRS**	2.0	MIDI, AMR, MP3, RMF
T310	2003.03	**GPRS**	2.0	MIDI, AMR
T610	2003.10	**GPRS**	2.0	MIDI, AMR
P900	2003.11	**GPRS**	2.0	MIDI, AMR, MP3, RMF
T230	2003.12	**GPRS**	2.0	MIDI, AMR
Z600	2003.12	**GPRS**	2.0	MIDI, AMR
T630	2004.02	**GPRS**	2.0	MIDI, AMR
Z200	2004.03	**GPRS**	1.2.1	MIDI, SMAF
K700i	announced	**GPRS**	2.0	MIDI, AMR, MP3
S700	announced	**GPRS**	2.0	MIDI, AMR, MP3
Z500	announced	**EDGE**	2.0	MIDI, AMR, MP3
Z1010	announced	**UMTS**	2.0	MIDI, AMR, MP3

SONY-ERICSSON	Screen Width:	Screen Height:	Colours:	MIDP Version:	Additional Java APIs:
T68i	101	80	**256**	**N/A**	
R600	101	65	**2**	**N/A**	
T200	101	67	**4**	**N/A**	
T300	101	80	**256**	**N/A**	
T600	101	65	**4**	**N/A**	
T100	101	67	**4**	**N/A**	
P800	208	320	**4096**	**1.0**	PersonalJava
T310	101	80	**256**	**N/A**	
T610	128	160	**65536**	**1.0**	JSR-135
P900	208	320	**65536**	**2.0**	JSR-(135,120,82), PersonalJava
T230	101	80	**4096**	**N/A**	
Z600	128	160	**65536**	**1.0**	JSR-135
T630	128	160	**65536**	**1.0**	JSR-135
Z200	128	160	**4096**	**N/A**	
K700i	176	220	**65536**	**2.0**	JSR-(135,120,184), CLDC 1.1
S700	240	320	**262144**	**2.0**	JSR-(135,120,184), CLDC 1.1
Z500	128	160	**65536**	**2.0**	JSR-(135,120,184), CLDC 1.1
Z1010	176	220	**65536**	**2.0**	JSR-(135,120), CLDC 1.1

SONY-ERICSSON	SymbianOS Version:	Additional Info:	Jbenchmark Rating:
T68i			N/A
R600			N/A
T200			N/A
T300			N/A
T600			N/A
T100			N/A
P800	**7.0**	UIQ 2.0 + specific APIs, 3GPP, MPEG-4, RMF	**2436**
T310			N/A
T610			**692**
P900	**7.0**	UIQ 2.1 + specific APIs, 3GPP, MPEG-4, RMF	**2203**
T230			N/A
Z600			**693**
T630			**803**
Z200			N/A
K700i		3GPP, MPEG-4	**2691**
S700		3GPP, MPEG-4	?
Z500		3GPP, MPEG-4	?
Z1010		3GPP, MPEG-4	**2572**

Appendix C – Mobile Phone Trends

The database contains the release date of each phone and the features relevant to data streaming, including Internet connectivity, audio playback capabilities, screen properties and the availability and version of Java implementation among others. The data was gathered from the respective companies website and from Jbenchmark.

	Release Date (YYYY.MM):	Data Transfer:	WAP Version:	Sound Formats:
T68i	2002.04	**GPRS**	2.0	mono
M50	2002.05	**GPRS**	1.2	MIDI (mono)
3410	2002.05	**CSD**	1.1	poly
7650	2002.06	**GPRS**	1.2.1	MIDI, AMR
C55	2002.09	**GPRS**	1.2.1	MIDI, WAV
R600	2002.09	**GPRS**	1.2.1	mono
T200	2002.10	**GPRS**	1.2.1	mono
T300	2002.10	**GPRS**	2.0	MIDI, AMR
7210	2002.11	**GPRS**	1.2.1	MIDI
T600	2002.11	**GPRS**	1.2.1	mono
A50	2002.12	**CSD**	1.2.1	mono
5100	2002.12	**GRPS**	1.2.1, TCP/IP	MIDI
6100	2002.12	**GPRS**	1.2.1	MIDI
6610	2002.12	**GPRS**	1.2.1	MIDI
S55	2003.01	**GPRS**	1.2.1	MIDI, SMAF
T100	2003.01	**CSD**	1.2.1	mono
3510i	2003.02	**GPRS**	1.2.1	MIDI
P800	2003.02	**GPRS**	2.0	MIDI, AMR, MP3, RMF
6800	2003.03	**GPRS**	1.2.1	MIDI
T310	2003.03	**GPRS**	2.0	MIDI, AMR
A55	2003.04	**CSD**	1.2.1	poly
SL55	2003.04	**GPRS**	1.2.1	MIDI, SMAF
3650	2003.04	**GPRS**	1.2.1	MIDI, AMR
7250	2003.05	**GPRS**	1.2.1	MIDI
8910i	2003.06	**GPRS**	1.2.1	mono
2100	2003.06	**N/A**	N/A	mono

	Screen Width:	Screen Height:	Colours:	MIDP Version:	Additional Java APIs:
T68i	101	80	**256**	**N/A**	
M50	101	64	2	**1.0**	Game API
3410	96	65	2	**1.0**	UI API, JSR-120
7650	176	208	**4096**	**1.0**	UI API
C55	101	64	2	**1.0**	Game API
R600	101	65	2	**N/A**	
T200	101	67	4	**N/A**	
T300	101	80	**256**	**N/A**	
7210	128	128	**4096**	**1.0**	UI API
T600	101	65	4	**N/A**	
A50	101	64	2	**N/A**	
5100	128	128	**4096**	**1.0**	UI API
6100	128	128	**4096**	**1.0**	UI API
6610	128	128	**4096**	**1.0**	UI API
S55	101	80	**256**	**1.0**	Game API
T100	101	67	4	**N/A**	
3510i	96	65	**4096**	**1.0**	UI API
P800	208	320	**4096**	**1.0**	PersonalJava
6800	128	128	**4096**	**1.0**	UI API, JSR-120
T310	101	80	**256**	**N/A**	
A55	101	64	2	**N/A**	
SL55	101	80	**4096**	**1.0**	Game API
3650	176	208	**4096**	**1.0**	UI API, JSR-(120,135)
7250	128	128	**4096**	**1.0**	UI API
8910i	96	65	**4096**	**1.0**	UI API
2100	96	65	2	**N/A**	

	SymbianOS Version:	Additional Info:	
T68i			N/A
M50			**552**
3410			**541**
7650	**6.1**	Series60 1.0	2524
C55			**489**
R600			N/A
T200			N/A
T300			N/A
7210		Series40 1.0	**692**
T600			N/A
A50			N/A
5100		Series40 1.0	**679**
6100		Series40 1.0	**760**
6610		Series40 1.0	**705**
S55			**763**
T100			N/A
3510i		Series40 1.0	**941**
P800	**7.0**	UIQ 2.0 + specific APIs, 3GPP, MPEG-4, RMF	2436
6800		Series40 1.0	**954**
T310			N/A
A55			N/A
SL55			**543**
3650	**6.1**	Series60 1.0, 3GPP	2229
7250		Series40 1.0	**752**
8910i		Series40 1.0	**838**
2100			N/A

	Release Date (YYYY.MM):	Data Transfer:	WAP Version:	Sound Formats:
M55	2003.07	**GPRS**	1.2.1 parts 2.0	MIDI, WAV
3300	2003.07	**GPRS**	1.2.1	MIDI, WB-AMR, [MP3, AAC]
ST55	2003.08	**GPRS**	2.0	MIDI, SMAF
A52	2003.09	**CSD**	1.2.1	poly
N-GAGE	2003.10	**GPRS**	1.2.1	MIDI, AMR, WAV, MP3, AAC
T610	2003.10	**GPRS**	2.0	MIDI, AMR
MC60	2003.11	**GPRS**	1.2.1 parts 2.0	MIDI, WAV
C60	2003.11	**GPRS**	1.2.1	MIDI
3100	2003.11	**GPRS**	1.2.1	MIDI
P900	2003.11	**GPRS**	2.0	MIDI, AMR, MP3, RMF
A60	2003.12	**GPRS**	1.2.1	poly
C62	2003.12	**GPRS**	1.2.1	MIDI
6220	2003.12	**EDGE**	1.2.1, TCP/IP	MIDI
T230	2003.12	**GPRS**	2.0	MIDI, AMR
Z600	2003.12	**GPRS**	2.0	MIDI, AMR
3200	2004.01	**EDGE**	1.2.1	MIDI, AMR, [MP3, AAC]
7600	2004.01	**UMTS**	2.0, TCP/IP	MIDI, AMR, MP3, AAC
SX1	2004.02	**GPRS**	2.0	MIDI, WAV
6600	2004.02	**GPRS**	2.0, TCP/IP	MIDI, WB-AMR
6820	2004.02	**EDGE**	2.0, TCP/IP	MIDI, AMR
1100	2004.02	**N/A**	N/A	mono
T630	2004.02	**GPRS**	2.0	MIDI, AMR
ST60	2004.03	**GPRS**	2.0	MIDI (and probably more)
Z200	2004.03	**GPRS**	1.2.1	MIDI, SMAF
CF62	2004.05	**GPRS**	1.2.1 parts 2.0	MIDI, WAV
CX65	2004.05	**GPRS**	2.0	MIDI, WAV
7200	2004.05	**EDGE**	2.0, TCP/IP	MIDI, AMR
6230	2004.05	**EDGE**	2.0, TCP/IP	MIDI, AMR, MP3, AAC
7610	2004.06	**GPRS**	2.0, TCP/IP	MIDI, WB-AMR, WAV, MP3, AAC, Real
C65	announced	**GPRS**	(probably 2.0)	(probably MIDI and more)
M65	announced	**GPRS**	(probably 2.0)	(probably MIDI and more)
S65	announced	**GPRS**	(probably 2.0)	(probably MIDI and more)
K700i	announced	**GPRS**	2.0	MIDI, AMR, MP3
S700	announced	**GPRS**	2.0	MIDI, AMR, MP3
Z500	announced	**EDGE**	2.0	MIDI, AMR, MP3
N-GAGE QD	announced	**GPRS**	1.2.1, TCP/IP	MIDI, AMR, WAV
3220	announced	**EDGE**	2.0, TCP/IP	MIDI
3660	announced	**GPRS**	1.2.1	MIDI, AMR
5140	announced	**EDGE**	2.0, TCP/IP	MIDI, AMR
7700	announced	**EDGE**	TCP/IP	MIDI, AMR, WAV, MP3, AAC, Real
9500	announced	**EDGE**	TCP/IP	MIDI, AMR, WAV, MP3, AAC

	Screen Width:	Screen Height:	Colours:	MIDP Version:	Additional Java APIs:
M55	101	80	**4096**	**1.0**	Game API
3300	128	128	**4096**	**1.0**	UI API, JSR-(120,135)
ST55	120	160	**65536**	**N/A**	
A52	101	64	2	**N/A**	
N-GAGE	176	208	**4096**	**1.0**	UI API, JSR-(120,135)
T610	128	160	**65536**	**1.0**	JSR-135
MC60	101	80	**4096**	**1.0**	Game API, Bitmap API
C60	101	80	**4096**	**1.0**	Game API
3100	128	128	**4096**	**1.0**	UI API, JSR-120
P900	208	320	**65536**	**2.0**	JSR-(135,120,82), PersonalJava
A60	101	80	**4096**	**N/A**	
C62	128	128	**4096**	**N/A**	
6220	128	128	**4096**	**1.0**	UI API, JSR-120
T230	101	80	**4096**	**N/A**	
Z600	128	160	**65536**	**1.0**	JSR-135
3200	128	128	**4096**	**1.0**	UI API, JSR-120
7600	128	160	**65536**	**1.0**	UI API, JSR-(120,135)
SX1	176	220	**65536**	**1.0**	Game API, JSR-(120,135,82)
6600	176	208	**65536**	**2.0**	UI API, JSR-(120,135,82)
6820	128	128	**4096**	**1.0**	UI API, JSR-120
1100	96	65	2	**N/A**	
T630	128	160	**65536**	**1.0**	JSR-135
ST60	120	160	**65536**	**2.0**	?
Z200	128	160	**4096**	**N/A**	
CF62	130	130	**65536**	**1.0**	(probably Game API)
CX65	132	176	**65536**	**2.0**	(probably JSR-184 and more)
7200	128	128	**65536**	**1.0**	UI API, JSR-120
6230	128	128	**65536**	**2.0**	UI API, JSR-(120,135,185,82), CLDC 1.1
7610	176	208	**65536**	**2.0**	UI API, JSR-(120,135,82)
C65	?	?	**65536**	? (2.0)	
M65	132	176	**65536**	**2.0**	
S65	132	176	**65536**	? (2.0)	
K700i	176	220	**65536**	**2.0**	JSR-(135,120,184), CLDC 1.1
S700	240	320	**262144**	**2.0**	JSR-(135,120,184), CLDC 1.1
Z500	128	160	**65536**	**2.0**	JSR-(135,120,184), CLDC 1.1
N-GAGE QD	176	208	**4096**	**1.0**	UI API, JSR-(120,135)
3220	128	128	**65536**	**2.0**	UI API, JSR-(120,135), CLDC 1.1
3660	176	208	**65536**	**1.0**	UI API, JSR-(120,135)
5140	128	128	**4096**	**2.0**	UI API, JSR-(120,135,185), CLDC 1.1
7700	640	320	**65536**	**2.0**	UI API, JSR-(120,135,82)
9500	640	200	**65536**	**2.0**	UI API, JSR-(120,135,82,185,36,46,75), File Connections, Personal Profile

	SymbianOS Version:	Additional Info:	
M55			**651**
3300		Series40 1.0	**700**
ST55			N/A
A52			N/A
N-GAGE	**6.1**	Series60 1.0, 3GPP	2169
T610			**692**
MC60			**549**
C60			**330**
3100		Series40 1.0	**794**
P900	**7.0**	UIQ 2.1 + specific APIs, 3GPP, MPEG-4, RMF	2203
A60			N/A
C62			N/A
6220		Series40 1.0	**768**
T230			N/A
Z600			**693**
3200		Series40 1.0	**869**
7600		Series40 1.0, 3GPP, GPRS	**1913**
SX1	**6.1**	Series60 1.2.2, 130Mhz TI OMAP 310	1971
6600	**7.0s**	Series60 2.0, 3GPP, MPEG-4, Real	2411
6820		Series40 1.0, 3GPP	**943**
1100			N/A
T630			**803**
ST60			**729**
Z200			N/A
CF62			**224**
CX65			**1426**
7200		Series40 1.0, 3GPP, MPEG-4	**614**
6230		Series40 2.0, 3GPP	**1953**
7610	**7.0s**	Series60 2.0, 3GPP, MPEG-4, Real	2704
C65			**800**
M65		3GPP	**1213**
S65		MPEG-4	**1376**
K700i		3GPP, MPEG-4	**2691**
S700		3GPP, MPEG-4	?
Z500		3GPP, MPEG-4	?
N-GAGE QD	**6.1**	Series60 1.0, 3GPP	?
3220		Series40 2.0, 3GPP	?
3660	**6.1**	Series60 1.0, 3GPP	?
5140		Series40 2.0, 3GPP	**768**
7700	**7.0s**	Series90 2.0, 3GPP, MPEG-4, Real, touchscreen	954
9500	**7.0s**	Series80 2.0, 3GPP, MPEG-4, Real	861

Appendix D – Mobile Phone 2005 Predicted Trends

Predicted average phone in the first half of 2005 in terms of graphics computation and display speed. Based on Mobile Phone Lifespan formula and data from Jbenchmark.

		data transfer			sound playback			MIDP version			colours				
		CSD/none	GPRS	EDGE+	no	just MIDI	yes	no Java	1.0	2.0	grey	256-4096	65536+	Symbian	total
no.	2H 2002	2	12	0	6	5	3	6	8	0	7	7	0	1	14
	1H 2003	3	9	0	4	3	5	3	9	0	3	9	0	2	12
	2H 2003	1	13	1	2	4	9	5	9	1	1	10	4	2	15
	1H 2004	1	8	5	1	0	13	2	7	5	1	3	10	3	14
	announced	0	7	5	0	1	11	0	2	10	0	2	10	4	12
%	**2H 2002**	**14.29**	**85.71**	**0.00**	**42.86**	**35.71**	**21.43**	**42.86**	**57.14**	**0.00**	**50.00**	**50.00**	**0.00**	**7.14**	
	1H 2003	**25.00**	**75.00**	**0.00**	**33.33**	**25.00**	**41.67**	**25.00**	**75.00**	**0.00**	**25.00**	**75.00**	**0.00**	**16.67**	
	2H 2003	**6.67**	**86.67**	**6.67**	**13.33**	**26.67**	**60.00**	**33.33**	**60.00**	**6.67**	**6.67**	**66.67**	**26.67**	**13.33**	
	1H 2004	**7.14**	**57.14**	**35.71**	**7.14**	**0.00**	**92.86**	**14.29**	**50.00**	**35.71**	**7.14**	**21.43**	**71.43**	**21.43**	
	announced	**0.00**	**58.33**	**41.67**	**0.00**	**8.33**	**91.67**	**0.00**	**16.67**	**83.33**	**0.00**	**16.67**	**83.33**	**33.33**	
	1H 2005	**10.34**	**72.46**	**17.21**	**16.95**	**17.46**	**65.60**	**23.29**	**54.52**	**22.18**	**14.14**	**47.52**	**38.34**	**18.03**	
	w/constant q	11.19	73.84	14.97	18.91	19.05	62.04	24.86	56.25	18.89	16.10	49.99	33.90	17.14	

	months till 1H 2005	q	q on 1H 2005	% in 1H 2005
2H 2002	30	100	31.64	**10.66**
1H 2003	24	114	57.00	**19.20**
2H 2003	18	128	87.50	**29.47**
1H 2004	12	146	85.34	**28.74**
2H 2004	6	164	35.47	**11.95**

		data transfer			sound playback			MIDP version			colours				% of
		CSD/none	GPRS	EDGE+	no	just MIDI	yes	no Java	1.0	2.0	grey	256-4096	65536+	Symbian	1H 2005
(%)	2H 2002	14.29	85.71	0.00	42.86	35.71	21.43	42.86	57.14	0.00	50.00	50.00	0.00	7.14	10.66
	1H 2003	25.00	75.00	0.00	33.33	25.00	41.67	25.00	75.00	0.00	25.00	75.00	0.00	16.67	19.20
	2H 2003	6.67	86.67	6.67	13.33	26.67	60.00	33.33	60.00	6.67	6.67	66.67	26.67	13.33	29.47
	1H 2004	7.14	57.14	35.71	7.14	0.00	92.86	14.29	50.00	35.71	7.14	21.43	71.43	21.43	28.74
	announced	0.00	58.33	41.67	0.00	8.33	91.67	0.00	16.67	83.33	0.00	16.67	83.33	33.33	11.95
(%)	1H 2005	10.34	72.46	17.21	16.95	17.46	65.60	23.29	54.52	22.18	14.14	47.52	38.34	18.03	

% of mobile phones with access to a given technology in the 1st half of 2005												
data transfer			sound playback			MIDP version			colours			
CSD/none	GPRS	EDGE+	no	just MIDI	yes	no Java	1.0	2.0	grey	256-4096	65536+	Symbian
10.34	72.46	17.21	16.95	17.46	65.60	23.29	54.52	22.18	14.14	47.52	38.34	18.03

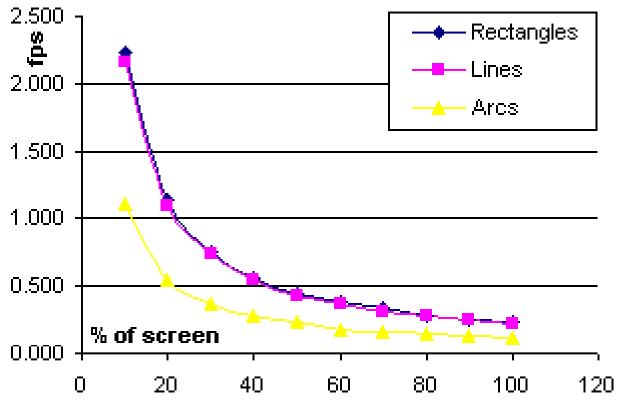
Jbenchmark:	2H'02	1H'03	2H'03	1H'04	2H'04		
	552	763	651	869	800		
	541	941	700	1913	1213		
	2524	*2436*	*2169*	*1971*	1376		
	489	954	692	*2411*	2691		
	692	543	549	943	768		
	679	*2229*	330	803	*954*		
	760	752	794	729	*861*		
	705	838	*2203*	224			
			768	1426			
			693	614			
				1953			
				2704			
Sum:	6942	9456	9549	16560	8663		
Average:	868	1182	955	1380	1238	**1124.4**	(JBenchmark rating for the average phone)
% in 1H'05:	10.66	19.2	29.47	28.74	11.95	**1145.4**	(JBenchmark rating for the average phone in 1H 2005)

Appendix E – Graphic Benchmarking Results

Benchmarking graphic results. Data collected from the tests that were done on the Siemens Model S55 mobile phone.

	fps (rectangles)			fps (arcs)			fps (lines)			rectangle numer equivalent		
% of screen	1	2	3	1	2	3	1	2	3	1	2	3
100	0.2	0.2	0.2	0.15	0.15	0.15	0.2	0.2	0.2	0.25	0.25	0.25
75	0.3	0.3	0.3	0.15	0.15	0.15	0.25	0.25	0.25	0.3	0.3	0.3
50	0.4	0.4	0.4	0.2	0.2	0.2	0.4	0.4	0.4	0.45	0.45	0.45
25	0.75	0.75	0.75	0.4	0.4	0.4	0.75	0.75	0.75	0.9	0.9	0.9
10	1.75	1.75	1.75	0.95	0.95	0.95	1.7	1.7	1.7	2.15	2.15	2.15
5	3.25	3.2	3.2	1.7	1.7	1.7	3.15	3.15	3.15	4.2	4.2	4.2
1	9.15	9.15	9.15	5	4.95	5	8.95	8.9	8.9	9.7	9.7	9.7

	Frames per second		
% of screen	**Rectangles**	**Lines**	**Arcs**
100	0.233	0.217	0.117
90	0.250	0.250	0.133
80	0.283	0.283	0.150
70	0.333	0.317	0.167
60	0.383	0.367	0.183
50	0.450	0.433	0.233
40	0.567	0.550	0.283
30	0.750	0.733	0.367
20	1.133	1.100	0.550
10	2.233	2.167	1.117



% of screen	**Fps.**
100	0.233
90	0.250
80	0.283
70	0.317
60	0.367
50	0.450
40	0.550
30	0.733
20	1.100
10	2.167