sonera | medialab

Mobile Java Application Development White Paper

Sonera MediaLab

www.medialab.sonera.fi
info@medialab.sonera.fi

September 12, 2002

Mobile Java Application Development

1 INTRODUCTION

Nowadays most new mobile phones are able to run downloaded external applications, not just the ones preinstalled on the device by the manufacturer. These applications can be native applications for the device's own operating system or platform independent applications that are run on special virtual machine (VM) software on the device. Java2 Micro Edition (J2ME) [1] offers a special environment for creating platform independent applications on mobile phones and other handheld devices. J2ME is divided into two parts: configurations and profiles. The configurations are specified by the memory size and computing power of the device, whereas the different profiles are specified by the user interface. The Mobile Information Device Profile (MIDP) [2,3] is targeted for mobile phones and is widely supported by all the major mobile phone manufacturers. Platform independence and wide support make J2ME and MIDP a interesting platform for mobile applications such as games. This white paper discusses developing applications for MIDP version 1.0. The upcoming version 2.0 will have many new features and extensions, but these are not within the scope of this document.

2 MIDP TECHNOLOGY

2.1 Overview

MIDP technology is based on Java2 Micro Edition and Connected, Limited Device Configuration (CLDC) by Sun Microsystems, Inc (see Fig. 1). CLDC is one of two configurations of Java2 Micro Edition. It is the foundation of the Java runtime environment targeting small, resource-constrained devices, such as mobile phones, personal digital assistants, and small retail payment terminals that have less than 512 kB of memory. MIDP is designed for creating applications and services to be run in small network-connectable handheld devices like mobile phones. The MIDP specification lists some requirements for the device and the MIDP application programming interface (API). MIDP applications are called MIDlets.

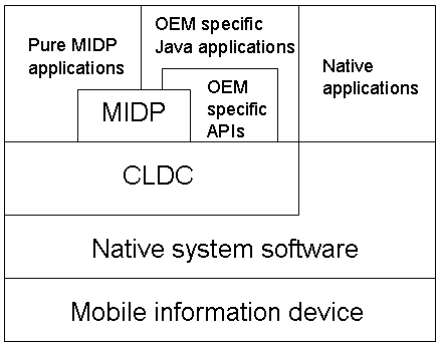

Figure 1: MIDP Architecture.

2.2 Mobile Information Device (MID)

MIDP compatible devices (MIDs) must satisfy some hardware and software requirements. These requirements make it possible to use a common MIDP API on all devices. The device itself can be a mobile phone, PDA, two-way pager or any other handheld device.

2.2.1 Minimum hardware requirements

The MIDP specification gives the following minimum requirements for the hardware of the device:

- **Display:** screen size 96x54 pixels, color depth: 1 bit, aspect ratio approximately 1:1.
- **Input:** (one or more of the methods must be supported): one-handed keyboard (ITU-T phone keyboard), two-handed keyboard (QWERTY keyboard), or touch screen.
- **Memory:** 128 kB of non-volatile[1] memory for MIDP components, 8 kB of non-volatile memory for application-created persistent data, 32 kB of volatile memory for the Java runtime.
- **Network:** two-way, wireless, limited bandwidth.

2.2.2 Minimum software requirements

MIDP also specifies some software requirements for the device's operating system and virtual machine. Some of the requirements are quite obvious: the device must have a kernel to manage hardware and scheduling, there must be a mechanism to read from and write to non-volatile memory, and there must be read and write access to the network interface. The system should also provide a time base for timer APIs and time stamping of persistent data. Software for application life cycle management should also be implemented. The system must also provide a minimal capability to write to a bit-mapped graphics buffer and to capture user input from one (or more) of the three input mechanisms (see minimum hardware requirements).

2.3 Application delivery and lifetime

One or more MIDlet applications can be packed to one MIDlet suite. MIDlet suites consist of two parts: the Java Application Descriptor (JAD) describes the applications in the suite and the Java Application Resource (JAR) holds the actual applications. The suites can be delivered to the mobile device by WAP, HTTP, IR, or Bluetooth, for example. The devices usually have special application management software to handle the downloading and life cycle of the applications. This software can also be called Java Application Manager (JAM) and it must be capable to browse or locate JADs, transfer JADs and JAR files to the device, install the downloaded MIDlet suite on the device, execute MIDlet suites, and allow the user to remove MIDlet suites stored on the device. A MIDlet's life cycle is illustrated in Figure 2.

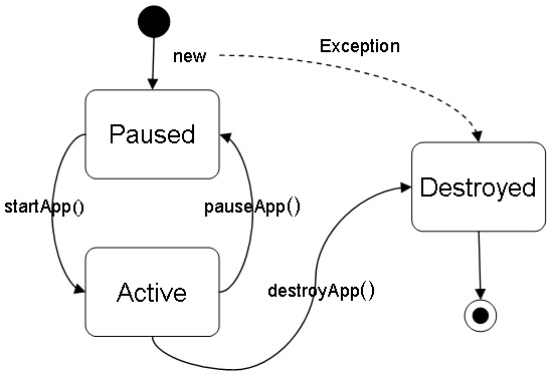

Figure 2: MIDlet life cycle.

[1] Volatile memory does not retain its contents when the user turns the device off. No special setup is needed to access the volatile memory. The most common type of volatile memory is DRAM. On the other hand, non-volatile memory retains its contents when the device is turned off. Non-volatile memory is usually accessed in read mode, a special setup may be required to write to it. Examples of non-volatile memory include ROM, flash, and battery backed SDRAM.

3 DEVELOPMENT TOOLS

3.1 Compiling and building

MIDP applications are Java applications, which means that Java 2 Platform, Standard Edition (J2SE) and its tools are needed to build them. It is also recommended that possible Integrated Development Environments (IDEs) are installed before installing the J2ME Wireless Toolkit (J2MEWT) [8].

3.2 J2ME Wireless Toolkit

J2MEWT is a set of tools that provides application developers with the emulation environment, documentation and examples needed to develop Java technology applications targeted for CLDC/MIDP compliant mobile phones and PDAs. J2MEWT can be tightly integrated with third party development environments such as Sun ONE Studio (formerly Forte for Java), providing a complete environment with which developers can write, test and debug applications from start to finish.

3.3 Integrated developing environments

J2MEWT provides tools and libraries for building and emulating MIDP applications. It does not contain any graphical user interface for application development. However, there are several IDEs that can be integrated with J2MEWT and can be used to develop MIDP applications. Some of the most common IDEs are compared in Table 1.

Product:	Sun ONE Studio 4.0 [9]	JBuilder 7 with MobileSet 3 [10]	Wireless Studio 7 [11]
Developer:	Sun Microsystems	Borland	CodeWarrior
Price:	Free	$399 (JBuilder7SE)	$599
System:	Windows, Solaris, Linux	Windows, Solaris, Linux, Mac	Windows
Free trial available:	Yes	Yes	Yes

Table 1: Some development environments.

3.4 Emulators

MIDP applications can be run on numerous emulators. J2MEWT and development tools support device emulators by device manufacturers and therefore testing applications on different platforms becomes somewhat easier. However, as the performance and features of the emulators may be different from the real device, applications should also be tested on the actual devices.

4 PROGRAMMING INTERFACES

4.1 MIDP API

The MIDP API consists of five different parts. The base is the core API that provides system level input and output, as well as the Java2 language and utility classes. These are mainly imported from J2SE, although they are somewhat slimmed down from the originals. Next, there is the MIDlet package that defines the application life cycle API and the possible interactions between the application and the environment in which the application runs. Persistent data storage is defined in the Record Management System API. The user interface API provides a minimal set of features for implementation of user interfaces for MIDP applications. There are

also functions to set timers and get notification when they expire in the user interface API. MIDP includes networking support based on the generic connection framework from the CLDC. In addition to the input/output classes specified in the CLDC, MIDP includes the interface for HTTP protocol access over the network.

4.2 Manufacturers' own APIs

The MIDP API was designed to be hardware independent, so there is no way to access the device's hardware directly. However, applications like action games need often low-level access to the device hardware such as the keyboard, display, lights, vibration and speaker. Applications could also want to use the device's other capabilities like GSM phone, SMS (or MMS) messaging, phone book, camera and so on. For these purposes, many device manufacturers have introduced their own APIs [4,5,6]. The upside is that these APIs give developers a lot of possibilities to take advantage of the devices' capabilities. On the other hand, these APIs usually work only on that particular device or maybe on a couple of other devices in the same product family. Using these APIs usually ruins the platform independency of MIDP.

5 RESTRICTIONS

Because MIDlets can be run on very different kinds of devices, there are quite a lot of restrictions to keep in mind when developing platform independent MIDlets. Limited memory size is one of the most significant restrictions. Some devices restrict the size of the MIDlet suite, so larger applications can not be downloaded or executed. The first Java capable phones were taken in public use in Japan in January 2001, where DoCoMo's i-mode phones had 10 kB of memory. In April 2002, the memory was upgraded to 30 kB in the new generation of i-mode Java phones, but the size of the memory is still very small. The first Java phones for the GSM/GPRS market were introduced in 2002 and have typically at least 30 kB of memory for Java applications. It is challenging to get all the data and code of graphical applications like games to fit into this size. Therefore, some game developers have introduced overlaying techniques similar to MS-DOS programming ten years ago. When you play one level of a game, the game automatically downloads – with the player's permission – the next level and the play can continue. Each level can be coded to fit into 30 kB, for example.

Processor speed may restrict some calculations on low end devices, but most recently released devices have ample processing power, so speed is not generally speaking a big problem. Because of the lack of floating point units on mobile devices, there is no implementation of floating point math in MIDP. All calculations must be performed using integer or fixed point math.

Different types of screens are probably the biggest problem when developing platform independent games or other graphical applications. The first issue is the screen size. The MIDP specification states only that the screen size should be at least 96x54 pixels. The developer must decide if he wants to take advantage of bigger screens or whether to design the application to fit a small screen. Another issue is color depth. It is very hard to produce good looking color graphics that also look good on black and white screens.

Different kinds of keyboards can also be a big problem when developing platform independent mobile games. The layout of the keyboard can be different on different devices, and devices with touch screens may not have a keyboard at all. An additional problem is the fact that some devices may not recognize key repeating.

Network sockets are not supported, as MIDP only uses the HTTP protocol. Also the bandwidth of the network interface can be very limited (for example, when using GSM data as the data bearer).

6 GAME APPLICATIONS

As an example of MIDP application development, we will now take a closer look on game applications. Playing games is very popular among the young, and mobile gaming has been estimated to rake in between 6 and 18 billion dollars by the year 2006, according to market analysis firms such as OVUM, ARC Group and Datamonitor.

From the programmer's point of view, games are interesting MIDP applications since they are usually technically demanding and use the hardware's capabilities to the maximum. Modern mobile devices are capable to run many kinds of games from simple puzzle games to complex action games, and even 3D games. Multiplayer games are also possible since MIDP specifies that devices must support at least the HTTP protocol.

6.1 Sprite based graphics

Action games based on 2D sprite graphics are one of the oldest and most widely used game types. The MIDP API doesn't support sprites or transparent images, so it is quite difficult to implement sprite graphics strictly using the MIDP API. However, many device manufacturers' own APIs support sprites and/or transparent images, and therefore it is possible to write a sprite API that wraps device specific APIs and use it to build different versions for different devices. Another problem is that sprite graphics need quite a lot of memory to save bitmaps.

6.2 Vector based graphics

Vector graphics can be implemented using only the MIDP API. There is a line drawing function in the MIDP API which can be used to render line vectors. There is no function to draw triangles or polygons, but these functions can be easily implemented using the line function. Vector graphics can save a lot of memory in some cases, compared to bitmap graphics. Vector graphics can also be used to create a more dynamic world than what is possible with bitmaps. A disadvantage is that somewhat more complex math is required to represent the graphics, which requires processor capacity. The lack of a floating point unit makes the use of fixed point math necessary.

6.3 3D graphics

Three dimensional vector graphics are possible on devices with bigger color screens and enough processing capacity. On small, low resolution, black and white screens 3D vector graphics might be too messy and slow. Some semi-3D techniques can also be used. These techniques include ray casting based dungeons (Wolfenstein 3D, Doom) and the rendering of a flat "3D" earth used in many driving games. Ray casting might also be used to create semi-3D voxel landscapes.

6.4 Sounds

There is no sound interface in MIDP API. The manufacturers' own APIs include different sound systems using device specific sound capabilities. These devices usually support at least monophonic melodies and beeps. Many new devices also support polyphonic melodies, MIDI, and even digitized sounds.

6.5 Game application developed at MediaLab

As an example of MIDP application development, we created a mobile version of the classic "Moon lander" (or "Lunar lander") game. "Moon lander" is a computer game classic where the player's mission is to control a small spacecraft and land it successfully on a landing platform. A screenshot of the game can be seen in Figure 3.

Figure 3: Screenshot of an MIDP game.

The application was developed using J2MEWT and device emulators provided by Sun Microsystems, Nokia [4], Ericsson [5], Motorola [6], and Siemens [7]. The game has been tested on numerous emulators, and on the Nokia 7650 camera phone and the Nokia 9210(i) communicator. The goal was to develop a simple game application that would run well on many different devices. "Moon lander" doesn't require a color display, but it is highly recommended, as the graphics quality may otherwise be unsatisfactory. The size of the whole application is 41 kB, but could easily be reduced by leaving out the splash screen and the background picture. As platform independence was one of the design goals, only MIDP APIs were used. No device specific APIs or features were used.

7 OTHER APPLICATIONS

Games are only one category of applications that are possible on the MIDP platform. The networking capability opens doors to developing, for example, mobile client programs for many common network applications such as browsers and chat clients. However, games and networking applications are just a small fraction of the possibilities. Many of the existing applications can be ported to the MIDP platform and whole new applications can be created for mobile devices.

8 SUMMARY

In this paper we have shown that Java 2 Micro Edition and Mobile Information Device Profile together give developers lots of possibilities to create applications and services for new mobile phones and other portable devices. There are many development environments, emulators, tools, programming examples and tutorials available on the Internet, so starting to develop software for mobile devices is not too difficult. Based on our experiences with MIDP application development, most problems are caused by differences between device properties, such as screen size and color depth. This paper only deals with version 1.0 of Mobile Information Device Profile. However, version 2.0 [3] has already been specified, and it will bring many improvements like sound and a game API. Also, support for floating point arithmetic and network sockets will be added. These improvements provide lots of new possibilities, but before MIDP version 2.0 appears in mobile devices, we have to do our best with the current version.

REFERENCES

[1] The Source for Java Technology, http://java.sun.com/ [Accessed September 4, 2002]

[2] J2ME Mobile Information Device Profile (MIDP), http://wireless.java.sun.com/midp/ [Accessed September 4, 2002]

[3] JSR 118: Mobile Information Device Profile 2.0, http://www.jcp.org/jsr/detail/118.jsp [Accessed September 4, 2002]

[4] Forum Nokia, http://www.forum.nokia.com [Accessed September 4, 2002]

[5] Ericsson Application Development, http://www.ericsson.com/mobilityworld/sub/development/index.html [Accessed September 4, 2002]

[6] Motorola Developer Resources, http://developers.motorola.com/developers/ [Accessed September 4, 2002]

[7] Siemens Mobile Partners, http://www2.siemens.fi/siemens.jsp [Accessed September 4, 2002]

[8] J2ME Wireless Toolkit, http://java.sun.com/products/j2mewtoolkit/ [Accessed September 4, 2002]

[9] Sun ONE Studio, http://wwws.sun.com/software/sundev/jde/index.html [Accessed September 4, 2002]

[10] Borland JBuilder MobileSet, http://www.borland.com/jbuilder/mobileset/ [Accessed September 4, 2002]

[11] CodeWarrior Wireless Studio, http://www.metrowerks.com/MW/Develop/Wireless/Wireless_Studio/Default.htm [Accessed September 4, 2002]

ADDITIONAL RESOURCES

Wireless Developer Homepage, http://wireless.java.sun.com/ [Accessed September 4, 2002]

JSR 37: Mobile Information Device Profile 1.0, http://www.jcp.org/jsr/detail/37.jsp [Accessed September 4, 2002]

Micro Java Network - Java Devices, http://www.microjava.com/devices [Accessed September 4, 2002]

DEFINITIONS, ACRONYMS AND ABBREVIATIONS

API	Application program interface
CLDC	Connected, Limited Device Configuration
FPU	Floating Point Unit
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
HTTP	Hyper Text Transfer Protocol
I/O	Input/Output
IDE	Integrated Development Environment
IR	Infrared
J2ME	Java 2 Micro Edition
J2MEWT	Java 2 Micro Edition Wireless Toolkit
J2SE	Java 2 Standard Edition
JAM	Java Application Manager
MID	Mobile Information Device
MIDlet	MIDP application
MIDP	Mobile Information Device Profile
MMS	Multimedia Message Service
OEM	Original Equipment Manufacturer
OTA	Over-The-Air
PDA	Personal Digital Assistant
SMS	Short Message Service
UI	User Interface
VM	Virtual Machine
WAP	Wireless Application Protocol

