# 11

# Embedded User Interface for Mobile Applications to Satisfy Design for All Principles

Evangelos Bekiaris, Maria Gemou and Kostantinos Kalogirou
*C.E.R.T.H. /Hellenic Institute of Transport*
*Hellas*

## 1. Introduction

This chapter describes the existing Java U.I. libraries available in the market. Some of them are under the GPL/ LGPL license and some other are commercial and a license is required to be purchased. There are also many others that are provided by the mobile device vendors.
This document separates these types of libraries, analyses each of them by elaborating their features and specifications. CERTH/ HIT tested most of the libraries included in this document. These evaluation tests have been taken place both on emulation environment and on mobile devices. Thus, the Sun Wireless Toolkit version 2.5.2 and Java ME SDK Device Manager emulator tools and also the following three mobile devices were used:

- Sony Ericsson C905 (Java platform Operating System)
- Nokia N82 and Nokia 95 8G (Symbian Operating System)
- HTC TyN II (Windows Mobile 6.x OS with IBM WEME(J9) version 6.1.1 with the support of Personal Profile 1.1)

The examples were run on the above devices are either samples provided by the company that support the particular library or they have created by CERTH/ HIT.

## 2. GPL/LGPL license UI libraries

### 2.1 AWT

The AWT stands for The Abstract Window Toolkit (AWT). It is the Java's original platform-independent windowing, graphics, and user-interface widget toolkit.
The AWT is now part of the Java Foundation Classes (JFC) — the standard API for providing a graphical user interface (GUI) for a Java program. AWT is supported by number of Java ME profiles such as the Connected Device Configuration with Personal Basis Profile or Personal Profile, which is the minimum requirement. The core package is the java.awt. Some of its features are:

- The rich set of user interface components;
- The robust event-handling model;
- Graphics and imaging tools, including shape, color, and font classes;
- Layout managers, for flexible window layouts that don't depend on a particular window size or screen resolution;
- Data transfer classes, for cut-and-paste through the native platform clipboard.

Source: User Interfaces, Book edited by: Rita Mátrai,
ISBN 978-953-307-084-1, pp. 270, May 2010, INTECH, Croatia, downloaded from SCIYO.COM

The following GUI components are supported by AWT:

- Button (java.awt.Button)
- Canvas (java.awt.Canvas)
- Checkbox (java.awt.Checkbox)
- Choice - Radio button (java.awt.Choice)
- Label (java.awt.Label)
- List (java.awt.List)
- Scrollbar (java.awt.Scrollbar)
- Text area (java.awt.TextArea)
- Text field (java.awt.TextField)
- Panel (java.awt.Panel)
- Frame (java.awt.Frame)
- Dialog (java.awt.Dialog)
- Popup menus (java.awt.PopupMenu)

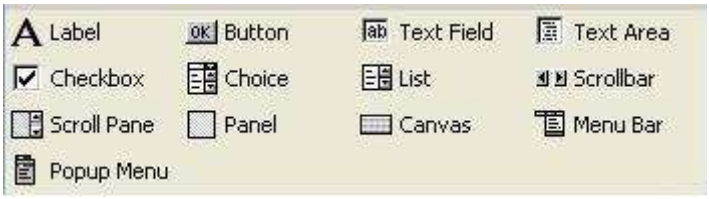


Fig. 1. AWT GUI components

The whole API of AWT can be found at the http:/ / java.sun.com/ javase/ 6/ docs/ api. AWT widgets provided a thin level of abstraction over the underlying native user interface. For example, creating an AWT check box would cause AWT directly to call the underlying native subroutine that created a check box. However, a check box on Microsoft Windows is not exactly the same as a check box on Mac OS or on the various types of UNIX and Linux distributions. Some application developers prefer this model because it provides a high degree of fidelity to the underlying native windowing toolkit and seamless integration with native applications. In other words, a GUI program written using AWT looks like a native Microsoft Windows application when run on Windows, but the same program looks like a native Apple Macintosh application when run on a Mac. However, some application developers dislike this model because they prefer their applications to look exactly the same on every platform.(Wikipedia , 2009)

### 2.2 LWUIT

"LWUIT is a UI library that is bundled together with applications and helps content developers in creating compelling and consistent Java ME applications. LWUIT supports visual components and other UI goodies such as theming, transitions, animation and more.
The Lightweight UI Toolkit is a lightweight widget library inspired by Swing but designed for constrained devices such as mobile phones and set-top boxes. Lightweight UI Toolkit supports pluggable theme-ability, a component and container hierarchy, and abstraction of the underlying GUI toolkit. The term lightweight indicates that the widgets in the library draw their state in Java source without native peer rendering. Internal interfaces and abstract classes provide abstraction of interfaces and APIs in the underlying profile. This

allows portability and a migration path for both current and future devices and profiles. For example, Graphics would be an abstraction of the graphics object in the underlying profile. The Lightweight UI Toolkit library tries to avoid the "lowest common denominator" mentality by implementing some features missing in the low-end platforms and taking better advantage of high-end platforms. The following figure shows the widget class hierarchy"( Sun Microsystems, 2008).

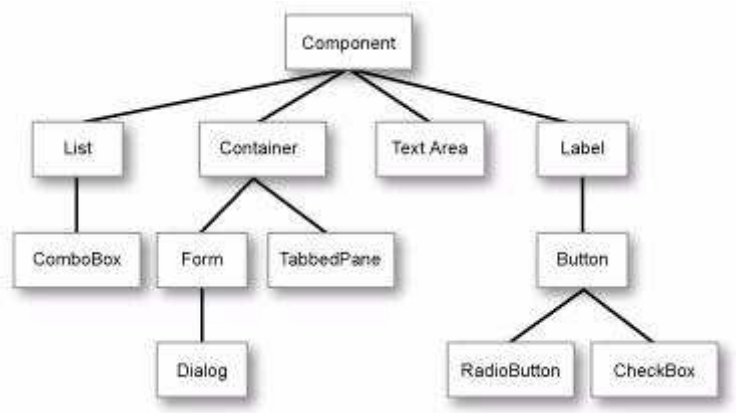


Fig. 2. LWUIT Widget library class hierarchy

Moreover, the Lightweight UI Toolkit library completely handles and encapsulates UI threading in order to increase compatibility. It has a single main thread referred to as the "EDT"(inspired by the Event Dispatch Thread in Swing and AWT). All events and (paint) calls are dispatched using this thread. This guarantees that event and paint calls are serialized and do not risk causing a threading issue. The following figure shows a screen dump of a LWUIT sample.

Fig. 3. LWUI sample

### 2.3 Light weight Visual Component Library (LwVCL)

The LwVCL library was created in order to support Graphical User Interfaces in different platforms. The LwVCL consists of the following:

- JSE LwVCL: Use it for desktop systems - Windows, UNIX, Mac OS, everywhere java can be installed. This is the base implementation of the library that is the "master" branch for all others. All new features come here first and after that are applied to other versions.
- JME Personal Profile (Personal Java) LwVCL: Use it for PDAs like to have the same capabilities as you have on the desktop systems. This version doesn't differ from JSE LwVCL
- .NET LwVCL: This version has the same JSE LwVCL capabilities.
- SWT LwVCL
- JME MIDP LwVCL Use it for the resources limited devices. This version is under development now.

Some of library's powerful characteristics are the following (lwvcl.com, 2007):

- Layered architecture.
  This library has a layered architecture where UI components set has minimal dependencies on a concrete platform. The LwVCL components are abstract as much as it is possible to be easily adapted to any other platforms and languages. See the library basic ideas page.
- Small size.
  The library packages are very small. The core Java package is about 160 Kb (the .NET DLL is about 200 Kb).
- Provides about 30 various GUI components.
  In spite of the small size of the library, LwVCL provides a huge number of UI components. In addition to simple components, you will get a grid, tree, tree grid, and other complex, flexible components.
- Dynamic and thrifty to system resources usage (CPU/ Memory/ Disk Space).
  The library is very fast and takes care of use of system resources.
- MVC (Model-View-Controller) compliance.
  The library components are designed according to the MVC concept that allows separating data models, views, and business logic.
- Flexible and customizable.

The library is customizable. It is very easy to extend the library with new components, or change its behavior according to your requirements.

### 2.4 Synclast

It is an extensible toolkit for creating colourful custom user interfaces on Java-enabled handheld devices. It is compatible with any MIDP 1.0 device, and is fully open source. It provides the same GUI components as the LCDUI library plus the following:

- Box
- Button
- Checkbox
- Colored Widget
- Flow
- Input

- Label
- Menu
- Popup
- Radio Button
- Radio Group
- Style Sheet
- Styled
- Synclast Canvas
- Synclast Full Canvas
- Synclast Image
- SynclastManager
- Synclast Task
- Table
- Tap Input Adapter
- Widget

Synclast was used mainly for creating games. The following figures shows the demo Synclast application running on Sun WTK.

Fig. 4. Synclast sample applications

### 2.5 Thinlet

Thinlet is a GUI toolkit based on XML structure. It supports both JME profiles, Personal and MID Profiles. Porsche Engineering developed a version of Thinlet based on MIDP.

"It is a single Java class, parses the hierarchy and properties of the GUI, handles user interaction, and calls business logic. Separates the graphic presentation (described in an XML file) and the application methods (written as Java code).Its compressed size is 39KB, and it is LGPL licensed. Thinlet runs with Java 1.1 (IE's default JVM) to 1.4, Personal Java, and Personal (Basis) Profile. Swing isn't required."(Robert Bajzat, 2002-2006) PDA application and mobile application GUI were built using Thinlet for the purposes of IM@GINE-IT European project. The following figures show some applications that uses Thinlet library.

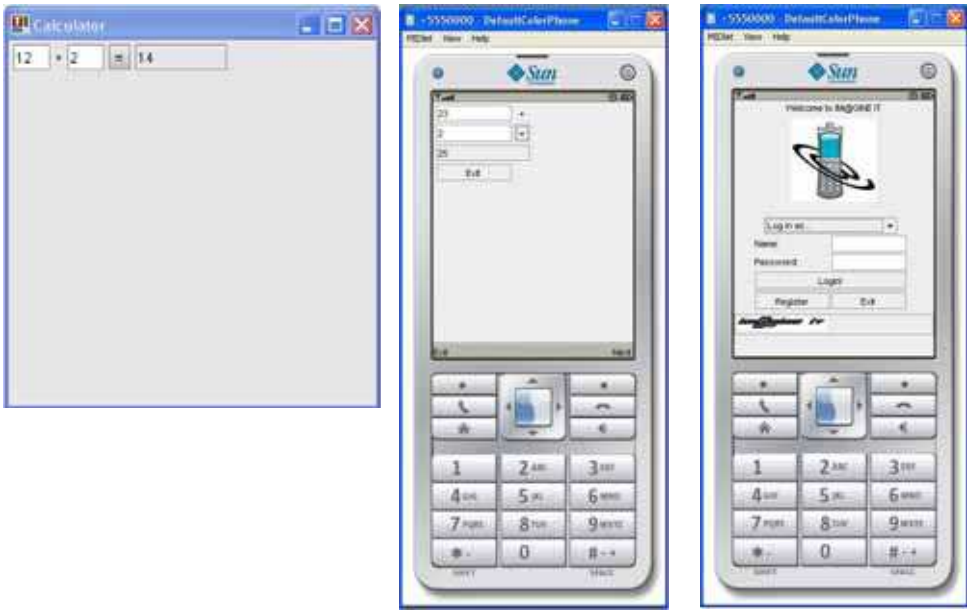

Fig. 5. Applications using Thinlet library


Fig. 6. Apime samples

### 2.6 Apime

"Apime is a framework to offer more functionality to mobile with Java enabled (JME). The core is the user interface, with basics components to make applications, and with possibility to

create news adapting to what each developer requires. Also it includes classes for file manage and customization (skins, internationalization, keyboards for different languages and mobiles. It is whole compatible with MIDP 1.0, although exists a version for MIDP 2.0 and other for Nokia, to use the full screen feature than MIDP 1.0 no offers. For all this it allow make different kind of applications easier and faster." (JAVA4EVER, 2005)

### 2.7 F.I.R.E (Flexible Interface Rendering Engine)

The basic set of Fire components offer all the functionality of the Java ME GUI components provided in the MIDP 2.0 profile (Forms, Items etc.) plus a much more appealing user interface, themes, animations, popup menus, and better component layout. The library can be downloaded from the http:/ / sourceforge.net/ projects/ fire-j2me/

Fig. 7. F.I.R.E. examples

### 2.8 Kuix

Kuix is an extensible and fully customizable JavaME UI framework that enables high end application development. The Kuix library provides wide device compatibility. From the beginning, maximizing compatibility level has lead the development of Kuix and it results today in a wide range of supported devices. Kuix is compliant with CLDC 1.0 and MIDP 2.0. Besides, it supports fast and easy application development. Forms and widgets components are organized through an XML approach that combined with CSS file, allow the programmers to build applications even faster. Kuix is an open source project licensed under GPL. As a strong copy left license, it requires that all derived works to be available under the same license. For professional developers that do not want to release their applications under GPL, it is invited to purchase a commercial license. The demo-sample was successfully downloaded and run on mobile devices. It is very interesting that is based on XML and CSS file approaches. However, this licensing may not meet the needs of professional developers; so for that reason Kalmeo release a commercial License to allow the non-disclosure of their source code. (http:/ / www.kalmeo.com/ products/ kuix). The following figures show the examples on Sun's WTK simulator.

Fig. 8. Kuix samples

### 2.9 MWT (Micro Window Toolkit)

It is inspired by its UI big brothers as AWT, Swing and SWT, MWT comes into the scene providing an UI framework designed and optimized for small devices. The MIDP high-level UI API was designed for applications portability, employing a high level of abstraction, however it provides very little control over look and feel, and sometimes, this is very important.

In the other hand, the low-level UI API provides a good control over graphics and input events, but the API lacks of UI components. The Java UI's (AWT) would be a solution, but it was not included because it was designed and optimized for desktop computers. MWT comes into the scene providing an UI framework designed and optimized for small devices."(J2ME-MWT Team, 2005-2007)

MWT is not one of these frameworks that takes control over your application. It was inspired by his UI big brothers as AWT, Swing and SWT and it was designed and optimized for small devices. MWT only requires the MIDP1 and CLDC1 APIs, so it's completely portable. The sample applications from MWT were tested on mobile devices and it is noticed that the user interface capabilities were limited.

### 2.10 OpenBaseMovil

In addition to database and scripting engine, OpenBaseMovil contains a declarative view definition language. With an XML file you can generate all of your views, and they are script and data aware: you can browse a set of results with less than ten lines of code. In case an application is released under the GNU GPL license, the OpenBaseMovil library can be

used freely with no limitations other than imposed by the GNU GPL license. Otherwise, a license must be purchased (http:/ / www.openbasemovil.org/ licensing/ ).

### 2.11 Swing ME

A Java ME implementation of Swing GUI, with Layouts, Borders, Renderers and lots of components including inline TextField, Buttons, Window, TabbedPane and many others. All visual and behavioural aspects can be fully customised of ANY component."(Yura.net, 2008).

The following figures show some features of the sample applications from yura.net.

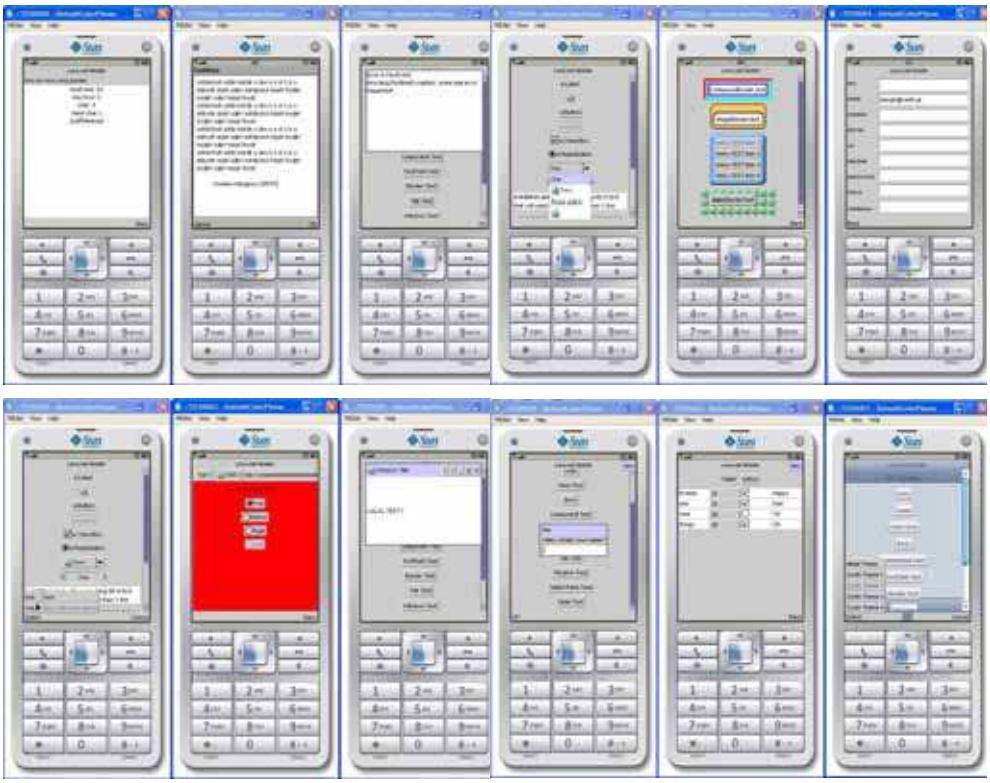

Fig. 9. SwingME examples

Some of the most interesting graphical features provided by SwingME are the border and Layouts (Border test example), the scroll bar component, the ability of using tab panels (Scroll Pane examples) and finally there are available themes for Menus.

## 3. Commercial UI libraries

### 3.1 TinyLine (Tinyline, 2002-2009)

TinyLine SVG implements an SVG Tiny 1.1+ engine for Android and Java platform (JME CLDC/ MIDP, CDC/ PP, JSE). TinyLine SVG allows incorporating SVG Tiny 1.1+ graphics

into Android and Java applications. The Tiny line SDK provides two products; TinyLine 2D (current version 2.1) and TinyLine SVG (current version 2.1). Each of them applies to different device platforms.

The TinyLine 2D implements a mobile 2D graphics engine for Java platform (JME CLDC/ MIDP, CDC/ PP, JOSÉ). Developers are easily able to incorporate high quality, scalable and platform-independent graphics into their Java applications. Some of its main features are:

- Small footprint (around 40K in jar file)
- Fast fixed-point numbers mathematics
- Paths, basic shapes and texts drawings
- Hit tests for paths and texts
- Solid colour, bitmap, pattern, gradient (radial and linear) paints
- Fill, stroke and dash
- Affine transformations
- Outline fonts
- Left-to-right, right-to-left and vertical text layouts
- Ant aliasing
- Opacity


Fig. 10. TinyLine examples

On the other hand, the TinyLine SVG implements an SVG Tiny 1.1+ engine for Android and Java platform (JME CLDC/ MIDP, CDC/ PP, and JSE). TinyLine SVG allows incorporating SVG Tiny 1.1+ graphics into Android and Java applications. It provides the following features:

- SVG Tiny 1.1+ engine
- Supports SVG fonts, raster image and text elements, paths
- Supports SMILE animations and events

- Allows both textual and gzipped SAG streams
- Compact code (around 100K in jar file)
- Easy to use API

The TinyLine library used to be free for JME until version 1.9. The screen dumps come from the sample of that version. From version 2.0, a license is needed to be purchased. However, it is free for Android platform. The examples from version 1.9 were tested successfully on real devices as well.

### 3.2 TWUIK

TWUIK Rich Media Engine is a technology that combines graphics, animation, rich-media user experience and interactivity for seamless deployment across an ever-wider range of supported JME devices. It supports the following platforms:

- JavaME CLDC 1.0/ 1.1 MIDP 2.0
- BREW 3.1
- Windows Mobile 5 and 6
- Symbian UIQ & Series 60
- DoCoMo Java 4.x & 5.x

"TWUIK™ Rich Media Engine (RME) is an UI technology that brings dazzling graphics, vibrant animation, engaging rich-media user experience and advanced interactivity to mobile application development for seamless deployment across an ever-wider range of supported JME devices. TWUIK™ enhances navigation, graphical display, and device functionality - all while reducing development cost and speeding time-to-market of new applications. TWUIK™ powered application makes your content and services available to the widest range of handsets without having to specifically re-develop for each specific handset, thereby reducing the cost of the development.

TWUIK™ is developed for JME devices and purposely optimized for the constrained environment of mobile devices. TWUIK's unique, flexible, modular architecture allows it to be easily integrated with low-level hardware, operating system, and software functionality.

TWUIK's cross-platform capabilities bridge the gap between different makes and models of handsets, making it possible for wireless operators and handset OEMs to enrich service offerings, maximizes expressiveness, and creates a customized, branded user experience that is uniform across all devices. This in turn simplifies the user experience, enables easy discovery of content & services, encourages consumption, promotes brand identity, and creates service/ device differentiation. By providing rich and visually appealing UI and more compelling mobile consumer experience, TWUIK™ dramatically boosts the consumption and stickiness of mobile content and applications." (Tricast Solutions Ltd., 2005-2007)

### 3.3 Paxmodept JavaME framework

"In Java ME the native MIDP GUI elements are ugly and inflexible. In order to create decent looking user interfaces for MIDP applications, developers must either write their own custom low level GUI components or use those from an existing library. The process of writing these components from scratch is a time consuming and expensive process and needs to take device fragmentation and variable screen sizes into account from the very beginning.

Fig. 11. TWUIK samples

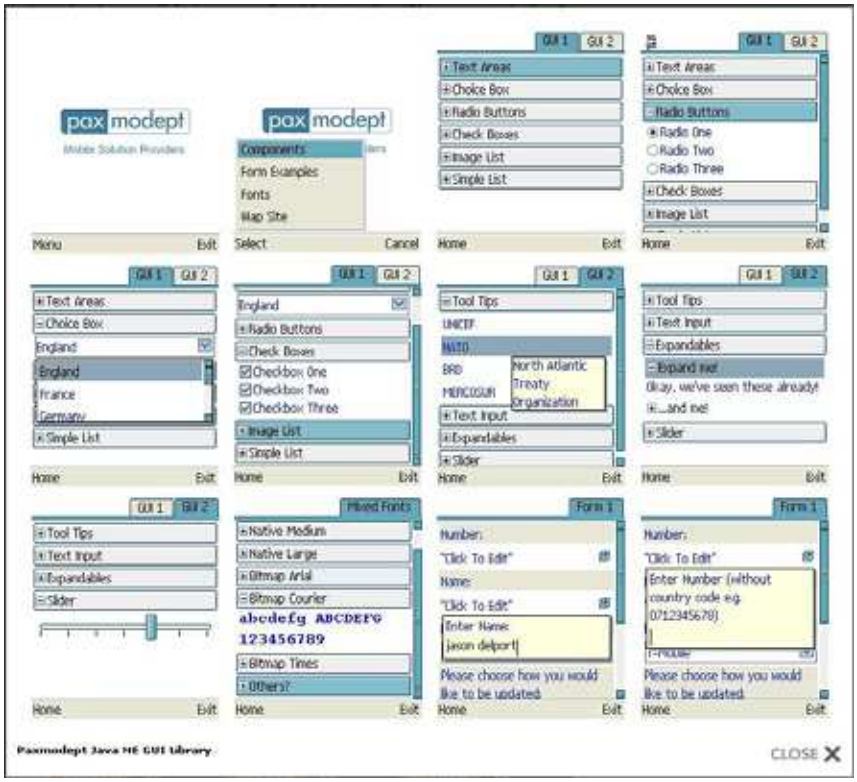


Fig. 12. Paxmodept GUI components

"The Paxmodept GUI library is intended for use on most Java ME capable devices and it has taken into account various device idiosyncrasies like screen size, keyboard mappings and input modes. The library has been designed to function much like existing Swing components so Java developers will feel very comfortable using the API programmatically" (Paxmodept, 2009). The library can be easily used in any IDE, such as Netbeans and Eclipse. It is provided a flexible layout manager which has support for a variety of different layout styles (Flow, Border and Grid) but also, allows developers to combine different layout styles on the same screen. In conjunction with this powerful layout manager there is a wide selection of GUI components which can act as either widgets or containers and be added to each other at, based on standard component tree architecture. Furthermore, the most important feature of the Paxmodept library is its speed and performance. It has been designed and optimized to work across a huge range of devices ensuring that the performance of every GUI component is lightning fast even on the most basic MIDP 2.0 devices. The following figure shows the Paxmodept GUI components and applications that uses these.


Fig. 13. Paxmodept sample (1)


Fig. 14. Paxmodept sample (2)

## 4. Device manufacturer's UI libraries

### 4.1 Advanced GUI (JSR 209)

AGUI is an optional package that sits on top of CDC at Foundation and Personal Basis Profile (PBP). PBP is supported on many CDC-based devices/ platforms. The current JME platforms such as Personal Profile and Personal Basis Profile are generally limited to the graphics and UI facilities found in only the core of AWT, as present in JDK 1.1.8. AGUI migrates the core APIs for advanced graphics and user interface facilities from the JSE platform to the JME platform. These facilities will include: Swing, Java 2D Graphics and Imaging, Image I/ O, and Input Method Framework.

Currently, there are not many devices supporting AGUI. However, JavaFX Mobile will fully support AGUI. The AGUI examples provided by Java ME SDK Device Manager emulator are displayed in the following figures.

Fig. 15. AGUI samples

### 4.2 LCDUI

The MIDP UI is composed of two core APIs, the high-level and the low-level. The high-level API is designed for business applications whose client parts run on MIDlets. For these applications, portability across devices is important. To achieve this portability, the high-level API employs a high level of abstraction and provides very little control over look and feel. The actual drawing to the MIDlet's display is performed by the implementation. Applications do not define the visual appearance (e.g., shape, color, font, etc.) of the components. Navigation, scrolling, and other primitive interaction are encapsulated by the implementation, and the application is not aware of these interactions. Applications cannot access concrete input devices like specific individual keys.

The low-level API, on the other hand, provides very little abstraction. This API is designed for applications that need precise placement and control of graphic elements, as well as access to low-level input events. Some applications also need to access special, device-specific features. A typical example of such an application would be a game.

On the other hand, using the low-level API, an application can have full control of what is drawn on the display, can listen for primitive events like key presses and releases access concrete keys and other input devices. The LCDUI library can be used by devices which are compatible with the CLDC configuration.

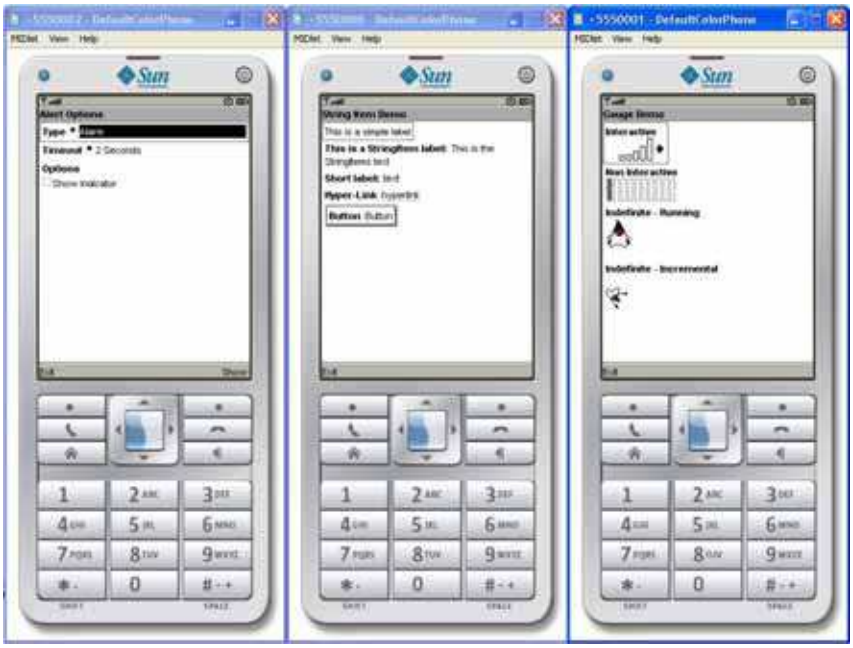

Fig. 16. JME examaples using LCDUI library

### 4.3 SWT

SWT stands for "Standard Widget Toolkit". SWT is an open source widget toolkit for Java designed to provide efficient, portable access to the user-interface facilities of the operating systems on which it is implemented. SWT is under Eclipse responsibility. Some screen dumps of SWT examples are appeared below.

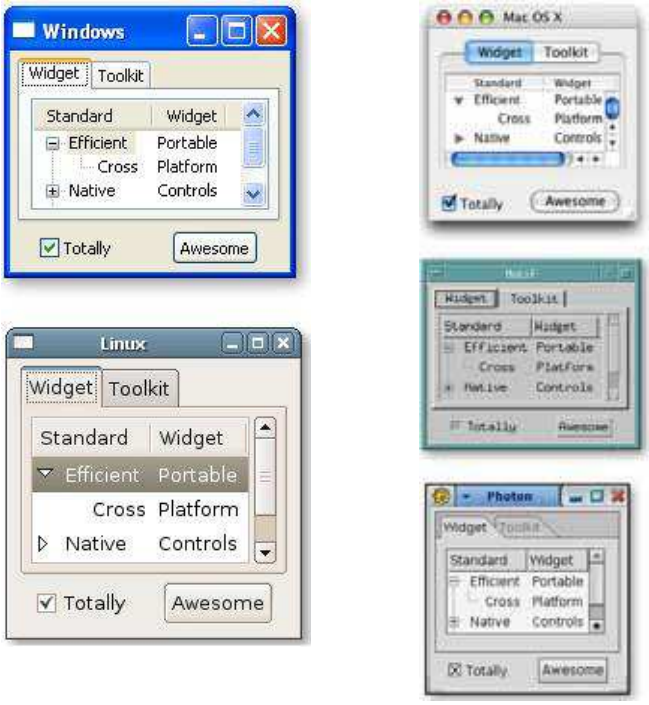


Fig. 17. SWT in deifferent platforms

Some of the SWT examples were successfully evaluated on Mobile Windows 6.0 devices, in which WEME (J9) JVM version 6.1.1 has been installed. Moreover, there is also a subset version of SWT, the eSWT library, which it is optimized for devices.

### 4.4 SVG

The SVG (JSR 226) defines an API for rendering scalable 2D vector graphics, including image files in W3C Scalable Vector Graphics (SVG) format. The API is targeted for JME platform, with primary emphasis on MIDP. The main use cases for this API are map visualization, scalable icons, and other advanced graphics applications. The SVG API includes:

- Ability to load and render external 2D vector images, stored in the W3C SVG Tiny format.
- Rendering of 2D images that are scalable to different display resolutions and aspect ratios.

The JSR 287 is package for rendering enhanced 2D vector graphics and rich media content based on select features from SVG Mobile 1.2 for Java ME platform, with primary emphasis on MIDP. This API will be designed as an extension to JSR 226, and therefore must remain to be fully backwards compatible with JSR 226 applications and Scalable Vector Graphics (SVG) rendering model. The scope of the API should include the following:

- Extend Features in JSR 226
- Support for select SVG Mobile 1.2 features


Fig. 18. SVG samples

### 4.5 Open GL ES

"The OpenGL ARB and the Khronos Group have long collaborated to ensure consistency in the OpenGL, OpenGL ES, OpenML, COLLADA and OpenGL SC standards. As a result of this transition all OpenGL- related activities will now occur under the single Khronos participation framework to enable fully- integrated cooperation between these related standards activities so that OpenGL may form the foundation for a coherent set of standards to bring advanced 3D graphics to all hardware platforms and operating systems - from supercomputers to jet fighters to cell phones. OpenGL® for Embedded System (ES) is a royalty-free, cross-platform API for full-function 2D and 3D graphics on embedded systems - including consoles, phones, appliances and vehicles. It consists of well-defined subsets of desktop OpenGL, creating a flexible and powerful low-level interface between software and graphics acceleration. OpenGL ES includes profiles for floating-point and fixed-point systems and the EGL™ specification for portably binding to native windowing systems. OpenGL ES 1.X is for fixed function hardware and offers acceleration, image quality and performance. OpenGL ES 2.X enables full programmable 3D graphics. OpenGL SC is tuned for the safety critical market." ( Khronos Group, 1997 – 2009)]

OpenGL ES (OpenGL for Embedded Systems) is a subset of the OpenGL 3D graphics API designed for embedded devices such as mobile phones, PDAs, and video game consoles. OpenGL ES is managed by the not-for-profit technology consortium, the Khronos Group, Inc. (Wikipedia, 2009)]


Fig. 19. Open GL example

### 4.6 JSR 184 Mobile 3D Graphics

The Mobile 3D Graphics API (M3G), is a specification defining an API for writing Java programs that produce 3D computer graphics. "It extends the capabilities of the Java Platform, Micro Edition, a version of the Java platform tailored for embedded devices such as mobile phones and PDAs. The object-oriented interface consists of 30 classes that can be used to draw complex animated three-dimensional scenes. M3G was developed under the Java Community Process as JSR 184. As of 2007, the current version of M3G is 1.1, but version 2.0 is in development as JSR 297. M3G was designed to meet the specific needs of mobile devices, which are constricted in terms of memory, and processing power, and which often lack an FPU and graphics hardware such as a GPU. The API's architecture allows it to be implemented completely inside software or to take advantage of the hardware present on the device.

M3G should not be mistaken for Java 3D, which extends the capabilities of the Java Platform, Standard Edition. Java 3D is designed for PCs that have more memory and greater processing power than mobile devices. M3G and Java 3D are two separate and incompatible APIs designed for different purposes. M3G provides two ways for developers to draw 3D graphics: immediate mode and retained mode.

In immediate mode, graphics commands are issued directly into the graphics pipeline and the rendering engine executes them immediately. When using this method, the developer must write code that specifically tells the rendering engine what to draw for each animation frame. A camera, and set of lights is also associated with the scene, but is not necessarily part of it. In immediate mode it is possible to display single objects, as well as entire scenes (or worlds, with a camera, lights, and background as parts of the scene).
Retained mode always uses a scene graph that links all geometric objects in the 3D world in a tree structure, and also specifies the camera, lights, and background. Higher-level information about each object — such as its geometric structure, position, and appearance — is retained from frame to frame."(Wikipedia, 2009)

Fig. 20. 3D applications

### 4.7 JSR 135 Mobile Media API

The Mobile Media API (MMAPI) is an API specification for the Java ME platform CDC and CLDC devices such as mobile phones. These APIs allow applications to play and record sounds and video, and to capture still images, depending on how it's implemented. It was developed under the Java Community Process as JSR 135. "The Multimedia Java API is based on the following types of classes from the javax.microedition.media package: Manager, Player, PlayerListener and various types of Control. Developers wishing to use JSR 135 would first make use of the static methods of the Manager class. Although there are other methods such as playTone, the main method used is createPlayer. This takes either a URI or an InputStream, or a MIME type. In most cases, URIs are used. The common URI protocols that are used, include: file, resource (which may extract a file from within the JAR of the MIDlet, but is implementation-dependent), http, rtsp, capture (used for recording audio or video).The MIME type is optional.

Fig. 21. Sun's Wireless Toolkit samples using JSR 135

### 4.8 JSR 234: Advanced Multimedia Supplements (AMMS)

The Advanced Multimedia Supplements (JSR-234 or AMMS) is an API specification for the Java ME platform. It is an extension to JSR 135 Mobile Media API providing new features, such as positional 3D audio processing, audio and video effects processing, better controls for digital camera, and better support for analog radio tuner including Radio Data System. It was developed under the Java Community Process as JSR 234.

JSR-234 defines six feature sets (Media Capabilities), and each of them defines minimum implementation requirements in order to try to avoid fragmentation and to define a common minimal base line for the implementations. Every JSR-234 implementation must support at least one Media Capability. These are music, 3D audio, camera, image encoding, image ports processing and tuner capabilities. It is noticed that many limitations such as taking snapshots were found while JSR 234 samples were evaluated on mobile devices.

### 4.9 BlackBerry UI library

"BlackBerry is a line of wireless handheld devices that was introduced in 1999 as a two-way pager. In 2002, the more commonly known as smart phone BlackBerry was released, which supports push e-mail, mobile telephone, text messaging, internet faxing, web browsing and other wireless information services as well as a multi-touch interface." ( Wikipedia, 2009)

BlackBerry includes the net.rim.device.api.ui.accessibility package to allow a BlackBerry device application that uses custom UI components to send information to an assistive technology application. When a custom UI component changes, an assistive technology application receives a notification about the change and can obtain more information about the change from the custom UI component. For example, if a BlackBerry device application uses a class called myTextField that extends the TextField class, when a BlackBerry device

Fig. 22. AMMS samples

user changes the text in a myTextField instance, an assistive technology application receives notification of the change and can retrieve data such as the text that the user selects or changes. The notification contains the information such as the name of the custom UI component and the type of event. For example, a change in the cursor position, or a change in the name of the custom UI component, a value of a custom UI component before the event, the value of a custom UI component after the event has taken place.

Moreover, UI component can support the usual and common states, such as focused, focusable, expanded, expandable, collapsed, selected, selectable, pushed, checked and more other.


Fig. 23. Samples on BlackBerry Java Development Environment

### 4.10 Java Speech API 2.0 (JSR 113)

The Java Speech API allows you to incorporate speech technology into user interfaces for your applets and applications based on Java technology. The Java Speech API specifies a cross-platform interface to support command and control recognizers, dictation systems and speech synthesizers. Although JSAPI defines an interface only, there are several implementations created by third parties, for example FreeTTS. This JSR extends the work of the Java Speech API 1.0 which was principally targeted at Java servers. Also, it targets embedded Java devices, and allows developers to incorporate speech technology into user interfaces for their Java programming language applets and applications.

The different classes and interfaces that form the Java Speech API are grouped into the following three packages:

- javax.speech: Contains classes and interfaces for a generic speech engine.
- javax.speech.synthesis: Contains classes and interfaces for speech synthesis.
- javax.speech.recognition: Contains classes and interfaces for speech recognition.

"The Central class is like a factory class that all Java Speech API applications use. It provides static methods to enable the access of speech synthesis and speech recognition engines. The Engine interface encapsulates the generic operations that a Java Speech API-compliant speech engine should provide for speech applications.

Speech applications can primarily use methods to perform actions such as retrieving the properties and state of the speech engine and allocating and deallocating resources for a speech engine. In addition, the Engine interface exposes mechanisms to pause and resume the audio stream generated or processed by the speech engine. The Engine interface is subclassed by the Synthesizer and Recognizer interfaces, which define additional speech synthesis and speech recognition functionality. The Synthesizer interface encapsulates the operations that a Java Speech API-compliant speech synthesis engine should provide for speech applications.

The Java Speech API is based on the event-handling model of AWT components. Events generated by the speech engine can be identified and handled as required. There are two ways to handle speech engine events: through the EngineListener interface or through the EngineAdapter class." (Wikipedia, 2009)

At Java One Conference 2008, there was a successful demonstration of JSR113 under a commercial version WEME (J9) JVM at HP PDA device. However, the demonstration was taken place for short sentences (e.g. "Open Agenda", "Call Kostas"). Currently, it is not aware if any mobile phone supports that API.

### 4.11 Java FX Mobile

JavaFX Mobile is the JavaFX application platform for mobile devices and a part of JavaFX platform. JavaFX Mobile applications can be developed in the same language, JavaFX Script, as JavaFX applications for browser or desktop, and using the same tools: JavaFX SDK and the JavaFX Production Suite. This concept makes it possible to share code-base and graphics assets for desktop and mobile applications. Through integration with Java ME, the JavaFX applications have access to capabilities of the underlying handset, such the file system, camera, GPS, bluetooth or accelerometer.

An independent application platform built on Java, JavaFX Mobile is capable of running on multiple mobile operating systems, including Android, Windows Mobile, and proprietary real-time operating systems.

JavaFX Mobile running on an Android was demonstrated at JavaOne 2008 and selected partnerships (incl. LG Electronics, Sony Ericsson) have been announced at the JavaFX Mobile launch in February, 2009. Sony Ericsson XPERIA X1 also runs JavaFX mobile platform including also CLDC 1.1, WMA (JSR-120), MMAPI (JSR-135) and File/ PIM(JSR-75). Some JavaFX Mobile samples were also tested on with Windows Mobile 6.x devices. The following figure shows some samples of JavaFX Mobile platform.


Fig. 24. JavaFX examples on Windows Mobile OS device

## 5. Conclusions

This chapter covers the existing UI libraries which are compatible with JME platform (CLDC and CDC configurations). Due to the big amount of existing UI libraries, they were divided into three categories, GPL/ LGPL, commercial and device's OEMs.

Most of the GPL/ LGP and commercial libraries were built as a separate software layers on top of device UI managers. They take advantage of UI manager's features and usually not of the Operating System's capabilities.

The UI libraries are provided by device manufacturer, they can be supported only by specific type of devices due to their specific Operating System features or due to some specific hardware capabilities. The level of support can be also provided by an external or by an integrated Java Virtual Machine within the device. For example, the IBM's WEME (or J9) virtual machine for Mobile Information Device Profile (MIDP) and Personal Profile (PP), the Jbed, crEMe and JBlend Java Virtual Machines. Unfortunately, these types of libraries offer the most appropriate and the most common solutions. They do not often support core features such as sound or localisation APIs.

The mobile device vendors try to cover and serve as much as possible a large area of customers; from simple-users to game-users and from elderly or disabled users to business users. For that reason, they focus to follow some standard (JSRs) and compatible libraries features which are available in the market. Such of these examples are the SVG (Scalable 2D Vector Graphics API- JSR 226) and OpenGL for ES libraries (Java Binding for the OpenGL® API – JSR 231).

However, additional effort is needed for these JSRs while they are deployed ("build" process) into specific device model due to its different Operating System and hardware demands. For that reason, some applications which do support specific UI standards can run successfully in one device and they can fail "**partially**" to run in other devices. Many UI libraries were tested in order to validate their consistency and their execution on devices and their UI features were examined. Some of them may be used for future research; some of these may need to be re-defined and re-implemented taking into account standard device's hardware capabilities and Operating System native features (advanced "hidden" facilities?). This approach is very close to define a new JSR, or update existing one as it is displayed in the following figures.

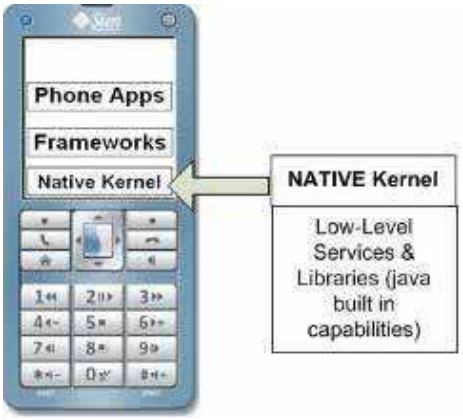


Fig. 25. Native Operating System

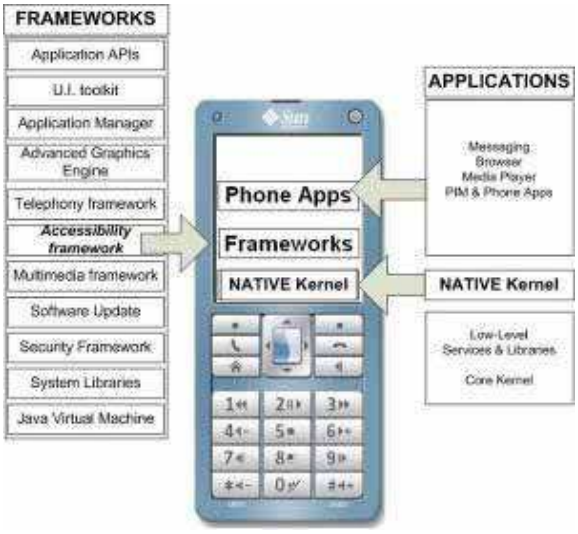


Fig. 26. Accessibility framework on top of JVM (as framework)

Alternatively, new additional bridge layers which reside on top of Operating Systems, are needed to expose both the simple and the complex accessible cross-platform applications to UI capabilities and moreover to native assistive technologies. The following figure shows the approach that will be used for the AEGIS European project in order to facilitate the accessibility support on various levels. Besides, this approach will take advantage of pre-built-in Java capabilities.

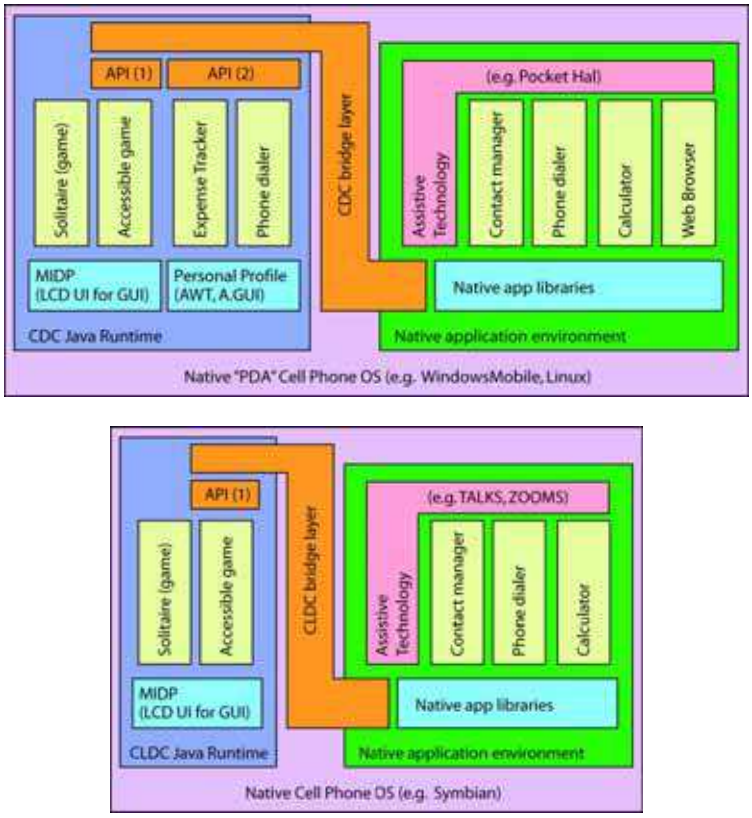


Fig. 27. CDC and CLDC with ÆGIS - accessibility API exposure to AT

## 6. References

JAVA4EVER (2005), *Apime*,
http:/ / www.java4ever.com/ index.php?section=j2me&project=apime&menu=main

J2ME-MWT Team (2005-2007), *microWindowToolkit: The open source framework for developing user interfaces in J2ME*, http:/ / j2me-mwt.sourceforge.net/

Khronos Group (1997 – 2009), *OpenGL*, http:/ / www.opengl.org/ about/ arb/

lwvcl.com (2007), *Light weight Visual Component Library*, http:/ / lwvcl.com/ index.php

Paxmodept (2009), *Pax Jwa ME (J2ME) Framework*,
http:/ / www.paxmodept.com/ paxmodept/ products.htm

Robert Bajzat (2002-2006), *Thinlet*,
http:/ / www.gplpedia.com/ referlink.php?id=1583&act=demo&referlink=http:
/ / thinlet.sourceforge.net/ demo.html
Sun Microsystems (2008), *Developer's Guide: Lightweight UI Toolkit*, page 13-16, Sun Microsystems, Santa Clara, California
TinyLine (2002-2009), *TinyLine 2.1 SDK Features*, http:/ / www.tinyline.com/ products.html
Tricast Solutions Ltd.( 2005-2007), *TWUIK*, http:/ / www.tricastmedia.com/ twuik/
Wikipedia (2009), *Abstract Window Toolkit*
http:/ / en.wikipedia.org/ wiki/ Abstract_Window_Toolkit
Wikipedia (2009), *Open GL ES*, http:/ / en.wikipedia.org/ wiki/ OpenGL_ES
Wikipedia (2009), *Mobile 3D Graphics*,
http:/ / en.wikipedia.org/ wiki/ Mobile_3D_Graphics_API
Wikipedia (2009), *BlackBerry* , http:/ / en.wikipedia.org/ wiki/ BlackBerry
Wikipedia (2009), *Java Speech API*, http:/ / en.wikipedia.org/ wiki/ Java_Speech_API
Yura.net (2008), *Swing ME*, http:/ / swingme.sourceforge.net/


**User Interfaces**

Edited by Rita Matrai

ISBN 978-953-307-084-1
Hard cover, 270 pages
**Publisher** InTech
**Published online** 01, May, 2010
**Published in print edition** May, 2010

Designing user interfaces nowadays is indispensably important. A well-designed user interface promotes users to complete their everyday tasks in a great extent, particularly users with special needs. Numerous guidelines have already been developed for designing user interfaces but because of the technical development, new challenges appear continuously, various ways of information seeking, publication and transmit evolve. Computers and mobile devices have roles in all walks of life such as in a simple search of the web, or using professional applications or in distance communication between hearing impaired people. It is important that users can apply the interface easily and the technical parts do not distract their attention from their work. Proper design of user interface can prevent users from several inconveniences, for which this book is a great help.



INTECH
open science | open minds

**InTech Europe**
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821