MiniGUI Technology White Paper

Version 3.0 (revised edition 2) for MiniGUI V3.0

Latest revision: 2017/11/07

Contents

1 Introduction

1.1 What Is MiniGUI

MiniGUI (http://www.minigui.com), developed by FMSoft, is a lightweight graphics user interface (GUI) support system for embedded systems and IoT devices. Since the first release under the GPL license at the beginning of 1999, MiniGUI has been widely used in handheld terminals (mobile phones and PDAs), set top boxes, industry control systems, industry instruments, portable media players, digital photo frames, information terminals, and so on.

At the same time, MiniGUI has become a cross-operating-system GUI system; it can run on Linux/uClinux, eCos, VxWorks, pSOS, ThreadX, Nucleus, OSE, and even uC/OS-II, also on the Win32 platform; the hardware platforms have been tested, including Intel x86, ARM (ARM7/ARM9/StrongARM/xScale), PowerPC, MIPS, and M68k (DragonBall/ColdFire). The MiniGUI V2.0.x provides the full multi-process support for high-end embedded devices based on embedded Linux, which brings MiniGUI into the high-end embedded market. As the most advanced version of MiniGUI after MiniGUI V2.0, the MiniGUI V3.0 has many important enhancements, such as BIDI text display support, transparent control, independent scroll control, unicode pre-rendered font(UPF), bitmap font, and new components including mGUtils, mGPlus, mGNCS, and mGEff.

MiniGUI is “a cross-operating-system graphics user interface support system for embedded devices”, and “an embedded graphics middleware”. So far, MiniGUI has been authorized by the most famous telecommunication equipment supplier in China, the biggest TV set manufacturer in China, the main TD-SCDMA standard maker, and the largest processor manufacturer in the world. MiniGUI has been widely adopted and applied by the leading manufacturers in the following fields: industry instrument, medical equipment, and the military industry. At the same time, MiniGUI has been recognized by global embedded devices developers, and is sold overseas including North America, Japan, Chinese Taiwan and Malaysia. MiniGUI has become the industrial de-facto standard in the field of embedded graphics middleware. It is worth mentioning that about 60 percent of net authenticated TD-SCDMA mobile use MiniGUI as their embedded graphical platform to support 3G application such as browser and videophone. Among these types of TD-SCDMA mobile include type T68 of Hisense and type U85 of ZTE, and so on. TD-SCDMA is a kind of 3G communication standard independently developed by China.

FMSoft not only releases some versions of MiniGUI under GPL[1], but also provides MiniGUI-VAR and other key software products for commercial customers. This document is organized to introduce features and apply areas of MiniGUI V3.0 in detail.

1.2 The Origin and Evolution of MiniGUI

Almost twenty years have passed since MiniGUI was launched at the end of 1998. Originally, MiniGUI was designed to provide a simple human-machine interface for a control system, which was

[1] The GPL is a widely used free software license, originally written by Richard Stallman for the GNU Project. Please visit http://www.gnu.org in detail.

based on Linux; no one foresaw that MiniGUI would become a cross-OS embedded GUI system. Fortunately, MiniGUI has been widely used in various projects since launched, and the increasing requirements from practical projects make MiniGUI grow into a cross-OS embedded GUI middleware product gradually.

In December 1998, the initiator of FMSoft, Wei Yongming, began to develop MiniGUI, and applied it in a computerized numerical control (CNC) system.

In March 2000, Lenovo adopted MiniGUI to develop the installer for HappyLinux V1.0 (a Linux distribution). At that time, MiniGUI had been a powerful embedded GUI support system for Linux.

From April 2000 to September 2002, as one of the famous free software, MiniGUI was developed and released under the GPL license.

In September 2002, the core developers of MiniGUI founded Beijing FMSoft Technology Co., Ltd., and started the commercial marketing with the free software.

MiniGUI V1.2.6 and MiniGUI V1.3.0 were released in May 2003 and September 2003 respectively.

In October 2003, MiniGUI was ported to the uClinux and eCos operating systems. Thus, MiniGUI had become a cross-OS embedded GUI system.

In August 2004, HUAWEI (the leading telecommunication equipment provider) used MiniGUI as the platform on set top box (STB), hand-held devices, etc.

In November 2004, FMSoft released MGIS solution for embedded geographic information system(GIS) application.

In January 2005, DaTang Mobile (the designer of TD-SCDMA technical standards) used MiniGUI as MMI solution[1] for TD-SCDMA mobile phone.

In March 2005, the Customized version of MiniGUI(MiniGUI-CMR) and Standard version of MiniGUI(MiniGUI-STD) had been released.

In May 2005, FMSoft and Intel reached an agreement on authorizing to develop its series of new products development and production of "Digital Family".

In July 2005, MiniGUI develop environment based on VxWorks Simulator had been released, which extended the cooperation between FMSoft and WindRiver.

In August 2005, Korea system programmer has become the distributor of FMSoft products officially in Korea, which marked a new stage in the progress for our products towards further internationalization.

In September 2005, FMSoft released the mSpider browser. In September 2005, FMSoft released the eDillo V0.4.0 underlying GNU GPL license.

In January 2006, FMSoft released the component products based on MiniGUI, including [2]mGp V1.0, mGi V1.0, and mG3d V1.0.

In February 2006, the embedded browser mSpider V1.6 was released.

In March 2006, FMSoft released MGDesktop V2.5. And Changhong Electric Co., Ltd. used MiniGUI and mSpider to develop DTV and IPTV, which indicated that FMSoft become the leading solution provider on DTV, IPTV, and its related areas.

In May 2006, FMSoft provide the port of MiniGUI for Vxworks.

In June 2006, FMSoft became a strategic partner of AMD; MiniGUI and Fhas were used by AMD as the graphics environment for the referenced design proposal of Argon PMP.

In June 2006, FMSoft integrated MiniGUI with the Real-Time Operating System OSE of Enea. MiniGUI can simplify the design and development of the graphical user interface, which runs on Hand-Held Terminal Device, telecom device, Medical Equipment, STB and Industrial Control system.

In August 2006, FMSoft cooperated with Atmel, which is an international leading semiconductor supplier. MiniGUI was used to develop MMI reference design for a wide range of chips in WiFi field, and it further strengthened FMSoft's world leading position in WiFi field.

In November 2006, Taiwan Inventec electronic company used MiniGUI to develop the visible IP phone supporting Skype, which promoted FMSoft's high-end status in digital-medium domain.

In December 2006, in HongKong ITU Telecom World, DaTang Mobile released TD-SCDMA 3G mobile software standard platform Arena, which used MiniGUI and MiniGUI Optional Components as their terminal application develop environment. This symbolized that MiniGUI already became handset de-facto industry standard in TD-SCDMA standard platform.

In December 2006, FMSoft released Embedded Flash Player: mSeal V1.0.

In March 2007, FMSoft released MGDesktop V3.0. In July 2007, FMSoft released the full-featured and high-end embedded browser mDolphin V1.0.

In February 2008, FMSoft released MGDesktop V4.0. In June 2008, FMSoft released mDolphin V2.0.

In October 2010, FMSoft released MiniGUI v3.0.12, miniStudio V1.0.8.

In August 2017, FMSoft released the latest source code of MiniGUI V3.0, all MiniGUI components, and mDolphin V3.0 under GPL 2.0/3.0, on GitHub.

In October 2017, WEI Yongming resumed the development of MiniGUI.

[2] It employs another product of FMSoft: Fhas.

At present, the latest MiniGUI V3.0.12 supports Linux/uClinux, eCos, uC/OS-II, VxWorks, pSOS, ThreadX, Nucleus, OSE and Win32 platform. FMSoft has taken an important step to his success with the commercial business mode based-on free software.

1.3 Typical Application Fields of MiniGUI

Since the original CNC system to the present popular smart hand-held devices, MiniGUI has been applied in many products. The main application fields of MiniGUI can be divided into the following categories:

Hand-held devices

Hand-held devices, including 2.5G/3G/4G feature phones, WiFi phones, portable media players, digital photo frames.

Figure 1.1 The left figure shows a TD-SCDMA 3G mobile end-software standard platform called Arena released by DaTang mobile communication equipment Co., Ltd, a leading corporation in mobile communication area of China. Arena uses ThreadX as its operating system, and adopts MiniGUI and Fhas, an application development platform, as the graphical system and application developing platform. The right figure shows a visual IP telephone supporting SKYPE developed by Inventec of Taiwan, which uses MiniGUI as the graphical system.

Figure 1.1 Typical application of MiniGUI: smart handheld devices

Digital-media devices and STBs

Figure 1.2 shows a web browser for a STB based on MiniGUI and an government information query terminal developed by FMSoft.

Figure 1.2 Typical application of MiniGUI: digital media devices and STBs

Industry instruments and control systems

Figure 1.3 illustrates some industry instruments and control systems based on Linux and MiniGUI.

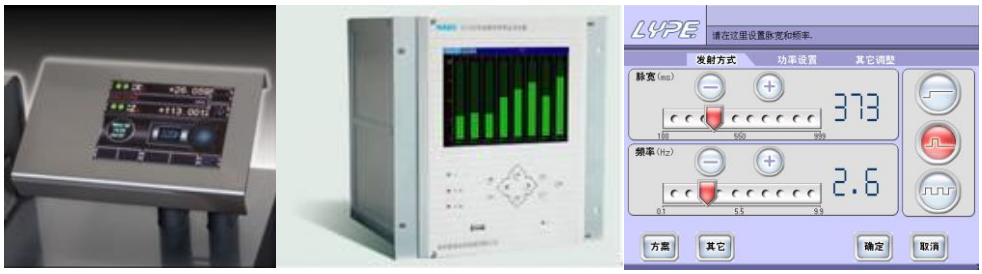

Figure 1.3 Typical application of MiniGUI: industry instruments and control systems

2 Features and Advantages of MiniGUI

2.1 Technical Features of MiniGUI

MiniGUI is one of the two commercial GUI system for embedded Linux in the world, providing fully graphical support for embedded Linux system. MiniGUI provides complete multi-process support for embedded system, and can run under three modes: MiniGUI-processes, MiniGUI-threads and MiniGUI-standalone. The main technical features of MiniGUI are as follow:

Hardware support

MiniGUI has been proven to be capable of running smoothly on the embedded systems with such CPUs/SoCs based on i386, ARM, MIPS, PowerPC etc.

- ✓ Support for low end vide devices such as monochromatic LCD and high end video devices. MiniGUI also provides support for special video devices (such as YUV device) by using graphics engine, which runs under the graphics abstract layer of MiniGUI. No limit for video resolution.
- ✓ Support for slave screens. If your system has multiple video devices, you can use one device as the master screen of MiniGUI to create main windows and controls and the other devices as the slave screens. By using GDI APIs of MiniGUI, you can also render text, output graphics to the slave screens.
- ✓ Support for all kinds of input devices, including PC keyboard, PC mouse, keypad, touch screen, remote controller, and so on.
- ✓ Support for multiple keyboard layouts, including American PC, French, German, Italian, Spanish, and so on.

Footprint

- ✓ At least 1024KB static memory (FLASH), varies from different configuration options.
- ✓ The typical RAM needed by MiniGUI itself are 1024KB. 8MB system RAM is recommended.

Operating Systems

- ✓ Support for Linux; MiniGUI can run in three runtime modes on Linux: MiniGUI-Processes, MiniGUI-Threads, or MiniGUI-Standalone .
- ✓ Support for RTOSes, such as VxWorks, ThreadX, eCos, uC/OS-II, pSOS, ThreadX, Nucleus, OSE; MiniGUI runs in two runtime modes on these RTOSes: MiniGUI-Threads or MiniGUI-Standalone .
- ✓ Support for built-in resources. You can compile the resources (bitmaps, icons, and fonts) into the library, so it is unnecessary to read the resources from files. Thus, MiniGUI can be used on some embedded systems without file systems.
- ✓ Special support for embedded systems, including the common I/O operations, byte-orders related functions, and so on.

Windowing sub-system

- ✓ Mature multi-window and messaging mechanism. By using the MiniGUI-Threads mode, it can create the main window in different threads and support for messaging among the threads. By using the MiniGUI-Processes mode, it supports for the multi-process windowing system.
- ✓ Support for dialog box and message box.
- ✓ Commonly used controls (widgets), including static label, button, single-line and multi-line edit boxes, list box, combo box, menu button, progress bar, property sheet, toolbar, track bar, tree view, list view, month calendar, grid view, animation, icon view, and so on.
- ✓ Support for other GUI elements, including menu, acceleration key, caret, timer, etc.

Graphics sub-system

- ✓ Support for enhanced GDI APIs. You can use these APIs to do raster operations, create complex regions, draw or fill ellipses, arcs, and polygons, etc. There are advanced 2D graphics functions available on C99 math library. We can even implement these advanced graphics interfaces on low-end video devices by using “Shadow” engine which runs under the graphics abstraction layer of MiniGUI.
- ✓ Support for almost all popular image file types including GIF, JPEG, PNG, Win32 BMP, etc. (JPEG and PNG are supported by using libjpeg and libpng respectively).
- ✓ Support for Windows resource files, for example, Windows bitmap, icon, cursor, etc.
- ✓ Support for multiple character sets and multiple font types. At present, what are supported include such character sets as ISO8859-1 ~ ISO8859-15, GB2312, BIG5, EUCKR, SHIFT-JIS, UNICODE (UTF-8 and UTF-16 encodings); bitmap fonts such as Qt Pre-rendered fonts, and vector fonts such as TrueType (by using FreeType library).
- ✓ Input method interface to support special embedded devices. Moreover, input methods for Simplified Chinese are built-in.

2.2 Advantages of MiniGUI

Comparing with other embedded graphics systems, MiniGUI has the advantages as follow:

Scalability

The abundant functions and configurability of MiniGUI makes it applicable in low-end products based on the CPU main frequency 30MHZ as well as high-end products. The developers can create dashy user interfaces by using the advanced control styles and the skin technology.

The feature of cross-operating-system makes it easy to run MiniGUI on the simplest embedded operating system, such as uC/OS-II, and the modern embedded operating system, such as Linux. Furthermore, MiniGUI provides complete and multi-window system for embedded Linux operating system.

These features make MiniGUI have strong scalability, which is considered at the begin time of designing MiniGUI, so MiniGUI can not only be applied in simple devices, but also be applied on complicated electronic products.

Light-Weight and Low Resources Consumption

MiniGUI is a light embedded graphical library, and we have considered the hardware situation of embedded devices and the requirement of system resources completely. The size of MiniGUI library can be reduced to about 500 KB or less, and this is very good for the embedded devices.

Besides these, the latest R&D result indicates, MiniGUI is capable of running on a system with 30 MHz CPU and 4MB RAM successfully (on uClinux), which cannot be reached by other embedded graphics systems.

High Performance and High Reliability

The good architecture and optimized graphics interfaces of MiniGUI lead a very fast graphics output. In fact, MiniGUI was designed for real-time systems, taking into consideration the compactness, high performance, and high efficiency from the beginning. MiniGUI has been widely used in many areas, especially in industry production systems. MiniGUI plays an important role in these products or projects.

Since the release of its first version in 1999, MiniGUI has been employed by many products and projects, which, in turn, drive MiniGUI to improve its reliability and robustness constantly.

For the latest successful cases, you could visit the “typical cases” area in the following web site:

http://www.minigui.com/

Configurability

GUI systems are expected to be configurable in order to satisfy the different requirements from the embedded systems. Like Linux kernel, MiniGUI have many compilation configuration options, though which we can designate MiniGUI libraries to include and exclude some features. In general, MiniGUI can be customized against the following aspects:

- ✓ The target operating system MiniGUI runs on.
- ✓ The target hardware platform runs on.
- ✓ The runtime mode: MiniGUI-Threads, MiniGUI-Processes, or MiniGUI-Standalone.
- ✓ GAL and IAL engines to be used.
- ✓ Font types to be supported.

- ✓ Charsets to be supported.
- ✓ Image file formats to be supported.
- ✓ Widgets to be used.
- ✓ Window/Widget appearance styles: classic, flat, or fashion style.

These configuration options increase the flexibilities of MiniGUI, and you can create the most suitable system based on your requirements.

In a word, MiniGUI is an embedded graphics support system for real time embedded products with high efficiency, reliability, scalability, and configurability. It brings the modern windowing and graphics technologies into the embedded devices. We can summarize the advantages of MiniGUI as follow:

- ✓ Support for multiple embedded OSes with great portability.
- ✓ Scalable architecture, easy to extend.
- ✓ Rich functions, flexible to customize.
- ✓ Optimal balance between low footprint and high performance.
- ✓ Wide application fields.

2.3 New feature of MiniGUI V3.0

MiniGUI V3.0 adds or improves the following features based on former releases:

Double Buffering Main Window

When a MiniGUI 3.0 main window has double buffer, you can get the rendering result of the main window in your own buffer. By using double buffer technology, you can use an advance 2D graphics interface or 3D rendering library to get the 3D user experience easily.

Look and Feel Render (LFRDR)

MiniGUI V3.0 abandons the old rendering mode which only supports three kinds control style and introduces a new mode of render, namely, Look and Feel Renderer. Renderer defines how to draw the window elements, which is modified based on MiniGUI V2.0. Window elements contain caption, caption button, scrollbar, selected item, invalid item, highlight item, hot item, object of 3D, etc. Apparent attributes of window elements contain color, size, font, etc. Renderer of window can customize the size, color, picture, font of window element to give user the convenience to design individual window apparent style. User can use a special renderer to draw a main window or a control and also can develop his/her own LF renderer by customizing the metrics, color, font, and icon of window elements to get more beautiful user interface.

At present, MiniGUI provides four apparent styles for the render of window and control. The four styles are: Classic, Flat, Fashion and Skin. User can configure MiniGUI to one of the styles through a configure option.

- ✓ Classic: The user interface of this style renderer is similar to Windows 95 window apparent style and is the most widely used renderer in all renderers. Figure 2.1 shows the user interface rendered by classic renderer.

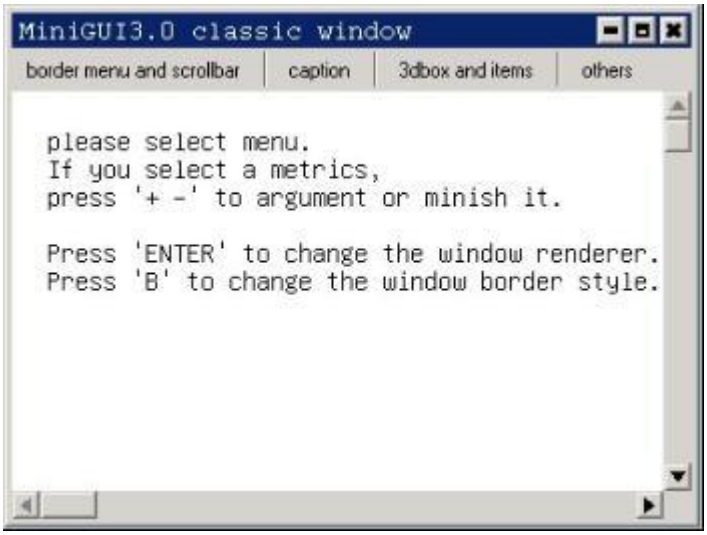

Figure 2.1 Default renderer of MiniGUI(Classic renderer)

- ✓ Fashion: This renderer uses the gradient filling function offered by mGPlus a component of MiniGUI V3.0 to rend the window and controls, so user can obtain very splendid graphics interface. Figure 2.2 shows the effect of fashion renderer.

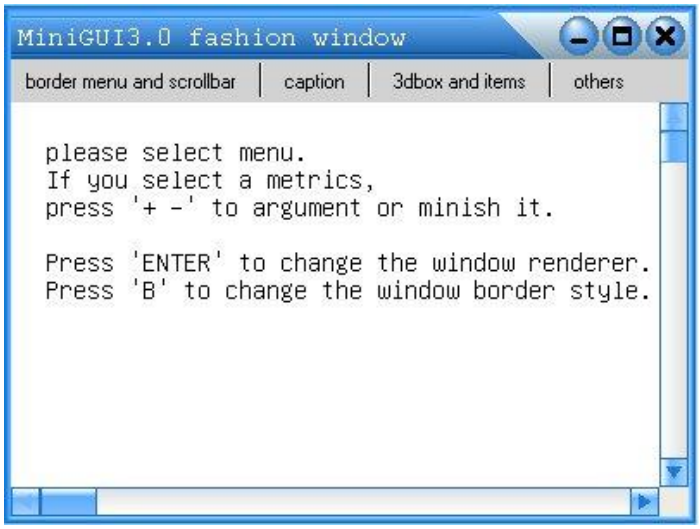

Figure 2.2 Fashion renderer of MiniGUI

- ✓ Flat: The user interface drawn by this renderer is very clear and concise. This renderer is fit for monochromatic screen and gray screen. Because of simple rendering and low resource consumption, the running speed of this renderer is the fastest in all renderers. Figure 2.3 shows a main window rendered by flat renderer.

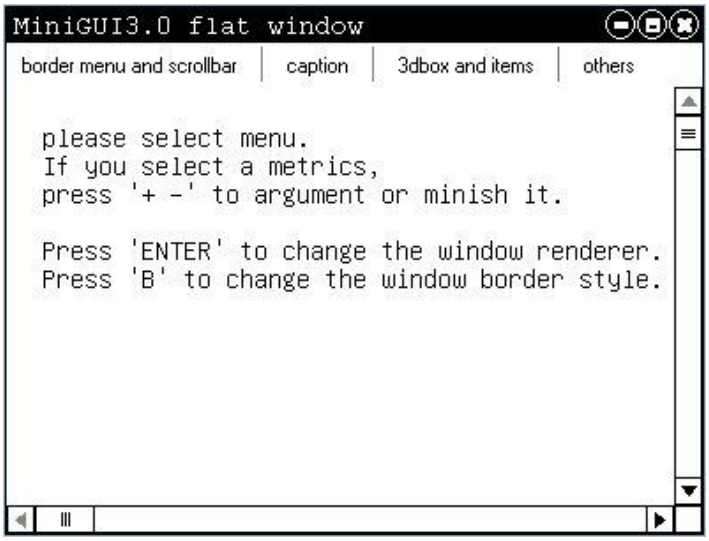

Figure 2.3 Flat renderer of MiniGUI

- ✓ Skin: Renderers introduced above is almost realized by code, so they are light and flexible. But the difference between the devices in embedded application area is giant. Some devices have very high performance. On these device, user can use skin renderer to beautify the user interface. Skin renderer needs a complete set of picture relevant to user interface that will consume some store resource. The advantage of skin renderer is that it makes user can custom the user interface by substituting the origin pictures of system with pictures designed by user himself/herself. Figure 2.4 shows the effect of skin renderer.

Figure 2.4 Skin renderer of MiniGUI

Support for BIDI Text

In addition to languages that we are familiar with are written from left to right (such as English and Chinese), there are some languages are written from right to left, like Arabic and Hebrew. To support these languages, MiniGUI 3.0 provides support for two language character sets for Arabic and

Hebrew and BIDI text rendering. MiniGUI 3.0 also provides the keyboard layouts for these two languages. Figure 2.5 shows an example of BIDI text.

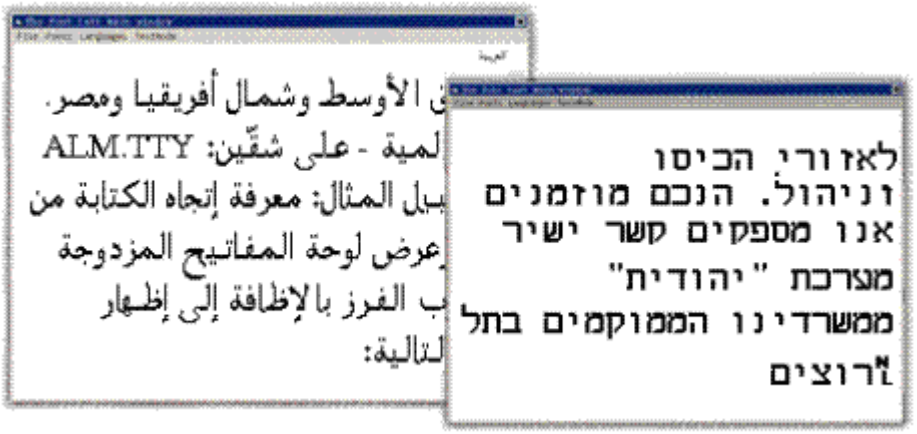

Figure 2.5 display of Arabic and Hebrew

Irregular window

In the previous version of MiniGUI, all the windows only can be rectangular. MiniGUI 3.0 provides support for irregular window, such as round rectangular window and non-rectangular window. By using the irregular window, the user interface display effect can be more amazing. Figure 2.6 shows a irregular window and irregular controls.

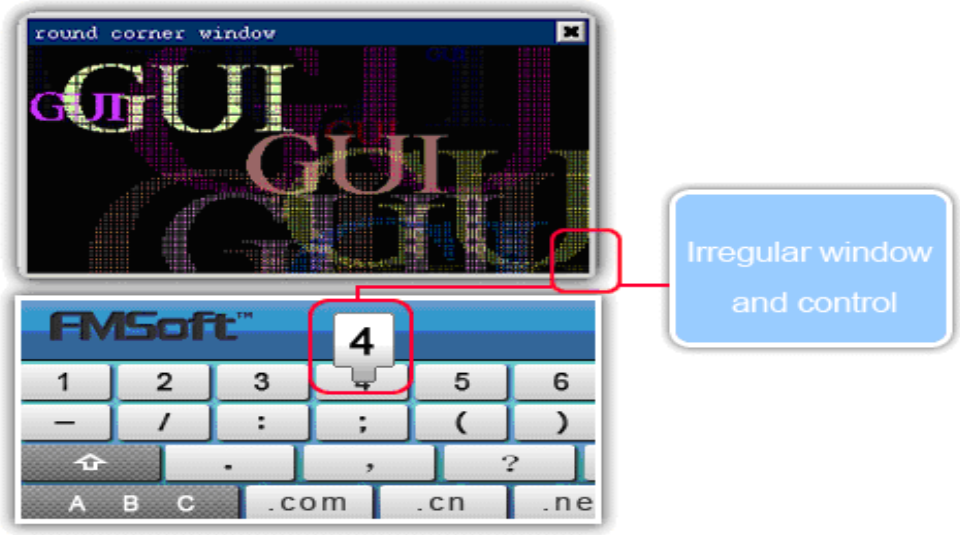

Figure 2.6 irregular window and irregular controls

Enhanced font support

In MiniGUI 3.0, FMSoft introduces a new UNICODE font file format, known as the "UPF" font. The most important feature of this font is to facilitate use in the multi-process environment. It can reduce the memory footprint of MiniGUI-Processes runtime mode. At the same time, FMSoft has enhanced VBF font format, VBF font has been upgraded to 3.0. VBF 3.0 can define the glyphs for Arabic and other languages.

Others

MiniGUI 3.0 offers support for the customization of desktop. Through the external programming interface, the user can place icons on the desktop and respond to the desktop events, then achieve an effect similar to the Windows desktop. In addition, MiniGUI 3.0 also enhances the control of transparency to make it more efficient and does not depend on the internal controls' implementation. At last, MiniGUI 3.0 provides independent scrollbar control and a unified virtual frame buffer support.

In addition, new components including mGUtils, mGPlus, mGNCS, and mGEff are added to MiniGUI. MiniGUI V3.0 also unifies the management of resource include font, bitmap, icon and cursor, because the inner resource mode and non-inner resource mode are the same effect to MiniGUI. Figure 2.7 shows a customized desktop and difference between transparent control and opaque control.

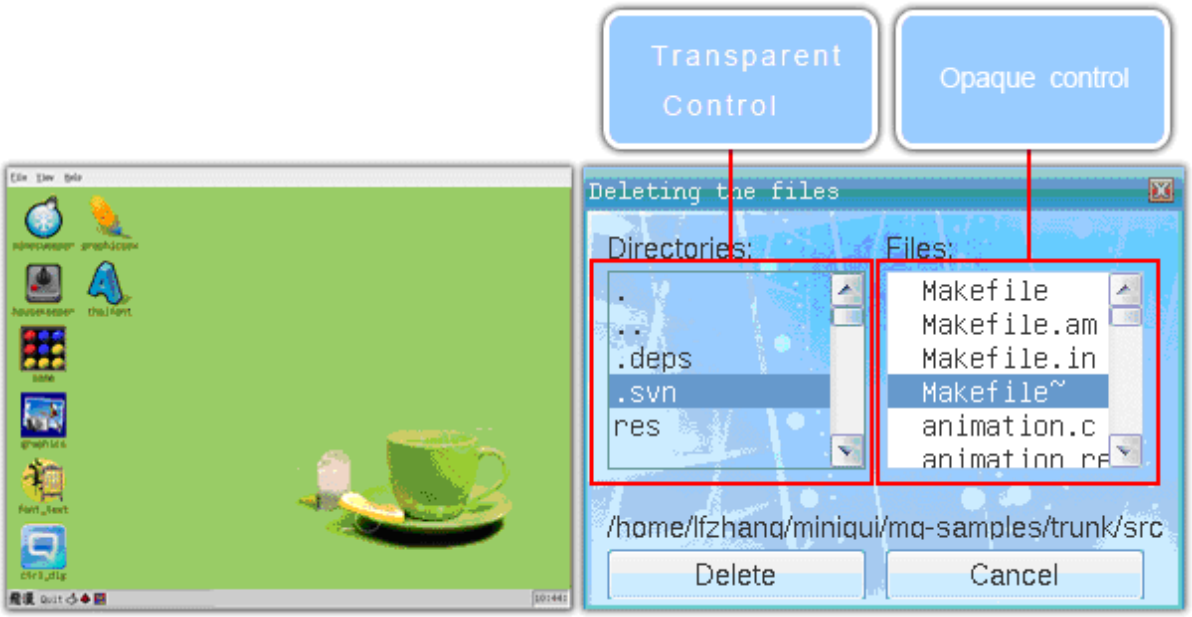

Figure 2.7 Customized desktop, transparent control and opaque control

3 System Requirements to Run MiniGUI

3.1 Operating Systems Supported by MiniGUI

MiniGUI can run on any embedded operating system which supports multi-tasks in theory; now it has been proved that MiniGUI can run on Linux, uClinux, VxWorks, eCos, uC/OS-II, pSOS, ThreadX, Nucleus, and OSE smoothly. MiniGUI can also run on Win32 platform. At the same time, MiniGUI provides identical APIs on different operating systems. MiniGUI V3.0 supports Linux/uClinux.

3.2 Hardware Platforms Running MiniGUI

In theory, running MiniGUI is independent of the underlying hardware platform; as long as there is one supported operating system running on one hardware platform, MiniGUI can run on the hardware platform. MiniGUI can run on many popular embedded hardware platforms such as Intel X86, ARM (ARM7/ARM9/StrongARM/xScale), PowerPC, MIPS, DragonBall, ColdFire, and so on.

3.3 Footprint of MiniGUI

MiniGUI takes few resources itself; with embedded Linux as an example, the storage and memory sizes typical used by MiniGUI are as follow:

- ✓ Linux kernel: 300KB~500KB (decided by the system requirements)
- ✓ File system: 500KB~2MB (decided by the system requirements)
- ✓ MiniGUI library: 500KB~900KB (decided by the configuration options)
- ✓ MiniGUI fonts, bitmaps, and other resources: 400KB (decided by the applications, the minimum is 200KB)
- ✓ Applications: 100KB~2MB (decided by the system requirements)

The total static memory used by MiniGUI will be about 2MB to 4MB. On some systems, especially on traditional real-time operating systems, full-featured MiniGUI only uses 1MB static memory.

For the detail footprint of MiniGUI V3.0, please refers to *MiniGUI Data Sheet* V3.0.

4 Software Architecture of MiniGUI

4.1 Software Architecture of MiniGUI

Why MiniGUI can run well on so many embedded system ? The reason is that MiniGUI has a good software architecture. MiniGUI separates the application layer and the system layer via the software abstract layer (also call portable layer). The relationship between MiniGUI and real-time operating systems is illustrated in figure 4.1. An application based on MiniGUI can implement its functions by calling APIs of ISO C library and MiniGUI libraries, and function interfaces of system and drivers. The graphics abstract layer and the input abstract layer of MiniGUI hide the details of underlying hardware and operating systems, so the applications need not to take care of the output and input devices. Besides, the special concept of MiniGUI runtime mode provides convenience to run MiniGUI on different operating systems.

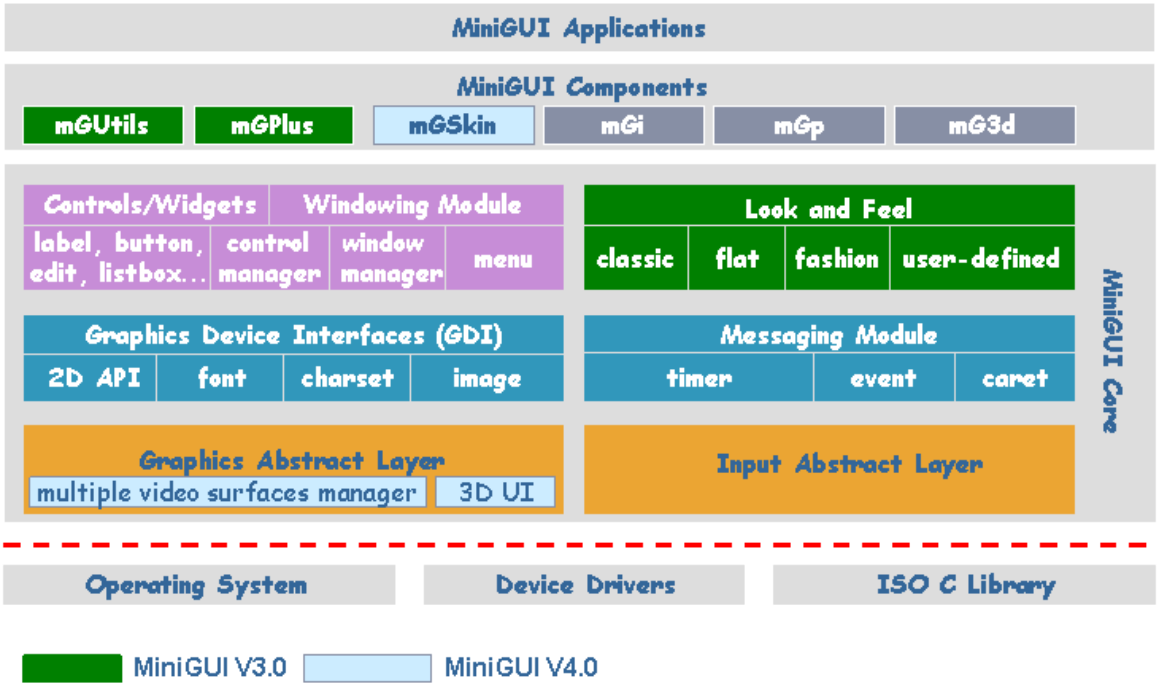

Figure 4.1 Software architecture of MiniGUI

From the above figure, we can find that from the bottom layer to top layer, MiniGUI is composed of several modules shown as below:

- ✓ Graphics abstract layer, GAL. GAL abstracts graphical interface of different device and system, providing uniform graphical interface for upper layer of MiniGUI. GAL includes software module aiming at FB device of Linux and LCD device of eCos etc.. This software modules call interfaces of low level device to realize concrete operations of GAL, such as open and close of device, set of resolution and display mode. We call this software module adapting interface of GAL as engine similar to device driver of system.
- ✓ Input abstract layer, IAL. Like GAL, IAL abstract all input device, such as keyboard, keypad, mouse, touch screen to provide uniform interface for upper layer. To support keyboard, touch screen and mouse, user needs to realize input engine for this device in IAL. Through IAL and the engines in IAL, MiniGUI can support input device such as, console(keyboard and mouse) , touch screen, remote controller, keypad etc..
- ✓ Graphics abstract interfaces, GDI. GDI is based on GAL providing graphical interface for application, such as curve drawing, text output, rectangle filling and so on. GDI also includes other independent sub-modules, for example, font and charset, image etc..
- ✓ Message process module. This module is on the basis of IAL. It realizes the message process mechanism and provides complete message manage interfaces for applications. As is well known, almost all GUI system is driven by event. The running of system and application is based on the message management module.
- ✓ Windowing module and control/widget module. On the basis of GDI and message process module, MiniGUI realizes the windowing module and control/widget module. This module supplies base interface for creating main window and control, and manages the control class. Control class is a important concept of reusing code of control. By using control class, user can create a number of instances of a control class. All this instances use same code of the control class. In this way, we achieve a similar concept of class and instance of C++ and able

to maximize reuse existing code, and improve the maintainability of the software. MiniGUI control module has realized common controls, such as static, buttons, edit box, list box, drop-down box, and so on.

- ✓ Look and Feel. This module is interface provided to application by MiniGUI V3.0 and can be used to customize MiniGUI window and the control rendering. In the previous version, the custom to main window and control has not yet been pulled out to form an independent module, but we can still let the main window and controls have three show style through the configure option of MiniGUI. The three show styles are a similar PC-style three-dimensional (PC3D), flat-style (FLAT), popular style (FASHION). In MiniGUI 3.0, the appearance of main window and controls can be entirely customized by applications. When create main window or control, user can let the main window or control has different appearance by specifying different renderer name for them.

Modules above comprise the core of MiniGUI. Based on the interface of MiniGUI, MiniGUI provides several components for application. The components are mGi, mGp, mG3d, mGUtils, mGPlus. These components provide extra function for application. We will make a detail introduce for these components.

4.2 The Runtime Modes of MiniGUI

Different from the general-purpose operating system like Linux, the traditional embedded operating systems have some particularities. For example, uClinux, uC/OS-II, eCos, and VxWorks usually run on non-MMU CPUs, without support for processes that have separate address spaces but only threads or tasks. Therefore, the runtime environments are entirely different. Therefore, we need configure and compile MiniGUI into three runtime modes for different operating systems:

- ✓ **MiniGUI-Threads:** A program running on MiniGUI-Threads can create multiple cascaded windows in different threads, and all the windows belong to a single process or run in the same memory address space. MiniGUI-Threads is fit for most traditional embedded systems such as VxWorks, ThreadX, Nucleus, OSE, pSOS, uC/OS-II, eCos and so on. Of course, MiniGUI can also run on Linux/uClinux in this mode.
- ✓ **MiniGUI-Processes**[3]: In opposition to MiniGUI-Threads, a program running on MiniGUI-Processes is an independent process, which can also create multiple windows. MiniGUI also has realized the multi-process window system. MiniGUI-Processes are fit for full-featured UNIX-like operating systems, such as Linux. This mode is provided in MiniGUI V2.0.X and is strengthened in MiniGUI V3.0. The details of this mode will be show in the below.
- ✓ **MiniGUI-Standalone**[4]**:** In this mode, MiniGUI is running in a single task, does not need either multithread support or multi-process support. This mode is fit for simple application area

[3] Before MiniGUI V2.0 this runtime mode is called “MiniGUI-Lite”. MiniGUI-Lite provides a tradeoff for Linux system in multi-process environment, but does not solve the overlap of windows that are from different processes. The MiniGUI-Processes runtime mode of MiniGUI V2.0 provides full-featured windowing solution for multi-processes environment.

[4] At present, we only offer the MiniGUI-Standalone runtime mode for Linux/uClinux operating system.

which support only one function at one time. Example, some products use uClinux as their operate system that lack multithread support for some reason. In this conditions, you can develop applications in this mode.

MiniGUI can almost run on all operating systems under MiniGUI-Standalone mode theoretically. MiniGUI-Threads is suitable for real-time operating systems, which provide support for multi-task, or general-purpose operating systems with full-featured UNIX-like such as Linux, UNIX. MiniGUI can only run on UNIX-like operating systems under MiniGUI-Processes mode, like Linux.

No matter which mode is used, MiniGUI provides for applications the furthest compatibility; only a few initialization interfaces are different among different runtime modes.

The Runtime Mode MiniGUI-Processes

MiniGUI-Processes is a successor of MiniGUI-Lite. It offers full-featured support for multi-process embedded operating systems, such as Linux. The runtime mode MiniGUI-Lite offered by MiniGUI V1.6.x and previous versions is designed for multi-process environment of Linux; we can run several client processes on the basis of efficient client/server architecture, and make use of superior features like address space protection. With MiniGUI-Lite runtime mode, the flexibility, stability, and scalability of embedded system based on MiniGUI will improve greatly. For example, we can run several MiniGUI client processes on MiniGUI-Lite, and if one process terminates abnormally, other processes will still run well. Moreover, on MiniGUI-Lite, it is convenient for us to integrate third-party applications. Actually, this is why many embedded device developers use Linux as their operating system.

Although MiniGUI-Lite runtime mode provides support for multi-process, it cannot manage windows created by different processes at one time. Therefore, MiniGUI-Lite distinguishes windows in different processes by layers. This method fits for the most embedded devices with low-resolution screen, but brings some problems for application development.

MiniGUI V2.0.x solves this problem completely. On the MiniGUI-lite mode, MiniGUI V2.0.X realizes a window system in complete multi-process environment, in which every window belongs to different process can be displayed on the same desktop at one time. MiniGUI V3.0.x not only entirely inherit the MiniGUI-processes mode of MiniGUI V2.0.x but also make user can define the icon on the desktop and respond the event of desktop. So, the user can customize the desktop conveniently. Figure 4.2 gives the screenshots of MiniGUI-Lite runtime mode of MiniGUI V1.6.x and MiniGUI-Processes runtime mode of MiniGUI 3.0.x.

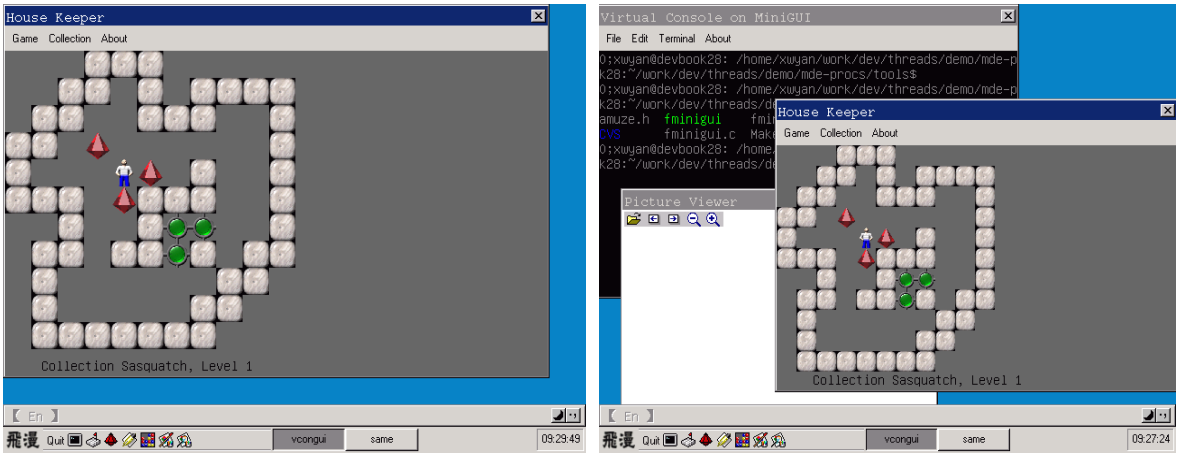

Figure 4.2 MiniGUI-Lite runtime mode of MiniGUI V1.6.x and MiniGUI-Processes runtime mode of MiniGUI 3.0.x

In the first screenshot of Figure 4.2, we run two client processes: vcongui and housekeeper. It is obvious that after housekeeper started, we could not see the windows of vcongui; in the second screenshot, we run three client processes: vcongui, picview, and housekeeper, but we can see all windows created by these three clients.

Compared with MiniGUI-Lite, MiniGUI-Processes runtime mode has obvious advantages. So MiniGUI is not only useful for traditional embedded system (MiniGUI-Threads), but also useful for embedded system with multi-process features, such as Linux. Besides, MiniGUI-Processes keeps the concept of layer in MiniGUI-Lite. You can put windows of different client processes into different layers. By using layer, we can create many workspaces for different processes like X Window, whereas, the footprint of MiniGUI-Processes is far less than X Window. With the MiniGUI-Processes runtime mode, MiniGUI will not only can extend its application fields in high-end embedded devices, but also can be applied in desktop environment.

Operating Systems and MiniGUI Runtime Modes

Table 4.1 illustrates the runtime mode(s) supported by MiniGUI on various operating systems.

Table 4.1 Operating Systems and MiniGUI Runtime Modes

Operating System	Runtime Mode(s) Supported
Linux	MiniGUI-Processes MiniGUI-Threads MiniGUI-Standalone
uClinux	MiniGUI-Threads MiniGUI-Standalone
VxWorks 6.x	MiniGUI-Threads
VxWorks 5.x	MiniGUI-Threads
ThreadX	MiniGUI-Threads
Nucleus	MiniGUI-Threads

OSE	MiniGUI-Threads
eCos	MiniGUI-Threads
uC/OS-II	MiniGUI-Threads
pSOS	MiniGUI-Threads

4.3 Windowing System

In MiniGUI, windows are generally organized in the form of hierarchy; the root window is the ancestor of all windows, all the other windows have parent windows except the root window, and each window may have child windows, brother windows, ancestor windows, or descendant windows, etc. Windows of the same level can be overlapped, but only one window can output to the overlapped region at one time.

MiniGUI has three window types: main window, dialog box, and control window (child window). The main window generally includes some child windows, and these child windows are generally MiniGUI built-in controls or user-defined controls. An application can also create main windows of other types such as dialog boxes or message boxes. A dialog box is actually a main window, and the application usually interacts with the user by using a dialog box.

4.4 Communication Mechanism

The communication mechanism of MiniGUI is similar to Win32's. The Figure 4.5 illustrates the mode of the way that how the messages transferring in MiniGUI-Thread runtime mode. Therein, Desktop thread acts as a micro-server, all of the messages fetched from Event thread will be sent to Desktop thread first, and then be dispatched to the target windows by Desktop thread.

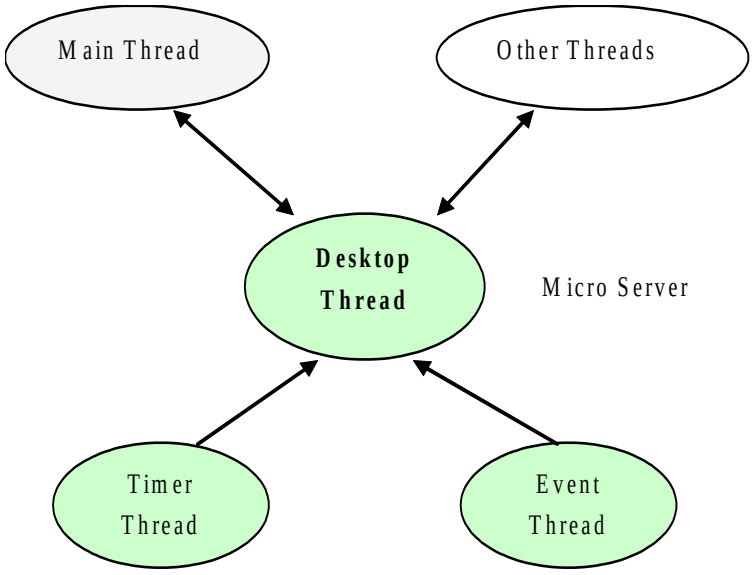

Figure 4.5 Communication mechanism of MiniGUI-Threads runtime mode

Comparing MiniGUI-Threads, the messages transferring mode of MiniGUI-Processes is implemented by UNIX socket. The communication mechanism is illustrated by Figure 4.6.

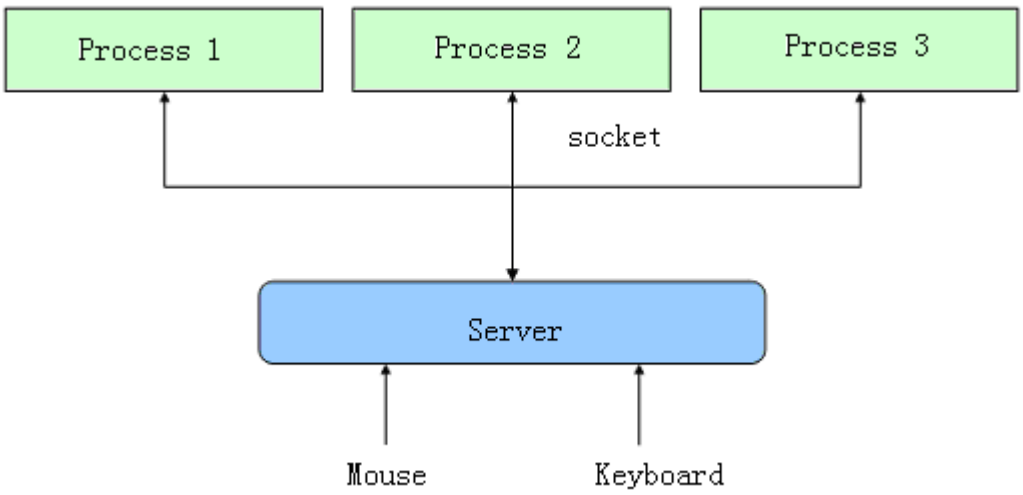

Figure 4.6 Communication mechanism of MiniGUI-Processes runtime mode

4.5 Fonts

MiniGUI Supports for multiple character sets and multiple fonts. At present, what are supported include such charsets/encodings as ISO8859-1 ~ ISO8859-15, GB2312, BIG5, EUCKR, SHIFT-JIS. The font types include bitmap fonts (RBF, VBF, QPF) and vector fonts, such as TrueType, Adobe Type1. MiniGUI's VBF font can be used to display special characters like Thai characters. MiniGUI can also zoom in the font glyphs automatically. For some special displaying equipment like TV set, MiniGUI also provides the anti-alias feature for font rendering.

MiniGUI V3.0 supports the new device font(bit-map font), which can provide other logical font APIs for applications and controls. The font glyphs can be defined by customized bitmaps. The font of VBF V2.0 has been updated to VBF V3.0. The VBF V3.0 hasn't been limited by supporting Latin character sets. The font-type number of single VBF font hasn't been limited to the inside of 255. The UPF font is the optimized release of QPF, which reduces the use of memory in the MiniGUI processes, and is convenient to using by the way of memory map. It provides BIDI text output and input for user, including Arabic (ISO8859-6), Hebrew (ISO8859-8).

4.6 Support of Input Devices

MiniGUI supports various common mouse/touch-panel devices. MiniGUI also provides excellent support with a calibration interface for touch-panel.

MiniGUI supports multiple keyboard layouts, including American PC, French, German, Italian, Spanish, and so on.

Except the common input devices, MiniGUI can provides support for various input devices, which are popular in embedded devices, such as remote controllers, keypads, buttons, and so on.

4.7 Input engines

The graphical engine of qvfb and wvfb brings some convenience to users in the developing MiniGUI. As an independent application program, the graphical engine causes a lot of inconveniences, such as virtual framebuffer program starting up single, users setting the display mode necessarily, and

only starting one MiniGUI application program in the mode of MiniGUI-Threads and MiniGUI-Standalone. Theoretically, the different application programs can run at different virtual framebuffers. MiniGUI V3.0 solves this problem by integrating the virtual framebuffer graphical engines into xvfb, which make users run programs easily by xvfb.

5 Development Environments

5.1. Current developing mode of MiniGUI

The MiniGUI application can be developed under Linux or Windows environment. Since MiniGUI is coded totally in C, it's an easy job to cross-compile and port MiniGUI and its applications to a different hardware platform.

An application for embedded devices can be cross-compiled in the host which installed the specified tool chain for a certain embedded device. The most common way is installing a cross compiler inherited from gcc and cross-compiling the application. If your operating system is VxWorks or UC/OS-II, you should install the relevant IDE (such as Tornado or ADS) in Windows, and compile your applications by using the IDE.

If you develop MiniGUI applications in Linux environment, there are two ways to run them after compiling, one is running MiniGUI on Linux console (you must make sure that the kernel built with the support for framebuffer), the other is running MiniGUI on qvfb or gvfb, which are a virtual framebuffer simulator running on X11).

If you develop MiniGUI applications under Windows environment, you can compile them by using Visual Studio IDE, then running the applications on wvfb (A framebuffer simulator running on Windows). Please see Figure 5.1.

Figure 5.1 A MiniGUI application running on wvfb simulator

Running MiniGUI applications in simulator or console directly, it is convenient to develop embedded application and you can avoid to run programs on embedded devices frequently. The standard debugger is also used to running embedded application in developing host. The xvfb of MiniGUI V3.0 is compatible with the qvfb/xvfb of MiniGUI 2.0.x/1.6.x.

5.2. miniStudio: the Integrated Development Environment of MiniGUI

miniStudio is an Integrated Development Environment of MiniGUI base on Eclipse. Developer can use miniStudio to make a fast customization of MiniGUI, design MiniGUI application user interfaces, and can also carry out resource management and debugging MiniGUI applications. miniStudio will provide user with the following application graphical user interface development tools:

- ✓ MiniGUI customization tool. It creates final MiniGUI libraries based on MiniGUI 3.0 BSP configuration.
- ✓ WYSIWYG (what you see is what you get) interface design tool. It helps user quickly design the user interface and makes it WYSIWYG.
- ✓ Font designer. It can carry out modification, addition and deletion operations on the existing font to make the font file to meet the needs of user.
- ✓ Bitmap Manager. It provides bitmap management functions.
- ✓ Text Manager. It unifies the management of text.

At present, the latest version of miniStudio is V1.0.8. You can download it on http://www.fmsoft.cn

6 Sample Program and Controls

6.1 "Hello world" Sample Program

The following code shows a "Hello World" program of MiniGUI. It creates an application window with size of 240x180 pixels, and displays "Hello world!" at the center of the window client region. In this example, renderers are used to control the appearance of the window. Figure 6.1.a, 6.1.b, 6.1.c show the appearance of the window separately rendered by three renderer: classic, flat, skin.

```
#include <stdio.h>

#include <minigui/common.h>
#include <minigui/minigui.h>
#include <minigui/gdi.h>
#include <minigui/window.h>

static int HelloWinProc(HWND hWnd, int message, WPARAM wParam, LPARAM lParam)
{
    HDC hdc;
    switch (message) {
        case MSG_PAINT:
            hdc = BeginPaint (hWnd);
            TextOut (hdc, 60, 60, "Hello world!");
            EndPaint (hWnd, hdc);
```

```
            return 0;
        case MSG_CLOSE:
            DestroyMainWindow (hWnd);
            PostQuitMessage (hWnd);
            return 0;
    }
    return DefaultMainWinProc(hWnd, message, wParam, lParam);
}


int MiniGUIMain (int argc, const char* argv[])
{
    MSG Msg;
    HWND hMainWnd;
    MAINWINCREATE CreateInfo;
    const char* old_renderer;

#ifdef _MGRM_PROCESSES
    JoinLayer(NAME_DEF_LAYER , "helloworld" , 0 , 0);
#endif

    CreateInfo.dwStyle = WS_VISIBLE | WS_BORDER | WS_CAPTION;
    CreateInfo.dwExStyle = WS_EX_NONE;
    CreateInfo.spCaption = "HelloWorld";
    CreateInfo.hMenu = 0;
    CreateInfo.hCursor = GetSystemCursor(0);
    CreateInfo.hIcon = 0;
    CreateInfo.MainWindowProc = HelloWinProc;
    CreateInfo.lx = 0;
    CreateInfo.ty = 0;
    CreateInfo.rx = 240;
    CreateInfo.by = 180;
    CreateInfo.iBkColor = COLOR_lightwhite;
    CreateInfo.dwAddData = 0;
    CreateInfo.hHosting = HWND_DESKTOP;

    old_renderer =SetDefaultWindowElementRenderer("classic");

    hMainWnd = CreateMainWindow (&CreateInfo);

    if (hMainWnd == HWND_INVALID)
        return -1;
    ShowWindow(hMainWnd, SW_SHOWNORMAL);

    while (GetMessage(&Msg, hMainWnd)) {
        TranslateMessage(&Msg);
        DispatchMessage(&Msg);
    }
    SetDefaultWindowElementRenderer(old_renderer);
    MainWindowThreadCleanup (hMainWnd);
```

```
    return 0;
}
```

Figure 6.1.a appearance of classic style renderer

Figure 6.1.b appearance of flat style renderer

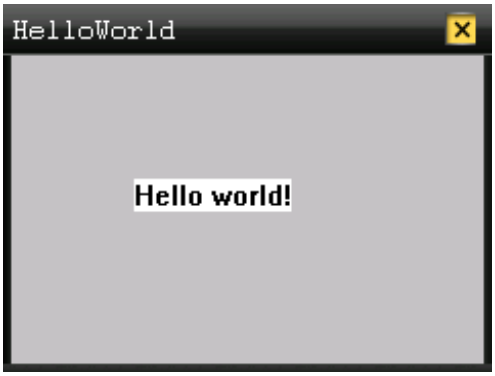

Figure 6.1.c appearance of skin style renderer

To achieve different styles appearance of window showed above, you only need to call *SetDefaultWindowElementRenderer* with different parameter. Corresponding code you have to change has been marked with red color. methods about how to call the function have been listed below.

Call for classic renderer：

```
old_renderer =SetDefaultWindowElementRenderer("classic");
```

Call for Flat renderer：

```
old_renderer =SetDefaultWindowElementRenderer("flat");
```

Call for skin renderer：

```
old_renderer =SetDefaultWindowElementRenderer("skin");
```

According to the above methods, application developers can easily choose the favorable window style. MiniGUI provides a variety of abundant controls, such as button, toolbar and so on, at the same time, it provides APIs to you to self-define controls or extend the built-in controls. We will introduce controls of MiniGUI in detail in the following. In order to brief the page, the appearance of all controls will only show the effect of classic style renderer.

6.2 Static Control

A static control is used to display information, such as text and digits in the specified position of a window, and can also be used to display some static image information, such as logos of your organization, product brands, etc. Figure 6.2 shows the static control of MiniGUI.

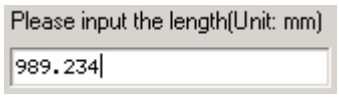

Figure 6.2 Static Control (used as a label of other control)

6.3 Button Control

Button control is the most frequently used control besides the static control. A button is usually used to provide switch selection for the user. The buttons of MiniGUI can be classified into push button, check box, radio button, etc. Figure 6.3 illustrates the button control of MiniGUI.

Figure 6.3 Button Control

6.4 List Box Control

A list box generally provides a series of options, which are shown in a scrollable window. The user can select one or more items with the keyboard or mouse. Figure 6.4 shows the list box control of MiniGUI.

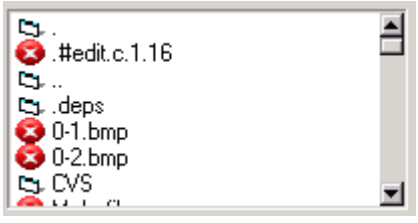

Figure 6.4 List box Control

6.5 Edit Box Control

Edit box provides an important approach for application to get the input data from the users. There are three kinds of edit control correspond to three control control class. These edit controls are: simple edit box (not recommended), single-line edit box, multiple-line edit box. In Figure 6.5, the top

picture illustrates three edit box controls defined by MiniGUI, and the bottom picture shows BIDI edit box, which can be input in left-to-right mode or right-to-left mode. BIDI edit box is provided by MiniGUI V3.0.

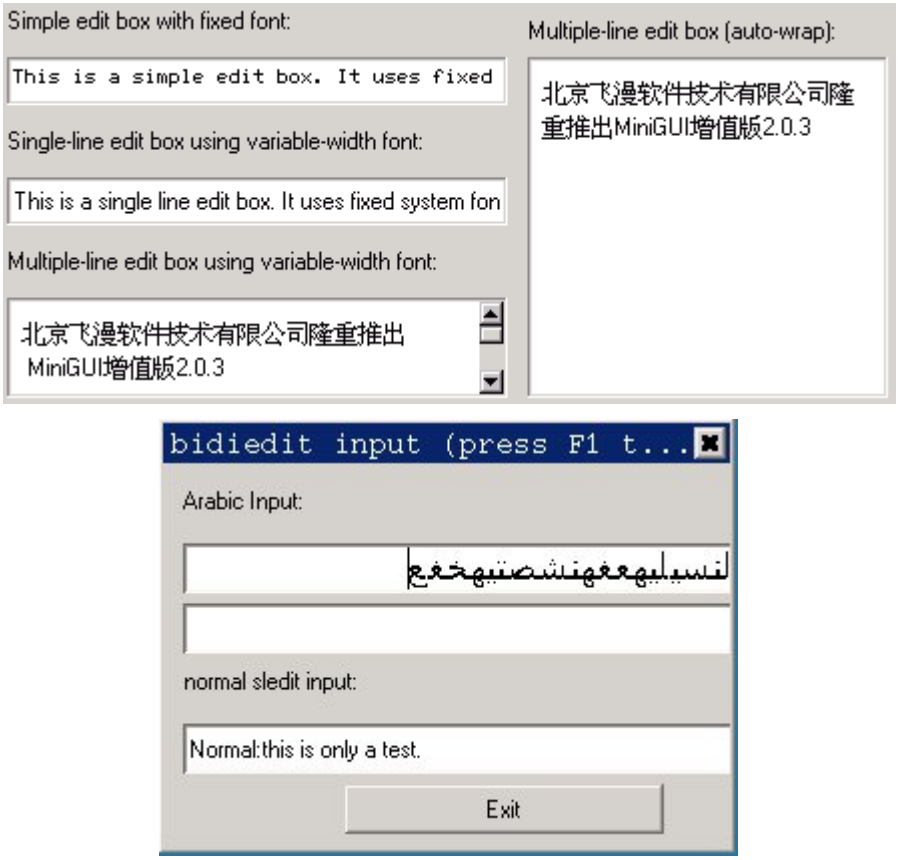

Figure 6.5 Edit boxes Control

6.6 Combo Box Control

In nature, a general combo box is the combination of an edit box and a list box. Users can input data in the edit box or select an item from the options listed in the list box. MiniGUI provides four combo boxes: simple combo box, dropped combo box, rotational combo box, and rotational digital combo box, illustrated in Figure 6.6 and Figure 6.7.

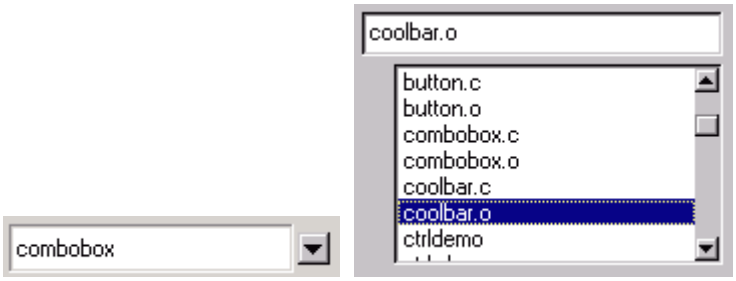

Figure 6.6 Combo Box

Figure 6.7 Rotational Combo Box Control

6.7 Menu Button Control

The menu button looks like a normal push button. The difference is that the menu button has a small rectangle (shown as a down arrow) on the right side of the rectangular button region. When the user clicks the control, a menu would pop up, and when the user clicks an item of the menu with the mouse, the button content will change to the content of this item. Figure 6.8 illustrates the normal status of a menu button control and the effect after the menu pops up; the left is in normal status, the right is the effect after the menu pops up.

Figure 6.8 MenuButton Control

6.8 Progress Bar Control

The progress bar is generally used to prompt the progress of a task for the users, and is frequently used for tasks such as copying file, installing software. Figure 6.9 shows a horizontal progress bar control and Figure 6.10 shows a vertical progress bar control.

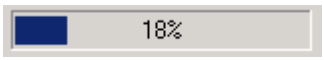

Figure 6.9 Horizontal Progress Bar Control

Figure 6.10 Vertical Progress Bar Control

6.9 Track Bar Control

The track bar is generally used for adjusting brightness, volume, etc. In the situation for adjusting the value in a range, track bar can be used. Figure 6.11 gives an instance of track bar control.

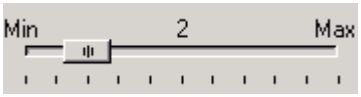

Figure 6.11 Track Bar Control

6.10 Toolbar Control

The use of toolbar is frequently seen in modern GUI applications. MiniGUI prepares the predefined toolbar control for the application as well. In fact, MiniGUI provides three different predefined toolbar control classes, namely TOOLBAR (not recommended), NEWTOOLBAR

(recommended), and COOLBAR (in MiniGUIExt library) control classes. Figure 6.12 shows an instance of NEWTOOLBAR control.

Figure 6.12 New Tool Bar

6.11 Property Sheet Control

The most familiar usage of property sheet is to place the interaction content belonging to different dialog boxes into one dialog box according to their catalogues. This can save space of the dialog box on the one hand, and can make the interaction interface more convenient to use on the other hand. Figure 6.13 illustrates a typical usage of property sheet control.

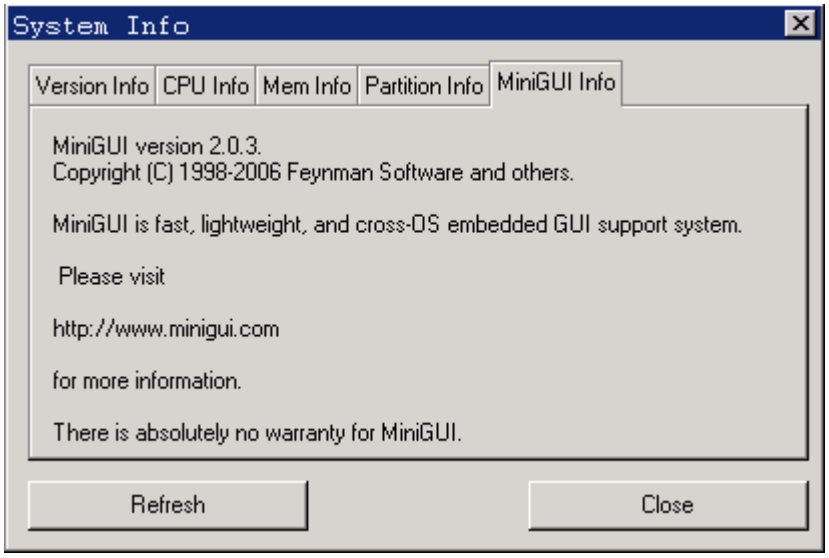

Figure 6.13 Property Sheet Control

6.12 Scroll View Control

The major usage of scroll view is to display and handle some list items. In this aspect, it is similar to the list box or list view control. However, the height of a list item in scroll view can be specified by the user, so different list item can have different height. The most important is that draw of list items in scroll view is completely determined by application. Totally speaking, scroll view is a control easy to be customized, and gives you great freedom. By using scroll view, you can perform much work that list box and list view control cannot do.

MiniGUI V3.0 provides more styles of scrollbar control and can show different style of scrollbar control in different Look and Feel renderer, illustrated in Figure 6.14.

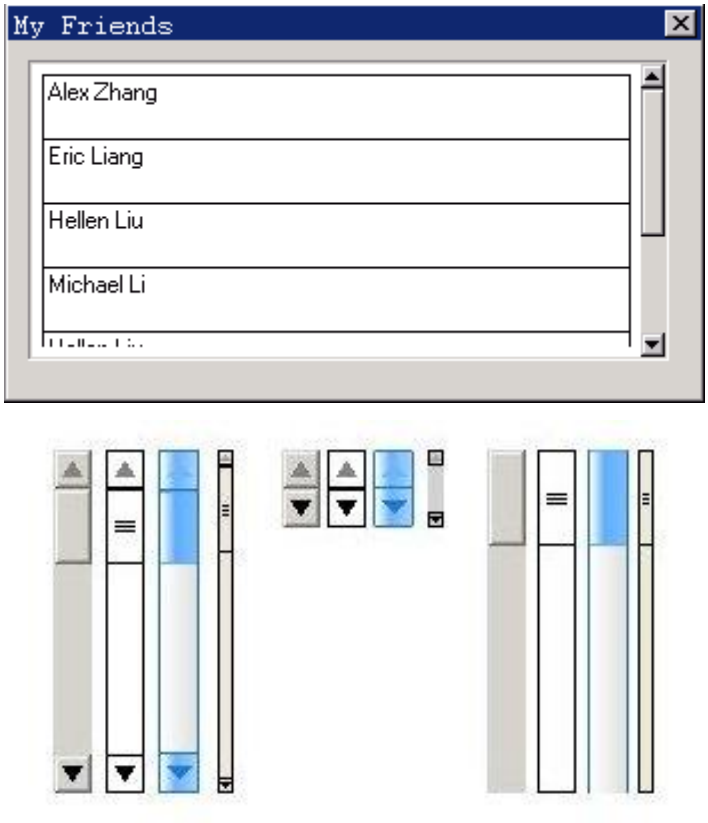

Figure 6.14 Scroll View Control

6.13 Tree View Control

The tree view control displays a hierarchy items in tree form, and each item (sub item) can include one or more child items. Each item or sub item includes the text title and an optional icon, and the user can unfold or fold the sub items by clicking it. The tree view control is fit to represent objects having affiliation relationship, such as file and directory structure, or organization of an institution. Figure 6.15 shows the contents of one book by using a tree view control.

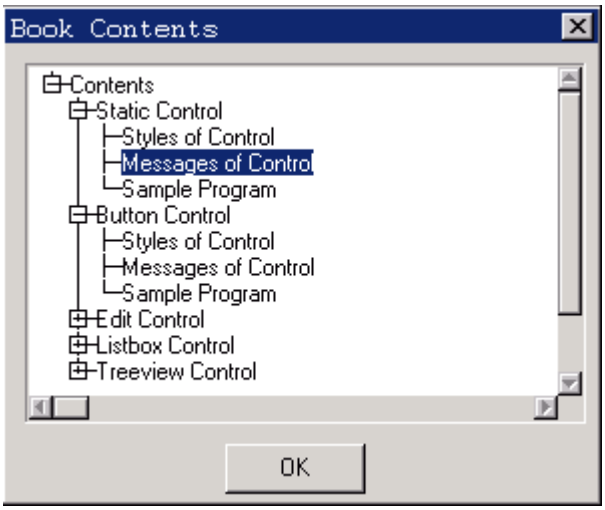

Figure 6.15 Tree View Control

6.14 List View Control

The list view control displays a series of data items (list item) in a table form. Every list item may be comprised of one or more sub items. All same sub items of different item are organized as column. Normally, the content of the list view header expresses the meaning of different column namely different sub item. On appearance, the list view control is a rectangle include list headers and data items. User can adjust the width of column in the list view through dragging the header of list view by mouse. When the content width or height is greater than the list view control, the control will display the content through scroll bar. Figure 6.16 gives an instance of list view control.

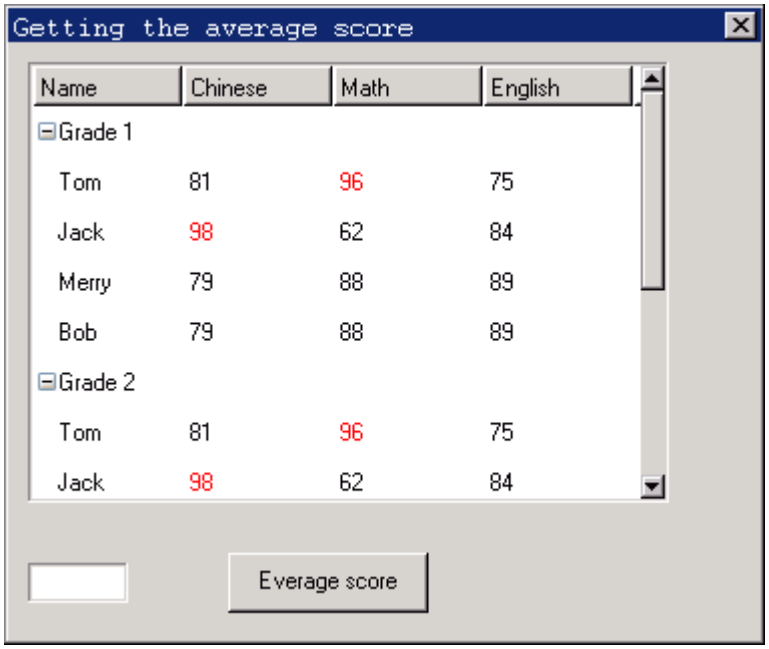

Figure 6.16 List View Control

6.15 Month Calendar Control

The month calendar control provides a user interface similar to a calendar, and makes the user be able to select and set the date conveniently. The application can get or set the date by sending message to a month calendar control. Figure 6.17 illustrates an instance of month calendar control.

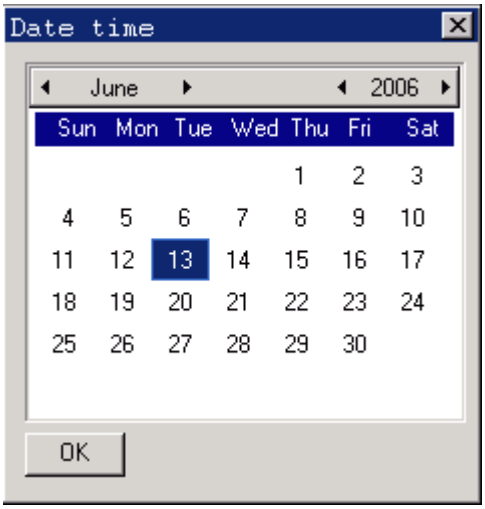

Figure 6.17 Month Calendar Control

6.16 Animation Control

Animation control can be used to display GIF animations, which is very simple and easy to use. Figure 6.18 shows two different frames of a GIF animation in two animation controls.

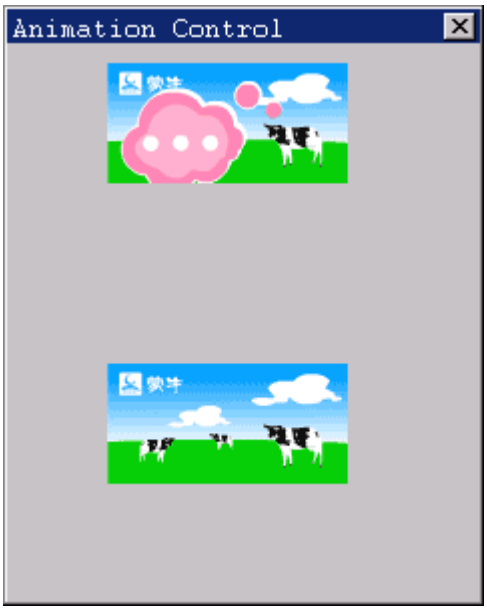

Figure 6.18 Animation Control

6.17 Grid View Control

A grid view control displays a series of data items (cells) in table form, and the contents of every cell are independent to each other. Normally, the header of grid view control(include header of a column and header of a line) expresses the meaning of a column or a line. On appearance, grid view control is a rectangle include headers and data cells. User can adjust the width of column or the height of line through dragging the header by mouse. Figure 6.19 shows an instance of grid view control.

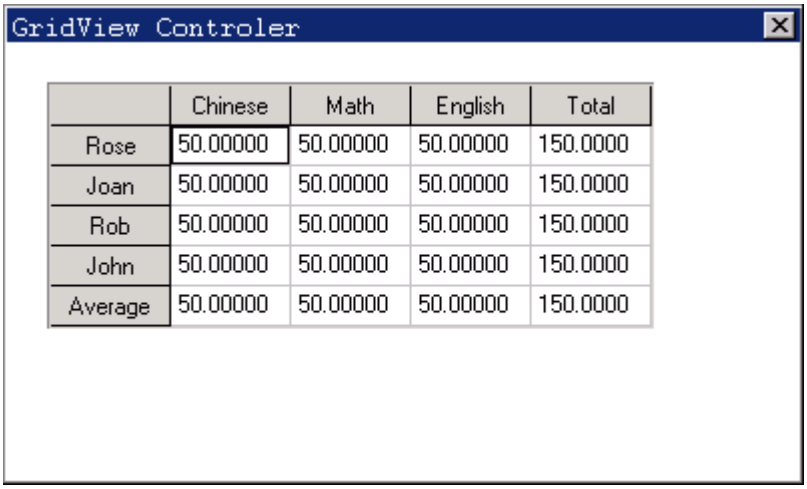

Figure 6.19 Grid View Control

6.18 Icon View Control

An icon view control offers an interface for user to navigate different entries in icon and/or label mode. The icon items will be shown in a scrollable child window. User can select one or more than one item by using keyboard and mouse, and the control will highlight the selected items. The typical use of

icon view control is to be container of desktop icons or to display the files in a directory in thumbnail mode. Figure 6.20 gives an instance of icon view control.

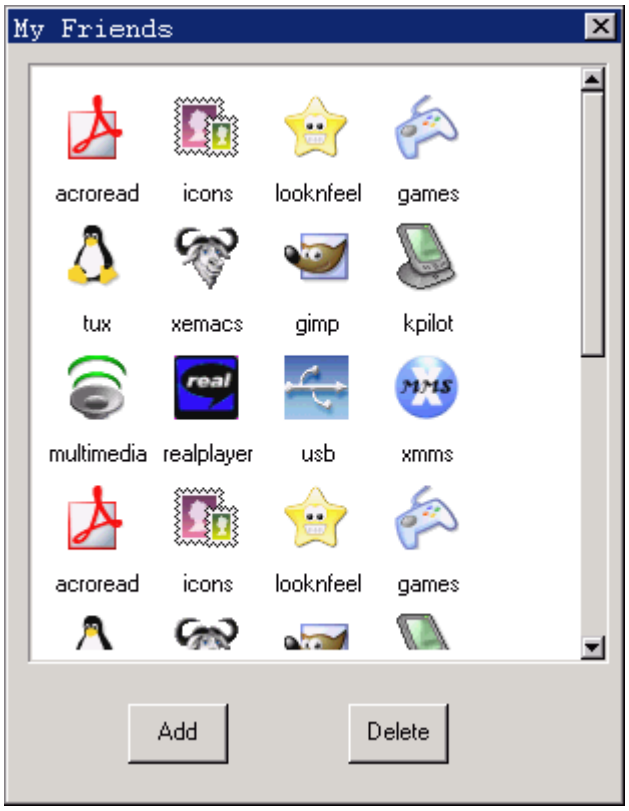

Figure 6.20 Icon View Control

7 Internationalization

MiniGUI supports charsets of ISO8859-1～ISO8859-15, GB2312, GBK, GB18030, BIG5, EUCKR, EUCJP, Shift-JIS, UNICODE, etc. The inner encode of support for these charsets is non-UNIOCDE, so the support consumes fewer resource and fits better for embedded system.

MiniGUI fully supports for UNICODE(a international standard charset), so it is convenient for developer to apply a lot of languages in their application such as English, Chinese, Korean, Japanese and other languages supported by UNICODE. MiniGUI V3.0 newly supports BIDI text, such as Arabic (ISO8859-6), Hebrew (ISO8859-8).

While developing MiniGUI applications, if only need to display words of one charset, for example, simplified Chinese (GB2312), you only need configure MiniGUI to enable GB2312 charset and some fonts for the charset. In this way, you can reduce the resources consumed. If you want to display characters in several charsets at the same time, such as Japanese and Korean, you can enable UNICODE in MiniGUI set font which support UNICODE, for example, TrueType fonts or QPF fonts. In this case, application needs to do nothing but to set logical font with right charset. MiniGUI can convert the certain characters in one charset(such as GB2312, GBK) to UNICODE automatically and select the relevant fonts to render the text. This realization mode is different from traditional mode which uses UNICODE as inner encode. The traditional mode need to translate characters from other charset to UNICODE before the characters being rendered rightly.

8 Components of MiniGUI

To meet the various needs of different embedded devices, FMSoft has developed a number of components based on MiniGUI. Users of these products, components, MiniGUI to expand the functions and can have a good MiniGUI application integration.

8.1 mGp

mGp is a printing component of MiniGUI, which provides a printing function to MiniGUI applications. At present, mGp supports Epson, HP and some other printers. Figure 8.1 shows the setup window of mGp.

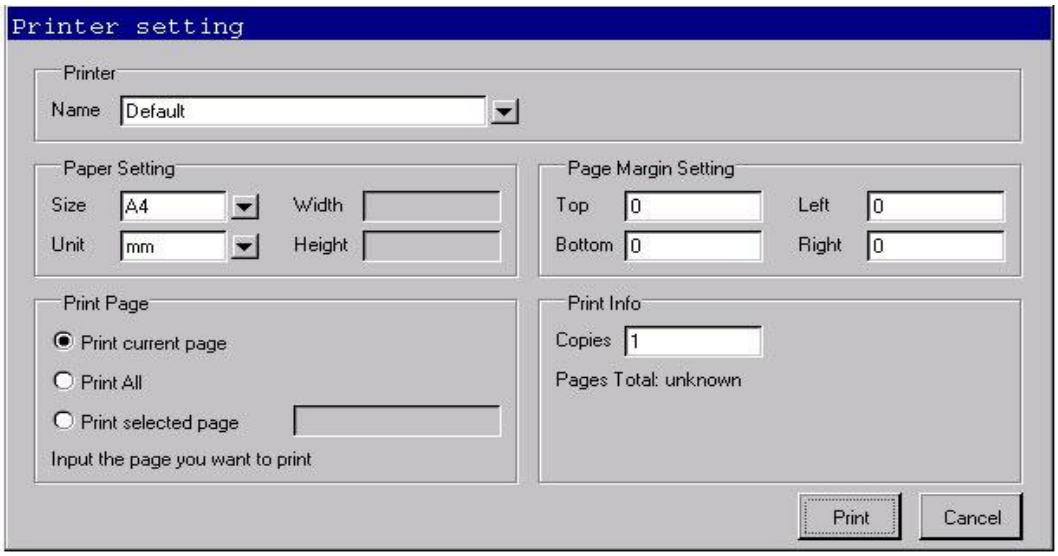

Figure 8.1 The setup window of mGp

8.2 mGi

mGi is a input component of MiniGUI, which provides the framework of soft-keyboard and handwrite input methods. It also supplies an IME container for users to add self-defined IME to it. In addition, you can use self-defined keyboard bitmap for the soft-keyboard and add self-defined translation method to it. Figure 8.2 and Figure 8.3 illustrate the user interfaces of input methods of soft_keyboard and handwrite respectively.

Figure 8.2 mGi: input with soft-keyboard

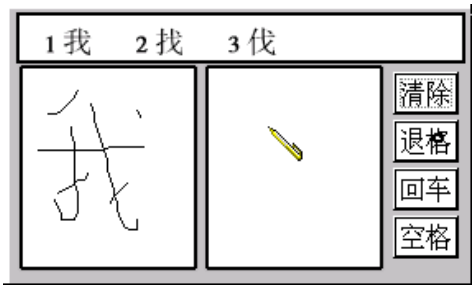

Figure 8.3 mGi: input with hand writing

8.3 mG3d

mG3d is a 3D rendering component of MiniGUI providing 3D interface for applications. Using this component, you can render 3D images, 3D words and 3D scene in your applications. Figure 8.4 shows some 3D objects created by mG3d.

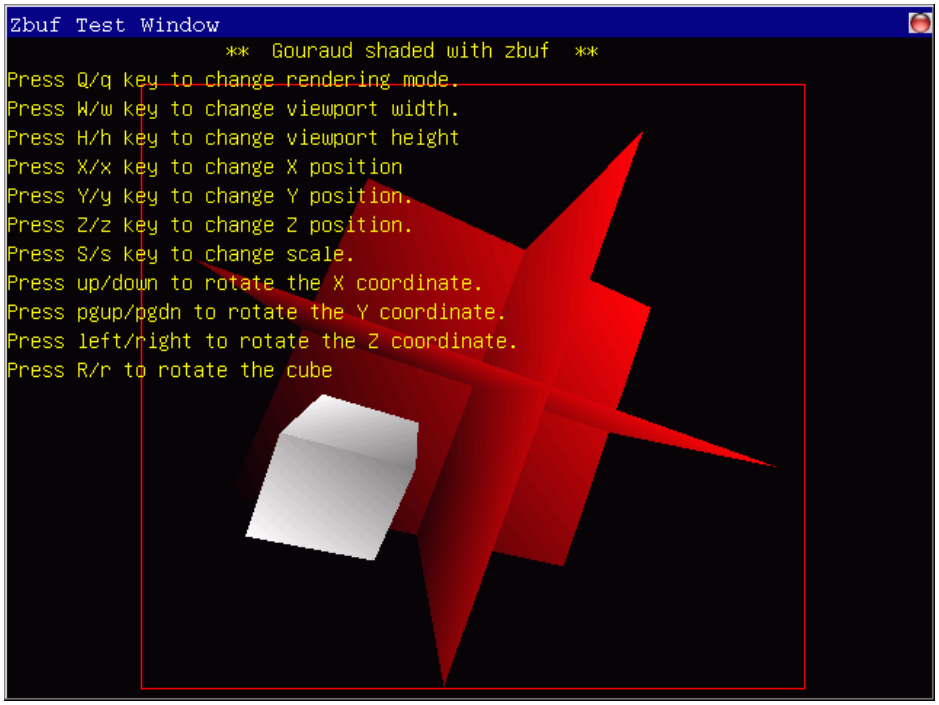

Figure 8.4 mG3d running effect

8.4 mGUtils

mGUtils provides users with several commonly used function templates, so user do not need to implement these functions again. Function templates are listed below:

- ✓ Common file dialog box. This template dialog box has file open, file save, and file save as function. It's appearance has two modes: simple model and PC model. As long as the user set up the FILEDLGDATA data structure, and call the relevant function, he/she can implement the corresponding function. It's easly to use.
- ✓ Color settings dialog box. This template dialog box is absolutely usful when you need to provide color palette for application. Color setting dialog box also has simple model and standard PC model. As long as the user set up the COLORDLGDATA data structure, and then

call the color setting dialog box interface function, user can get a beautiful color setting user interface. It's a piece of cake to realize.

- ✓ Font settings dialog box. This template provides a font settings dialog box same as Windows's font set dialog box. The role of font setting dialog box in the application development is well known to very developer. To realize the font setting dialog box, user only need to set up the FONTDLGDATA data structure at first, and then call the font setting function.
- ✓ Information set dialog box. This template provides a dialog box to show special information. With this template, users do not have to develop a dialog box to show pop-up information. Setting up the INFODLGDATA data structure, then user can call the template function shows the information.

Figure 8.5 shows the color setting dialog box and file dialog box.

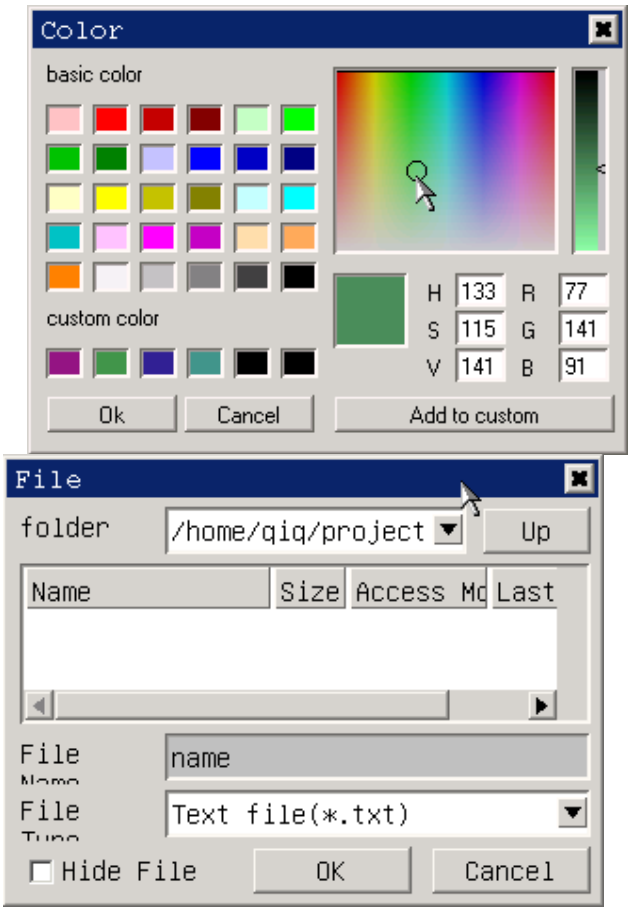

Figure 8.5 mGUtils: Color Setting Dialog Box and File Dialog Box

8.5 mGPlus

mGPlus is an expanding and enhancement to graphical interface of MiniGUI, mainly on the provision of two-dimensional vector graphics and advanced graphics algorithm support, such as path, gradient filling and color composite.

- ✓ Path. The path is comprised of a set of broken lines and curves with strict order. Paths can be used for filling and cutting. By providing support for the path, we can realize many functions

on vector graphics such as drawing, limitless zooming and rotate etc., and also we can provide better support for vector font.

- ✓ Gradient fill. Gradient fill is a kind of filling which uses a painting brush that it's color changes linearly or according to a path to fill in a designated area or a region of a path or a graphic. With gradient filling, we can implement more beautiful and more stereo control. At present, MiniGUI provides two modes of gradient filling: linear gradient filling and radial gradient filling.
- ✓ Color composition: In today, product packaging has been attached great importance. Every application developer hopes to develop a very pretty sophisticated user interface to obtain user's first good impression. Color composition can be a potent weapon in this regard. it can make an ever-changing composition between pictures, giving you a amazing user interface effect. MiniGUI 3.0 has implemented 12 kinds of patterns of color composition.

Figure 8.6, 8.7 illustrates path, color composition, color gradient filling separately.

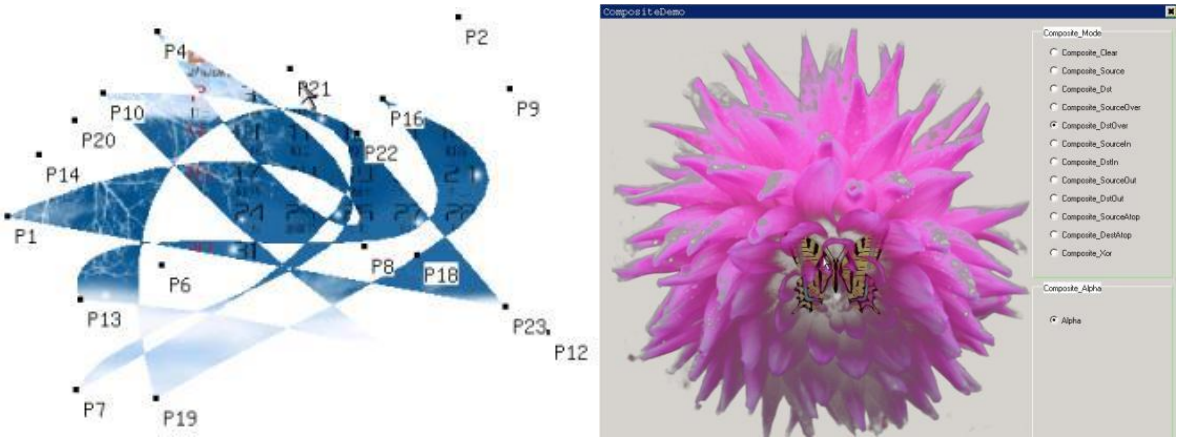

Figure 8.6 path and color composite

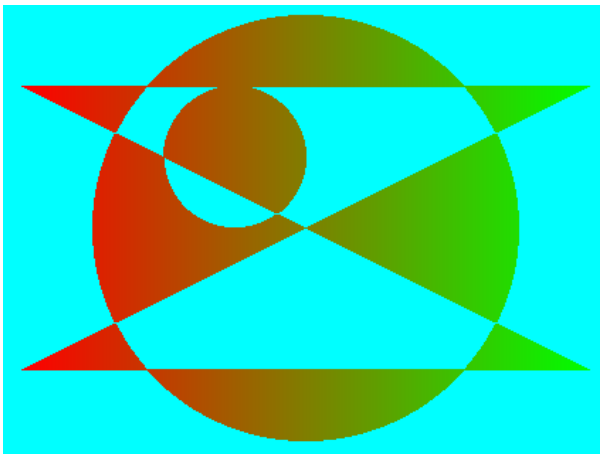

Figure 8.7 color gradient filling

8.6 mGNCS

In the development of miniStudio, in order to realize the design of visual graphic interface, based on MiniGUI existing interfaces, FMSoft developed a new control set. The new control set introduced by miniStudio is developed on the foundation of the original MiniGUI control set. In order to

distinguish it from MiniGUI intrinsic control set, it is referred to as “new control set”, and we release it as a new MiniGUI component: mGNCS.

Features of the new control set are as follows:

- ✓ Object-oriented programming idea is adopted to rearrange the inheritance relations between the controls. C language is used to realize concepts similar to C++ class, and C language interface is provided externally. In this respect, the new control set introduced by mStudio is similar to the control set of Gtk+.
- ✓ Interfaces and styles of the controls are standardized, enabling all controls to access with uniform interface.
- ✓ Based on MiniGUI 3.0 look and feel renderer, the concept of look and feel renderer is further expanded. Each control in the new control set has its own look and feel renderer and exclusive renderer interface of each control is defined according to the inheritance rule of the controls, enabling them to change along with the changes of controls.

Through developing the new control set, we settle the deficiencies of MiniGUI intrinsic control set in the following aspects simultaneously:

- ✓ The interfaces are not uniform and standardized;
- ✓ Operating controls through messages, it is difficult to maintain codes and easy to make mistakes;
- ✓ Hierarchical relation of intrinsic control is unclear, there are many repeated codes, and it is inconvenient to customize and extend control functions;
- ✓ Drawing efficiency of the intrinsic control is low, and twinkling problem is serious when running on the development board.

You can mix mGNCS controls and MiniGUI original controls in your applications. ***However, we strongly encourage and recommend that you use mGNCS as the application framework for your new MiniGUI application.***

8.7 mGEff

mGEff provides an animation framework for MiniGUI applications. mGEff provides a lot of stable and efficient effectors, which can be used to implement the animations like flipping, zooming, scrolling, and so on. Furthermore, by using the double buffer provided by MiniGUI 3.0, mGEff provides the animation APIs for MiniGUI main windows.

9 MiniGUI Resources

You can visit the following web site to download the official release tarballs of MiniGUI and its components, as well as the samples and some open source software run on MiniGUI:

```
http://www.minigui.com
```

FMSoft releases MiniGUI and MiniGUI components under GPL on GitHub:

```
https://github.com/VincentWei
```

You could get the latest MiniGUI documents at the following link:

```
http://wiki.minigui.com
```

10 Licensing Policy of GPL'd MiniGUI Versions

FMSoft releases some versions of MiniGUI under GPL (General Public License). So any application linking to GPL'd MiniGUI must follow GPL. If you cannot accept GPL or don't want accept GPL, you need to be licensed from FMSoft.

10.1 Free Use for Those Who Are 100% GPL

If your application is licensed under GPL, you need not to be authorized by Feynman. Any one is welcome to copy, modify and release MiniGUI following GPL. For doing this, you do not need a separate signed agreement with FMSoft, because the GPL text is sufficient. But you also need to pay a attention to that Feynman will not give any kinds of warranty or support service in this case.

10.2 Free Use for Those Who Never Copy, Modify or Distribute

As long as you never distribute (internally or externally) the MiniGUI in any way, you are free to use it for powering your application, irrespective of whether your application is under GPL license or not. Of course, there is on warranty and support service from Feynman.

More specifically:

- ✓ Modifying - You are allowed to modify MiniGUI source code any way you like. If you distribute the modified version, all changes, all interface code and all code that connects directly or indirectly to the interface code fall under GPL.
- ✓ Copying - You are allowed to copy MiniGUI binaries and source code, but when you do so, the copies will fall under the GPL.

10.3 Commercial Use for Everyone Else

If your application is not licensed under GPL and you intend to distribute MiniGUI software or function library (be that internally or externally), you must first obtain a commercial license to the MiniGUI software in question.

More specifically:

- ✓ If you link MiniGUI in your non-GPL application, you need a commercial license for the MiniGUI library.

- ✓ If you use MiniGUI library within your organization and you don't want to risk it falling under the GPL license, you are welcome to purchase a commercial license.
- ✓ Many users may choose the commercial license simply because in this way FMSoft takes responsibility for its products. Under the GPL license, there are no warranties or representations from the developer (i.e. from FMSoft).

Mode of commercial authorization, price and way to purchase are shown in the following web:

```
http://www.minigui.com
```

For commercial application software developers, we provide value-added release version of MiniGUI which is used to support non-GPL application during developing phase of software product.

11 Contact Us

Detailed information is available on the following website for those who are interested in MiniGUI:

```
http://www.minigui.com
```

For complete products information of FMSoft, please refer to:

```
http://www.fmsoft.cn
```

About products by FMSoft and problem of MiniGUI licensing, please sent email to:

```
consult@minigui.com
sales@minigui.com
```