MiniGUI Technology White Paper

Version 2.0 (revised edition 4) for MiniGUI V2.0.4/1.6.10
Copyright © 2002~2007, Beijing Feynman Software Technology Co., Ltd.
Latest revision: 2007/07/10

You are permitted to copy and redistribute this document,
provided that the document and this announcement are clean and complete.

Contents

0 Foreword

With the development and widely use of high-end consumer electronic products (smart phones, portable media players, etc.), more and more developers pay their attention to a new system software component besides the operating system: embedded graphics user interface (GUI) support system.

All man-machine interaction oriented embedded products must take the graphics and/or text output into account. With the mobile phone as an example, multimedia messages, WAP and the coming 3G applications need support from a full-featured embedded GUI system, in order to run the multimedia applications, J2ME applications, and 3D applications. However, the development of GUI applications on an embedded system is not as easy as on a PC platform. The first reason is the limited resources (low-speed processor, small dynamic and static storage spaces, etc.) of such devices; the second reason is the weak support from the underlying operating system. Therefore, there are many technical challenges to provide a full-featured and suitable GUI support system for embedded products with different hardware configurations and various real-time embedded operating systems.

Embedded software suppliers including many RTOS vendors have been providing their proprietary graphics solutions for embedded systems. Consequently, there are many embedded graphics systems with different features, and the wide use of Linux in embedded systems promotes this situation. For these reasons, it is hard for a device vendor to choose a suitable embedded graphics support system for his embedded product.

Fortunately, Beijing Feynman Software Technology Co., Ltd (Feynman Software) has nine years experience in the development of cross-operating-system GUI support system. The core product of Feynman Software, MiniGUI, has become the de-facto industrial standard in the embedded graphics field.

This white paper introduces the cross-operating-system GUI support system, MiniGUI, developed by Feynman Software.

1 Introduction

1.1 What Is MiniGUI

MiniGUI (http://www.minigui.com), developed by Feynman Software, is one of the world famous free software projects. MiniGUI aims to provide a lightweight graphics user interface (GUI) support system for real-time embedded systems. 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, and so on.

At the moment, 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 tested include Intel x86, ARM (ARM7/ARM9/StrongARM/xScale), PowerPC, MIPS, and M68k (DragonBall/ColdFire). The newly released 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.

MiniGUI is "a cross-operating-system graphics user interface support system for embedded devices", and "an embedded graphics middleware". So far, MiniGUI has gained recognition from 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. In August of 2005, KSP from Korea has officially become the agent of Feynman products in Korea. With the support of the leading enterprises, MiniGUI has become the de-facto standard in the field of embedded graphics middleware.

Feynman Software not only releases some versions of MiniGUI under GPL, but also provides software products for commercial customers. This white paper introduces the features and advantages of MiniGUI version 2.0.4/1.6.10 in detail.

1.2 The Origin and Evolution of MiniGUI

Nine 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 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 Feynman Software, Wei Yongming, began to develop MiniGUI, and applied it in a computerized numerical control (CNC) system. In March 2003, 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 Feynman Software 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.

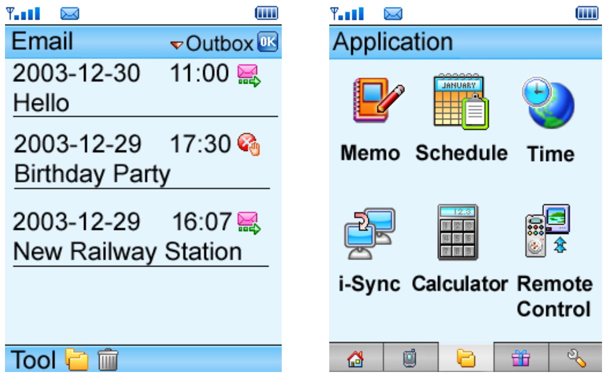
In August 2004, the most famous telecommunication equipment supplier (HUAWEI) in China used MiniGUI as the platform on STBs, hand-held devices, etc. In January 2005, the main TD-SCDMA standard maker (Datang Mobile) chose MiniGUI as the MMI solution[1] for its TD-SCDMA mobile phones.

At present, the latest release of MiniGUI is version 2.0.4/1.6.10[2], which supports Linux/uClinux, eCos, uC/OS-II, VxWorks, pSOS, ThreadX, Nucleus, OSE and Win32 platform. Feynman Software 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 (including 2.5G/3G smart phones, feature phones, WiFi phones, portable media players, PDAs)
  Figure 1.1 illustrates the MiniGUI-based application interfaces for a WiFi phone. This device is based on the eCos operating system.



2003-12-30 11:00
Hello

2003-12-29 17:30
Birthday Party

2003-12-29 16:07
New Railway Station

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 information terminal developed by Feynman Software.

[1] It employs another product of Feynman Software: Fhas.

[2] MiniGUI V2.0.x provides support for multi-process-based operating systems, like Linux; MiniGUI v1.6.x provides support for traditional real-time embedded operating systems, which are multi-thread- or multi-task- based.

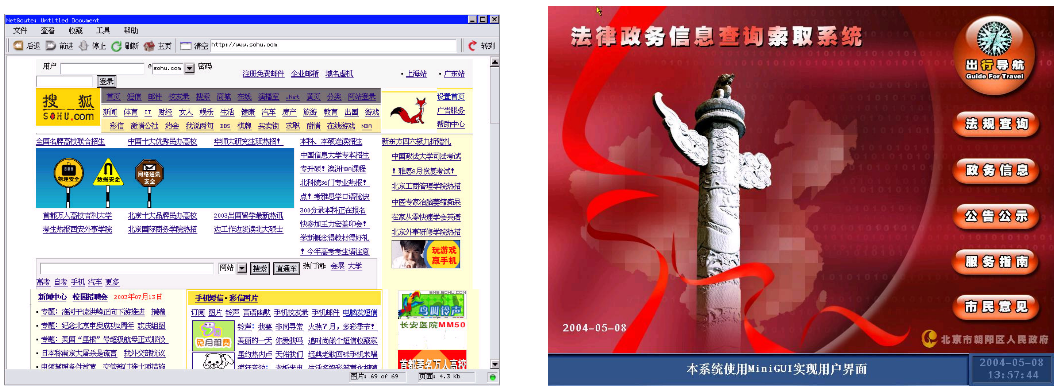


Figure 1.2 Typical application of MiniGUI: digital media devices and STBs

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

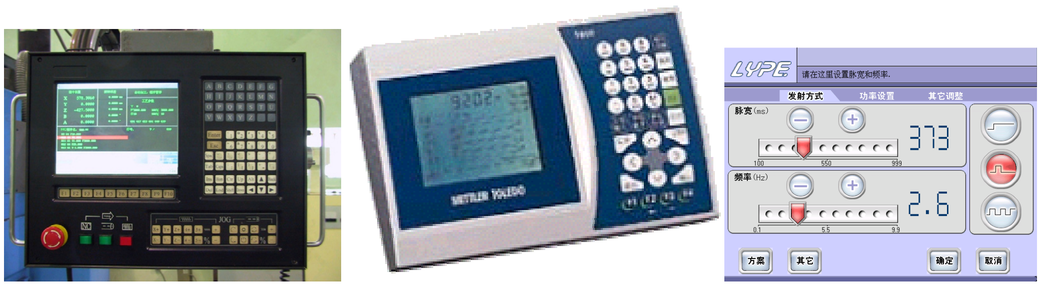


Figure 1.3 Typical application of MiniGUI: industry instruments and control systems

2 Features and Advantages of MiniGUI

2.1 Technical Features of MiniGUI

MiniGUI is a complete and self-contained embedded graphics support system, which is designed and optimized for embedded systems. As a middleware between operating system and applications, MiniGUI hides the diversities of different underlying OSes and hardware platforms, and provides identical APIs for top-level applications. The main technical features of MiniGUI are as follow:

1) Support for many different realtime embedded operating systems, including Linux, uClinux, eCos, uC/OS-II, VxWorks, pSOS, ThreadX, Nucleus, and OSE. SDK on Win32 platform is available also; it can facilitate the development and debugging of embedded applications.
2) Support for three runtime modes. You can configure and compile MiniGUI as one of three runtime modes: MiniGUI-Threads, MiniGUI-Processes[3], and MiniGUI-Standalone.
3) 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.
4) Mature multi-window and messaging mechanism.
5) Commonly used controls (widgets), including static label, button, single-line and multi-line edit boxes, list box, combo box, progress bar, property sheet, toolbar, track bar, tree view, list view, month calendar, grid view, animation, icon view, and so on.
6) Support for dialog box and message box.
7) Support for other GUI elements, including menu, acceleration key, caret, timer, etc.
8) Support for skin UI. You can use skin APIs to build your dashy user interfaces.
9) Support low end devices such as single color LCD and high end devices, and also support specific video devices (such as YUV device) by using graphics engine, which runs under the graphics abstract layer of MiniGUI.
10) 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 abstract layer of MiniGUI.
11) Support for Windows resource files, for example, Windows bitmap, icon, cursor, etc.
12) 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).
13) Support for multiple character sets and multiple font types. At present, what are

[3] This runtime mode is provided by MiniGUI V2.0.x for Linux; On Linux, MiniGUI V1.6.8 and earlier version provide another runtime mode called ‘MiniGUI-Lite’, which is a simplified multi-process runtime mode.

supported include such character sets as ISO8859-1 ~ ISO8859-15, GB2312, BIG5, EUCKR, SHIFT-JIS, UNICODE (UTF-8 and UTF-16 encondings); bitmap fonts such as Qt Pre-rendered fonts, and vector fonts such as TrueType as well as Adobe Type1. MiniGUI also provides auto zoom in function, anti-alias function for TV and other video equipments.

14) Support for multiple keyboard layouts, including American PC, French, German, Italian, Spanish, and so on.
15) Input method interface to support special embedded devices. Moreover, input methods for Simplified Chinese are built-in.
16) Special support for embedded systems, including the common I/O operations, byte-orders related functions, touch screen calibration interface, and so on.
17) 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.

Secondly, many technical innovations have been made in the course of nine years of development. These technical innovations enable MiniGUI to become more suitable for realtime embedded systems and exhibit agreeable flexibility, applicable to various high-end or low-end embedded systems including handheld devices, set-top-boxes, game terminals and so forth. These technical innovations include:

1) Graphics and input abstract layer (GAL and IAL) that, placing no influence on API of the top-level, greatly facilitate the porting and debugging of MiniGUI itself as well as applications. At present, MiniGUI has been proven to be capable of running smoothly on the embedded systems based on i386, ARM (44B0, MX1, StrongARM, xScale), MIPS, PowerPC, DragonBall, ColdFire, etc. By using the graphics and input abstract layer, we also could implement some software engines; the "Random" IAL engine simulating the real input can be used to do auto-test of MiniGUI and its applications. For another example, we could support output devices in YUV color space like TV set by using the "Shadow" GAL engine. This engine can also provide support for those graphics chipsets whose frame buffer cannot be accessed directly, provide support for the devices with a color depth less than 8-bpp (bits per pixel), and implement the screen rotation.
2) An optimized architecture to support multiple charsets and multiple fonts. It is very easy to add a new font type and/or a new charset support in MiniGUI. Support for various charsets is achieved by creating specific logical fonts in different charsets and/or encodings such as GB2312, BIG5, EUCKR, Shift-JIS, UNICODE (UTF-8 and UTF-16 encodings), etc. In a single MiniGUI application, it is very easy to display characters in different languages. Different from the traditional multi-charset implementations, which are achieved by UNICODE, MiniGUI's implementation consumes fewer resources and has more flexibility.
3) Three runtime modes. Different from the general-purpose operating systems 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, those runtime environments are entirely different from Linux. We can configure and compile MiniGUI into three runtime modes for different operating systems: MiniGUI-Threads, MiniGUI-Processes, and MiniGUI-Standalone.

At last, in MiniGUI V2.0.x, we not only provide full support for Linux, which has a multi-process environment by the runtime mode of MiniGUI-Processes, but also keep the concept of layer. 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. MiniGUI-Processes can run very well on high-end embedded devices.

The enhancement and improvement of the new version of MiniGUI, especially the full-featured multi-process support in version 2.0.x, will remarkably drive applications of MiniGUI. The new version of MiniGUI will make it easy to support multi-media applications, game applications, and other advanced and complex embedded applications.

2.2 Advantages of MiniGUI

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

1) 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.

2) 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.

3) 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 following web page:

```
http://www.minigui.com/project/index.shtml
```

4) 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 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.

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. MiniGUI SDK for Win32 can facilitate the development and debugging of embedded applications. At the same time, MiniGUI provides identical APIs on different operating systems.

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, M68K (DragonBall/ColdFire), FRV, 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 realtime operating systems, full-featured MiniGUI only uses 1MB static memory.

For the detail footprint of MiniGUI, please refer to *MiniGUI Data Sheet* V2.0-4.

4 Software Architecture of MiniGUI

4.1 Software Architecture of Embedded Systems Based on MiniGUI

The relationship between MiniGUI and realtime operating systems is illustrated in Figure 4.1. An application on MiniGUI can implement its functions by calling APIs in ANSI C library and MiniGUI libraries. The portable layer of MiniGUI hides the details of underlying hardware and operating systems, and the applications need not to take care of the output and input devices.

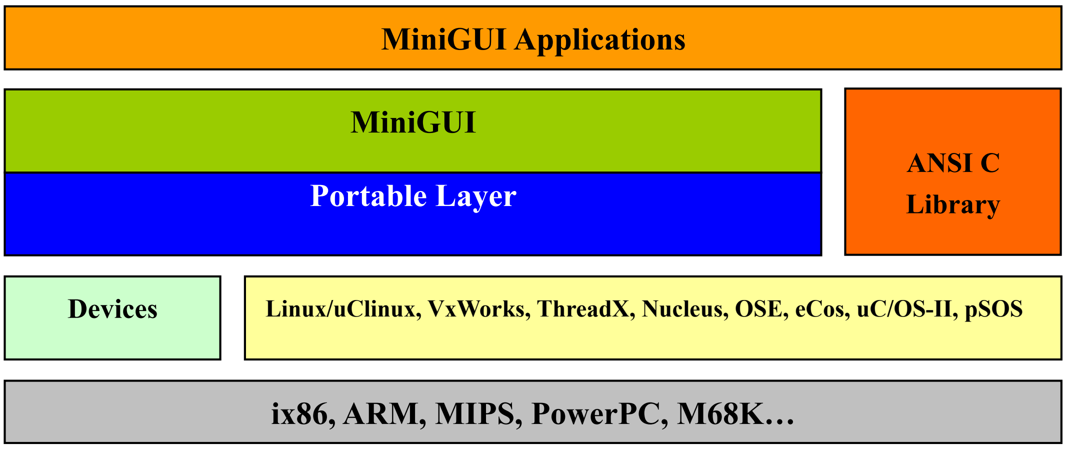

MiniGUI Applications
MiniGUI
Portable Layer
ANSI C Library
Devices
Linux/uClinux, VxWorks, ThreadX, Nucleus, OSE, eCos, uC/OS-II, pSOS
ix86, ARM, MIPS, PowerPC, M68K…

Figure 4.1 MiniGUI and realtime operating systems

Besides, the special concept of MiniGUI runtime mode provides convenience to run MiniGUI on different operating systems.

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. MiniGUI-Threads is fit for some real time systems on Linux/uClinux, eCos, uC/OS-II, and VxWorks.
- **MiniGUI-Processes**[4]: In opposition to MiniGUI-Threads, a program running on MiniGUI-Processes is an independent process, which can also create multiple

[4] 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.

windows. MiniGUI-Processes are fit for full-featured UNIX-like operating systems, such as Linux.

- **MiniGUI-Standalone:** This is a single process/thread/task runtime mode of MiniGUI. This mode is useful for some systems that lack PThread support, like some buggy uClinux systems.

MiniGUI can almost run on all operating systems[5] 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 like Linux/UNIX. Moreover, MiniGUI can run on only 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.

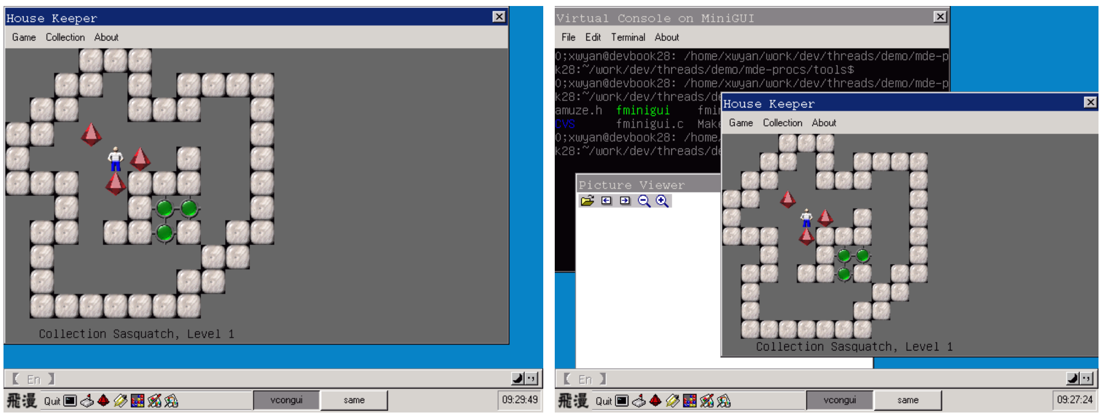
4.2.1 The Runtime Mode MiniGUI-Processes

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. A window created by a client of MiniGUI-Lite is not a global object, i.e., a client does not know the windows created by others. However, windows created by MiniGUI-Processes are all global objects, and windows created by MiniGUI-Processes can clip each other. Figure 4.2 gives the screenshots of MiniGUI-Lite runtime mode of MiniGUI V1.6.x and MiniGUI-Processes runtime mode of MiniGUI 2.0.x.

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



```
0;xwyan@devbook28: /home/xwyan/work/dev/threads/demo/mde-p
k28:~/work/dev/threads/demo/mde-procs/tools$
0;xwyan@devbook28: /home/xwyan/work/dev/threads/demo/mde-p
k28:~/work/dev/threads/de
amuze.h   fminigui    fmi
CVS       fminigui.c  Make
0;xwyan@devbook28: /home
k28:~/work/dev/threads/de
```

Figure 4.2 MiniGUI-Lite runtime mode of MiniGUI V1.6.x and MiniGUI-Processes runtime mode of MiniGUI 2.0.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 extend its application fields in high-end embedded devices, but also have an application in desktop environment.

4.2.2 Operating Systems and MiniGUI Runtime Modes

Table 4.1 illustrates the runtime mode(s) supported by MiniGUI V2.0.x and V1.6.x on various operating systems.

Table 4.1 Operating Systems and MiniGUI Runtime Modes

Operating System	MiniGUI Version	Runtime Mode(s) Supported
Linux	MiniGUI V2.0.x	MiniGUI-Processes MiniGUI-Threads MiniGUI-Standalone
uClinux	MiniGUI V1.6.x	MiniGUI-Threads MiniGUI-Standalone
VxWorks	MiniGUI V1.6.x	MiniGUI-Threads
ThreadX	MiniGUI V1.6.x	MiniGUI-Threads
Nucleus	MiniGUI V1.6.x	MiniGUI-Threads
OSE	MiniGUI V1.6.x	MiniGUI-Threads
eCos	MiniGUI V1.6.x	MiniGUI-Threads
uC/OS-II	MiniGUI V1.6.x	MiniGUI-Threads
pSOS	MiniGUI V1.6.x	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 just like Win32's. The Figure 4.3 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.

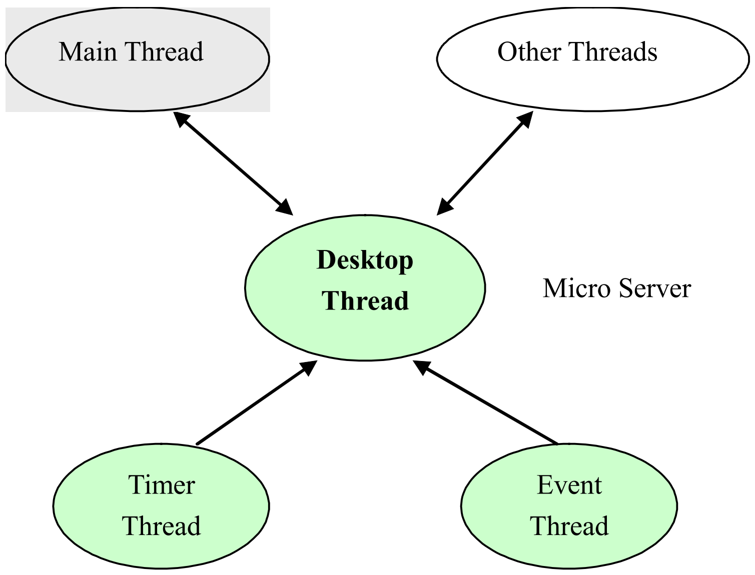


Figure 4.3 Communication mechanism of MiniGUI-Threads runtime mode

Comparing MiniGUI-Threads, the messages transferring mode of MiniGUI-Processes is implemented by UNIX socket. The communication mechanism is illustrated by Figure 4.4.

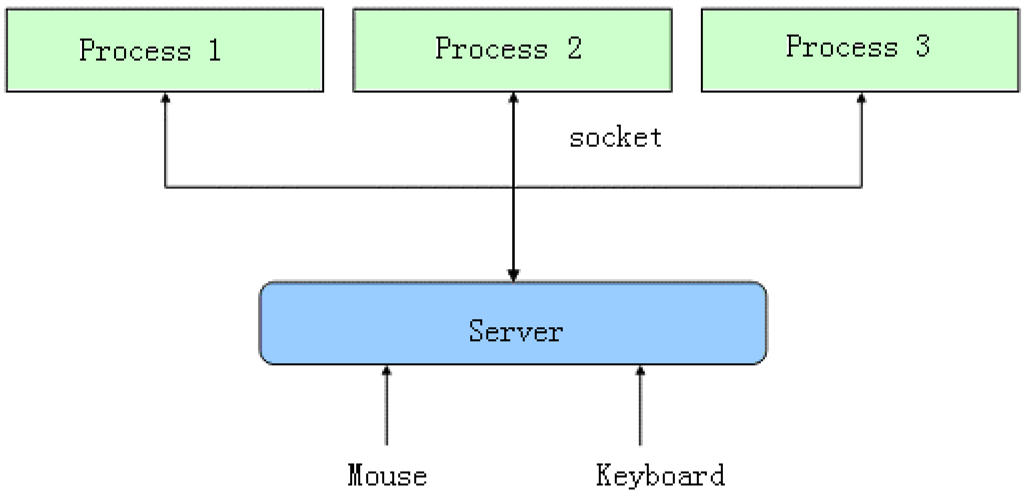


Figure 4.4 Communication mechanism of MiniGUI-Processes runtime mode

4.5 Font

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 as well as 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.

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 Method

MiniGUI supports for input method for multi-byte languages like Chinese, Korean, and Japanese. And the input method for simplified Chinese (GB2312) has been built-in.

5 Development Environments

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, which is 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.

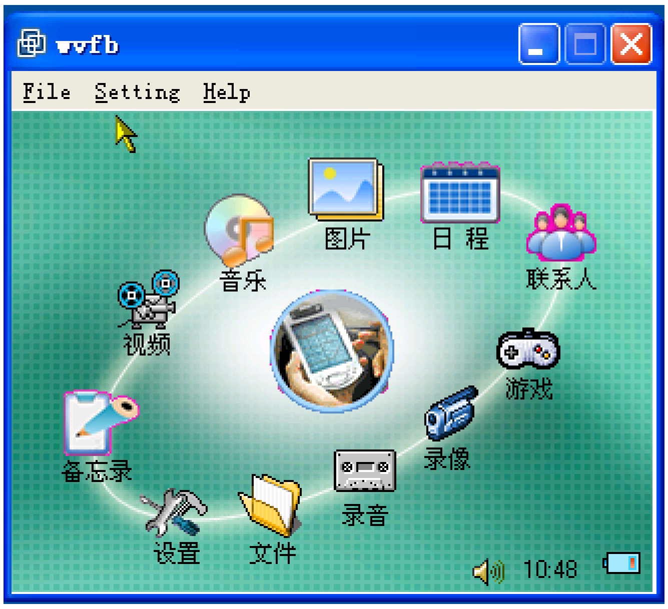


Figure 5.1 A MiniGUI application running on wvfb simulator

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.

```
#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;

#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;

    hMainWnd = CreateMainWindow (&CreateInfo);

    if (hMainWnd == HWND_INVALID)
        return -1;

    ShowWindow(hMainWnd, SW_SHOWNORMAL);

    while (GetMessage(&Msg, hMainWnd)) {
        TranslateMessage(&Msg);
        DispatchMessage(&Msg);
    }

    MainWindowThreadCleanup (hMainWnd);

    return 0;
}
```

The Figure 6.1 shows the window created by the "Hello world" program.

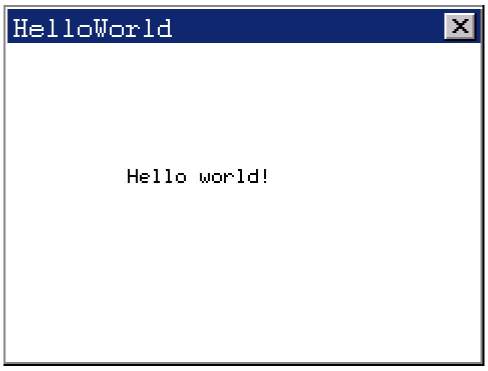


Figure 6.1 The window created by "Hello world" program

You can develop MiniGUI applications by using the GDI APIs of MiniGUI, create a main window and render graphics and fonts into the main window. However, you can also use the built-in controls of MiniGUI to build your MiniGUI applications in a fast way. 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.

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.

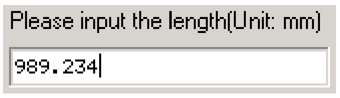


Figure 6.2 Static control (used as a label of other control)

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.

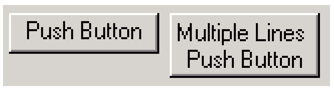


Figure 6.3 Button control

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.

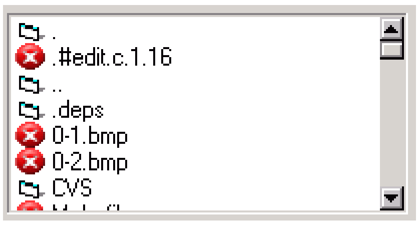
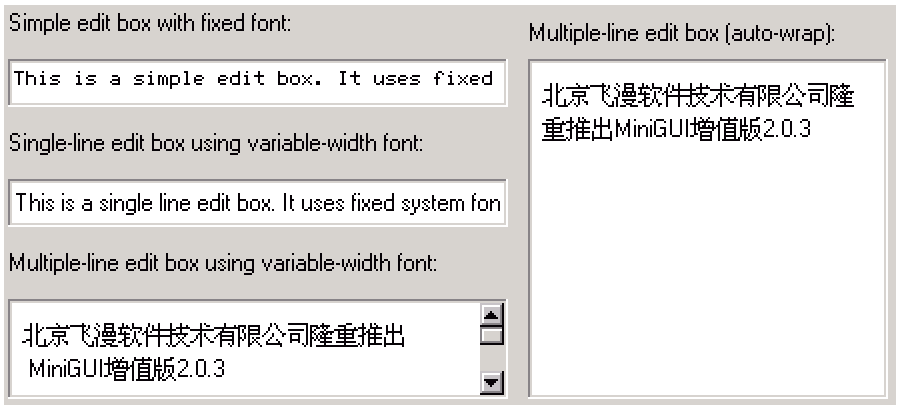

.
.#edit.c.1.16
..
.deps
0-1.bmp
0-2.bmp
CVS

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. Figure 6.5 illustrates three edit box controls defined by MiniGUI: simple edit box (not recommended), single-line edit box, and multiple-line edit box.


Simple edit box with fixed font:

This is a simple edit box. It uses fixed

Single-line edit box using variable-width font:

This is a single line edit box. It uses fixed system fon

Multiple-line edit box using variable-width font:

北京飞漫软件技术有限公司隆重推出
MiniGUI增值版2.0.3

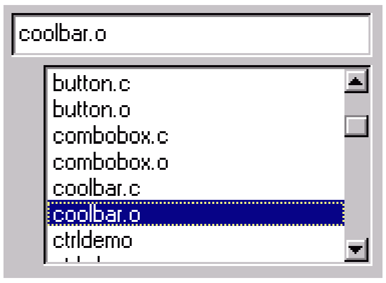
Multiple-line edit box (auto-wrap):

北京飞漫软件技术有限公司隆
重推出MiniGUI增值版2.0.3

Figure 6.5 Edit boxes

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. Figure 6.6 shows the combo box control of MiniGUI.


coolbar.o

button.c
button.o
combobox.c
combobox.o
coolbar.c
coolbar.o
ctrldemo

Figure 6.6 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 6.7 illustrates the normal status of a menu button control and the effect after the menu pops up.

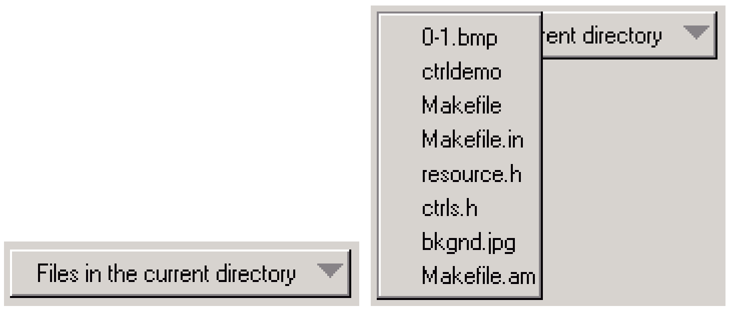


Figure 6.7 MenuButton control (The left is in normal status, the right is the effect after the menu pops up)

6.8 Progress Bar Control

The progress bar is generally used to prompt the progress of a task for the user, and is frequently used for tasks such as copying file, installing software. Figure 6.8 shows a horizontal progress bar control and a vertical progress bar control.

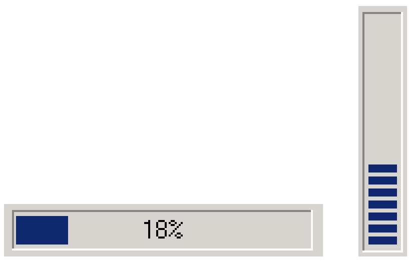


Figure 6.8 Progress bar control

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 6.9 gives an instance of track bar control.

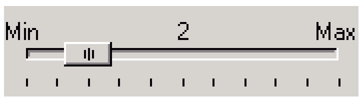


Figure 6.9 Track bar control

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 6.10 shows an instance of NEWTOOLBAR control.

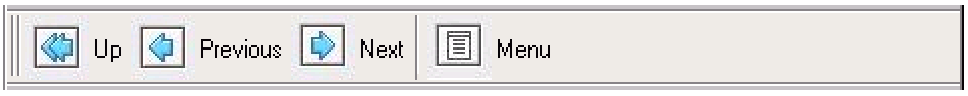
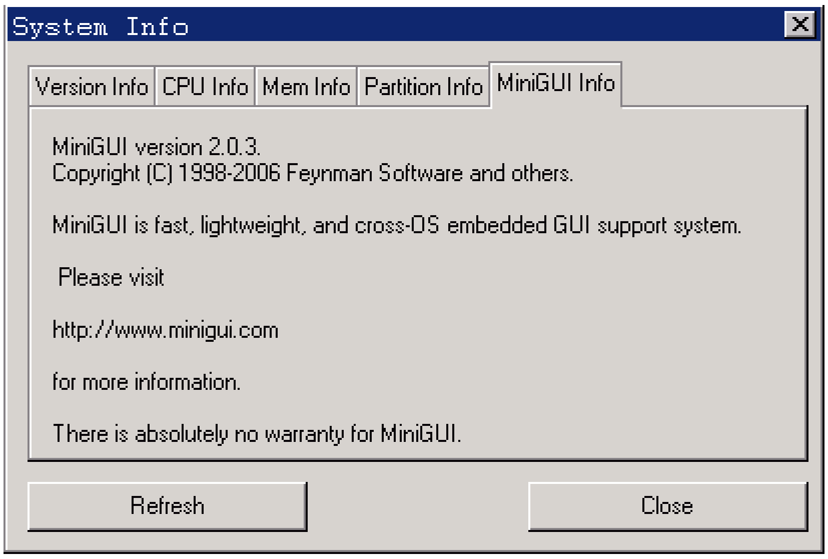


Figure 6.10 New tool bar control

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 6.11 illustrates a typical usage of property sheet control.



MiniGUI version 2.0.3.
Copyright (C) 1998-2006 Feynman Software and others.

MiniGUI is fast, lightweight, and cross-OS embedded GUI support system.

Please visit

http://www.minigui.com

for more information.

There is absolutely no warranty for MiniGUI.

Figure 6.11 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. Figure 6.12 shows an instance of scroll view control.

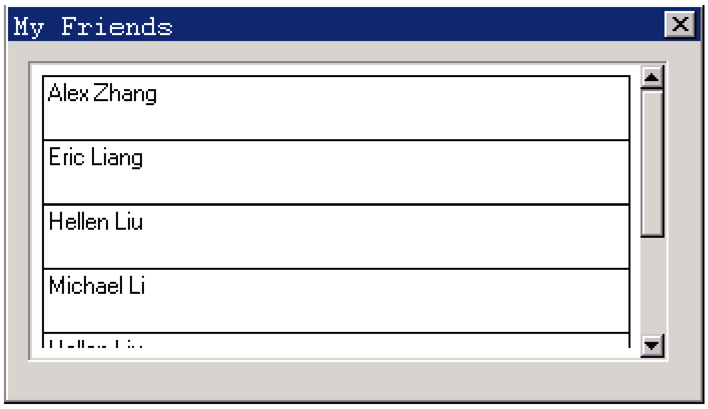


Figure 6.12 Scroll view control

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 6.13 shows the contents of one book by using a tree view control.

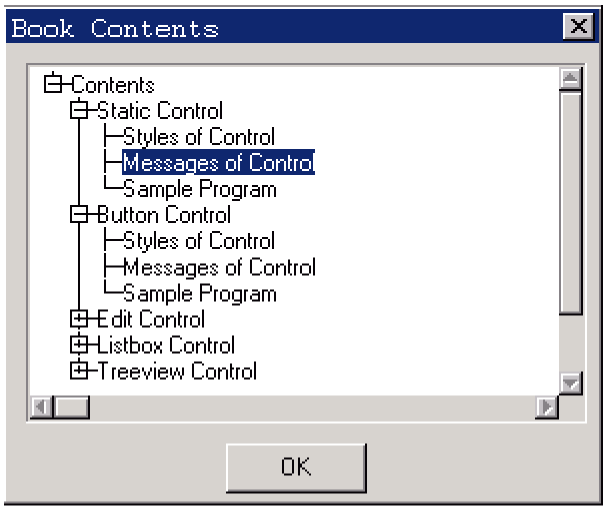
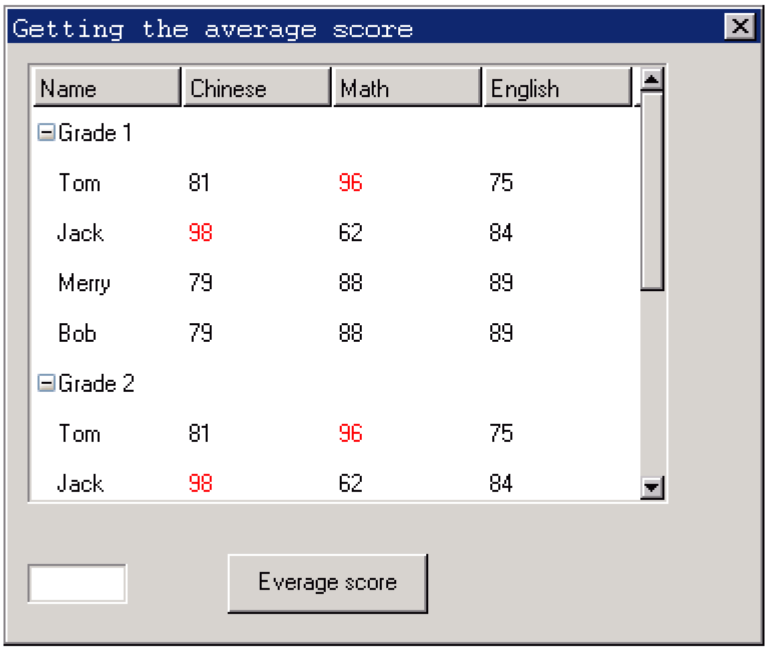


Figure 6.13 Tree view control

6.14 List View Control

The list view control displays a series of data items (list item) in a table form, and the content of each list item may be comprised of one or more sub items. Figure 6.14 gives an instance of list view control.

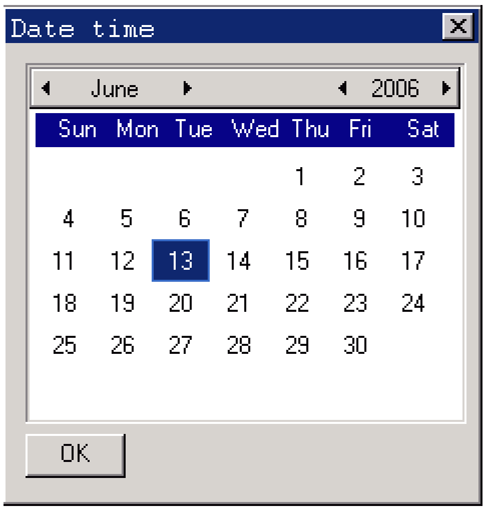


Name	Chinese	Math	English
⊟Grade 1			
Tom	81	96	75
Jack	98	62	84
Merry	79	88	89
Bob	79	88	89
⊟Grade 2			
Tom	81	96	75
Jack	98	62	84

Figure 6.14 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 6.15 illustrates an instance of month calendar control.



Sun	Mon	Tue	Wed	Thu	Fri	Sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Figure 6.15 Month calendar control

6.16 Animation Control

Animation control is one control, which can be used to display animations; it is very simple and easy to use. You can build an ANIMATION object from a GIF89a file by using APIs of MiniGUI, and then create an animation control to play the animation. Figure 6.16 shows two different frames of a GIF89a image in two animation controls.

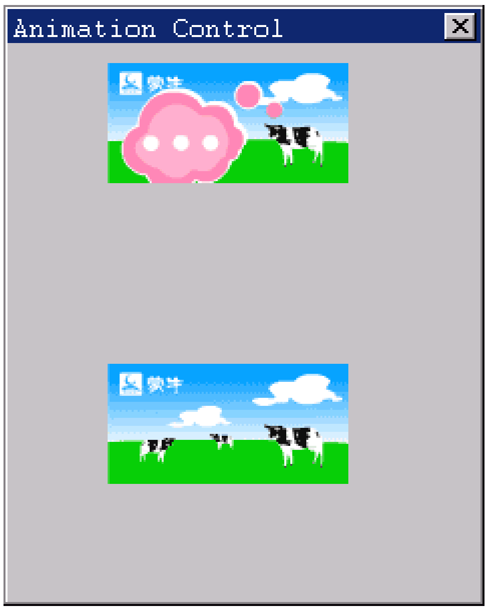
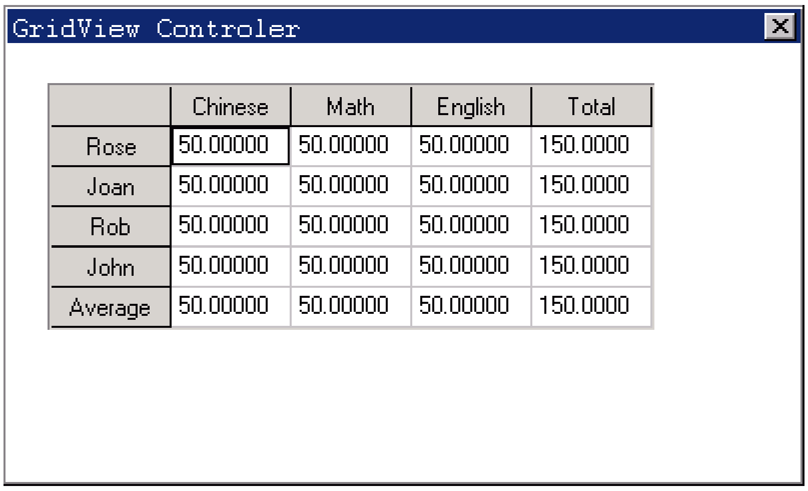


Figure 6.16 Animation control

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 each other. Figure 6.17 shows an instance of grid view control.



	Chinese	Math	English	Total
Rose	50.00000	50.00000	50.00000	150.0000
Joan	50.00000	50.00000	50.00000	150.0000
Rob	50.00000	50.00000	50.00000	150.0000
John	50.00000	50.00000	50.00000	150.0000
Average	50.00000	50.00000	50.00000	150.0000

Figure 6.17 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. Figure 6.18 gives an instance of icon view control.

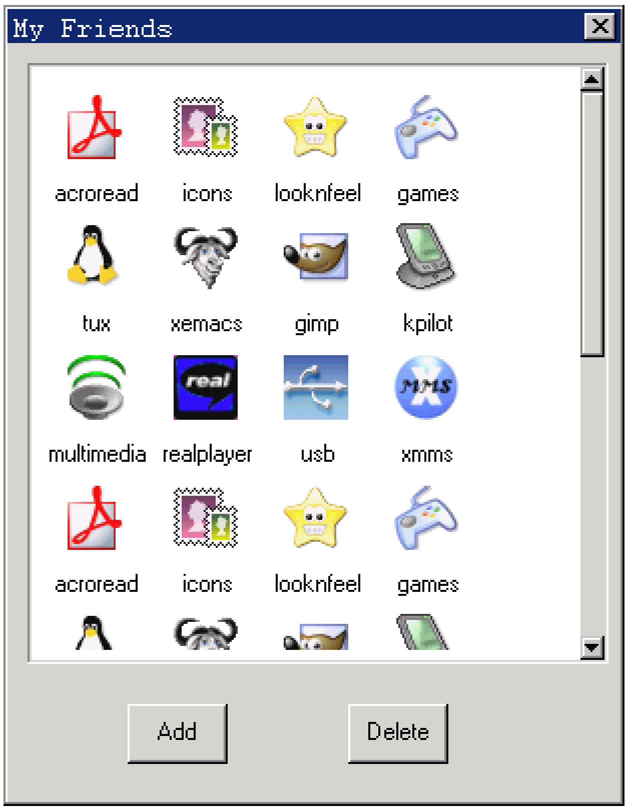
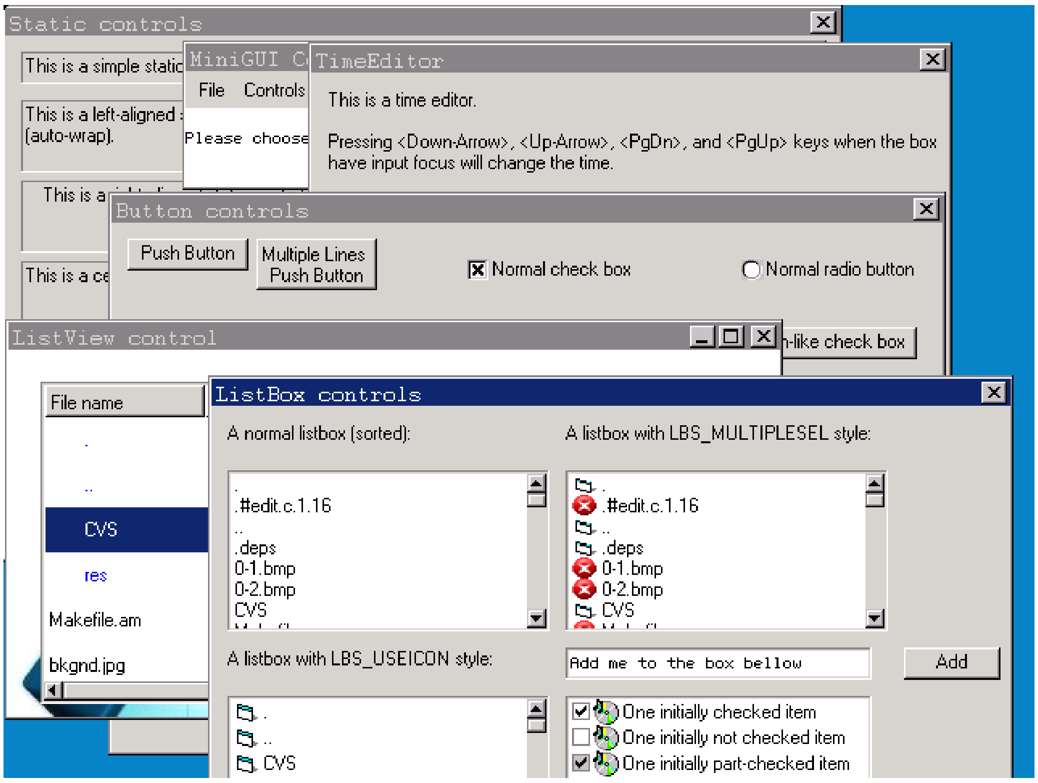


Figure 6.18 Icon view control

7 Appearance Styles of MiniGUI Windows/Controls

Up to now, MiniGUI provides three appearance styles of windows or controls, which are CLASSIC, FLAT, and FASHION. These styles can be used in different displaying devices; you can specify the style you want to use while configuring MiniGUI.

The CLASSIC style is familiar with Windows classic style. Please see Figure 7.1.



This is a simple static

This is a left-aligned (auto-wrap).

This is a

This is a ce

Please choose

This is a time editor.

Pressing <Down-Arrow>, <Up-Arrow>, <PgDn>, and <PgUp> keys when the box have input focus will change the time.

A normal listbox (sorted):

```
.
.#edit.c.1.16
..
.deps
0-1.bmp
0-2.bmp
CVS
```

A listbox with LBS_MULTIPLESEL style:

```
.
.#edit.c.1.16
..
.deps
0-1.bmp
0-2.bmp
CVS
```

A listbox with LBS_USEICON style:

Add me to the box bellow

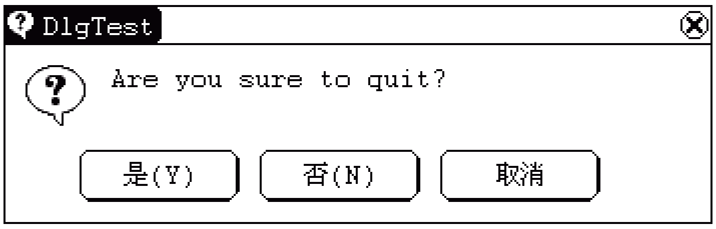
One initially checked item

One initially not checked item

One initially part-checked item

Figure 7.1 The CLASSIC appearance style of MiniGUI

The FLAT style of MiniGUI can be used in gray screen. Please see Figure 7.2.



Are you sure to quit?

Figure 7.2 The FLAT style of MiniGUI

The FASHION style is a handheld devices suitable style of MiniGUI. Please see Figure 7.3.

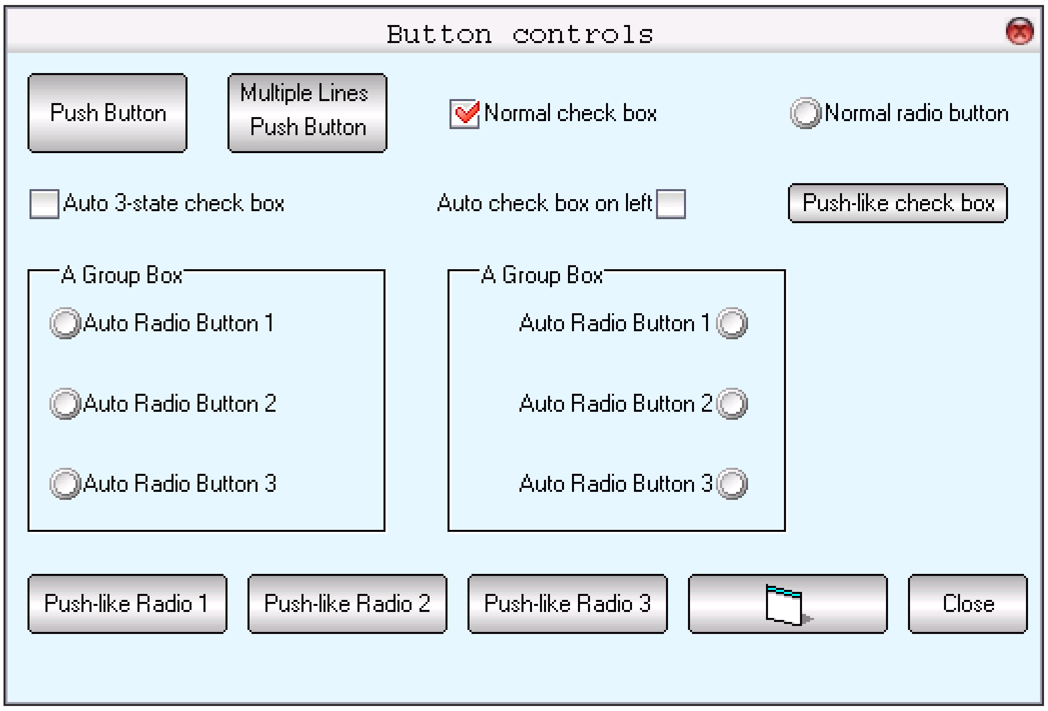


Figure 7.3 The FASHION style of MiniGUI

8 Internationalization

MiniGUI provides support for multiple character sets and multiple fonts. At present, MiniGUI supports ISO8859-1 ~ ISO8859-15, GB2312, BIG5, EUCKR, SHIFT-JIS, UNICODE (UTF-8 and UTF-16 encodings), bitmap fonts, and vector fonts such as TrueType as well as Adobe Type1.

MiniGUI's support for multi-charset and multi-font is achieved by the concept of logical font. It is possible to display characters in different languages in one window. Different from the traditional implementation of the support for multi-charset (which is achieved by UNICODE), MiniGUI's implementation consumes fewer resources, and therefore the better for embedded devices.

While developing MiniGUI applications, if you want to display single specified font, for example simplified Chinese (GB2312), you only need configure MiniGUI to enable GB2312 charset and some fonts for the charset. In this way, we 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 by using TrueType fonts or QPF fonts. In this case, MiniGUI can convert the certain characters in one charset to UNICODE automatically and select the relevant fonts to render the text.

9 Feynman Software's Products Related to MiniGUI

9.1 MiniGUI-VAR Product

MiniGUI Value-Added Release (MiniGUI-VAR) is a value-added product for customers who develop commercial, proprietary (non-GPL) software based on MiniGUI, and is licensed by the number of developers. If you want to develop commercial products based on some embedded operating systems, and want to use MiniGUI as embedded graphics support system, then MiniGUI will be the best choice.

Two versions of MiniGUI value-added products are available: MiniGUI-VAR V2.0.x and MiniGUI-VAR V1.6.x.

MiniGUI-VAR V2.0.x is based on MiniGUI V2.0.x, mainly provides support for those embedded operating systems with multi-process like Linux.

MiniGUI-VAR V1.6.x is based on MiniGUI V1.6.x, mainly supporting those embedded operating systems with multi-thread like uClinux, VxWorks, pSOS, eCos, ThreadX, uC/OS-II, Nucleus, OSE and other operating systems. Feynman Software also provides customers who using MiniGUI-VAR V1.6.x with the optional component MiniGUI SDK for Windows. It can be used with Visual Studio on the Win32 platform to develop MiniGUI applications.

For complete information about MiniGUI-VAR, please visit:

```
http://www.minigui.com/product/index.shtml
```

Except for the MiniGUI-VAR product, Feynman Software also provides some MiniGUI component products and other MiniGUI applications such as mSpider. Figure 9.1 shows the product line of Feynman Software.

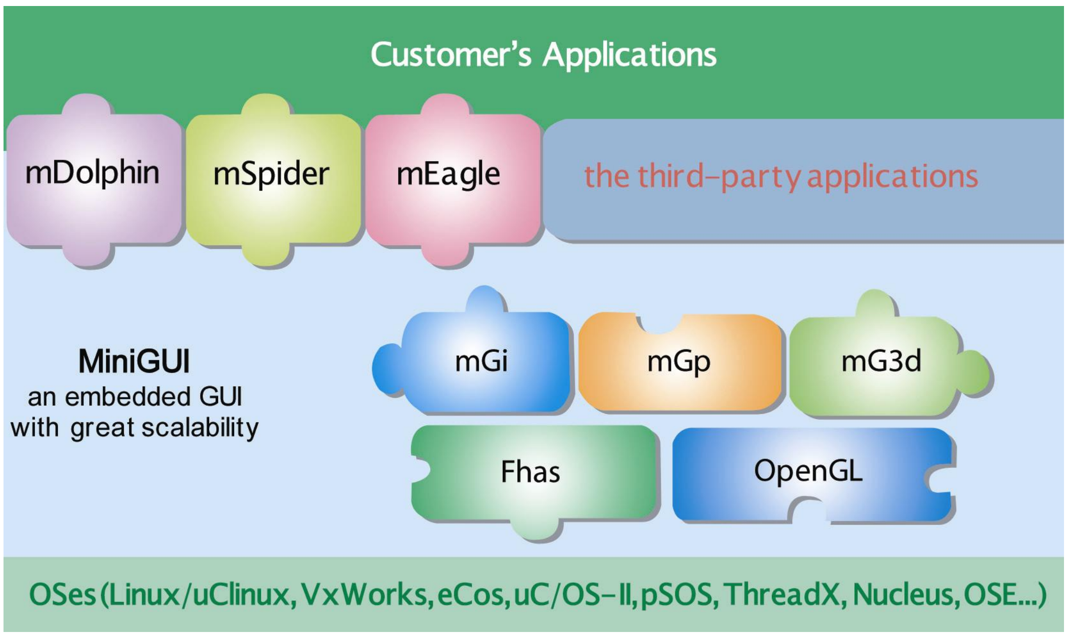
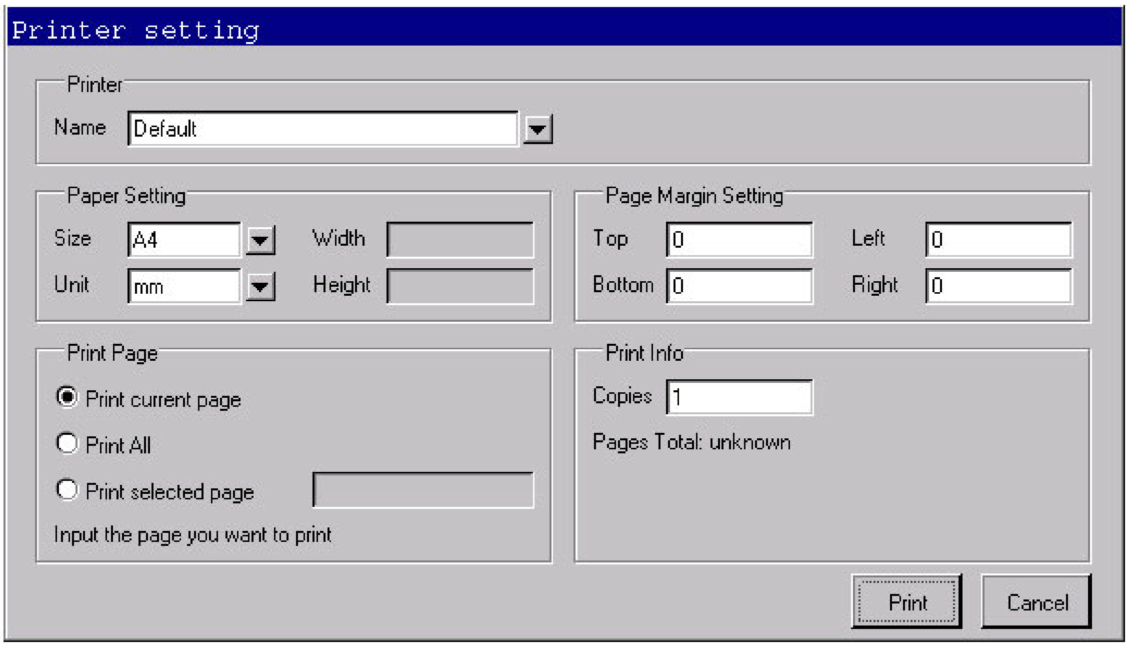


Figure 9.1 Product line of Feynman Software

9.2 MiniGUI Component Products

9.2.1 mGp

mGp provides a printing engine for applications based on MiniGUI so that applications using mGp will have the printing function. At present, mGp provides printing support for Epson, HP and some other printers. Figure 9.2 shows the setup window of mGp.



Printer setting

Printer

Name: Default

Paper Setting

Size: A4 | Width:
Unit: mm | Height:

Page Margin Setting

Top: 0 | Left: 0
Bottom: 0 | Right: 0

Print Page

(●) Print current page
( ) Print All
( ) Print selected page
Input the page you want to print

Print Info

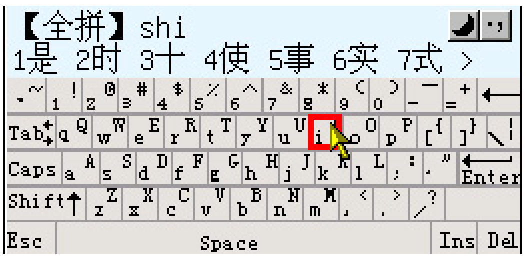
Copies: 1
Pages Total: unknown

Print | Cancel

Figure 9.2 The setup window of mGp

9.2.2 mGi

mGi provides input method framework for applications based on MiniGUI. mGi now provides the framework for soft-keyboard and hand writing input methods. mGi also provides an IME container for user to add self-defined IME to it. On the other hand, you can use self-defined keyboard bitmap for the soft-keyboard and add your self-defined translation method to it. Figure 9.3 and 9.4 give the UIs of mGi.



【全拼】shi
1是 2时 3十 4使 5事 6实 7式 >

Figure 9.3 mGi: input with soft-keyboard

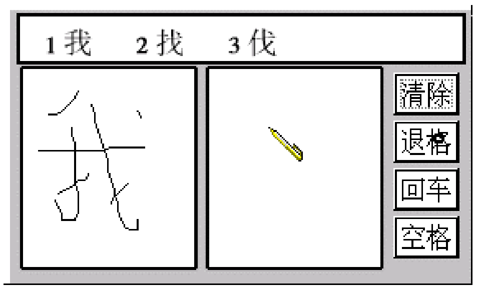
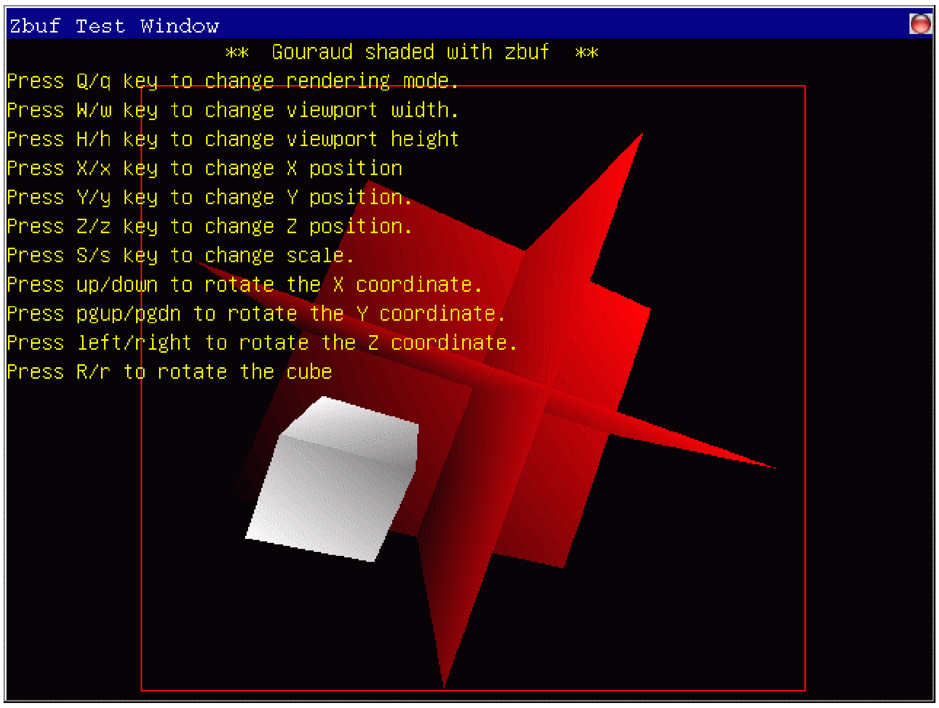


Figure 9.4 mGi: input with hand writing

9.2.3 mG3d

mG3d is a 3D rendering library for applications based on MiniGUI. By using this library, you can render 3D objects in your applications. Figure 9.5 gives some 3D objects created by using mG3d.



```
Zbuf Test Window
               ** Gouraud shaded with zbuf **
Press Q/q key to change rendering mode.
Press W/w key to change viewport width.
Press H/h key to change viewport height
Press X/x key to change X position
Press Y/y key to change Y position.
Press Z/z key to change Z position.
Press S/s key to change scale.
Press up/down to rotate the X coordinate.
Press pgup/pgdn to rotate the Y coordinate.
Press left/right to rotate the Z coordinate.
Press R/r to rotate the cube
```

Figure 9.5 mG3d running effect

9.3 Full-Featured Embedded Browser: mDolphin

mDolphin is developed by Beijing Feynman Software Technology Co., Ltd.. mDolphin, as an embedded browser based on MiniGUI graphic platform, is a modular, scalable, full-featured browser that supports powerful Internet browsing experience.

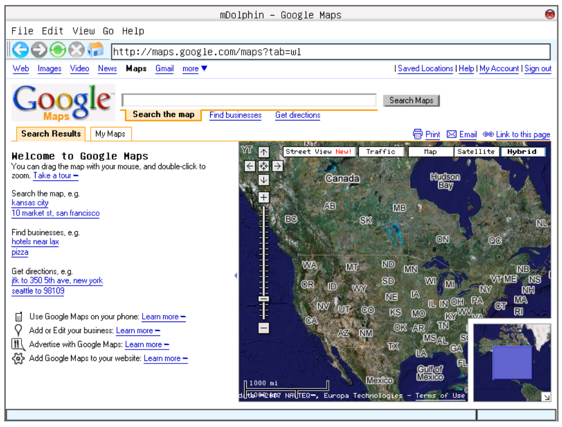
mDolphin's modular and scalable architecture allows for custom configurations to meet specific device requirements. To make browsing convenient, mDolphin includes three rendering technologies, such as Smart-Fit Rendering, which intelligently adapts standard Web pages to the screen width of the device and eliminates the need for horizontal scrolling.

At present, mDolphin can be used in mobile phone (3G phone, WiFi phone), IPTV, information terminal and IP-based device. It can run on Linux/uClinux, eCos. Now, mDolphin has been successfully used in mobile communication device.

The main features of mDolphin are as follow:

- Support for the popular W3C standards: HTML 4.01, XHTML 1.0, XML, CSS 2, XSLT, DOM 2, XPATH, and JavaScript 1.5.
- Support for HTTP 1.1, HTTPS, Cookie and Proxy.
- Support for AJAX applications.
- Support for three rendering modes to meet your needs of the different screen sizes.
- Support for multiple charsets and encodings by using UNICODE kernel; support for bidirectional text.
- Support for Linux/uClinux, eCos and other operating systems.
- Modular and extensible software architecture.
- Support for customization of user interface.

Figure 9.6 illustrates some screenshots of mDolphin.



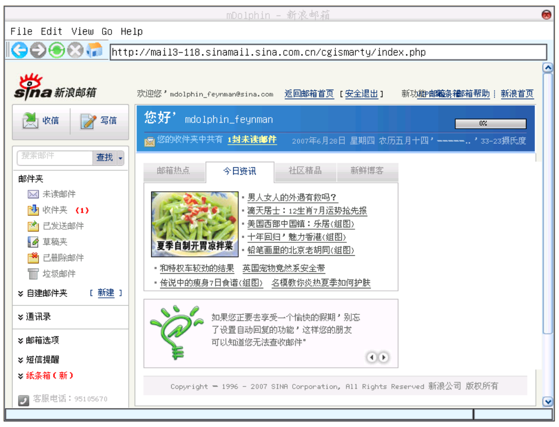
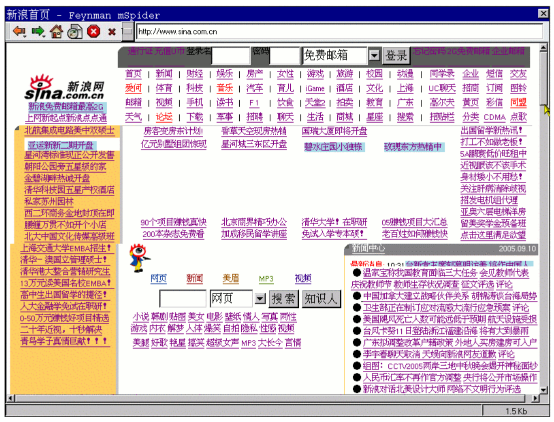


Figure 9.6 Screenshots of mDolphin

9.4 Light-Weight Embedded Browser: mSpider

mSpider is a lightweight browser based on MiniGUI, which is developed by Feynman Software. Figure 9.7 illustrates some screenshots of mSpider.



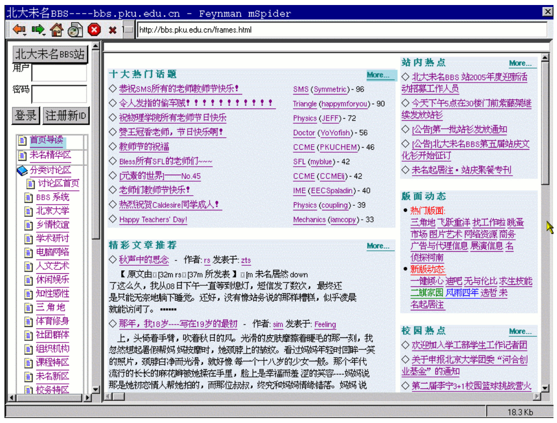

北大未名BBS----bbs.pku.edu.cn - Feynman mSpider
http://bbs.pku.edu.cn/frames.html
北大未名BBS站
十大热门话题
精彩文章推荐
站内热点
版面动态
校园热点

Figure 9.7 Screenshots of mSpider (Visit sina.com and the BBS of Peking University)

Features and advantages of mSpider are as follow:

- Compliant to popular web standards. mSpider supports the most tags and attributes of HTML 4.01, Cookies, Frame/iFrame, HTTP 1.1, as well as the familiar JavaScript 1.5 objects.
- Low footprint and high navigation speed. The size of executable of mSpider is about 2MB large, and the minimal dynamic memory to run mSpider is about 4MB.
- Support for various image formats. Right now, mSpider can display JPEG, PNG, BMP, GIF (animated or not) images.
- Great text rendering ability. Because mSpider is developed based on MiniGUI, mSpider takes good support for the multi-charset and various fonts, and gives us great text render ability.
- Support for multiple operating systems. Derived from the excellent portability of MiniGUI, mSpider can run on many operating systems. By now, mSpider can run on Linux/uClinux and VxWorks smoothly.
- Easy to customize user interface. mSpider V2.0 provides you the ability to customize the user interface of mSpider.

For complete information about mSpider, please visit:

```
http://www.minigui.com/product/mspider.shtml
```

9.5 Geography Information System: mEagle

mEagle is a geography information system for embedded system, which is developed by Feynman Software. mEagle is based on MiniGUI embedded graphics middleware, so it is able to run on more than 10 embedded OSes which MiniGUI supported.

In view of the common application of geographic information system in embedded devices, for example, electronic map operations (roam, zoom, rotation), GPS positioning, path tracking, and so on, Feynman Software has carried on a careful optimized design to mEagle, and makes it have a fast speed, low resource consumption.

The main features of mEagle:

- Based on MiniGUI graphics library, it is suitable for all kinds of embedded systems, and it has good portability, as well as fast running speed.
- Mainly support for MapInfo, ARCInfo and other popular map formats to meet the requirements of different users.
- Support for common GIS functions, including map roam, zoom, distance measurement, map rotation, and so on.
- Support for eagle eye function.
- Support for keyword search and regional search for map information, and it can locate the searching result as well.
- Support for the map coordinates data processing function, and it can be integrated with GPS module to provide GPS navigation and location functions.
- The tri-layer software architecture of mEagle can facilitate the users to do second development greatly to meet all kinds of possible special requirements.

Figure 9.8 illustrates some screenshots of mEagle.

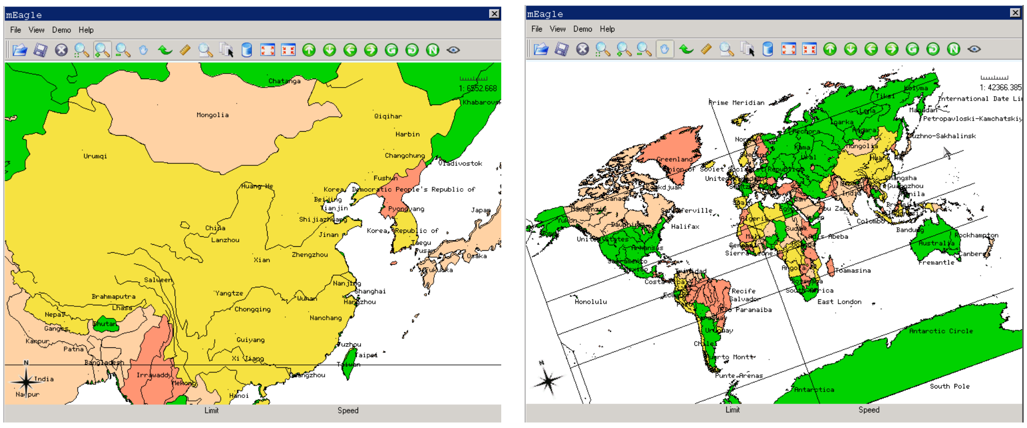


Figure 9.8 Screenshots of mEagle

9.6 Other MiniGUI-Based Applications and Solutions

Feynman Software also provides other MiniGUI-based applications or solutions:

- **mGallery** is a full-featured, dapper, and high performance PMP solution, which is built on MiniGUI. mGallery provides a suit of complete PMP application framework and the main applications, to help the design companies and manufacturers develop PMP solutions and PMP products rapidly and conveniently.
- **mSeal** is a flash player, which is the two-Dimensional vector graph render engineer based on MiniGUI , and developed by Feynman Software. At present, mSeal is a player which most supports for flash characteristics in embedded Linux system, has the highest-quality graphics output, and has the fastest running speed. mSeal will integrate in form of plug-in into the full-featured and new-generation web browser of Feynman.
- **Fhas** is an application platform product designed by Feynman Software, supporting the development of products and applications on all kinds of intelligent appliances. Fhas has been used widely in TD-SCDMA phone and WiFi phone.

For more information about the products above, please visit the website of Feynman Software:

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

10 MiniGUI Resources

10.1 Open Source Releases and Development Packages

Feynman Software has released two versions of MiniGUI under GPL: MiniGUI V1.3.3 and MiniGUI-STR V1.6.2. You can visit the following web page to download the source codes:

```
http://www.minigui.com/download/index.shtml
```

If you want to evaluate the full features of MiniGUI V2.0 x, please download the development package in the above web page. Install the development package on a PC box running Linux, and then you can develop MiniGUI V2.0.x applications. The development package for MiniGUI V1.6.x is also available on the above web page.

```
http://www.minigui.com/download/index.shtml
```

You could get the on-line API reference documentations of MiniGUI V2.0.x, MiniGUI V1.6.x, and MiniGUI V1.3.x at the following link:

```
http://www.minigui.com/api_ref/index.html
```

If you have any questions on using MiniGUI, you can visit our technology forum to tell us your opinions and questions. The link is:

```
http://www.minigui.org/cgi-bin/lb5000/leoboard.cgi
```

10.2 Other Open Source Softwares Released by Feynman Software

You could get our open source software from our website, all of these can run on MiniGUI:

- mGIS 1.0: This is a Geography Information System on MiniGUI (GPL).
- eDillo 0.4.0: This is an embedded browser for MiniGUI. It is based-on Dillo 0.8.x (GPL).
- Monqueror 0.6:This is an browser which is ported from Konqueror (GPL).
- MGXine 0.9.12: This is a multi-media player for MiniGUI. It is based-on the famous Xine (GPL/LGPL).
- FlashPlayer 0.2: This is a flash (SWF) player for MiniGUI (GPL).

The download page of all above open source software is:

```
http://www.minigui.com/download/mgother.shtml
```

10.3 MiniGUI Demos

You could download MiniGUI demos from the following web page:

```
http://www.minigui.com/download/mgdemo.shtml
```

There are MiniGUI demos on Linux PC platform, Windows platform, VxWorks platform and other popular hardware development boards.

11 Licensing Policy of GPL'd MiniGUI Versions

Feynman Software releases some versions of MiniGUI under. So any application linking to GPL'd MiniGUI must follow GPL. If you cannot accept GPL, you need to be licensed from Feynman Software.

11.1 Free Use for Those Who Are 100% GPL

If your application is licensed under GPL, you are free and welcome to ship any GPL software of Feynman Software with your application. By "application" we mean any type of software application, system, tool or utility. For doing this, you do not need a separate signed agreement with Feynman Software, because the GPL text is sufficient. But we do recommend you to be in touch with us as there usually are good opportunities for partnership and co-marketing.

11.2 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.

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.

11.3 Commercial Use for Everyone Else

If your application is not licensed under GPL and you intend to distribute MiniGUI software (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 Feynman Software takes responsibility for its products. Under the GPL license, there are no warranties or representations from the developer (i.e. from Feynman Software).

12 Contact Us

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

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

For complete products information of Feynman Software, please refer to:

```
http://www.minigui.com/product/index.html
```

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

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