

miniStudio 用户手册

版本 1.0 修订号 0 适用于 miniStudio Ver 1.0.x

北京飞漫软件技术有限公司 2010年5月

版权所有 (C) 2008~2010, 北京飞漫软件技术有限公司, 保留所有权利。

无论您以何种方式获得该指南的全部或部分文字或图片资料,无论是普通印刷品还是电子文档,北京飞漫 软件技术有限公司仅仅授权您阅读的权利,任何形式的格式转换、再次发布、传播以及复制其内容的全部 或部分,或将其中的文字和图片未经书面许可而用于商业目的,均被视为侵权行为,并可能导致严重的民 事或刑事处罚。

目录

_Too	c263773467	
第一	─章 开篇	1
	概述	1
	用 VI 编辑器实现示例程序	2
	用 miniStudio 实现示例程序	9
	传统的 VI+makefile 传统方式和 miniStudio 的方式对比	
第二	二章 miniStudio 开发环境	
	嵌入式环境搭建准备工作	
	嵌入式环境搭建	
	在 Linux 下搭建嵌入式开发环境	
	Eclipse 搭建嵌入式开发环境	15
第三	三章 miniStudio 实例开发	
	实例开发	24
	PC 平台的实例开发	
	君正 4740 平台的实例开发	
	实例下载	40
第四	马章 miniStudio 实现多语言	41
	不同字体输入功能。包括,中文,英文,繁体。	41
	翻译功能 实现多语言更容易	42
	字体调整	45
第王	互章 Connect Event 的实例应用-秒表	47
	Connect Event 的介绍	47
	Connect Event 的应用	47
	新建窗口	47
	添加消息事件	48
	添加控件	49
	添加 connect event 事件	
	添加代码	54
	编译运行	55
	实例下载	56
第六	、章 数据绑定与数据源的应用	57
	数据绑定与数据源的介绍	57
	数据绑定实例应用	57
	数据绑定的功能	57
	数据绑定的实例	57
	数据源实例应用	59
	数据源的功能	59
	静态数据源实例应用	60
	实例包下载	62
第七	上章 渲染器及其应用	63
	渲染器介绍	63

染器的使用6	3
這染器和渲染器集的创建6	3

第一章 开篇

概述

刚开始使用一个产品的时候,每个人都会有相同的疑问:

- 为什么我要用这个新东西?
- 这个新东西能给我带来什么?

在嵌入式产品的开发中,我们最长见的开发工具就是 VI+Makefile 了。那现在我们就用 VI 和 miniStudio 两 个不同的工具分别来实现"两个窗口层次调用"的示例程序。通过这个示例,可以回答大家上面的两个问题,同时还可以帮助大家尽快的体会到 miniStudio 的便捷之处!

本示例具体需求是通过点击第一个窗口 Window-I 中的按钮"Window-II"来调出第二个窗口 Window-II 。其中图 1-1 为 Window-I 窗口效果图, 图 1-2 为 Window-II 窗口效果图。

Window-I		
	Window-II	

图 1-1





用 VI 编辑器实现示例程序

1、在 vi 编辑器中键入如下代码,保存文件名为 window1.c

```
window1.c 具体代码如下:
```

```
/*
** $Id: windowl.c 2009-10-27 05:22:47 $
**
** Copyright (C) 1998 ~ 2009 Feynman Software.
**
** License: GPL
*/
#include <minigui/common.h>
#include <minigui/minigui.h>
#include <minigui/minigui.h>
#include <minigui/minigui.h>
#include <minigui/minigui.h>
#include <minigui/minigui.h>
```

```
#define IDC_BUTTON 1000
extern Window2;
HWND Window1;
static int MiniGUIProc(HWND hWnd, int message, WPARAM wParam, LPARAM 1Param)
{
        HDC hdc;
        static int number = 0;
        switch (message) {
                case MSG_CREATE:
                CreateWindow (CTRL_BUTTON,
                "Window-II",
                WS_CHILD | BS_PUSHBUTTON | BS_CHECKED | WS_VISIBLE,
                IDC_BUTTON,
                75, 150, 150, 50, hWnd, 0);
                break;
                case MSG_COMMAND:
                switch(wParam)
                {
                        case IDC_BUTTON:
                        WindowSecond(hWnd);
                        return 0;
```

```
}
               break;
               case MSG_CLOSE:
               DestroyMainWindow (hWnd);
               PostQuitMessage (hWnd);
               return 0;
       }
       return DefaultMainWinProc(hWnd, message, wParam, 1Param);
int MiniGUIMain (int argc, const char* argv[])
{
       MSG Msg;
       MAINWINCREATE CreateInfo;
       #ifdef _MGRM_PROCESSES
       JoinLayer(NAME_DEF_LAYER, "MiniGUI", 0, 0);
       #endif
       CreateInfo.dwStyle = WS_VISIBLE | WS_MAXIMIZEBOX| WS_MINIMIZEBOX|WS_THINFRAME |
WS_CAPTION;
```

CreateInfo.dwExStyle = WS_EX_NONE; CreateInfo.spCaption = "Window-I"; CreateInfo.hMenu = 0; CreateInfo.hCursor = GetSystemCursor(0); CreateInfo.hIcon = 0; CreateInfo.hIcon = 0; CreateInfo.MainWindowProc = MiniGUIProc; CreateInfo.lx = 0; CreateInfo.ty = 0; CreateInfo.ty = 0; CreateInfo.rx = 300; CreateInfo.by = 400; CreateInfo.iBkColor = COLOR_lightwhite; CreateInfo.dwAddData = 0; CreateInfo.hHosting = HWND_DESKTOP;

Window1 = CreateMainWindow (&CreateInfo);

if (Window1 == HWND_INVALID)

return -1;

ShowWindow(Window1, SW_SHOWNORMAL);

while (GetMessage(&Msg, Window1)) {

TranslateMessage(&Msg);

DispatchMessage(&Msg);

```
}
MainWindowThreadCleanup (Window1);
return 0;
}
#ifdef _MGRM_THREADS
#include <minigui/dti.c>
#endif
```

2、在 vi 编辑器中键入如下代码,保存文件名为 window2.c

```
window2.c 具体代码如下:
```

```
/*
** $Id: window2.c 2009-10-27 05:22:47 $
**
** Copyright (C) 1998 ~ 2009 Feynman Software.
**
** License: GPL
*/
#include <minigui/common.h>
#include <minigui/minigui.h>
```

#include <minigui/gdi.h>

#include <minigui/window.h>

#include <minigui/control.h>

#define IDC_BUTTON 100

extern Window1;

HWND Window2;

static int MiniGUIProc(HWND hWnd, int message, WPARAM wParam, LPARAM 1Param)

```
{
HDC hdc;
static int create_num = 0;
switch (message) {
    case MSG_CLOSE:
    DestroyMainWindow (hWnd);
    PostQuitMessage (hWnd);
    return 0;
}
```

return DefaultMainWinProc(hWnd, message, wParam, 1Param);

void WindowSecond(HWND hWnd)

MSG Msg;

{

MAINWINCREATE CreateInfo;

#ifdef _MGRM_PROCESSES

```
JoinLayer(NAME_DEF_LAYER, "MiniGUI", 0, 0);
```

#endif

CreateInfo.dwStyle = WS_VISIBLE | WS_MAXIMIZEBOX| WS_MINIMIZEBOX|WS_THINFRAME | WS_CAPTION;

```
CreateInfo.dwExStyle = WS_EX_NONE;
CreateInfo.spCaption = "Window-II";
CreateInfo.hMenu = 0;
CreateInfo.hCursor = GetSystemCursor(0);
CreateInfo.hIcon = 0;
CreateInfo.hIcon = 0;
CreateInfo.MainWindowProc = MiniGUIProc;
CreateInfo.lx = 50;
CreateInfo.ty = 100;
CreateInfo.ty = 100;
CreateInfo.rx = 250;
CreateInfo.by = 300;
CreateInfo.iBkColor = COLOR_yellow;
CreateInfo.dwAddData = 0;
CreateInfo.hHosting = HWND_DESKTOP;
```

Window2 = CreateMainWindow (&CreateInfo);

ShowWindow(Window2, SW_SHOWNORMAL);

}

#ifdef _MGRM_THREADS

#include <minigui/dti.c>

#endif

3 编译运行,根据运行效果修改 window1.c 和 window2.c 中相关窗口和按钮的属性。

用 miniStudio 实现示例程序

1、GuiBuilder中设置两个窗口以及一个按钮的属性。

首先,在图形开发界面 GuiBuilder 里设置两个窗口以及一个按钮的相关属性,比如"窗口""按钮"的大小、背景色等相关属性。同时可以通过 GuiBuilder 的预览功能及时的看到最终的图形效果。

操作界面如下:

在 GuiBuilder 中创建窗口 Window-I、 Window-II 并在 Window-I 中添加"按钮"控件。之后直接在 GuiBuilder 中按照自己需要修改窗口和控件的属性值。通过预览功能还可以看到窗口运行时的最终图形效 果。如图 1-3 为窗口 Window-I 的最终设计界面,图 1-4 为窗口 Window-II 的最终设计效果,图 1-5 是"按钮"控件相关属性的设置。

[StartWnd]Window–I.xml Window–II.xml		Property Event	Renderer	
Window-I		Name	Value	
		ID	ID_MAINWND1	
		х	0	
		Y	0	
	=	Width	300	
		Height	400	
		Text	Window-I	
Window-II		Renderer	0	
	Ц	BgColor	0×FFFFFFFF	
		Notify	True	
		Border	True	
		Visible	True	
		Enabled	True	
		HasCaption	True	
		SystemMenu	True	
		BorderType	DlgFrame	
«I w I	┝	MinimizeBox	True	
[2] [2] [2] [2] [2] [2] [2] [2] [2] [2]			i	
I~1 I-3				
图 1-3				
হা I-3 [StartWnd]Window–I.xml Window–II.xml		Property Event	Renderer	
StartWnd]Window-I.xml Window-II.xml		Property <u>Event</u> Name	Renderer Value	
[StartWnd]Window-I.xml Window-II.xml	4	Property <u>Event</u> Name ID	Renderer Value ID_MAINWND2	
[StartWnd]Window-I.×ml Window-II.×ml	A	Property Event Name ID X	Renderer Value ID_MAINWND2 50	
SartWnd]Window-I.xml Window-II.xml		Property <u>Event</u> Name ID X Y	Renderer Value ID_MAINWND2 50 100	
StartWnd]Window-I.xml Window-II.xml		Property <u>Event</u> Name ID X Y Width	Renderer Value ID_MAINWND2 50 100 200	
In the second se		Property <u>Event</u> Name ID X Y Width Height	Renderer Value ID_MAINWND2 50 100 200 200	
[StartWnd]Window-I.xml Window-II.xml	III	Property Event Name ID X Y Width Height Text	Renderer Value ID_MAINWND2 50 100 200 200 Window-II	
In the second se		Property Event Name ID X Y Width Height Text Renderer	Renderer Value ID_MAINWND2 50 100 200 200 Window-II 0	
[StartWnd]Window-I.xml Window-II.xml		Property Event Name ID X Y Width Height Text Renderer BgColor	Renderer Value ID_MAINWND2 50 100 200 200 Window-II 0 0xFF00FAFF	
[StartWnd]Window-I.xml Window-II.xml		Property Event Name ID X Y Width Height Text Renderer BgColor Notify	Renderer Value ID_MAINWND2 50 100 200 200 200 Window-II 0 0xFF00FAFF True	
[StartWnd]Window-I.xml Window-II.xml		Property Event Name ID X Y Width Height Text Renderer BgColor Notify Border	Renderer Value ID_MAINWND2 50 100 200 200 200 Window-II 0 0xFF00FAFF True True	
[StartWnd]Window-I.xml Window-II.xml		Property Event Name ID X Y Width Height Text Renderer BgColor Notify Border Visible	Renderer Value ID_MAINWND2 50 100 200 200 Window-II 0 0xFF00FAFF True True True	
[StartWnd]Window-I.xml Window-II.xml		Property Event Name ID X Y Width Height Text Renderer BgColor Notify Border Visible Enabled	RendererValueID_MAINWND250100200200Window-II00xFF00FAFFTrueTrueTrueTrueTrueTrue	
[StartWnd]Window-I.xml Window-II.xml		Property Event Name ID X Y Width Height Text Renderer BgColor Notify Border Visible Enabled HasCaption	Renderer Value ID_MAINWND2 50 100 200 200 Window-II 0 0xFF00FAFF True True	
[StartWnd]Window-I.xml Window-II.xml		Property Event Name ID X Y Width Height Text Renderer BgColor Notify Border Visible Enabled HasCaption SystemMenu	RendererValueID_MAINWND250100200200Window-II00xFF00FAFFTrue	
[StartWnd]Window-I.xml Window-II.xml		Property Event Name ID X Y Width Height Text Renderer BgColor Notify Border Visible Enabled HasCaption SystemMenu BorderType	Renderer Value ID_MAINWND2 50 100 200 200 Window-II 0 0xFF00FAFF True True	
[StartWnd]Window-I.xml Window-II.xml		Property Event Name ID X Y Width Height Text Renderer BgColor Notify Border Visible Enabled HasCaption SystemMenu BorderType	RendererValueValueID_MAINWND250100200200Window-II00xFF00FAFF00xFF00FAFFTrueTrueTrueTrueTrueTrueTrueDigFrameTrue	

[StartWnd]Window–I.xml Window–II.xm	1	_[Property Event	Renderer
Window-I			Name	Value 📤
			ID	ID_BUTTON1
			х	75
	=	= [Y	150
			Width	150
			Height	50
· · · ·			Text	Window-II
▪ Window-II			Renderer	0
			BgColor	0×FFCED3D6
			Font	
			Notify	True
			Border	False
			Visible	True
▲ III	▼		Enabled	True

图 1-5

2、添加第一个窗口"按钮"的 onClicked 事件来调用第二个窗口

因为部分代码会自动生成,所以如果我们要完成通过按钮调用第二个窗口的功能,只需要少量的代码。

具体步骤如下,添加一个"按钮"的 onClicked 事件,双击 onClicked (如图 1-6), 之后会自动跳转到要 添加代码的相应行,此时添加相应代码即可。

<pre>*[StartWnd]Window-I.xml Window-II.x</pre>	ml	Property	Event	Renderer	
Window-I	<u> </u>	Name	;	Value	_
		onClicked		onClicked	
		onPushed			
	=	onCreate			
		onSizeChar	ging		
		onSizeChar	iged		
· · · · · · · · · · · · · · · · · · ·		onCSizeCha	inged		
• Window-II	•	onFontChar	ging		
		onFontChar	iged		
		onEraseBkg	(rnd		
		onPaint			
		onClose			
		onKeyDown			
	► ►	onKeyUp			

图 1-6

此示例中会跳转到 Window-I.c 文件的如下函数中, Copyright © by the Feynman Software. All contents is the property of Feynman Software.

Button1_onClicked (mWidget* self, int id, int nc)

我们需要在这个函数内添加代码如下:

ntCreateMainwnd2Ex(hPackage, HWND_DESKTOP, 0, 0, 0);

因为要引用变量 "hPackage", 所以在 Window-I.c 文件中先引入外部变量就可以了。

extern HPACKAGE hPackage;

因此我们只需要自己添加两行代码,同样的功能就可以实现了。

传统的 VI+makefile 传统方式和 miniStudio 的方式对比

下面是完成一个同样功能 GUI 的产品从开发到发布,使用传统的方式(vi+makefile)和 miniStudio 的所需要的步骤和总时间对比表

步骤序号	<u>VI+makefile 传统方式</u>	<u>miniStudio</u>	<u>注解</u>
1	<u>MiniGUI</u> 主框架代码编写	guibuilder 可以生成所有的框架代码	guibuilder 本身可以生成所有 框架代码和时间的 handle 代 码,更方便,更安全,更稳定
2	界面设计,通过代码设置每个控件的 大小位置	guibuilder 可视化的设计,随 时预览各个控件的位置大小	miniStudio 方便的界面设计功 能,同时提供 MiniGUI3.0 支持 的所有控件的支持,完全放心 使用
3	使用 VI 进行业务逻辑代码编写,需要 自己在窗口过程函数中设计每个控 件和窗口, 控件和控件之间的消息相 应,需要对 Minigui 的每个消息都有 详细了解	guibuilder 可以直接选择每个 控件的事件,同时跳转到代码 相应的位置,用户直接可以编 写这块事件相应代码,无须了 解消息是怎么传递	独有的 ConnectEvent 功能,可 以把两个控件通过一个消息 链接起来,帮助更方便的使用 控件对象
4	通过代码设置窗口字体,渲染器等。 需要加载设备字体,创建逻辑字体, 然后对控件或者 HDC 设置字体后才 能正常使用字体。渲染器需要对窗口 中的每个控件单独设置渲染器功能。	guibuilder 可视化设计字体, 渲染器,所见即所得	miniStudio 的渲染器设计功 能,可以灵活的设计每个控件 的每个线条,边边角角都可以 随意改动,并可以动态替换, 方便换肤
5	文件多的时候,需要编写 Makefile 才可以正常编译自己东西	eclipse 编译环境,自动生成 makefile,同时支持交叉编译	miniStudio 架构在 eclipse 基础 之上,充分利用了 eclipse 的强 大功能配合

6	自己制作发布包,把所有资源放在一 起打包,放到板子上	miniStudio 可以支持打包发布	miniStudio本身提供打包发布 的功能,更快捷的把你放置在 不同位置的图片,字体等资源 打包。
总 时 间	>2 天	<4 /	小时

通过上面两个示例的对比,可以看出,使用 vi+makefile 的传统方法,需要手工键入所有的代码,根据需求 来设计窗口大小,button 控件位置,每次要看效果,需要先编译运行,之后根据实际情况做出相应调整, 修改属性不直观,更不要说设计时候字体渲染器等高级应用,动辄几百行代码,都没有设计一点业务逻辑。 而使用 miniStudio 的时候,我们有直观的 guibuilder 界面编辑器,可以随时预览窗口效果,同时生成大部 分代码。我们所要作的,就是从窗口 gui 设计中跳出来,专心的投入到业务逻辑的设计中。

所以, 赶快开始你的 miniStudio 之旅吧。

第二章 miniStudio 开发环境

嵌入式环境搭建准备工作

- Linux 操作系统: ubuntu 8.04
- 准备好 miniStudio 安装文件和运行 miniStudio 的第三方软件
- 在 Eclipse 官方网站 http://www.eclipse.org/downloads/下载支持 C/C++开发的 IDE
- 安装 Eclipse 运行的所需要插件: sudo apt-get install sun-java6-jre sun-java6-plugin sun-java6-fonts sun-java6-jdk
- 准备交叉编译工具链: mipseltools-gcc412-lnx26.tar.gz

嵌入式环境搭建

在 Linux 下搭建嵌入式开发环境

- 根据安装 README,安装 miniStudio
- 解压交叉编译工具链到具体路径。例如: tar xvf mipseltools-gcc412-lnx26.tar.gz -C /opt/toolchain/
- 添加交叉编译工具的环境变量:
 - o 如果你只想临时添加交叉编译器的环境变量,只需在终端输入命令: export
 - PATH=\$PATH:/opt/toolchain/mipseltools-gcc412-Inx26/bin 即可
 - 。 如果你想把环境变量永久的添加到系统的环境变量中,只需把 export

PATH=\$PATH:/opt/toolchain/mipseltools-gcc412-Inx26/bin,添加到~/.bashrc 文件即可

- 在编译脚本中指定头文件和库文件的路径。如下所示:
- #! /bin/sh
- CFLAGS="-g -02 -I/opt/mipseltools-gcc412-lnx26/mipsel-linux/include"
- LDFLAGS="-L/opt/mipseltools-gcc412-lnx26/mipsel-linux/lib -lts"
- arm-linux-gcc \$CFLAGS \$LDFLAGS -o NcsDemo AdvanceControls.c containers.c \
- Lables.c main.c NcsDemo main.c scrollbar.c trackbar.c buttons.c edits.c \backslash
- listbox.c main_welcome.c progressbar.c spinner.c mgb_tswin.c -lmgncs \
- -lmgutils -lmgplus -lminigui_ths -lpthread -lpng -ljpeg -lz -lm -lstdc++

•

• 在 miniStudio 生成的代码中,按照上面的脚本编写即可

Eclipse 搭建嵌入式开发环境

- 根据 README 安装 miniStudio
- 解压交叉编译工具链到具体路径。例如: tar xvf mipseltools-gcc412-lnx26.tar.gz -C /opt/toolchain/
- 把交叉工具链的环境变量添加到系统的环境变量中
- 启动 Eclipse 并且设置 Eclipse 工作空间,就是选择个目录,然后点击 OK 按钮。如图 2-1 所示

eclipse	
Eclipse stores your projects in a folder called a workspace	
Choose a workspace folder to use for this session.	
Workspace: /home/licaijun/workspace/mStudio-Test/ncs-v6-demo Structure Browse	
□ <u>U</u> se this as the default and do not ask again Cancel OK	



• 新建一个 MiniGUI 工程. 如图 2-2 和图 2-3 所示

New New	X
Select a wizard	
New MiniGUI Application Project	
<u>W</u> izards:	
type filter text	
👂 🗁 General	
▷ 🗁 C/C++	
CVS	
🗢 🗁 MiniGUI	
MiniGUI Application Project	
Tasks	
	Tinich
< <u>Back</u> <u>Next</u> Cancel	-inisn

图 2-2

🗧 MiniGUI	Project
MiniGUI Project	
Create MiniGUI project of selected type	
Project name: MiniGUI	
✓ Use <u>d</u> efault location	
Location: //home/licaijun/workspace/mStudio	-Test/ncs-v6-demo/MiniGUI B <u>r</u> owse
Project type:	Toolchains:
🗢 🗁 MiniGUI Project	Linux GCC
MG 3.0.x Project With NCS	
Empty MiniGUI Project	
Show project types and toolchains only if	they are supported on the platform
? < <u>B</u> ack <u>N</u>	ext > Cancel <u>Finish</u>

图 2-3

• 打开工程的属性对话框进行编译选项配置。如图 2-4

~	S MiniGUI	1	
	🕨 🐮 Binaries	New	,
	lncludes	Go Into	
8	🗸 🥝 res	Open in <u>N</u> ew Window	
	👝 image	Сору	Ctrl+C
	v Brenderer	📄 Paste	Ctrl+V
	V Biexi	💢 Delete	Delete
	MiniGUL rev	3. Remove from Context	Shift+Ctrl+Alt+Down
	in res project	Maye	
	b Gase	Rena <u>m</u> e	F2
	Include	🚵 Import	
	🕨 👝 Debug	A Export	
	mgncs.cfg	Build Project	
	MiniGUL.crg	Clean Project	
		8 Refresh	F5
		Close Project	
		Close Unrelated Projects	
		Exclude from build	
		Build Configurations	>
		Make Targets	>
		Index	>
		Convert To	
		<u>R</u> un As	>
		Debug As	>
		Profile As	>
		Team	>
		Comp <u>a</u> re With	>
		Restore from Local History	
		Properties	AltEnter
		17	

图 2-4

• 点击"Properties for MiniGUI"对话框中的"Manage Configurations"按钮. 如图 2-5

•	Properties for Min	igui 💮
type filter text	Settings	↓ · → · ·
Resource Builders ▼ C/C++ Build Build Variables Discovery Options	Configuration: Debug [Active]	ild Artifact Binary Parsers S Error Parsers
Environment Settings	GCC C Compiler	Command: gcc
Tool Chain Editor ▷ C/C++ General MiniGUI Properties Project References Refactoring History Run/Debug Settings ▷ Task Repository WikiText	 Preprocessor Symbols Directories Optimization Debugging Warnings Wiscellaneous SCC C Linker General Libraries 	All options: -I/usr/include/ -I/usr/local/ include/ -I/include/ -O0 -g3 - Wall -c -fmessage-length=0 Expert settings: Command line pattern: \${COMMAND} \${FLAGS} \${C
?		OK Cancel

图 2-5

• 新建一个 configuration。点击"New"按钮,在对话框中的"name"栏输入工具链名称,"Description" 栏输入描述。 如图所示: 2-6

•		Properties fo	or MiniGUI			×
type filter text	Setting	S		¢	• => •	•
Resource Builders ▼ C/C++ Build	Configu	Iration: Debug [Active]	\$	Manage Configura	itions	^
Build Variables Discovery Options Environment	⊛ Tool	Cro Note: The configuration n system. Please ensure th	eate New Configuration ame will be used as a directory r at it is valid for your platform.	name in the file		1
Settings Tool Chain Editor ▷ C/C++ General MiniGUI Properties Project References Refactoring History Run/Debug Settings ▷ Task Repository	▽ 89	Name: mipsel-linux Description: mipsel Copy settings from © Existing configuration © Default configuration	Debug		JT_F	Ξ
WikiText	⊽ እ	 Import predefined 	not selected Cancel	ОК ¢		
() () () () () () () () () () () () () (Cancel	ОК	

图 2-6

• 选中"Manage Configurations"对话框中新增加的工具链,点击"Setactive"按钮,设置为当前工具链; 如图 2-7 所示

🍵 MiniGUI: Manage Configurations 🛛 🗙						
Configuration	Description	Status				
Debug		Active				
mipsel-linux-gcc	mipsel					
Release						
<	111	>				
Set Active New Delete Rename						
Cancel OK						

图 2-7

• 修改 GCC C compiler, 把"Command"编辑框内的 gcc 的改成 mipsel-linux-gcc 编译器。 如图 2-8 所示

•	Properties for MiniGUI
type filter text	Settings 🗢 🗘 🗸
Resource Builders ▼ C/C++ Build Build Variables Discovery Options Environment Settings	Configuration: mipsel-linux-gcc [Active] Stool Settings Build Steps Build Steps Build Artifact GCC C Compiler Command: Mipsel-linux-gcc
Tool Chain Editor ▷ C/C++ General MiniGUI Properties Project References Refactoring History Run/Debug Settings ▷ Task Repository WikiText	Image: Preprocessor All options: -!/usr/include/ -!/usr/include/ -!/ include/ -OO -g3 -Wall -c -fmessage- length=0 Image: Preprocessor Image: Preprocessor Image: Preprocessor Image: Preprocessor Preprocessor Preprocessor Image: Preprocessor Prepreprocesor Preprocessor
?	Cancel OK

图 2-8

• 指定编译程序时候,所需要链接的头文件目录。如图所示: 2-9

Properties for MiniGUI						
type filter text Resource Builders C/C++ Build Build Variables Discovery Options Environment Settings Tool Chain Editor C/C++ General MiniGUI Properties Project References Refactoring History Run/Debug Settings Task Repository WikiText	Settings ▼ Settings GCC C Compiler Preprocessor Symbols Directories Optimization Debugging Warnings Warnings Warnings Warnings Secollaneous GCC C Linker General Cibraries Miscellaneous Shared Library Settings GCC Assembler General General	Include paths (-I) /usr/include/ /usr/local/include/ /include/				
?			Cancel	OK		

图 2-9

• GCC C Linker 中的"Command"编辑框内的 gcc,也要改成 mipsel-linux-gcc。 如图 2-10 所示

🖨 Properties for MiniGUI					
type filter text	Settings	↔ · ⇒ ·	-		
Resource Builders ⊂ C/C++ Build Build Variables Discovery Option Environment Settings Tool Chain Edito ▷ C/C++ General MiniGUI Properties Project References Refactoring History Run/Debug Setting: ▷ Task Repository WikiText	 ▼ SGCC C Compiler Preprocessor Symbols Directories Optimization Debugging Warnings Warnings Miscellaneous S GCC C Linker General Libraries Miscellaneous Shared Library Settings S GCC Assembler General 	Command: gcc All options: -L/usr/local/lib Expert settings: Command line pattern: \${COMMAND} \${FLAGS} \${OUTF	×		
?		Cancel OK			

图 2-10

• 设置应用程序所需要的链接库路径以及库的名称,也就说在编译的时候,按照你指定的路径寻找库。 例如: pthread, minigui, mgncs 等等。如图 2-11 所示



图 2-11

- 选中工程名,右键,在弹出的菜单中选择 Build Project 编译工程。
- 编译好后的工程,目标文件在 Debug 目录下。

第三章 miniStudio 实例开发

实例开发

- PC 平台
- 君正 4740 平台

PC 平台的实例开发

新建工程

• 启动 eclipse,新建一个 HelloWorld 工程。如图 3-1,图 3-2,并在图 3-2 中 Project name 中添加工 程的名称,点击 Finish 按钮。

	New		×
Select a wizard New MiniGUI Application Project			
<u>W</u> izards:			
type filter text			
🕨 🗁 General			
▷ > c/c++			
🕨 🗁 CVS			
🗢 🗁 MiniGUI			
MiniGUI Application Project			
👂 🗁 Tasks			
(?)	< <u>B</u> ack <u>N</u> ext >	Cancel	Einish

图 3-1

e MiniGUI	Project
MiniGUI Project	-
Create MiniGUI project of selected type	
Project name: HelloWorld	
🗐 Lise default location	
Location: //home/fm-test/Install-Test/mStudio/HeiloWorld	Browse
Project type:	Toolchains:
▽ 🗁 MiniGUI Project	Linux GCC
MG 3.0.x Project With NCS	
Empty MiniGUI Project	
Show project types and toolchains only if they are supp	orted on the platform
	Next > Cancel Einish
-	



窗口设计

新建窗口

• 打开工程目录,在目录 res 中双击 res.project, 打开 guibuilder 界面,如图 3-3,图 3-4。



图 3-3



• 新建一个窗口,选择一个窗口,并给窗口命名 helloWorld,最后点击 ok。如图 3-5:

New File			×
Please select a wir	ndow templat	e:	
templates user—t	emplates		
Malt Frame FFF	Mah Frane E E I OK Cancel	OK Cancel	=
empty_window	dialog	dialog3	
Page1 Page2 Page3		GroupBox	•
4			
Input File Name:	helloWorld		
∟Overwrite exist	†11e	ОК	Cancel

图 3-5

• 选中窗口,设置窗口的大小,修改属性值 Width 和 Height 分别为 420,320,并设置窗口的标题,修 改属性值 Text 为"HelloWorld!",最后保存窗口。如图 3-6:

*[StartWnd]helloWorld.xml		_[Property Event	Renderer
HelloWorld!			Name	Value 📤
		-	ID	ID_MAINWND1
		>	x	0
	=	Ī	Y	0
		ŀ	Width	420
		ł	Height	320
	' [-	Text	HelloWorld!
		F	Renderer	0
		E	BgColor	0×FFCED3D6
		ľ	Notify	True
		E	Border	True
		١	Visible	True
	' L	E	Enabled	True
◀ III	▼ ↓	1	HasCaption	True

添加控件

• 添加静态框。用鼠标选 Toolbox 中 Label 控件,并将此控件拖至窗口中心,并适当地拖放 Label 控件 的大小。设置 Label 的 Text 为"Hello World !!",如图 3-7:

<pre>{ *[StartWnd]hell</pre>	loWorld.xml				Property Event	Renderer
HelloWorld!			×		Name	Value
					ID	ID_STATIC2
					х	67
				≡	Y	98
					Width	263
	Halla Hapld II				Height	68
	HEITO MOUTU !!		P		Text	Hello World !
•	•	•			Renderer	0
					BgColor	0×FFCED3D6
					Font	
					Notify	True
					Border	False
				Ш	Visible	True
∢ ₩			Þ	F	Enabled	True

图 3-7

设置字体

• 选中 Label 控件,点击属性列表中 font,弹出对话框,设置字体,最后点击 ok,如图 3-8 所示;字 体的设置效果图,如图 3-9 所示。最后保存。



图 3-8

[StartWnd]hell	oWorld.xml	
HelloWorld!		
	•	
1.11	Hello World !!	•
•	•	

图 3-9

加载图片

选中 Toolbox 中 Image 控件,并将此控件拖至窗口中适当位置,如图 3-10 所示;并点击 Image 属性列表中的 Image,弹出窗口,打开本地需载入的图片目录,如图 3-11 所示。点击鼠标右键弹出菜单,选择 import,弹出对话框,点击 import,如图 3-12 所示。选中需载入的图片,点击 ok,如图 3-13 所示。添加图片的效果,如图 3-14 所示。最后保存。

[*[StartWnd]helloWorld.xml			Property Even	t Renderer
HelloWorld!		*	Name	Value
• •			ID	ID_IMAGE3
• 対 •			х	7
· · ·		=	Y	8
Hello World !!			Width	175
		Н	Height	50
	P		Renderer	0
			BgColor	0×FFCED3D6
			Notify	True
			Border	False
			Visible	True
			Enabled	True
		TabStop	False	
▲ III III III III III III III III III I	}	◄	Align	Center
Navigator Struct View			Valign	Middle
• 1013 CCC	-	*	Image	•
et en			DrawMode	Normal
			Transparent	False

图 3-10



图 3-11

Import Image Resource	×
Input Image ID Name:	
IDB_MINIGUI_LOGO	
Import Cancel	





图 3-13

*[StartWnd]helloWorld.xml	_		Property [Event	Renderer
HelloWorld!		*	Name		Value
			ID		ID_IMAGE3
·MINIGUI·			x		7
Hello Horld !!		≡	Y		8
			Width		175
			Height		50
	•		Renderer		0
			BgColor		0xFFCED3D6
			Notify		True
			Border		False
			Visible		True
			Enabled		True
		TabStop		False	
I III III III III III III III III III	•	▼	Align		Center
Navigator Struct View			Valign		Middle
1.00m		*	Image		IDB_MINIGL()
Mum GU helloWorld.xml			DrawMode		Normal
			Transparen	t	False

图 3-14

预览窗口

• 选择界面设计窗口的菜单 Action 中的 Preview,如图 3-15。预览效果,如图 3-16。
File Edit	t Form	nat A	ction Window He	∋lp					
Minigul			Preview		Ctrl+Shift+P		<u> II 11</u> H	+ 🗱 👄	1
	Tool	xoc	Save Dialog Tem	nplate	Source	rld.xml			
	k	No	Set Default Ren	nderer	Ctrl+Shift+D				٩.
	₽	Lal	Set Screen Size	;	Ctrl+Shift+S				
	A	Lal	Connect Events		Ctrl+Shift+E	JI •			
	8	LEI Rei —	Font Management	:	Ctrl+Shift+F	•			
		Im:	Set As Start Wi	.ndow		lello Hr	arld II		
8	ΞI	Gr	Show and Snap G	arid	Ctrl+Shift+G				
		Horz	Separator =	■		1			
		Vert	Separator						
	₽	Butto	ons						
	ok	Push	Button						
		Check	Button						
		Radio	Button				•		
		Menu	Button						

图 3-15



编译运行

切换到 eclipse 界面,选中工程,点击编译按钮 ,注意查看是编译否有错误,如果没有则点
 击运行按钮 ○▼。运行效果图如图 3-17:



目标部署

• 选中工程,按鼠标右键,选择弹出菜单中 Export,弹出对话框。如图 3-18,图 3-19 所示。需要注意的是在图 3-19 中,需要选中 Target 的目标工程,并在 Target Information 中设置 IAL, GAL 和 bpp,以及 Resource Packagesr, Binary File 和 TargetDerictary。

		Export			2
elect					
elect an export destinat	ion:				
🕨 🗁 General					
> 🗁 C/C++					
🔻 🗁 MiniGUI					
💭 Deploy MiniGUI	Project				
> 🗁 Run/Debug					
> 🗁 Tasks					
> 🗁 Team					
	< <u>B</u> ack	<u>N</u> ext >	Cancel	Eir	nish

图 3-18

•	Deploy MiniGUI Project		×
Deploy Project			
Deploy MiniGUI Proj	act Resource		
🗹 😂 HelloWorld			
Target Information			
IAL pc_xvfb	GAL pc_xvfb Resolution(bpp) 16		•
Resource Packages	/home/fm-test/Install-Test/mStudio/HelloWorld/res/HelloWorld.res		Browse
Binary File	/home/fm-test/Install-Test/mStudio/HelloWorld/Debug/HelloWorld	-	Browse
Target Directory	/home/fm-test/Install-Test/mstudio/Target	•	Browse
?	Kenter State St		Einish
Ŭ			

注: PC 版目标部署的目的是使本机上的应用程序 Target 包在其他 PC(linux)上正常运行。

目标运行

切换到终端,进入 TargetDerictary 的路径后,在工程目录下执行./Debug/HelloWorld。目标工程运 • 行的效果。如图 3-20:

fm-test@fm-test-desktop: ~/Install-Test/miniStudio/Target\$ cd HelloWorld/

fm-test@fm-test-desktop:~/Install-Test/miniStudio/Target/HelloWorld\$ ls

Debug MiniGUI.cfg res

fm-test@fm-test-desktop:~/Install-Test/miniStudio/Target/HelloWorld\$./Debug/HelloWorld



图 3-20

说明: HelloWorld 目标部署的目录结构

-- Debug

可执行文件的目录;

可执行文件:

`-- HelloWorld

-- MiniGUI.cfg

配置文件;

-- mgncs.cfg

`— res	资源目录;
HelloWorld.res	
- image	图片目录;
`— minigui-logo.gif	加载的图片;

君正 4740 平台的实例开发

交叉编译

• PC 上实例开发完后,参考第二章搭建君正的交叉编译的环境,再进行交叉编译。交叉编译后在工程目录中生成交叉编译后的可执行文件目录:

fm-test@fm-test-desktop: //Install-Test/miniStudio/HelloWorld\$ ls

mipsel-linux-gcc Debug include MiniGUI.cfg Release res src

fm-test@fm-test-desktop:~/Install-Test/miniStudio/HelloWorld\$ cd mipsel-linux-gcc/

fm-test@fm-test-desktop:~/Install-Test/miniStudio/HelloWorld/mipsel-linux-gcc\$ ls

HelloWorld makefile objects.mk sources.mk src

目标部署

• 编译通过后,是目标部署。部署参考 PC 上部署,需注意的是在 Deploy MiniGUI Project 的对话框 中的设置,如图 3-21 所示。

	Deploy MiniGUI Project		×
Deploy Project			
Deploy MiniGUI Proje	act Resource		
l			
🗹 😂 HelloWorld			
Target Information:			
IAL custom	GAL fbcon Resolution(bpp) 16		•
Resource Packages	/home/fm-test/Install-Test/mStudio/HelloWorld/res/HelloWorld.res	-	Browse
Binary File	/home/fm-test/Install-Test/mStudio/HelloWorld/mipsel-linux-gcc/HelloWorld	-	Browse
Target Directory	/home/fm-test/CrossBuild/4740	•	Browse
Ø			Einish

注: 交叉编译后,目标部署中的 Binary File 的文件路径

• 切换到终端,进入 TargetDerictary 路径下的工程目录,修改 MiniGUI.cfg 文件中 gal_engine 的 defaultmode。

[system]

gal_engine=fbcon

defaultmode=480x272-16bpp

ial_engine=jz4740

mdev=/dev/input/mice

mtype=IMPS2

[fbcon]

defaultmode=480x272-16bpp

• • •

注意: minigui 的资源文件放置的位置应与 MiniGUI.cfg 文件中指定的路径一致。

目标运行

• 部署完了之后,将实例移植到君正 4740 上,到此君正 4740 实例实例开发完成。移植后的效果,如 图 3-22 所示。



图 3-22

实例下载

实例包下载:

HelloWorld.tar.gz: An example for eclipse on PC

http://wiki.minigui.com/twiki/pub/Products/MiniStudioSTSP3/HelloWorld.tar.gz

 $\operatorname{Copyright} {\textcircled{\sc opt}}$ by the Feynman Software. All contents is the property of Feynman Software.

第四章 miniStudio 实现多语言

不同字体输入功能。包括,中文,英文,繁体。

- 新建一个 MiniGUI 窗口,并且在窗口中画一个有 Text 属性的控件。例如: Label
- 选中 Label 控件,选择 Label 控件属性中的 Text 属性,会弹出 Input Text 对话框。如图:3-1 与图 3-2

about us		Name	Value
MiniGUI		ID	ID_STATIC2
		х	12
	=	Y	84
		Width	235
		Height	50
		- Text	
		Renderer	0
		BgColor	0×FFCED3D6
		Font	∗–unifont–brn∷
		Notify	True
		Bonden	False

图 3-1

about us 🗖 🗖 🗙		Name	Value
MiniGUI		ID	ID_STATIC2
Input Text	×	х	29
		Y	90
about us 🛰		Width	235
		Height	50
		Text	
		Renderer	0
		BgColor	0xFFCED3D6
		Font	*–unifont–brn:
UKCancel		Notify	True
		Border	False

 • 切换输入法,向 input text 对话框输入字符。如图: 3-3

about us 📃 🗖 🗙	*	Name	Value
MiniGUI		ID	ID_STATIC2
Input Text		х	29
家油 徳 田 mc + ud i o	F	Y	90
从建设用 mstudio		Width	235
-		Height	50
		Text	
		Renderer	0
		BgColor	0×FFCED3D6
		Font	*−unifont−brn∷
		Notify	True
		Border	False

图 3-3

翻译功能 实现多语言更容易

• 首先制作一个纯英文版本。如图: 3-4

about us		Name	Value
MiniGUI		ID	ID_MAINWND1
		x	0
	=	Y	0
		Width	323
about us		Height	310
		Text	about us
		Renderer	0
\$		BgColor	0×FFCED3D6
		Notify	True
		Border	True
		Visible	True

图 3-4

• 在 windows 菜单栏里选择 text 选项 切换到 text 视图。如图: 3-5

Minigul	🕒 🐴 ·	A 🎤		
_	Name	Id	Default Text	Current Text (en_US)
E		10002		about us
		10004		about us
		10006		
		10007		
		10008		
		10009		
8		10010		
		10011		

Text Window Help



• 在 text 菜单栏里选择 profile 选项 打开 profile 对话框。如图: 3-6

,			
	Text	Window	Help

Minigul	<u>₽</u>	🏂 🥕		
_	Name	Id	Default Text	Current Text (en_US)
E	Profile .			×
	langua	ige	status	Add
	en_US		current	= Delete
				Set Dewiult
			OK	Cancel

- 点击 add 按钮选择 Chinese(CN)语言,按 ok 按钮回到 profile 设置界面,选择 zh_CN 项点击 Set Current 按钮,
- 选择 en_US 项点击 set Default 按钮,程序总是把 default 项设置的语言,翻译成 current 项设置的语言。
- 如图: 3-7

Text Wind	dow Help					
Minigul	📔 🏄 d	26 🥕				
_	Name	Id	ault Text	(en_l	Current	Text (zh_CN)
	Profile .		8			×
	langua	age	status			Add
	en_US		default			
00	zh_CN		current			Delete
						et Default
						at Current
	4		Ш	•	E C	
				OK		Cancel

- 按 ok 按钮,回到 text edit 主界面,选择你想翻译的行,在 text 菜单栏里选择 translate 项,完成翻译,
- 也可选择 translate all 翻译全部 (如果你觉得翻译不准确还可以在 current text 栏里纠正)如图: 3-8

Text Wind	low He	1p						
New Text	t	<u>k</u> -	A 🎸					
Del Text	t	9	Id	ault Text	(en_l	Current	Text	(zh_CN)
Translat	te		10002	about us				
Translat	te All		10004	about us				
			10006					
Protile	•••		10007					
00			10008					
			10009					
<u>&</u>			10010					
			10011					

• 在 windows 菜单栏里选择 UI 选项 切换回 UI 视图,可以看到,翻译结果 如图: 3-9

关于我们			-	Name	Value
Mini	IGUI			ID	ID_STATIC2
				х	29
			=	Y	90
1.1	100 C			Width	235
1.1	关于我们	1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 -		Height	50
		• [Text	关于我们
				Renderer	0
				BgColor	0×FFCED3D6
				Font	*−unifont−brn:
				Notify	True

图 3-9

字体调整

- 点击 Font 属性打开字体设置对话框,Family 里可以设置字体类型, Style 栏里设置字体样式, Size 栏设置字体大小
- Charset 里设置字符集, Effects 栏里设置加斜, 下划线, 删除线等效果
- flip 栏里设置翻转效果 如图: 3-10 和 3-11

关于我们			Name	Value
MiniGUI			ID	ID_STATIC2
Font Selecting	4	×	х	29
Family:	Style:	Size:	Y	90
unifont 💌	Book 🔽	16 💌	Width	235
Effects			Height	50
Italic			Text	关于我们
Underline	Sample		Renderer	0
			BgColor	0xFFCED3D6
Flip None 💌			Font	*-unifont
Car	cel	Ok 🔰	Notify	True
			Border	False

关于我们			Name	Value
Mini	IGUI		ID	ID_STATIC2
			х	29
		=	■ Y	90
1.1	100 C	1.1	Width	235
- 1 -1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-	<u>、老王我们</u>	1.1	Height	50
	· · · · ·	•	Text	关于我们
		M3	Renderer	0
			BgColor	0×FFCED3D6
			Font	*−unifont−cin
			Notify	True
			Border	False

图 3-11

第五章 Connect Event 的实例应用-秒表

Connect Event 的介绍

本章我们将通过秒表的实例来学习如何使用 Connect Event。什么是 Connect Event?也就是将一个对象的 事件链接到任意一个对象上。

Connect Event 的应用

新建窗口

• 在 guibuilder 的窗口设计界面中新建一个窗口,并设置窗口 width, height, Border 和 HasCaption 等属性,如图 5-1 所示,再给添加窗口的背景图片,如图 5-2 所示。

*[StartWnd]watch.xml		Property Ev	vent	Renderer
	4	Name		Value
		ID		ID_MAINWND1
		Х		0
	=	Y		0
		Width		240
		Height		320
	\square	Text		Main Frame
		Renderer		0
		BgColor		0xFFCED3D6
		Notify		True
		Border		False
		Visible		True
		Enabled		True
	•	HasCaption		False

图 5-1



图 5-2

添加消息事件

• 选中窗口,打开右边的 Event 列表,添加 OnCreate 消息事件,如图 5-3 所示。最后保存。

Property	Event	Renderer	
Name	;	Value	-
onCreate		onCreate	
onSizeChar	ging		
onSizeChar	iged		
onCSizeCha	inged		
onFontChar	ging		
onFontChar	iged		
onEraseBkg	rnd		
onPaint			
onClose			
onKeyDown			
onKeyUp			_
onChar			
onSysKeyDo	ιwn		
onSysKeyUp	I		
onSysChar			
onKeyLongF	ress		

图 5-3

添加控件

• 添加 Label 控件。将 Label 拖放到适当的位置,并修改 Text 和 font,如图 5-4 所示。

*[StartWnd]watch.xml		Property	Event	Renderer
		Name	:	Value
		ID		ID_STATIC1
		х		80
	=	Y		144
		Width		79
		Height		23
	\square	Text		00:00
		Renderer		0
		BgColor		0×FFCED3D6
		Font		∗–unifont–brn∷
		Notify		True
		Border		False
a a	Ц	Visible		True
	F	Enabled		True
I III IIII III IIII III III	•	Enabled		True



• 添加 timer 控件。修改 Timer 属性中 Interval 为 100ms, 如图 5-5 所示。



图 5-5

添加 connect event 事件

 选中 Label 控件,选择 guibuilder 菜单 Action 中的 Connect Events,弹出对话框;再点击 Add 按 钮,弹出 Select Event 对话框,选择 timer 的 ID,以及 timer 的消息事件 OnTimer,点击 ok;选中 添加的 Connect Events 事件,点击 Source,生成代码。如图 5-6,图 5-7,图 5-8,图 5-9,图 5-10 所示。



图 5-6



图 5-7



图 5-8



图 5-9

```
💽 watch.c 🕺
  //$func @3690618880 onCreate -- Need by merge, don't modify
  static BOOL Mainwndl_onCreate (mWidget* self, DWORD dwAddData)
  {
       //TOD0:
2
       return TRUE;
  }
  //$handle @3690618880 -- Need by merge, don't modify
  static NCS_EVENT_HANDLER Mainwnd1_handlers [] = {
       {MSG_CREATE, Mainwnd1_onCreate},
🖉 //$user -- TODO add your handlers hear
       {-1, NULL}
  };
  //$func #3690618880 MSG TIMER 3750375424 4294787072 -- Need by merge, don't modify
  static BOOL staticl_on_timer1_timer (mStatic *self, mTimer* sender, int id, DWORD param)
2
      //TOD0:
       return TRUE; /* allow the event to go next */
  }
  //$connect #3690618880 -- Need by merge, don't modify
  static NCS_EVENT_CONNECT_INF0 Mainwnd1_connects[] = {
       {ID_TIMER1, ID_STATIC1, MSG_TIMER, (NCS_CB_ONOBJEVENT)static1_on_timer1_timer},
//$user -- TODO add your handlers hear
    {-1, -1, 0, NULL}
  };
  //$mainhandle -- Need by merge, don't modify
  NCS_EVENT_HANDLER_INFO mainwnd_Mainwnd1_handlers [] = {
       {ID_MAINWND1, Mainwnd1_handlers},
//$user --TODO: Add your handlers here
    {-1, NULL}
  };
```

图 5-10

添加代码

• 在 watch.c 文件中 OnCreate 和 static1_on_timer1_timer 函数中添加代码,如图 3-11 所示。

```
🖻 watch.c 🕺
  //$func @3690618880 onCreate -- Need by merge, don't modify
  static BOOL Mainwndl_onCreate (mWidget* self, DWORD dwAddData)
  {
      //TODO:
2
      mTimer * timer = (mTimer*)_c(self)->getChild(self, ID_TIMER1);
      if(timer)
          _c(timer)->start(timer);
      return TRUE;
  }
  //$handle @3690618880 -- Need by merge, don't modify
  static NCS_EVENT_HANDLER Mainwnd1_handlers [] = {
      {MSG_CREATE, Mainwnd1_onCreate},
  //$user - TODO add your handlers hear
1
      {-1, NULL}
  };
  //$func #3690618880 MSG_TIMER_3750375424_4294787072 -- Need by merge, don't modify
  static BOOL static1_on_timer1_timer (mStatic *self, mTimer* sender, int id, DWORD param)
  {
2
      //TODO:
      char szText[100];
      time t tim;
      struct tm *ptm;
      static int show dot=0;
      time(&tim);
      ptm = localtime(&tim);
      sprintf(szText, "%02d%s%02d",ptm->tm_hour, show_dot?" ": ":", ptm->tm_min);
      SetWindowText(self->hwnd, szText);
      show_dot = (show_dot+1)%2;
      return TRUE; /* allow the event to go next */
  }
```

图 5-11

编译运行

• 编译运行。运行的效果图,如图 3-12 所示。



图 5-12

实例下载

实例包下载:

• watch.tar.gz: An example for Connect Event

http://wiki.minigui.com/twiki/pub/Products/MiniStudioSTSP5/watch.tar.gz

第六章 数据绑定与数据源的应用

数据绑定与数据源的介绍

数据绑定和数据源是 mGNCS 提供给应用程序的两个非常重要的机制,这两个机制均有助于实现程序逻辑和它所处理的数据之间的分离,且便于类似 miniStudio 这样的可视化 GUI 设计工具来设计界面。

mGNCS 的数据源和数据绑定功能的思想来源是 VC++将对话框中的控件内容和给定的类成员变量绑定起来的一种机制。但是,mGNCS 提供的数据源和数据绑定功能更加强大。利用 mGNCS 的数据绑定功能,当 mGNCS 控件的值发生变化时,我们可以自动更新其他控件,或者将数据再次保存到期望的数据源中;通过 mGNCS 的数据源,我们可以定义不同格式的数据来源,如程序定义的字符串数组、文本文件、配置 文件,甚至数据库等等,并使用这些数据自动填充到 mGNCS 控件中。

数据绑定实例应用

数据绑定的功能

数据绑定是使图形用户界面和内部逻辑之间传递数据。其优点:

- 解耦图形用户界面和内部逻辑的处理,使开发人员更易于更换界面
- 规范化和模块化应用程序,提高程序的可扩展性
- 简化编程,把程序员从繁琐的 Get/Set 操作中解脱出来
- 统一接口,有利于 miniStudio 等工具进行可视化的操作,也有利于用户抽象

数据绑定的实例

下面这个实例就是将编辑框中的内容和拖动条的位置绑定在了一起:

- 拖动拖动条,编辑框中的内容将自动改变,反映当前的拖动条位置;
- 在编辑框中键入一个整数值(0~10),则拖动条的当前位置也将发生对应的变化。

如图 6-1 所示:





```
数据绑定的代码添加在窗口的 onCreate 中,如下:
```

static BOOL Mainwnd1_onCreate (mWidget* self, DWORD dwAddData)

{

//TOD0:

mTrackBar * tb = (mTrackBar*)_c(self)->getChild(self, ID_HTRACKBAR1);

mSlEdit * se = (mSlEdit*) _c(self)->getChild(self, ID_SLEDIT1);

ncsConnectBindProps(NCS_CMPT_PROP(tb, NCSN_TRKBAR_CHANGED, NCSP_TRKBAR_CURPOS, NCS_BT_INT, NCS_PROP_FLAG_READ | NCS_PROP_FLAG_WRITE),

NCS_CMPT_PROP(se, NCSN_EDIT_CHANGE, NCSP_WIDGET_TEXT, NCS_BT_STR, NCS_PROP_FLAG_READ|NCS_PROP_FLAG_WRITE),

NCS_BPT_DBL);

ncsAutoReflectObjectBindProps((mObject *)se);

return TRUE;

上面的代码段完成了两件事情:

- 调用 ncsConnectBindProps 函数,将拖动条的当前位置值属性(NCSP_TRKBAR_CURPOS)和 编辑框的文本属性(NCSP_WIDGET_TEXT)绑定在了一起,并且当拖动条产生位置改变事件 (NCSN_TRKBAR_CHANGED),或者编辑框产生内容改变事件(NCSN_EDIT_CHANGE)时触 发绑定。
- 调用 ncsAutoReflectObjectBindProps 函数,使得编辑框可以自动响应数据绑定带来的内容改变。

假设没有数据绑定功能,我们就需要编写两个事件处理器分别响应两个控件的两个事件,而现在,一切都 变得非常简单。

需要注意的是,拖动条的位置值属性是整数类型,而编辑框的内容属性是字符串类型,这两者之间的数据 类型转换,将由 mGNCS 自动完成。

数据源实例应用

数据源的功能

数据源就是数据的集合,通过抽象的数据源接口,我们可以为一些大型控件,如列表框、列表型等控件提供良好的数据交换机制。定义抽象的数据源接口之意义在于:

- 统一管理数据,是界面和数据分离
- 统一数据访问接口,便于程序的开发和维护,也便于 miniStudio 工具进行可视化处理

目前, mGNCS 实现了对如下几类数据源的支持:

- 应用程序定义的 C 语言数据, mGNCS 称为 Static Data Source (静态数据源);
- 来自 MiniGUI 配置文件格式的文本数据(ini 文件);
- 来自类似 UNIX passwd 文件的以行为单位的文本域数据(txt 文件)

静态数据源实例应用

使用 C 程序定义的静态数据初始化了一个列表项控件,如图 6-2 所示:

NameSexAgeJimeMale15LilyFemale12TomMale11	a Source			
NameSexAgeJimeMale15LilyFemale12TomMale11				
Jime Male 15 Lily Female 12 Tom Male 11	Name	Sex	Age	
Lily Female 12 Tom Male 11	Jime	Male	15	
Tom Male 11	Lily	Female	12	
	Tom	Male	11	
	1			

图 6-2

要实现上面的数据源功能,该程序完成了如下三个方面的工作:

```
{"Tom", "Male", "11"}
```

};

说明:上述代码分别定义了列表型控件的表头信息以及内容数据。

第二步,注册数据源。注册的代码在 DataSource_main.c 中紧接着 ncsInitialize()函数之后。

```
ncsRegisterStaticData ("/listview/header", (void*)_header, 3,
sizeof(NCS_LISTV_CLMRD)/sizeof(DWORD), sizeof(DWORD),
```

gListVColumnRecordTypes);

```
ncsRegisterStaticData ("/listview/content", (void*)_content, 3, 3, sizeof(char*), NULL);
```

说明:上述代码将上述两种数据分别定义成了 "listview/header" 和 "listview/content",在需要引用这 两种数据时,使用这里给出的名称即可。

第三步,使用数据源。此处的代码添加在窗口的 onCreate 函数中。

```
static BOOL Mainwnd1_onCreate (mWidget* self, DWORD dwAddData)
```

//TOD0:

{

mListView *lv = (mListView*)_c(self)->getChild (self, ID_LISTVIEW1);

if (1v) {

mRecordSet *rs;

rs = _c(g_pStaticDS)->selectRecordSet (g_pStaticDS, "/listview/header", NCS_DS_SELECT_READ);

_c(lv)->setSpecificData (lv, NCSSPEC_LISTV_HDR, (DWORD)rs, NULL);

rs = _c(g_pStaticDS)->selectRecordSet (g_pStaticDS, "/listview/content",
NCS DS SELECT READ);

```
_c(lv)->setSpecificData (lv, NCSSPEC_OBJ_CONTENT, (DWORD)rs, NULL);
}
return TRUE;
}
```

说明:上述代码从数据源中获取对应的数据,然后调用列表型控件的 setSpecificData 方法进行设置。

从创建 listView 的过程,我们只是用了只是这么几个函数和数据定义就创建一个完整的 ListView 控件。有此,我们可以看到采用数据源的方式免除了许多手工代码的麻烦。

实例包下载

数据绑定实例包下载:

• DataBinding.tar.gz: An example for data binding

http://wiki.minigui.com/twiki/pub/Products/MiniStudioSTSP6/DataBinding.tar.gz

数据源实例包下载:

• DataSource.tar.gz: An example for data source

http://wiki.minigui.com/twiki/pub/Products/MiniStudioSTSP6/DataSource.tar.gz

第七章 渲染器及其应用

渲染器介绍

渲染器分为: classic、fashion、flat和 skin 四种。它主要为用户提供了多种风格的主窗口和控件界面外观风格。应用程序在这几种风格的窗口界面之间进行切换非常容易,只要在创建窗口时传递不同的参数,你就可以变换出不同风格的界面。另外,还进一步统一了窗口元素的属性,如颜色、尺寸、字体等,通过简单的接口,应用程序就可以方便地控制窗口元素的上述属性。

渲染器的使用

渲染器和渲染器集的创建

通过属性新建渲染器

 选中要渲染的窗口或者控件。例如设置一个 button 控件的渲染器,点击 Property->Renderer 中的 "New Renderer"选项(如图: 7-1 所示),此时会弹出一个新建渲染器的对话框,可以选择渲染器的种 类以及对渲染器的"ID Name"进行命名(如图: 7-2 所示)

*[StartWnd]1.xml			Property	Event	Renderer	
Main Frame			Nam	e	Value	
			ID		IDOK	
		=	х		28	
			Υ		182	
			Width		80	
			Height		30	
			Text		ок	
			Renderer			▼
• ок	Cancel		BgColor		[New Render	rer
			Font			~
			Notify		True	
			Border		False	

Renderer	Type:	Classic 🔽
Control	Type:	Button
ID	Name:	IDR_

图 7-2

 新建渲染器后,在 Property->Renderer 中会出现该类型控件的渲染器(例如:你选中 button 控件, Renderer 中只会出现 button 类型的渲染器),选择该渲染器,就可以看到控件被渲染器后的效果。 如图: 7-3 所示

Main Frame 📃 🗖 🗙 -	*	Name	Value
		ID	IDOK
		х	128
	535	¥	83
		Width	160
<mark>е ок з</mark> е		Height	71
		Text	ОК
		Renderer	IDR_fashion_bເ▼
		BgColor	IDR_fashion_butt
		Font	inco nender er

图 7-3

- 修改渲染器属性
 - o 通过属性栏进行修改,选中被渲染过的控件或者窗口,点击属性栏中的"Renderer"。例如:选中"Ok" button,点击属性栏中的"Renderer",可以对 button 的 fashion 渲染器每个属性进行设置。如图 7-4 所示:

[StartWnd]1.xml	Property Event	Renderer
Main Frame 📃 🗖 📥	Name	Value
	ID	IDR_fashion_b
	ColorBorderInAct	0xFFC8D0D4
	ColorFg3DBody	0xFF000000
	ColorTextDisable	0xFF99A8AC
• OK •	ColorBg3DBody	0xFFFF3000
	ColorBgDisable	0xFFFFFFFF
	RoundX	з
	RoundY	3
	GradientMode	Vert

图 7-4

通过渲染器管理器,修改渲染器的属性,即修改渲染器的 Value 值,当修改 Value 值后,右
 边的控件也随着变化。如图 7-5 所示:

Minigul	📔 🚍 🎽 👿 💥 📮 🗖 🔜 👘						
	E-Root	*	Name	Value	*	<u> </u>	
			ID	IDR_fashion_b			
			ColorBorderInAct	0xFFC8D0D4		Button	
00			ColorFg3DBody	0xFF000000			
			ColorTextDisable	0xFF99A8AC			
00			ColorBg3DBody	0xFFFF3000			
			ColorBgDisable	0×FFFFFFFF			
8			RoundX	10			
			RoundY	10			
a (*			GradientMode	Vert			

图 7-5

• 删除渲染器

。 在渲染器管理器中,选中要删除的渲染器,可以通过鼠标右键或者工具栏的删除按钮删除渲染器。如图:7-6,7-7 所示

RenererSet	Renderer	Window	Help		
MiniGUI	🗐 🧮 🎽	Сору		Ctrl+C	
		Cut		Ctrl+X	A
Final		Delete	Э	Ctrl+D	BULL
	_			(*)	
00					
00					
				N	





图 7-7

通过渲染器管理器创建渲染器和渲染集

通过渲染器管理器创建渲染器

- 创建渲染器
 - 。 使用菜单创建渲染器。进入渲染器管理,点击菜单按钮的 Renderer->New Renderer,如图 7-8 所示:



图 7-8

此时会弹出一个新建渲染器的对话框,如图 7-9 所示:

×
Classic 💌
Animate
IDR_
Cancel



"Renderer Type"有四中类型渲染器: classic, flat, fashion 和 skin。 "Control Type"包括所有的控件,选择哪个控件,就会渲染到哪个控件或者窗口。例如,选择 button 控件,在渲染 button 控件的时候(参考通过属性创建渲染器),就直接可以使用该渲染器,不需要再新建了。 "ID name"是新建渲染器的名称

。 使用 ToolBar 创建渲染器。进入渲染器管理,点击 ToolBar 中的"新建按钮",如图 7-10 所示:



图 7-10

点击该按钮后,会弹出一个新建的渲染器的对话框,可以参考通过 Renderer->New Renderer 创建新的渲染器方法 * 使用鼠标右键创建渲染器。 选中 root 节点,然后点击鼠标右键,选择"New Renderer"。如图 7-11 所示:



图 7-11

当点击"New Renderer"选项后,会弹出新建属性对话框,根据需要可以选择不同渲染器和 控件,具体的可以参考通过 Renderer->New Renderer 创建新的渲染器方法

- 渲染器属性的修改。 具体可以参考上面的渲染器修改方法
- 渲染器删除。具体操作方法可以参考上述删除渲染器方法
- 渲染器复制。 选中要复制的渲染器,可以通过快捷键,菜单和鼠标右键来复制渲染器

	⊟—Root IDR_fas	hion_button <u>(Fashion.Butt</u>	n)
	Сору	Ctrl+C	
	Cut	Ctrl+X	
00	Delete	Ctrl+D	
00			

图 7-12

点击鼠标右键,点击"COPY"选项。如图 7-12 所示
RenererSet	Renderer 🕨	∤indow Help	
Minigui	New Rende	erer 🕴 🗔 🗔	
	Delete Re	enderer	A
	Сору	Ctrl+C	ton)
	Paste	Ctrl+V	
88			

图 7-13

点击菜单 Renderer->Copy。如图 7-13 所示



图 7-14

点击 ToolBar 中的复制按钮,如图 7-14 所示

• 渲染器剪切。选中要剪切的渲染器,可以通过快捷键,ToolBar,菜单和鼠标右键来剪切

	E-Root	shion_button	(Fashion.Button)	A
ton Q	Сору	Ctrl+C		
	Cut	Ctrl+X		
00	Delete	Ctrl+D		
00				

图 7-15

点击鼠标右键,点击"CUT"选项。如图 7-15 所示

RenererSet	Renderer W	Nindow Help		
MiniGUI	New Rende	erer 🎍	-	
	Delete Re	enderer		
	Сору	Ctrl+C	NO(Fashion.Button)	
	Paste	Ctrl+V		
88		21		

图 7-16

点击菜单 Renderer->Copy。如图 7-16 所示

•



点击 ToolBar 中的复制按钮。如图所示 7-17

• 渲染器的复制。完成复制或者剪切动作以后,选中"root"或者是渲染器集,可以通过鼠标右键,菜单和 ToolBar。



图 7-18

点击鼠标右键,点击"Paste",如图 7-18 所示:

RenererSet	Renderer W	lindow Hel	p	
Minigul	New Rende	rer		
	Delete Re	nderer	(Fashion.Button)	A
	Сору	Ctrl+C		
	Paste	Ctrl+V		
88				

图 7-19

通过菜单 Renderer->Paste 粘贴,如图 7-19 所示:

5	
图 7-20	

通过 ToolBar 粘贴渲染器. 如图 7-20 所示:

通过渲染器管理器创建渲染集

• 创建渲染器集。 可以通过菜单和鼠标右键来创建渲染器集



图 7-21

点击菜单 RendererSet->New RdrSet,如图 7-21 所示



图 7-22

选中"root"节点,点击鼠标右键,点击"New RdrSet"。如图 7-22 所示 当点击新建渲染器集选项后, 会弹出新建渲染器集的对话框。如图 7-23 所示:

Create New Renderer	Set [×
New Renderer Set		
Renderer Type:	Classic 💌	
Set ID Name:	IDRS_	
OK	Cancel	
(*****		

图 7-23

"Renderer Type"是渲染器类型,可以选择 skin, flat, fashion 和 classic 四中渲染器。"Set ID Name" 是渲染器集的名称,可以根据自己需要命名 * 渲染器集添加渲染器。可以通过鼠标右键和 Toolbar 按钮,添加渲染器

New Rende	Renderer	
Add Rende	Renderer	
Paste	Ctrl+V	
Delete	Ctrl+D	



选中要渲染器集,点击鼠标右键,点击"Add Renderer",如图 7-24 所示

Main Frame	X
IDR_d	
ОК	Cancel

图 7-25

当点击"Add Renderer"按钮后,会弹出添加渲染器对话框,对话中会显示出所有和渲染器集类型相同的渲染器。如图 7-25 所示 值得注意地方,当你选择某种渲染器集后,该渲染器集只能添加相同 类型的渲染器,同种类型控件渲染器,只能被添加一个。

• 渲染器集的删除。 可以通过菜单 RendererSet->delete 和 ToolBar 中的删除按钮。

New Rende	Renderer	
Add Rende	Add Renderer	
Paste	Ctrl+V	
Delete	Ctrl+D	

图 7-26

Copyright © by the Feynman Software. All contents is the property of Feynman Software.

选中要删除的渲染器集,点击 RendererSet->delete。如图 7-26 所示



图 7-27

选中要删除的渲染器集,点击 ToolBar 中的删除按钮。如图 7-27 所示



图 7-28

选中要删除的渲染器集,点击 ToolBar 中的删除按钮。如图 7-28 所示