

第34章 图形开发界面

- 
-
- ◆ 嵌入式GUI概述
 - ◆ MiniGUI DEMO
 - ◆ MiniGUI的编译和安装
 - ◆ MiniGUI
 - ◆ MiniGUI编程基础

嵌入式GUI概述

- **GUI(Graphics User Interface)**, 图形用户界面, 借助菜单、按钮等标准界面元素和鼠标进行操作, 帮助用户方便地向计算机系统发出指令, 并将系统运行的结果同样以图形方式显示给用户。
- **嵌入式GUI**: 轻型、占用资源少、高性能、高可靠性、可配置

	MiniGUI	MicroWindows	OpenGUI	QT/Embedded
可配置和可定制性	好，大量编译配置选项	一般	差	差
系统资源消耗	小	较大（基于UNIX套接字，进程间通讯）	最小（不支持多进程）	最大（C++）
效率	好	较差	最好	差
操作系统支持	Linux/uClinux, uC/OSII, VxWorks等	Linux	Dos、Linux、QNX	Linux
硬件平台支持	X86、ARM、MIPS、PowerPC	X86、ARM、MIPS	X86	X86、ARM
主要应用区域	中国大陆、台湾地区	美国，及国内少数用户	欧洲	欧美、韩国

5.3 MiniGUI概述

- MiniGUI (`:// minigui`) 原是由魏永明主持和开发的一个自由软件项目，现由北京飞漫软件技术有限公司维护并开展后续开发。
- 遵循GPL 条款
- 支持操作系统Linux、uClinux、VxWorks、eCos、uC/OS-II、pSOS、ThreadX 等
- MiniGUI 为应用程序定义了一组轻量级的窗口和图形设备接口。

MiniGUI 典型应用

- 手机/PDA



MiniGUI 典型应用

- 数字媒体/机顶盒



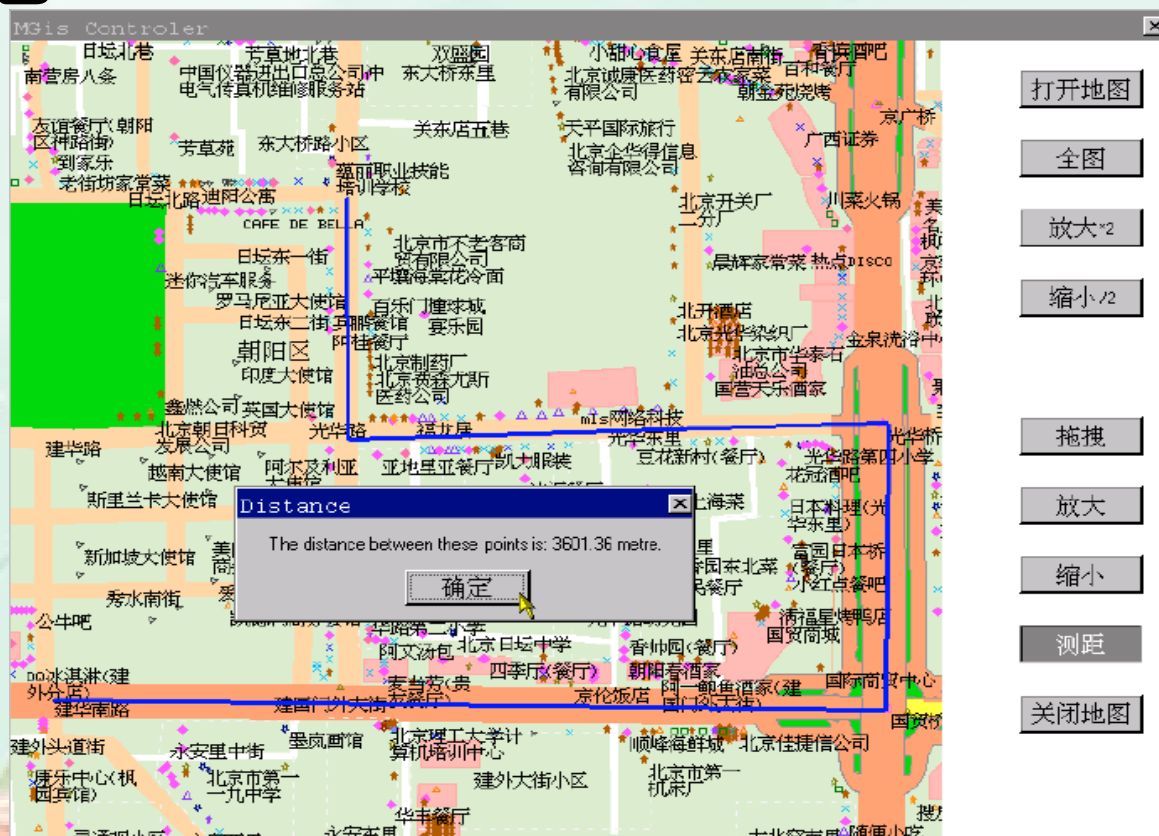
MiniGUI 典型应用

- 工控仪表



MiniGUI 典型应用

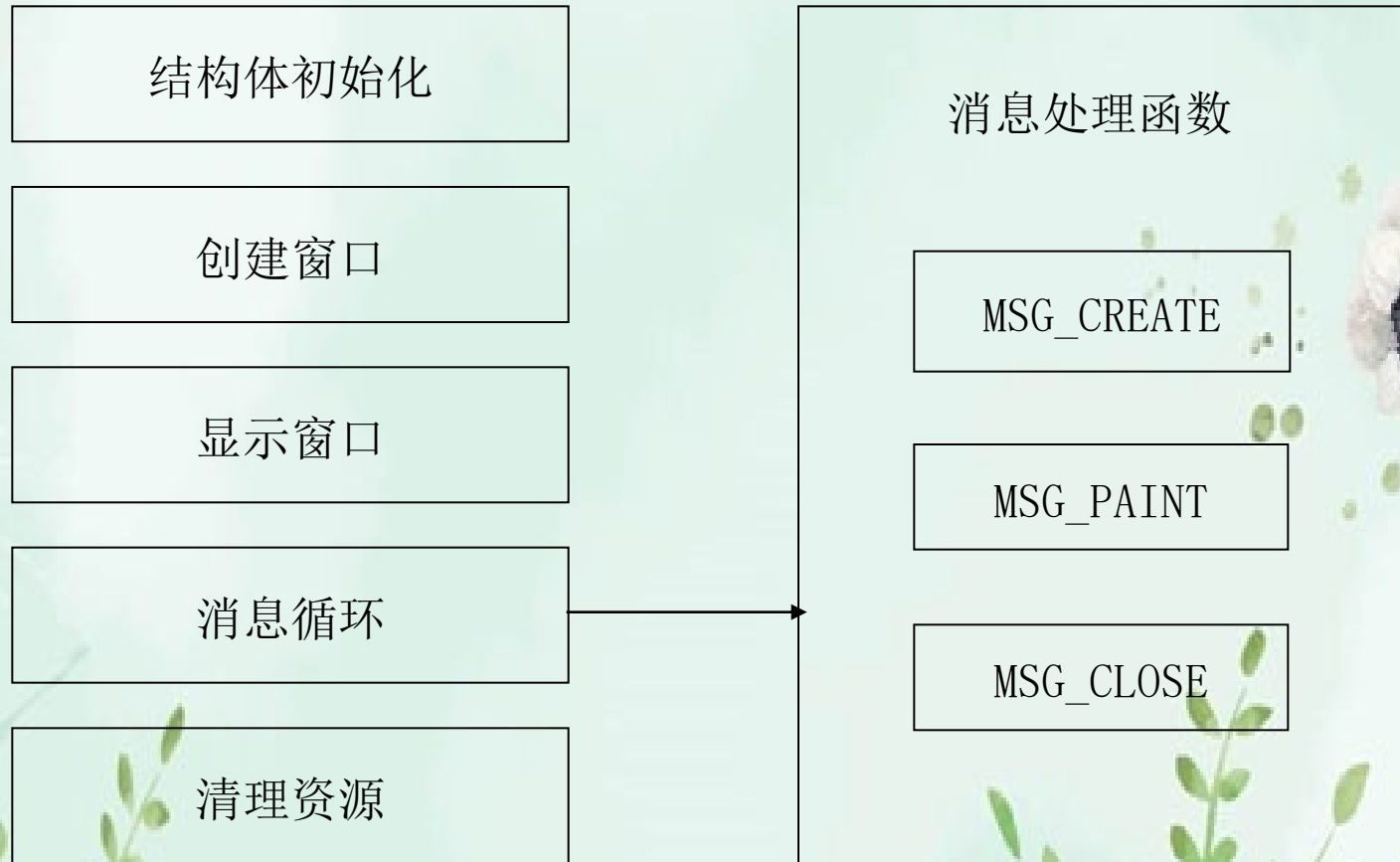
- 其他



miniGUI系统架构



程序架构分析：



程序架构分析：

1) 头文件（前四个为必须）

- `#include <minigui/common.h>`
 - `#include <minigui/minigui.h>`
 - `#include <minigui/gdi.h>`
 - `#include <minigui/window.h>`
 - `#include <minigui/control.h>`
-
- `common.h` 包括 MiniGUI常用的宏及数据类型的定义。
 - `minigui.h` 包含了全局的和通用的接口函数以及某些杂项函数的定义。
 - 包含了绘图函数的接口的定义。
 - `window.h` 包含了窗口有关的宏、数据类型、数据结构的定义以及函数接口声明。
 - `control.h` 包含了 所有内建控件的接口定义。

程序架构分析：

2) MiniGUI程序入口点

```
int MiniGUIMain (int argc, const char*  
argv[]) {  
    }
```

- 函数原形： `int MiniGUIMain (int argc, const char* argv[])`
- 参数argc 和argv 与C 程序main 函数的参数argc 和argv 的含义是一样的，分别为命令行参数个数和参数字符串数组指针。
- main 函数已经在 MiniGUI的函数库中定义了，该函数在进行一些 MiniGUI的初始化工作之后调用 MiniGUIMain函数。

程序架构分析：

3) 设置主窗口属性 ——MAINWINCREATE 结构描述

```
//窗口信息
CreateInfo.dwStyle = WS_VISIBLE | WS_BORDER |
WS_CAPTION; //风格
CreateInfo.dwExStyle = WS_EX_NONE;
CreateInfo.spCaption = "Hello, world!";
//标题名
CreateInfo.hMenu = 0;
//无菜单
CreateInfo.hCursor = GetSystemCursor(0);
CreateInfo.hIcon = 0;
CreateInfo.MainWindowProc = HelloWinProc;
//消息处理过程函数
CreateInfo.lx = 0;
CreateInfo.ty = 0;
CreateInfo.rx = 320;
CreateInfo.by = 240;
CreateInfo.iBkColor = COLOR_lightwhite;
CreateInfo.dwAddData = 0;
CreateInfo.hHosting = HWND_DESKTOP; //托管窗口
```

程序架构分析：

3) 设置主窗口属性 —— **MAINWINCREATE** 结构描述

- `CreateInfo.dwStyle = WS_VISIBLE | WS_BORDER | WS_CAPTION;`
- 设置主窗口风格，这里把窗口设为初始可见的，并具有边框和标题栏。
- `CreateInfo.deExStyle = WS_EX_NONE;`
- 设置主窗口的扩展风格，该窗口没有扩展风格。
- `CreateInfo.spCaption = “HelloWorld”`
- 设置主窗口的标题为” HelloWorld”。

程序架构分析：

3) 设置主窗口属性 ——MAINWINCREATE 结构描述

- `CreateInfo.hMenu = 0;`
- 设置主窗口的主菜单，该窗口没有主菜单。
- `CreateInfo.hCursor = GetSystemCursor(0);`
- 设置主窗口的光标为系统缺省光标。
- `CreateInfo.hIcon = 0;`
- 设置主窗口的图标，该窗口没有图标。

程序架构分析：

3) 设置主窗口属性 —— **MAINWINCREATE** 结构描述

`CreateInfo.iBkColor = PIXEL_lightwhite;`

- 设置主窗口的背景色为白色，
`PIXEL_lightwhite` 是 MiniGUI 预定义的像素值。
- `CreateInfo.dwAddData = 0;`
- 设置主窗口的附加数据，该窗口没有附加数据。
- `CreateInfo.hHosting = HWND_DESKTOP;`
- 设置主窗口的托管窗口为桌面窗口。

程序架构分析：

3) 设置主窗口属性 —— **MAINWINCREATE** 结构描述

- `CreateInfo.MainWindowProc = HelloWinProc;`
- 设置主窗口的窗口过程函数为 `HelloWinProc`，所有发往该窗口的消息由该函数处理。
- `CreateInfo.lx = 0;`
- `CreateInfo.ty = 0;`
- `CreateInfo.rx = 320;`
- `CreaetInfo.by=240;`
- 设置主窗口在屏幕上的位置，该窗口左上角位于(0, 0)，右下角位于(320, 240)。

程序架构分析：

4) 创建和显示主窗口

- //创建主窗口
- `hMainWnd = CreateMainWindow (&CreateInfo);`
- CreateMainWindow创建主窗口，参数是指向MAINWINCREATE结构的指针— CreateInfo，返回值为所创建的主窗口的句柄。
MAINWINCREATE 结构描述一个主窗口的属性。
- //显示主窗口
- `ShowWindow(hMainWnd, SW_SHOWNORMAL);`
- 显示窗口在屏幕上。

程序架构分析：

5) 进入消息循环

- MiniGUI 对 MiniGUI 程序维护一种名叫消息队列的数据结构。在发生事件之后，MiniGUI 将事件转换为一个消息，并将消息放入目标程序的消息队列之中。应用程序现在的任务就是执行消息循环代码，不断地从消息队列中取出消息，进行处理。

- //消息循环

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

程序架构分析:

6) 窗口过程函数

```
• static int HelloWinProc (HWND hWnd, int message, WPARAM wParam, LPARAM  
  lParam)  
• {   HDC hdc;  
•     RECT rc;  
•     char msg_text[30];  
•     switch (message) {  
•         //创建时发送的消息, 用来做初始化处理, 这里是初始化字符串内容  
•         case MSG_CREATE:                break;  
•         //窗口绘画时发送的消息, 用来在界面绘图或者输出文字  
•         //注意这里的return 0 表明不需要进入默认处理函数中。  
•         case MSG_PAINT:  
•             printf ("BeginPaint.\n");  
•             hdc = BeginPaint (hWnd);  
•             TextOut (hdc, 10, 50, , "Hello, world!");  
•             EndPaint (hWnd, hdc);  
•             printf ("EndPaint.\n");  
•             return 0;  
•         //结束时一些处理函数, 主要是销毁主窗口和发送quit消息以退出消息循环。  
•         case MSG_CLOSE:  
•             DestroyMainWindow (hWnd);  
•             PostQuitMessage (hWnd);  
•             return 0;  
•     }  
•     return DefaultMainWinProc (hWnd, message, wParam, lParam);
```

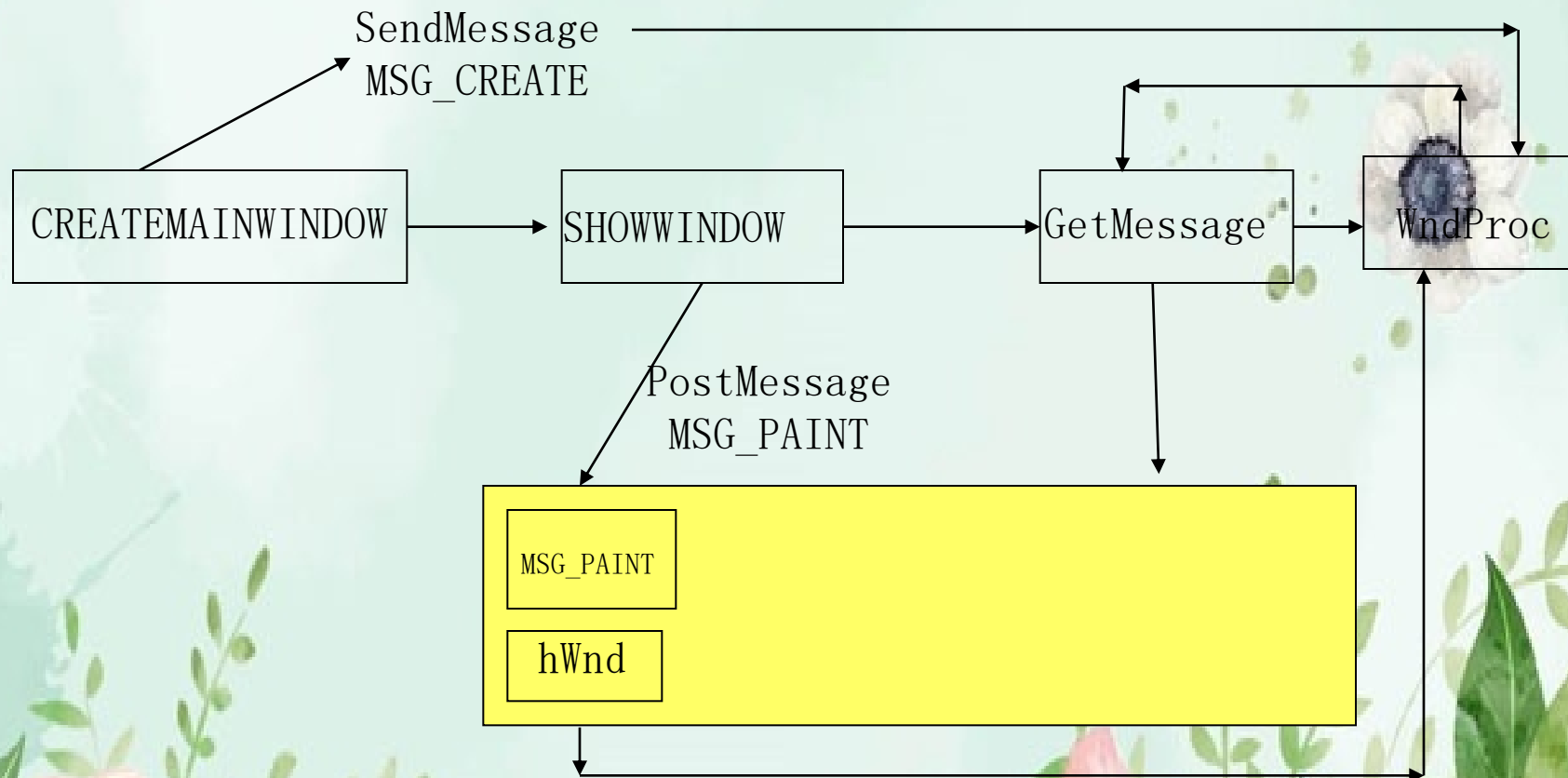
程序架构分析：

6) 窗口过程函数

- 在 程序中， 窗口过程是HelloWinProc。
- CreateMainWindow 函数根据 MAINWINCREATE 结构类型的参数中指定的窗口过程创建主窗口。
- 窗口过程函数由 MiniGUI进行调用。
-
- 窗口过程函数不予处理的消息应该传给 DefaultMainWinProc 函数进行缺省处理。

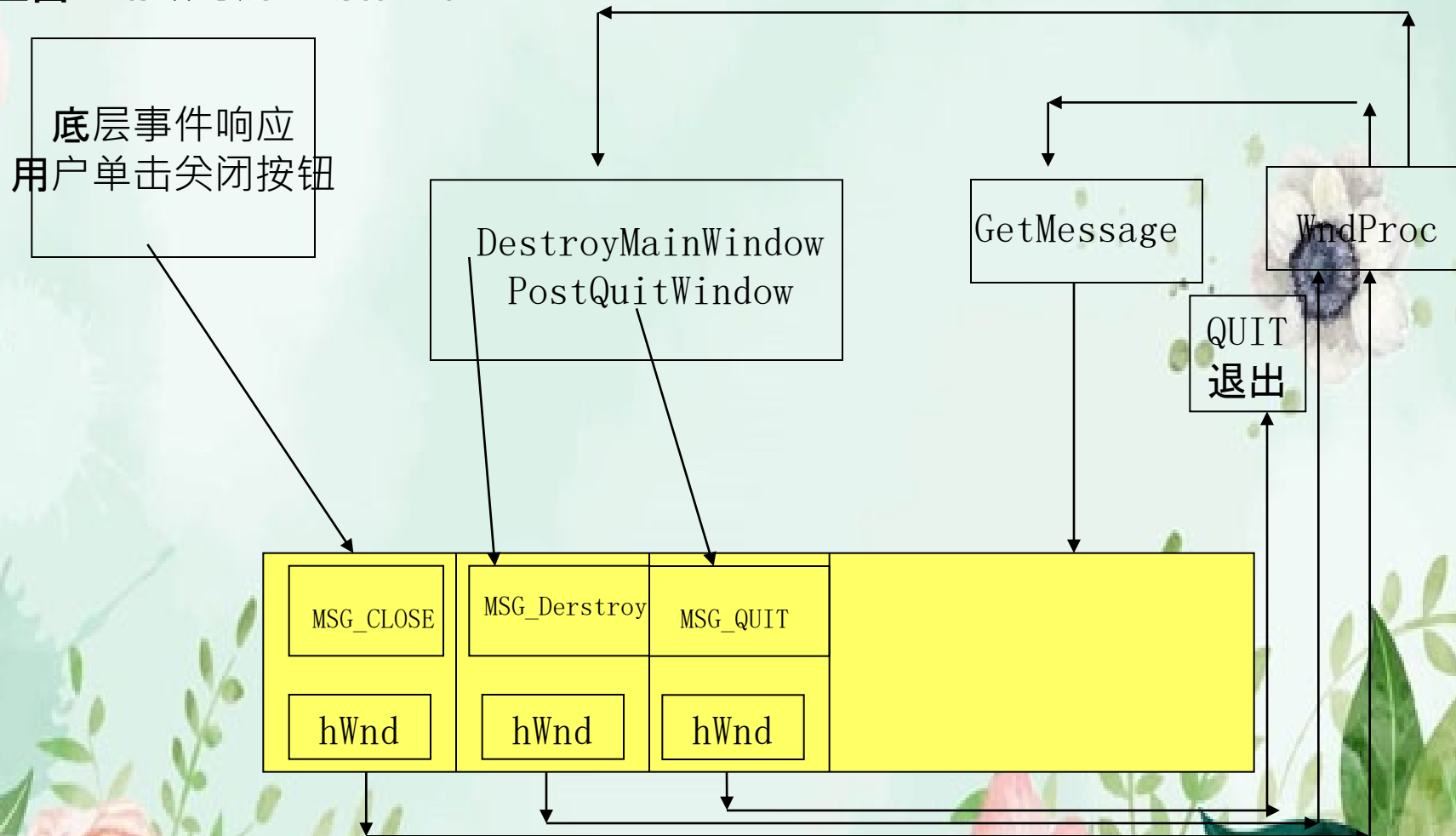
程序架构分析：

主窗口创建时消息动作过程：



程序架构分析：

主窗口销毁时消息动作过程：



程序架构分析：

7) 屏幕输出

- **MSG_PAINT**消息-进行屏幕输出。
- `case MSG_PAINT:`
 - `hdc = BeginPaint (hWnd);`
 - `TextOut (hdc, 60, 60, "Hello world!");`
 - `EndPaint (hWnd, hdc);`
 - `return 0;`

程序架构分析：

8) 程序的退出

- 单击窗口右上角关闭按钮，发送 **MSG_CLOSE** 消息。
- `case MSG_CLOSE:`
- `DestroyMainWindow (hWnd);`
- `PostQuitMessage (hWnd);`
- `return 0;`

程序架构分析：

9) 程序编译

- MiniGUI-Threads 模式：

- `$ gcc -o helloworld helloworld.c -lpthread -lminigui`

- pthread 线程库
- minigui minigui支持库

程序架构分析：

- 10) 程序执行
- **./helloworld**



MiniGUI函数库组成

- **libminigui** : 核心库，标准控件
- **libmgext** : 扩展库，一些有用的控件
- **libvcongui** : 虚拟控制台窗口

MiniGUI运行模式

- **MiniGUI-Threads**
- **MiniGUI-Lite**
- **MiniGUI-Standalone**
- **注意：**
 - 以下讲述基于MiniGUI1.6；
 - MiniGUI 2.0.x 中Lite模式升级为Processes模式。
 - 默认模式是MiniGUI-Threads。

MiniGUI 的主要特征(1/4):

- **1) 提供了完备的多窗口机制和消息传递机制。**
- **2) 提供常用的控件类。**
- **3) 对话框和消息框支持。**
- **4) 其它 GUI 元素，包括菜单、加速键、插入符、定时器等。**
- **5) 界面皮肤支持。**

MiniGUI 的主要特征(2/4):

- 6) 支持低端显示设备（比如单色 **LCD**）和高端显示设备（比如彩色显示器。
- 7) **Windows** 的资源文件支持，如位图、图标、光标等。
- 8) 各种流行图像文件的支持。
- 9) 多字符集和多字体支持。
- 10) 多种键盘布局的支持。

MiniGUI 的主要特征(3/4):

- 11) 汉字（**GB2312**）输入法支持。
- 12) 针对嵌入式系统的特殊支持，包括一般性的 I/O 流操作，字节序相关函数等。
- 13) 层的支持。（**MiniGUI-Lite** 版本）。
- 14) 提供快速和增强的位块操作，视频加速支持以及 **Alpha** 混合等功能。

MiniGUI 的主要特征(4/4):

- **15)** 增强的新 **GDI** 函数。
- **16)** 图形抽象层（**GAL**）以及输入抽象层（**IAL**）。

MiniGUI 优势：

- 轻型、占用资源少
- 高性能
- 高可靠性
- 可伸缩性强
- 跨操作系统支持
- 可配置

MiniGUI编程基础

- MiniGUI窗口
- MiniGUI消息及消息队列
- MiniGUI对话框
- MiniGUI菜单
- MiniGUI基本控件
 - 按钮
 - 静态框
 - 编辑框

窗口

- 窗口是屏幕上的一个矩形区域。应用程序的可视部分由一个或多个窗口构成。窗口系统控制窗口的位置、大小、和可见区域。
- 一个应用程序窗口一般包括如下部分：
 - 一个可视的边界
 - 一个窗口ID，客户程序使用该ID操作窗口，“窗口句柄”
 - 一些其它特性：高、宽、背景色等
 - 可能有菜单和滚动条等附加窗口元素

窗口

- 屏幕上的重叠窗口通常具有如下的关系：



窗口一般组织为层次体系结构的形式（或者说，树的形式）。

根窗口（root window）是所有窗口的祖先，占满整个屏幕的表面，也称

为桌面窗口。

除了根窗口以外的所有窗口都有父窗口，每一个窗口都可能有子窗口、

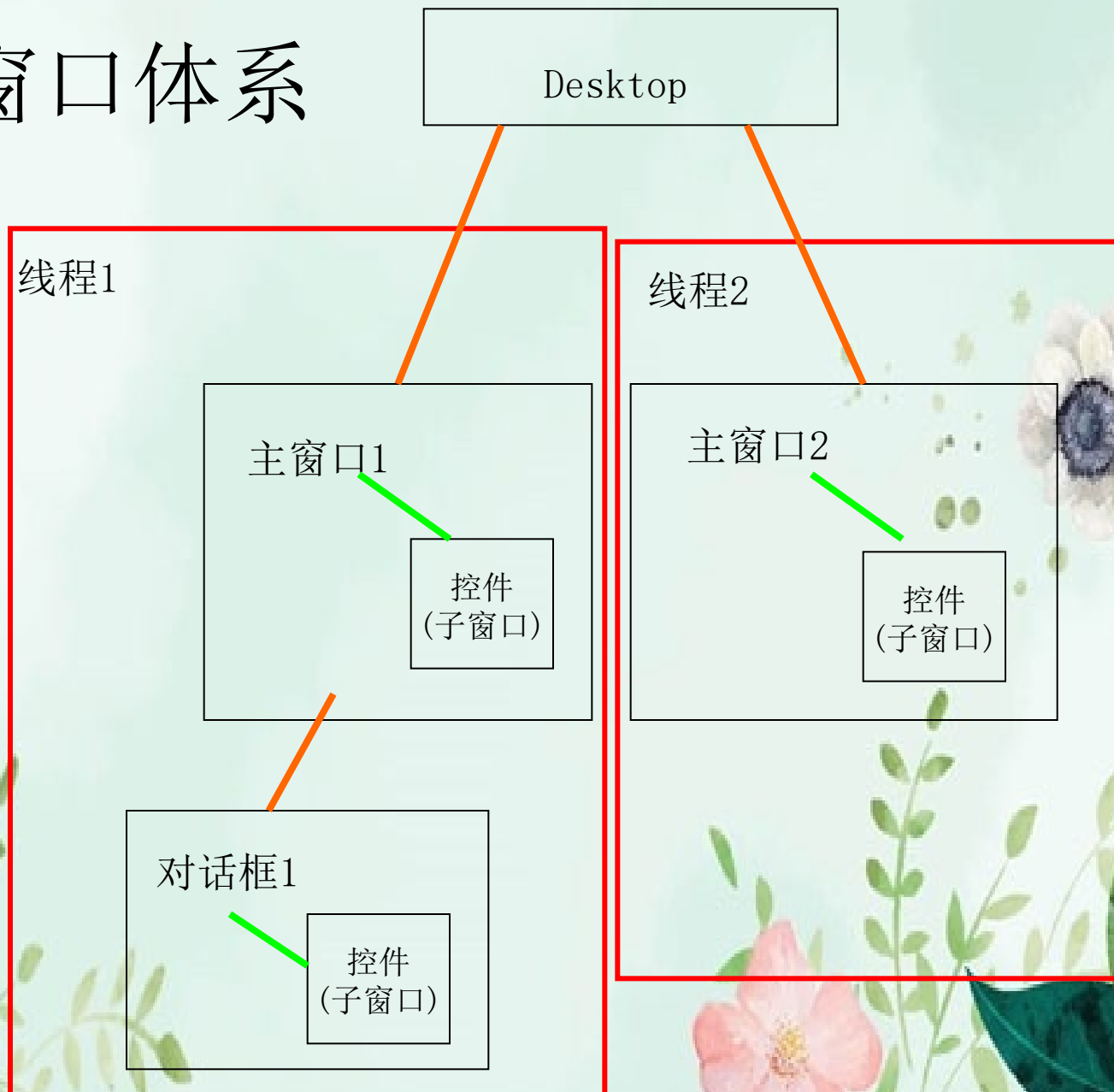
兄弟窗口、祖先窗口和子孙窗口等

子窗口含在父窗口内，同一个父窗口内的子窗口为同级窗口。

窗口有从属关系，也就是说，某些窗口的生命周期和可见性由它的所有

者决定。父窗口通常拥有它们的子窗口。

窗口体系



主窗口：

- MiniGUI 中有三种窗口类型：主窗口、对话框和控件窗口（子窗口）。
- 每一个MiniGUI应用程序一般都要至少创建一个主窗口，作为应用程序的主界面或开始界面。
- 主窗口通常包括一些子窗口，这些子窗口通常是控件窗口，也可以是自定义窗口类。
- 应用程序还会创建其它型的窗口，例如对话框。对话框本质上就是主窗口，应用程序一般通过对话框提示用户进行输入操作。

主窗口：

- 主窗口没有很精确的定义，目前可以认为其特征主要有父窗口为**Desktop**的，拥有子窗口，在创建后可以进入消息循环。
- **MiniGUI** 为每个托管窗口为**Desktop**的主窗口建立单独的消息队列，在该主窗口上派生出的主窗口、对话框及控件（子窗口）均使用同一消息队列。
- 在调用 **MiniGUIMain**之前，**MiniGUI** 启动桌面窗口**Desktop**。桌面窗口作为 **MiniGUI** 的窗口管理器而存在，也是事实上根窗口。

主窗口属性：

- **MAINWINCREATE**结构体指定窗口信息：
- 窗口风格
- 窗口标题
- 窗口菜单
- 窗口图标
- 窗口背景色
-

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/478060033036006036>