MiniGUI 界面设计一例

（陈云川 ybc2084@163.com UESTC,CD 2007-4-4）

1 引

我这里采用的是 MiniGUI 的非商业版本——MiniGUI ver 1.3.3。由于该版本的 MiniGUI 采用 GPL 条款发布，故我不必担心侵权之嫌。老实说，学习 MiniGUI 编程方法的过程是颇有一点痛苦的，但是在窥探明白其门径之后，我不得不说，我已经喜欢上了这个界面系统。

话休絮烦，这里给出一个我刚拼好的界面，如图 1 所示。其作用很简单，实时采集 GPS 数据，并在电子地图上显示，同时能够上下左右平移电子地图，也能够实时显示当前的经纬度信息。本来我还想把地图缩放之类的功能加进去的，但是限于开发板上的 Flash 空间实在是捉襟见肘，放不下那么多地图，于是只好作罢。所以，你会发现图上显示的【Zoom Out】和【Zoom In】两个按钮是灰色的。右边的 map 框内显示的红点是当前的位置，实际上的位置应该是在成都市一环路上，但是在这副图上存在一定的误差。

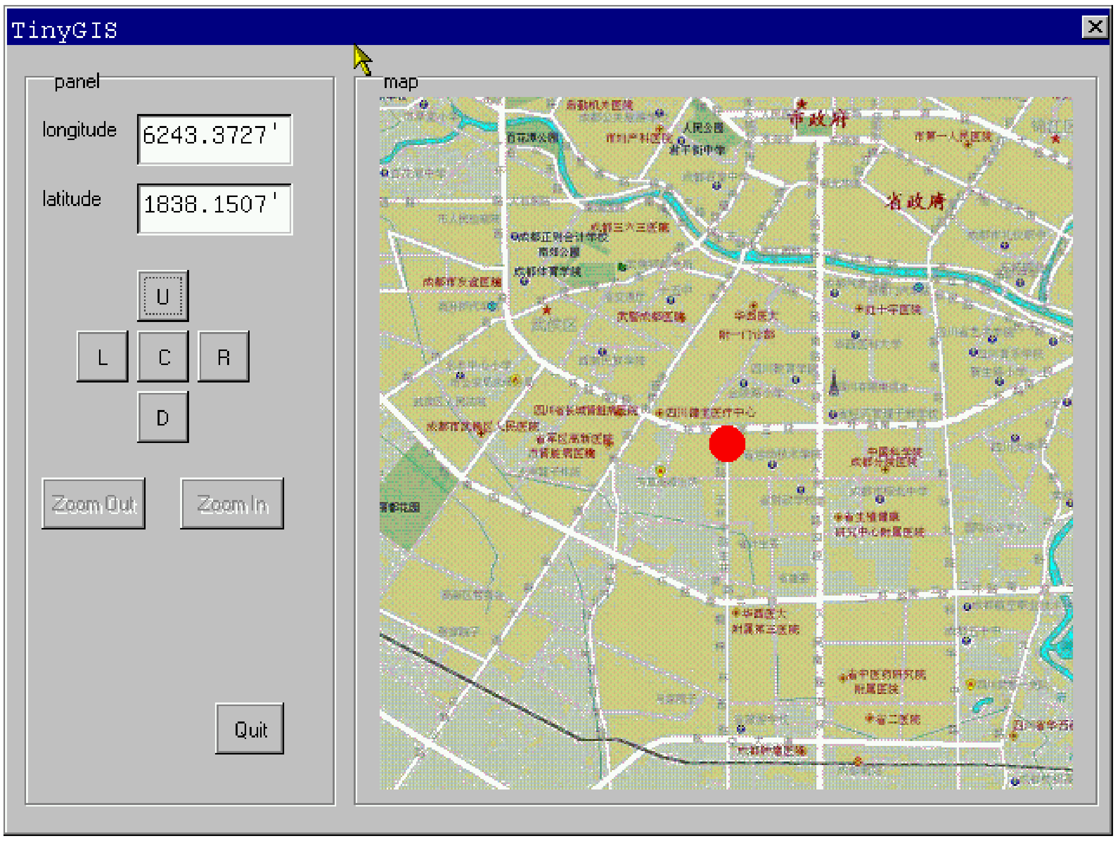

图 1 基于 MiniGUI 的 TinyGIS 界面

只要是曾经关注过我的博客的朋友都知道，这是我去年参加 Intel 杯电子设计竞赛时候的作品的一个完全翻版。但是不同的是，当时用的是 Windows CE，而现在我用的是 ARM-Linux+MiniGUI。

既然 MiniGUI 都是在 GPL 下发布的，那么本程序的源代码自然也是以 GPL 发布的。所以，下面给出完整的源代码。

2 源代码

首先是头文件 gui.h：

```
/*
 * $file    :     gui.h
 * $desc  :
 * $author     :      rockins(ybc2084@163.com)
 * $date  :
 * $copyright  :      all copyrights(c) reserved by rockins.
 */

#ifndef    _GUI_H_
#define    _GUI_H_

#define    IDC_L_BTN          100                              // left shift button
#define    IDC_R_BTN          101                              // right shift button
#define    IDC_U_BTN          102                              // up shift button
#define    IDC_D_BTN          103                              // down shift button

#define    IDC_ZOOMOUT_BTN 104                                     // zoom out(fang da) button
#define    IDC_ZOOMIN_BTN      105                                 // zoom in(suo xiao) button
#define    IDC_CENTER_BTN      106                                 // concentrate to center button
#define    IDC_QUIT_BTN    107                                 // quit button

#define    IDC_PANEL_STC 200                                   // panel for other widgets
#define    IDC_MAPPANEL_STC201                                 // panel for map
#define    IDC_LON_STC            202                              // longitude static text box
#define    IDC_LAT_STC            203                              // latitude static text box

#define    IDC_LON_EDT            300                              // longitude edit
#define    IDC_LAT_EDT            301                              // latitude edit

//#define  IDC_MAP_BMP            400                              // raster map bitmap

typedef struct _RASTER_MAP
{
#define    MAP_FILE_LEN    100                             // map file path len
       char *bmpname;                                   // .bmp file name
       BITMAP map;                                      // map object
       unsigned int tw, th;                          // total width and height
       unsigned int lx, ty;                          // left x and top y
       unsigned int dw, dh;                             // display width and height
}RASTER_MAP;

#endif
```

接下来是相应的实现，全部位于 gui.c 中：

```
/*
  * $file   :       gui.c
  * $desc  :
  * $author      :       rockins(ybc2084@163.com)
  * $date  :
  * $copyright :       all copyrights(c) reserved by rockins.
  */

#include <stdio.h>
#include <stdlib.h>

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

#include "gui.h"

HWND          hMainWnd;                                  // main window
RASTER_MAP      * pMap = NULL;                                  // map

//
// shift button(left, right, up, down, center) notification
//
static void
ShiftBtnNotifyProc(HWND hWnd, int id, int nc, DWORD add_data)
{
        if (id == IDC_CENTER_BTN && nc == BN_CLICKED) {
                pMap->lx = (pMap->tw - pMap->dw) * 0.5f;
                pMap->ty = (pMap->th - pMap->dh) * 0.5f;
        }

        if (id == IDC_L_BTN && nc == BN_CLICKED) {
                if ((signed int)pMap->lx - 10 > 0)
                        pMap->lx -= 10;
                else
                pMap->lx = 0;
        }
        if (id == IDC_R_BTN && nc == BN_CLICKED) {
                if (pMap->lx + pMap->dw + 10 < pMap->tw)
                        pMap->lx += 10;
                else
                        pMap->lx = pMap->tw - pMap->dw;
        }
        if (id == IDC_U_BTN && nc == BN_CLICKED) {
                if ((signed int)pMap->ty - 10 > 0)
                        pMap->ty -= 10;
                else
                        pMap->ty = 0;
        }
        if (id == IDC_D_BTN && nc == BN_CLICKED) {
                if (pMap->ty + pMap->dh + 10 < pMap->th)
                        pMap->ty += 10;
                else
                        pMap->ty = pMap->th - pMap->dh;
        }

        SendMessage(hMainWnd, MSG_PAINT, 0, 0);
}

//
// quit button notification
//
static void
QuitBtnNotifyProc(HWND hWnd, int id, int nc, DWORD add_data)
{
        if (id == IDC_QUIT_BTN && nc == BN_CLICKED) {
                PostMessage(hMainWnd, MSG_CLOSE, 0, 0);
        }
}

//
// create controls in main window
//
static int
CreateControls(HWND hMainWnd)
{
        HWND    hPanelStcWnd;
        HWND    hLonStcWnd;
        HWND   hLatStcWnd;
        HWND   hLonEdtWnd;
        HWND   hLatEdtWnd;
        HWND   hLBtnWnd;
        HWND   hRBtnWnd;
        HWND   hUBtnWnd;
        HWND   hDBtnWnd;
        HWND   hZoomOutBtnWnd;
        HWND   hZoomInBtnWnd;
        HWND   hCenterBtnWnd;
        HWND   hQuitBtnWnd;
        HWND   hMapPanel;

        hPanelStcWnd = CreateWindow(CTRL_STATIC, "panel",
                        SS_GROUPBOX | WS_CHILD | WS_VISIBLE,
                        IDC_PANEL_STC, 10, 10, 180, 430, hMainWnd, 0);

        hLonStcWnd = CreateWindow(CTRL_STATIC, "longitude",
                        SS_LEFT | WS_CHILD | WS_VISIBLE,
                        IDC_LON_STC, 20, 40, 50, 30, hMainWnd, 0);

        hLatStcWnd = CreateWindow(CTRL_STATIC, "latitude",
                        SS_LEFT | WS_CHILD | WS_VISIBLE,
                        IDC_LAT_STC, 20, 80, 50, 30, hMainWnd, 0);

        hLonEdtWnd = CreateWindow(CTRL_EDIT, "",
                        ES_READONLY | WS_CHILD | WS_BORDER | WS_VISIBLE,
                        IDC_LON_EDT, 75, 40, 90, 30, hMainWnd, 0);

        hLatEdtWnd = CreateWindow(CTRL_EDIT, "",
                        ES_READONLY | WS_CHILD | WS_BORDER | WS_VISIBLE,
                        IDC_LAT_EDT, 75, 80, 90, 30, hMainWnd, 0);

        hUBtnWnd = CreateWindow(CTRL_BUTTON, "U",
                        BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE,
                        IDC_U_BTN, 75, 130, 30, 30, hMainWnd, 0);

        hLBtnWnd = CreateWindow(CTRL_BUTTON, "L",
                        BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE,
                        IDC_L_BTN, 40, 165, 30, 30, hMainWnd, 0);

        hCenterBtnWnd = CreateWindow(CTRL_BUTTON, "C",
                        BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE,
                        IDC_CENTER_BTN, 75, 165, 30, 30, hMainWnd, 0);

        hRBtnWnd = CreateWindow(CTRL_BUTTON, "R",
                        BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE,
                        IDC_R_BTN, 110, 165, 30, 30, hMainWnd, 0);

        hDBtnWnd = CreateWindow(CTRL_BUTTON, "D",
                        BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE,
                        IDC_D_BTN, 75, 200, 30, 30, hMainWnd, 0);

        hZoomOutBtnWnd = CreateWindow(CTRL_BUTTON, "Zoom Out",
                        BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_DISABLED,
                        IDC_ZOOMOUT_BTN, 20, 250, 60, 30, hMainWnd, 0);

        hZoomInBtnWnd = CreateWindow(CTRL_BUTTON, "Zoom In",
                        BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_DISABLED,
                        IDC_ZOOMIN_BTN, 100, 250, 60, 30, hMainWnd, 0);

        hQuitBtnWnd = CreateWindow(CTRL_BUTTON, "Quit",
                        BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE,
                        IDC_QUIT_BTN, 120, 380, 40, 30, hMainWnd, 0);

        hMapPanel = CreateWindow(CTRL_STATIC, "map",
                        SS_GROUPBOX | WS_CHILD | WS_VISIBLE,
                        IDC_MAPPANEL_STC, 200, 10, 430, 430, hMainWnd, 0);

        // set notification callback for essential buttons
        SetNotificationCallback(hLBtnWnd, ShiftBtnNotifyProc);
        SetNotificationCallback(hRBtnWnd, ShiftBtnNotifyProc);
        SetNotificationCallback(hUBtnWnd, ShiftBtnNotifyProc);
        SetNotificationCallback(hDBtnWnd, ShiftBtnNotifyProc);
        SetNotificationCallback(hCenterBtnWnd, ShiftBtnNotifyProc);
        SetNotificationCallback(hQuitBtnWnd, QuitBtnNotifyProc);

        return (0);
}

//
// load raster map
//
static int
LoadMap(HDC hdc, RASTER_MAP * map)
{
        return LoadBitmap(hdc, &map->map, map->bmpname);
}

//
// unload raster map
//
static void
Unloadmap(RASTER_MAP * map)
{
        UnloadBitmap(&map->map);
}

static int
MainWinProc(HWND hWnd, int message, WPARAM wParam, LPARAM lParam)
{
        switch (message) {
                case MSG_CREATE:
                        CreateControls(hWnd);
                        return (0);
                case MSG_CLOSE:
                        DestroyMainWindow(hWnd);
                        DestroyAllControls(hWnd);
                        PostQuitMessage(hWnd);
                        break;
                case MSG_SIZECHANGING:
                {
                        const RECT * rcExpect = (RECT *)wParam;
                        RECT * rcResult = (RECT *)lParam;

                        rcResult->left = 0;
                        rcResult->top = 0;
                        rcResult->right = 640;
                        rcResult->bottom = 480;
                        return (0);
                }
                case MSG_PAINT:
                {
                        HDC hdc = BeginPaint(hWnd);

                        // map not loaded, load first
                        if (pMap == NULL) {
                                pMap = (RASTER_MAP *)malloc(sizeof(RASTER_MAP));
                                pMap->bmpname = (char *)malloc(MAP_FILE_LEN * sizeof(char));
```

```
                    strncpy(pMap->bmpname, "res/chengdu_map.bmp", MAP_FILE_LEN);
                    pMap->lx = 0;
                    pMap->ty = 0;
                    pMap->dw = 400;
                    pMap->dh = 400;
                    LoadMap(hdc, pMap);
                    pMap->tw = pMap->map.bmWidth;
                    pMap->th = pMap->map.bmHeight;
                }
                FillBoxWithBitmapPart(hdc, 215, 30, pMap->dw, pMap->dh,
                        0, 0, &pMap->map, pMap->lx, pMap->ty);

                EndPaint(hWnd, hdc);
                break;
            }
            default:
                break;
        }

        return (DefaultMainWinProc(hWnd, message, wParam, lParam));
}

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

#ifdef    _LITE_VERSION
        SetDesktopRect(0, 0, 640, 480);
#endif

        CreateInfo.dwStyle = WS_VISIBLE | WS_BORDER | WS_CAPTION;
        CreateInfo.dwExStyle = WS_EX_NONE;
        CreateInfo.spCaption = "TinyGIS";
        CreateInfo.hMenu = 0;
        CreateInfo.hCursor = GetSystemCursor(0);
        CreateInfo.hIcon = 0;
        CreateInfo.MainWindowProc = MainWinProc;
        CreateInfo.lx = 0;
        CreateInfo.ty = 0;
        CreateInfo.rx = 640;
        CreateInfo.by = 480;
        CreateInfo.iBkColor = COLOR_lightgray;
        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);
}
```