CVR MINIGUIUI 开发说明

Sep.27, 2016

林清文

Agenda

- MINIGUI 和QT比较
- APP和LIBMINIGUI框架
- 显示说明
- 消息说明
- 菜单
- 参数保存
- 对话框创建
- 创建等待显示
- Q&A

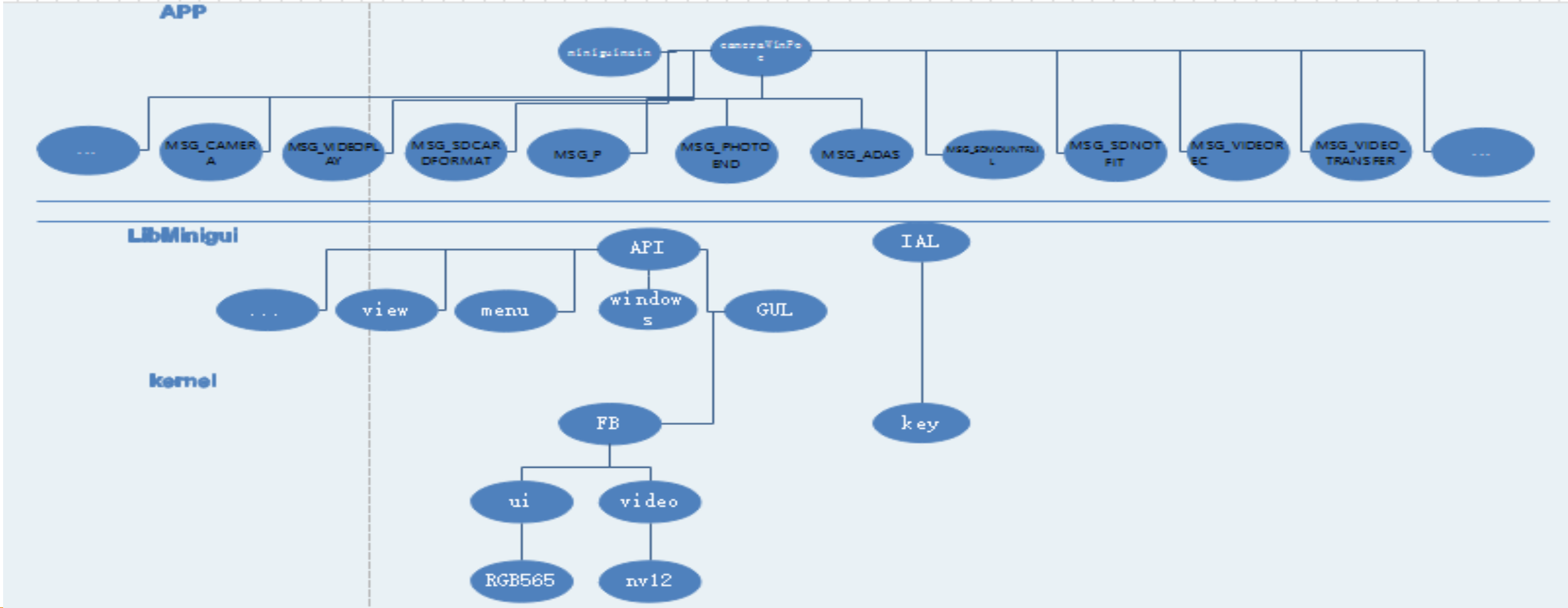
MINIGUI和QT比较

MINIGUI和QT比较

- QT当初是为PC设计的桌面环境，而且架构过于复杂，很难进行系统裁剪，扩充、定制和移植。
- 因为对硬件加速支持的匮乏，很难应用到对图形速度、功能和效率要求较高的实时性嵌入式系统，因此QT/E大多运行在strongARM,xscale的IPAQ等之上。
- 可应用于汽车DVR，运动DV，安防摄像机，无人机摄像等设备。
- MiniGUI跨多种操作系统的支持。目前MiniGUI已支持LINUX、UCLINUX、eCOS、VxWORKS、threadx,nucleus和UC/OS-II等主流的嵌入式操作系统。QT主要用于LINUX.
- 多语种尤其是中文的支持。MiniGUI的语言支持是做得非常好的，它能支持各种语言。QT也提供了中文的支持，但是还需要做很多工作，搞不好会影响整个项目的成败。
- MiniGUI提供了从用户使用手册、编程指南，再到API各个环节的文档齐全，对开发人员来说，可谓是得心应手！

APP和LIBMINIGUI框架

APP和LIBMINIGUI框架



显示说明

显示说明

- Mingui的所有显示都是在camera_ui.c中的cameraWinProc函数的MSG_PAINT消息来处理。
- 在MSG_PAINT中，界面显示是分模块显示的。有MODE_USBCONNECTION：usb插入模式，MODE_PLAY：播放视频模式，MODE_RECORDING：录像模式，MODE_PHOTO：拍照模式，MODE_PREVIEW：资源管理模式。
- 显示的内容基本以图片形式显示。比如：信息栏的电池，sd卡，模式等信息图片。
- 显示函数：FillBoxWithBitmap（）

```
if (SetMode < MODE_PLAY) {
    char tf_cap[20];
    long long tf_free;
    long long tf_total;
    FillBoxWithBitmap(hdc, TOPBK_IMG_X, TOPBK_IMG_Y, g_rcScr.right, TOPBK_IMG_H, &topbk_bmap);
    FillBoxWithBitmap(hdc, BATT_IMG_X, BATT_IMG_Y, BATT_IMG_W, BATT_IMG_H, &batt_bmap[battery]);
    FillBoxWithBitmap(hdc, TF_IMG_X, TF_IMG_Y, TF_IMG_W, TF_IMG_H, &tf_bmap[sdcard]);
    FillBoxWithBitmap(hdc, MODE_IMG_X, MODE_IMG_Y, MODE_IMG_W, MODE_IMG_H, &mode_bmap[SetMode]);
```

资源加载/卸载

- 定义存储图片资源变量（全局变量）。
- 资源路径为：/app/video/res/

```
#define TOPBK_IMG_X 0
#define TOPBK_IMG_Y 0
#define TOPBK_IMG_W 1
#define TOPBK_IMG_H 61
const char* topbk_img = "top_bk.bmp";
BITMAP topbk_bmap;
```

- 加载图片资源，在loadres()函数中。

```
for (i = 0; i < (sizeof(png_menu_debug) / sizeof(BITMAP)); i++) {
    sprintf(img, "%s%s", respath, debug_img[i]);
    if(LoadBitmap(HDC_SCREEN, &png_menu_debug[i], img))
        return -1;
}
```

- 显示，在MSG_PAINT消息中调用FillBoxWithBitmap函数，显示资源。

```
FillBoxWithBitmap(hdc, TOPBK_IMG_X, TOPBK_IMG_Y, g_rcScr.right, TOPBK_IMG_H, &topbk_bmap);
```

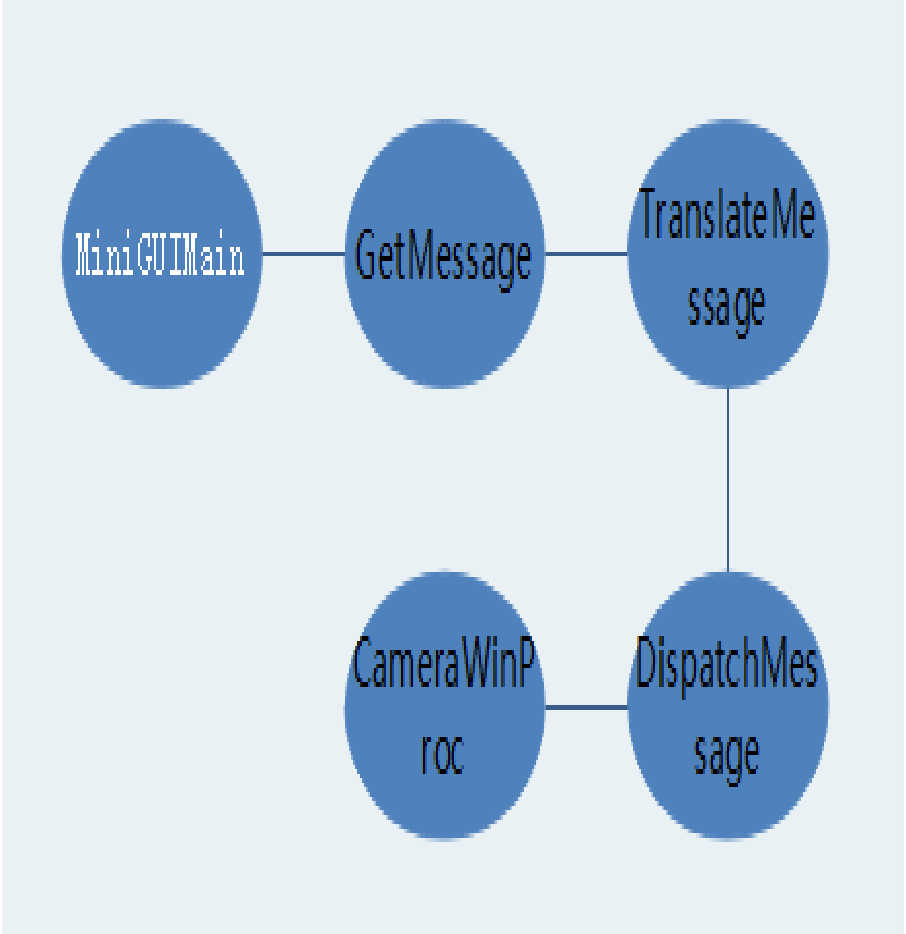
- 释放资源，在函数unloadres中。

```
UnloadBitmap(&play_bmap);
```

消息说明

消息说明（minigui自带）

- MSG_PAINT :显示消息
- MSG_CREATE :创建窗口消息
- MSG_TIMER： MSG_CREATE定时器
- MSG_KEYDOWN： 按键消息
- MSG_KEYLONGPRESS： 按键消息
- MSG_KEYUP： 按键消息
- 消息都是在MiniHUIMain函数中进行消息的获取，翻译，并传给cameraWinproc进行处理的。



消息说明（自定义）

- MSG_ADAS：ADAS
- MSG_SDMOUNTFAIL：sd卡加载失败
- MSG_SDCARDFORMAT：sd卡格式化
- MSG_USBCHAGE：usb消息
- MSG_SDCHANGE：sd卡消息
- MSG_BATTERY：电池状态消息

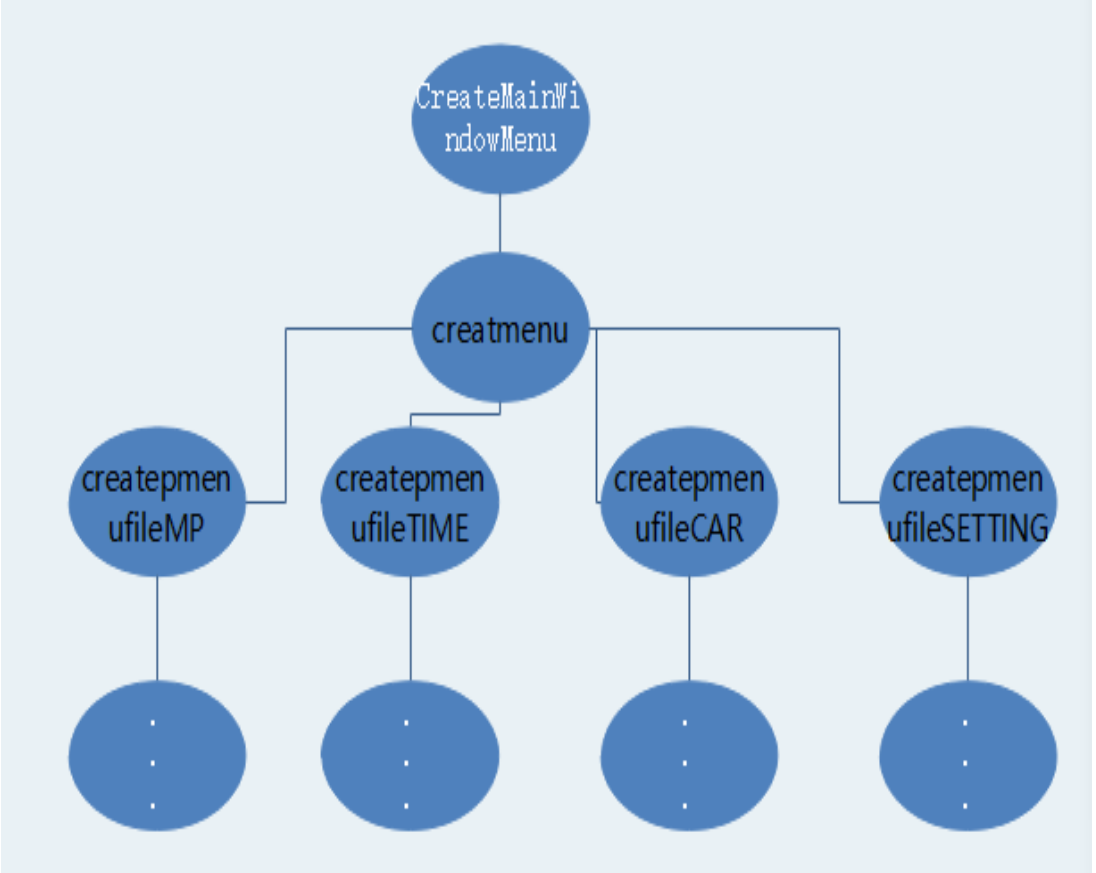
消息添加过程：

- 在camera_ui_def.h添加消息定义 #defineMSG_SDCARDFORMAT (MSG_USER+6)。
- 在cameraWinProc函数中添加 case MSG_SDCARDFORMAT，用于处理自定义的这个消息。
- 发送消息： PostMessage(hWnd, MSG_SDCARDFORMAT, 0, 1)。

菜单

菜单创建过程

- 创建菜单函数：CreateMainWindowMenu().
- 菜单显示函数： popmenu（）。
- 菜单销毁： DestroyMainWindowMenu(hWnd)。
- 在模式切换的过程中，如果切换到的那个模式不需要菜单，需要销毁菜单，等切回到需要菜单的模式，再调用菜单创建函数创建菜单。



菜单创建举例说明（主菜单）

- 主菜单的各个选项都是图片的形式的，图片的资源加载在前面说过。不过现实跟之前说的过程不一样。
- MFT_BITMA：菜单类型，这位图片菜单。
- Id：创建的图片菜单的id号
- Checkedbmp和uncheckedbmp：为菜单要现实的图片， 分别为点击时候显示图片和没有点击时候显示图片。
- Hsubmenu：创建下一级菜单的函数，如果为空，就没有下一级菜单。
- MFT_SEPARATOR：创建的分割菜单，不会被选中，只是起到分割作用。

```
memset(&mii,0,sizeof(MENUITEMINFO));
mii.type=MFT_BITMAP;//类型
mii.state=0;
mii.id=IDM_ABOUT_MP;//ID
mii.typedata=(DWORD)MP[0];
mii.uncheckedbmp      = &bmp_menu_mp[0];
mii.checkedbmp        = &bmp_menu_mp[1];

for (i = 0; i < LANGUAGE_NUM; i++)
    mii.str[i] = (char *)MP[i];

mii.hsubmenu=createpmenufileMP();
InsertMenuItem(hmnu,0,TRUE,&mii);

mii.type=MFT_SEPARATOR;//类型
mii.state=0;
mii.id=0;//ID
mii.typedata=0;
InsertMenuItem(hmnu,1,TRUE,&mii);
```

菜单创建举例说明（一级菜单）

- CreatePopupMenu：创建字符形式的说明栏，不会被选中。
- MFT_STRING：字符菜单类型
- Str[]：要显示的字符。
- LANGUAGE_NUM：中英文切换

- InsertMenuItem：创建菜单

```
memset(&mii,0,sizeof(MENUITEMINFO));
mii.type=MFT_STRING;
mii.id=0;
mii.typedata=(DWORD)MP[0];

for (i = 0; i < LANGUAGE_NUM; i++)
     mii.str[i] = (char *)MP[i];

hmnu=CreatePopupMenu(&mii);


memset(&mii,0,sizeof(MENUITEMINFO));
mii.type=MFT_STRING;
mii.state=0;
mii.id=IDM_FONTCAMERA;
mii.typedata=(DWORD)fontcamera_ui[0];
mii.hsubmenu=createpmenufileFONTCAMERA();
for (i = 0; i < LANGUAGE_NUM; i++)
     mii.str[i] = (char *)fontcamera_ui[i];
InsertMenuItem(hmnu,cnt++,TRUE,&mii);

mii.type=MFT_SEPARATOR;//类型
mii.state=0;
mii.id=0;//ID
mii.typedata=0;
InsertMenuItem(hmnu,cnt++,TRUE,&mii);
```

菜单创建举例说明（二级菜单）

- MFT_RADIOCHECK：可显示选中状态的菜单项
- State： parameter_get_video_fontcamera都有记录这个菜单的选中状态，如果选中parameter_get_video_fontcamera值为1，没选中就为0.

```
memset(&mii,0,sizeof(MENUITEMINFO));
mii.type=MFT_STRING;
mii.id=0;
mii.typedata=(DWORD)MP[0];

for (i = 0; i < LANGUAGE_NUM; i++)
     mii.str[i] = (char *)MP[i];

hmnu=CreatePopupMenu(&mii);
```

```
memset(&mii,0,sizeof(MENUITEMINFO));
mii.type=MFT_RADIOCHECK;
mii.state=(parameter_get_video_fontcamera()== 0) ? MFS_CHECKED : MFS_UNCHECKED;
mii.id=IDM_FONT_1;
mii.typedata=(DWORD)mpstr;

for (i = 0; i < LANGUAGE_NUM; i++)
    mii.str[i] = mpstr;

InsertMenuItem(hmnu,cnt++,TRUE,&mii);
```

菜单消息

- 菜单的点击消息在cameraWinProc中的MSG_COMMAND进行处理。
- 根据收到的菜单的ID号进行对应的处理。

```
case IDM_DEBUG_PHOTO_ON:
    if (video_rec_state == 0)
        {
            if(sdcard == 1)
            {
                startrec(hWnd);
            }
        }
    parameter_save_debug_photo(1);
    break;
```

参数保存

参数保存

- 由于关机一些菜单设置的参数和一些应用程序参数会丢失，因此要将参数保存到一个特定的文件里。
- 功能实现在parameter.c中
- 获取参数函数：parameter_get_*函数
- 设置参数函数： parameter_save_*函数

```
struct _SAVE
{
    char pararater_version[12];
```

```
int parameter_sav_wifi_mode(char mod)
{
    parareter.wifi_mode = mod;
    return parameter_save();
}
char parameter_get_wifi_mode(void)
{
    return parareter.wifi_mode;
}
```

对话框创建

对话框创建

- 简单的对话框主要是为了实现人机交互，用MessageBox函数的形式实现简单对话框的创建。

自定义对话框创建（例如时间设置对话框）：

- DialogBoxIndirectParam：创建时间设置窗口
- MyDateBoxProc：消息处理函数
- SendDlgItemMessage：添加控件

```
if(parameter_get_video_lan()==1)
    DlgMyDate.controls = CtrlMyDate;
else if (parameter_get_video_lan()==0)
    DlgMyDate.controls = CtrlMyDate_en;
DialogBoxIndirectParam (&DlgMyDate, HWND_DESKTOP, MyDateBoxProc, 0L);

break;
```

```
static int MyDateBoxProc (HWND hDlg, int message, WPARAM wParam, LPARAM lParam)
{
    int i;
    time_t t;
//  char bufff[512] = "";
    struct tm * tm;
    switch (message) {
        case MSG_INITDIALOG:
        {
            HWND hCurFocus;
            HWND hNewFocus;

            time (&t);
            tm = localtime (&t);
            hCurFocus = GetFocusChild (hDlg);
            SendDlgItemMessage(hDlg, IDL_YEAR, CB_SETSPINFORMAT, 0, (LPARAM)"%04d");
            SendDlgItemMessage(hDlg, IDL_YEAR, CB_SETSPINRANGE, 1900, 2100);
            SendDlgItemMessage(hDlg, IDL_YEAR, CB_SETSPINVALUE, tm->tm_year+1900, 0);
            SendDlgItemMessage(hDlg, IDL_YEAR, CB_SETSPINPACE, 1, 1);
```

创建等待显示

创建等待显示

- 在程序子星过程中有些功能的执行需要一点时间。因此需要在显示上显示一些正在处理等gif动画，用来标志处理完成。
- loadingWaitBmp：创建动画窗口
- ANS_AUTOLOOP：自动播放
- ANM_STARTPLAY：开始播放
- DestroyAnimation：摧毁动画窗口

```
void loadingWaitBmp(HWND hWnd)
{
        ANIMATION* anim = CreateAnimationFromGIF89aFile (HDC_SCREEN, "/usr/local/share/minigui/res/imag
        if (anim == NULL) {
            printf("anim=NULL\n");
            return;
        }
        SetWindowAdditionalData (hWnd, (DWORD) anim);
        CreateWindow (CTRL_ANIMATION,"", WS_VISIBLE | ANS_AUTOLOOP,
                      190, (WIN_W - 98) / 2, (WIN_H - 98) / 2,  98, 98, hWnd, (DWORD)anim);
        SendMessage (GetDlgItem (hWnd, 190), ANM_STARTPLAY, 0, 0);

        return;
}
```

```
    DestroyAnimation ((ANIMATION*)GetWindowAdditionalData (hWnd), TRUE);
    DestroyAllControls (hWnd);
```

Q&A

Thanks!