CVR MINIGUIUI Development Introduction

Sep.27, 2016

林清文

Agenda

- Compare MINIGUI with QT
- APP and LIBMINIGUI Framework
- Display Description
- Message Description
- Menu
- Parameter Save
- Creat Message Box
- Create Wait Message
- Q&A

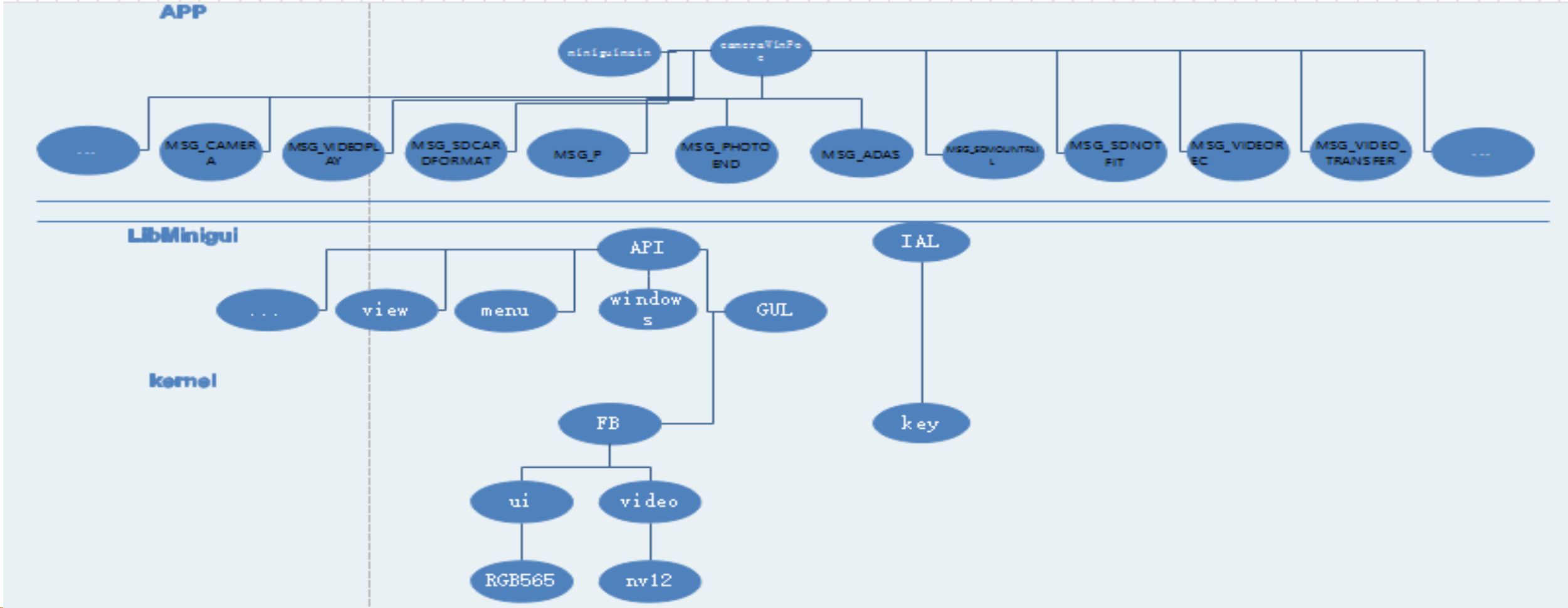
Compare MINIGUI with QT

Compare MINIGUI with QT

- QT is originally designed for PC desktop environment, and its structure is too complex to systematically cut, extend, customization and transplantation.
- Because of weak support for hardware acceleration, it is difficult to meet the requirements for graphic speed, function and efficiency of higher real-time embedded systems, thus QT/E mostly run in IPAQ such as strongARM, xscale.
- Apply in equipment such as car DVR, DV, security camera and UAV camera.
- MiniGUI can be used in multiple operating system. Currently MiniGUI supports mainstream embedded operating systems such as LINUX, UCLINUX, eCOS, VxWORKS, threadx, nucleus and UC/OS – II. QT is mainly used for LINUX.
- Multilingual support, especially support Chinese. MiniGUI language support is very good, it can support various languages. QT also offers Chinese, but still have a lot of work, which may influence the success of entire project.
- MiniGUI provides user manual, programming guide, complete API documentations, it is handy for developers!

APP and LIBMINIGUI Framework

AAPP and LIBMINIGUI Framework



Display Description

Display Description

- All Display of Mingui processed by MSG_PAINT info in function cameraWinProc of camera_ui.c.
- MSG_PAINT interface display module is divided. MODE_USBCONNECTION: usb insert mode, MODE_PLAY: Video play mode, MODE_RECORDING: Recording mode, MODE_PHOTO: Photo mode, MODE_PREVIEW: Resource management mode.
- Basic display format is image. Image information such as batteries in information column , sd card and modes.
- Display function: 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]);
```

Resource Loading/Unloading

- Define Stored Image of Resource Varible (Global variable)
- Resource Path： /app/video/res/
- Loading Resource Image, in Function loadres()
- Display, call Function FillBoxWithBitmap in message MSG_PAINT to display resource
- Unloading resource, realized in Function unloadre

```
#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;

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;
}
```

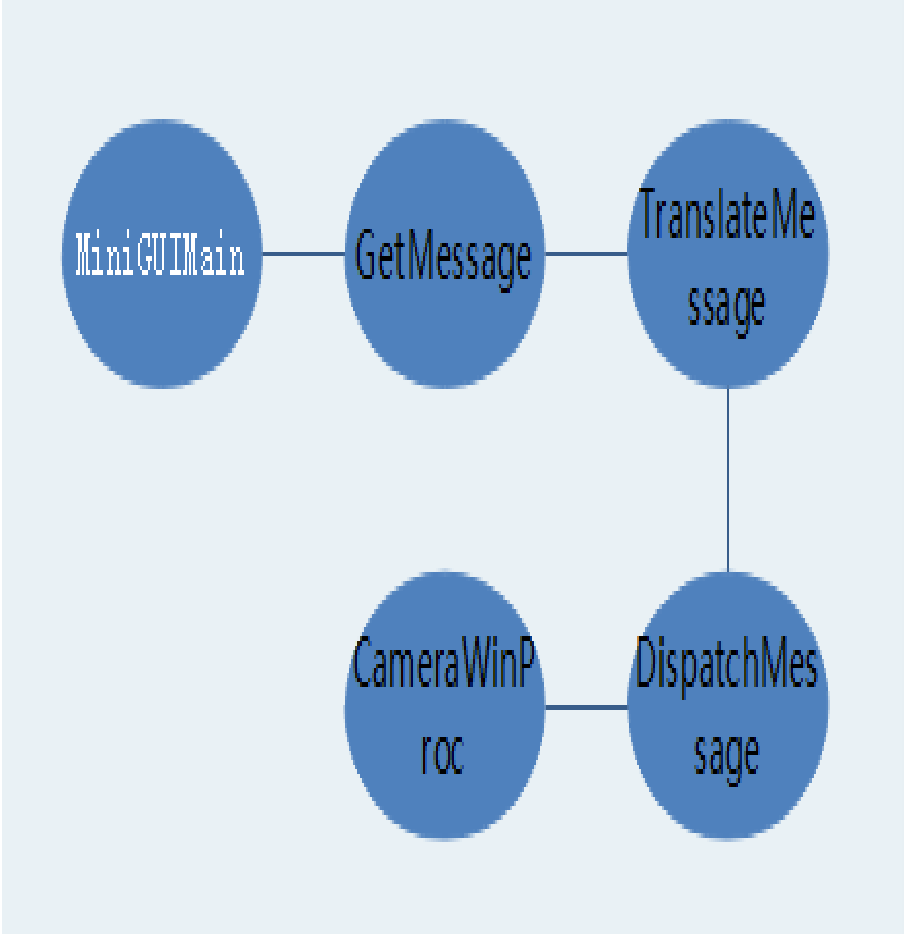
```
FillBoxWithBitmap(hdc, TOPBK_IMG_X, TOPBK_IMG_Y, g_rcScr.right, TOPBK_IMG_H, &topbk_bmap);
```

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

Message Description

Message Description(minigui)

- MSG_PAINT : Display Message
- MSG_CREATE : Create Window Message
- MSG_TIMER： MSG_CREATE Timer
- MSG_KEYDOWN： Key Message
- MSG_KEYLONGPRESS： Key Message
- MSG_KEYUP： Key Message
- Messages are carried out from function MiniHUIMain, translated, and to be processed in cameraWinproc.



Message Description (User-defined)

- MSG_ADAS：ADAS
- MSG_SDMOUNTFAIL：sd card mounted fail
- MSG_SDCARDFORMAT：sd card formatting
- MSG_USBCHAGE：usb message
- MSG_SDCHANGE：sd card message
- MSG_BATTERY：Battary Status Message

Message Adding Process：

- Add message definition in camera_ui_def.h

  #defineMSG_SDCARDFORMAT (MSG_USER+6)。
- Add case MSG_SDCARDFORMAT in function cameraWinProc，

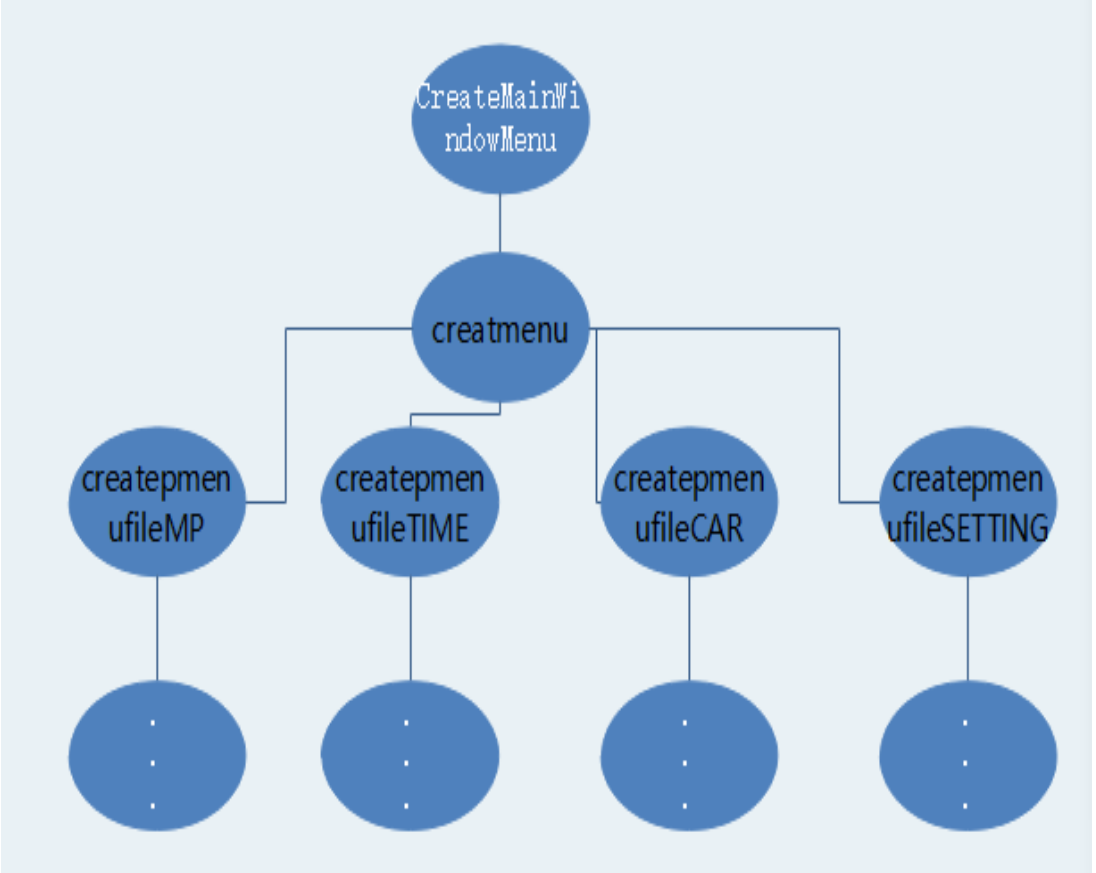
to process user-defined message.

- Send Message： PostMessage(hWnd, MSG_SDCARDFORMAT, 0, 1)。

Menu

Menu Creating Process

- Create Menu Function： CreateMainWindowMenu()
- Menu Display Function： popmenu()
- Menu Destroy： DestroyMainWindowMenu(hWnd)
- In mode switching process, if the mode need to be switched doesn't need menu, need to destroy menu first, and create menu again after switching back to mode requires menu.



Menu Create Example（Main Menu）

- All main menu options are in images, those images had been mentioned in Resource Loading before. But in reality, they are not the same.
- MFT_BITMA： Menu Type of this menu image
- Id： Create id of image Menu
- Checkedbmp and uncheckedbmp： For real menu images, respectively display image when click the image and display image when didn't click the image
- Hsubmenu： Function to create submenu, if it is empty, means no submenu
- MFT_SEPARATOR： Create split menu, won’t be selected, only used for split

```
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);
```

Menu Create Example （First-Level Menu）

- CreatePopupMenu：Create Character Form description column, it won't be selected
- MFT_STRING：Character Menu Type
- Str[]：Display Character
- LANGUAGE_NUM：Chinese-English Switch

- InsertMenuItem：Create Menu

```
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);
```

Menu Create Example （Second-Level Menu

- MFT_RADIOCHECK：Display Selected Menu
- State： parameter_get_video_fontcamera recorded the selected Menu，if the Menu is selected, parameter_get_video_fontcamera value is 1，if not selected, the value is 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);
```

Menu Message

- Menu Click Message is processed in MSG_COMMAND of cameraWinProc.
- Process according to received Menu ID

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

Parameter Save

Parameter Save

- Some Settings Menu parameters application parameters will be lost after shut down, so need to save parameters to a specific file.
- Function Realization in parameter.c
- Parameter Get Function : parameter_get_*Function
- Parameter Set Function : parameter_save_*Function

```
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;
}
```

Create Message Box

Create Message Box

- Simple Message Box is used to realize man-machine interaction. Use MessageBox Function to create simple Message box.

Define Message Box Creation (eg. Create time setting Message Box):

- DialogBoxIndirectParam : Create time setting box
- MyDateBoxProc : Message process function
- SendDlgItemMessage : Add control

```
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);
```

Create Wait Message

Create Wait Message

- Some program need some time for execution. Therefore need to display the program in processing such as GIF animation, used for symbolizing the process is completed.
- loadingWaitBmp : Create Animation Box
- ANS_AUTOLOOP : Autoplay
- ANM_STARTPLAY : Start Play
- DestroyAnimation : Destroy Animation Box

```
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!