简介：

LightningUI工程是一个集GUI和车载业务模块于一体的项目。主要的模块目录结构见下表：

|  |  |
| --- | --- |
| 模块目录/文件 | 备注 |
| LightningUI/Acceleboot | 快照模式启动模块（暂未使用） |
| LightningUI/Activate | 设备激活模块 |
| LightningUI/AndroidAuto | AndroidAuto模块 |
| LightningUI/AppConfigProxy | 参数配置模块 |
| LightningUI/Audio | 音频管理模块 |
| LightningUI/Backlight | 背光控制模块 |
| LightningUI/Bluetooth | CSR蓝牙模块（暂未使用） |
| LightningUI/Bluetooth FLC | 慧翰蓝牙模块 |
| LightningUI/Carplay | CarPlay模块 |
| LightningUI/Communication | 通信模块，包括跟MCU的串口通信以及一些器件IIC通信 |
| LightningUI/Config | 客户配置文件 |
| LightningUI/CSRPlayerSDK | 多媒体播放模块 |
| LightningUI/DAB | DAB数字收音模块 |
| LightningUI/Dial-Up Network | 拨号网络设置模块 |
| LightningUI/DIB | 图片转换模块 |
| LightningUI/FantasySDK | GUI图形库 |
| LightningUI/Gesture | 手势管理模块 |
| LightningUI/GFX | 图形头文件 |
| LightningUI/GPSAPI | GPS模块 |
| LightningUI/GStreamer | Gstreamer库 |
| LightningUI/GUI | 所有的GUI页面文件 |
| LightningUI/iAP | iAP1模块 |
| LightningUI/Include | 通用的头文件目录 |
| LightningUI/Key | 按键管理模块 |
| LightningUI/LED | LED面板灯模块 |
| LightningUI/MobileData | 手机号码归属地查询模块 |
| LightningUI/ModulVer | 版本管理模块 |
| LightningUI/OpenGL-ES | OpenGL-ES sdk |
| LightningUI/PWM | PWM头文件 |
| LightningUI/Res | 语言等资源文件目录 |
| LightningUI/SR | 讯飞语音识别模块 |
| LightningUI/VIP | 视频输入模块 |
| LightningUI/Wave | wave音频播放工具 |
| LightningUI/WebLink | WebLink模块 |
| LightningUI/Wifi | Wifi设置模块 |
| LightningUI/LightningUI.cpp | LightningUI工程主入口文件，WinMain函数在此文件定义 |

2. LightningUI启动流程

2.1整个GUI和业务模块逻辑封装在一个名为CGUI\_Fantasy的类中，在LightningUI/LightningUI.cpp文件的WinMain函数中被创建，初始化和销毁。WinMain函数流程：

设置主线程优先级

创建窗口InitInstance

创建CGUI\_Fantasy并初始化

创建事件监控线程

进入消息循环（处理触摸和其他输入信号）

资源销毁和退出

2.2 CGUI\_Fantasy初始化流程

CGUI\_Fantasy类定义在LightningUI\GUI\Main\GUI\_Fantasy.cpp。

CGUI\_Fantasy初始化操作定义在CGUI\_Fantasy::Initialize函数中。

|  |  |
| --- | --- |
| 动作 | 代码 |
| 创建事件变量 | m\_hEventSD = CreateEvent(NULL,FALSE,FALSE,NULL);  m\_hEventMute = CreateEvent(NULL,FALSE,FALSE,NULL); |
| 初始化OpenglES环境和Com组件 | CGUI\_OpenglES::Initialize(window,entry,iPriority);  if(FAILED(CoInitializeEx(NULL,COINIT\_MULTITHREADED)))  {  return FALSE;  } |
| 初始化按键模块 | #if CVTE\_EN\_KEY\_PROXY  //Init KeyHandler  CKeysHandler::GetInstance()->Initialize(window, WM\_KEY);  CKeysHandler::GetInstance()->AddKeys(g\_keyparam, \_countof(g\_keyparam));  #endif |
| 初始化配置模块 | //get config:  m\_pConfig = new CAppConfigProxy; //CProConfig;  m\_pConfig->GetOemSector();  UINT idLang=m\_pConfig->GetCurParam()->idLanguage;  g\_idLanguageOffset=(idLang==0)?0:idLang\*1000;  CReg reg(HKEY\_LOCAL\_MACHINE, RK\_CLOCK);  reg.SetDW(RV\_AUTODST, m\_pConfig->GetCurParam()->bDSTAutoAdjust);  #if CVTE\_EN\_NAVI  m\_bAutoStartNavi = (m\_pConfig->GetCurParam()->bAutoStartNavi);  #else  m\_bAutoStartNavi = FALSE;  #endif  g\_bKeyBeep=(m\_pConfig->GetCurParam()->bKeyBeep);  //test:  m\_pConfig->GetCurParam()->bHasCanBus=0; |
| 初始化开关机logo | #if CVTE\_EN\_IDLE\_LOGO  const int cx=800,cy=480;  if(m\_tagLogo.Initialize(0,NULL,m\_pConfig->GetCurParam()->strLogoName.String(),cx,cy))  {  VIEW\_STATE vs=m\_tagLogo.GetCurViewState();  vs.fRotateX=-180;  m\_tagLogo.SetIniViewState(&vs);  m\_tagLogo.SetScale(GLfloat(g\_iScreenWidth)/cx,GLfloat(g\_iScreenHeight)/cy);  }  if(m\_tagLogo2.Initialize(0,NULL,m\_pConfig->GetCurParam()->strLogoName2.String(),cx,cy))  {  VIEW\_STATE vs2=m\_tagLogo2.GetCurViewState();  vs2.fRotateX=-180;  m\_tagLogo2.SetIniViewState(&vs2);  m\_tagLogo2.SetScale(GLfloat(g\_iScreenWidth)/cx,GLfloat(g\_iScreenHeight)/cy);  }  //ActivateRenderLogo();  #endif |
| 初始化皮肤管理器 | //skin manager:  m\_pSkinManager=new CSkinManagerGL(this,m\_pConfig);  //skins 1:  if(!IsFileExists(m\_pConfig->GetCurParam()->strSkinFileName.String()))  {  //m\_pConfig->GetCurParam()->strSkinFileName = \_T("\\Windows\\skin\_lightning.skn");  m\_pConfig->GetCurParam()->strSkinFileName = \_T("\\ResidentFlash\\LightningUI\\Skins\\skin\_lightning.skn");  }  m\_pSkinManager->LoadFromFile(m\_pConfig->GetCurParam()->strSkinFileName.String()); |
| 初始化状态栏和文字管理器 | //text:  m\_pTextGL = new CTextGL();  //m\_pTextGL->SetFontFile(\_T("\\ResidentFlash\\LightningUI\\glFont\\font\_xihei.bin"));  //status bar:  m\_pStatusBar = new CStatusBarEx;  m\_pStatusBar->Initialize(0,NULL,m\_pSkinManager,m\_pTextGL,window,entry);  //text:  if(m\_pConfig->GetCurParam()->idLanguage == LANGUAGE\_RUSSIAN)  {  m\_pConfig->GetCurParam()->idFont = FONT\_Arial;  }  GLfloat pix\_ratio=GLfloat(CVTE\_DEF\_SCREEN\_PHYSICALWIDTH)/GLfloat(CVTE\_DEF\_SCREEN\_PHYSICALHEIGHT)\*GLfloat(g\_iScreenHeight)/GLfloat(g\_iScreenWidth);  m\_pTextGL->Initialize(m\_pSkinManager,FontIDGL(m\_pConfig->GetCurParam()->idFont)/\*FONT\_Helvetica\*//\*FONT\_Segoe\_UI\*/,pix\_ratio);  //m\_pTextGL->SetFont(m\_pConfig->GetCurParam()->idFont);  //set RIGHT\_TO\_LEFT reading order:  if(m\_pConfig->GetCurParam()->idLanguage == LANGUAGE\_PERSIAN)  {  m\_pTextGL->SetReadingOrder(FALSE);  } |
| 初始化视频输入窗口 | //video window:  m\_pVideoWindow = new CVideoWindow;  if(m\_pVideoWindow)  {  m\_pVideoWindow->Initialize(entry.hInstance,m\_pConfig);  } |
| 初始化Overlay显示条 | //smart bar window:  m\_pSmartBarVol = new CSmartBarVol;  if(m\_pSmartBarVol)  {  m\_pSmartBarVol->Initialize(m\_pSkinManager,SMART\_BAR\_VOLUME,window,entry.hInstance);  m\_listSmartBars.Add(m\_pSmartBarVol,SMART\_BAR\_VOLUME);  }  m\_pSmartBarPhone = new CSmartBarPhone;  if(m\_pSmartBarPhone)  {  m\_pSmartBarPhone->Initialize(m\_pSkinManager,SMART\_BAR\_PHONE,window,entry.hInstance);  m\_listSmartBars.Add(m\_pSmartBarPhone,SMART\_BAR\_PHONE);  }  m\_pSmartBarMedia = new CSmartBarMedia;  if(m\_pSmartBarMedia)  {  m\_pSmartBarMedia->Initialize(m\_pSkinManager,SMART\_BAR\_MEDIA,window,entry.hInstance);  m\_listSmartBars.Add(m\_pSmartBarMedia,SMART\_BAR\_MEDIA);  }  m\_pSmartBarDrop = new CSmartBarDrop;  if(m\_pSmartBarDrop)  {  m\_pSmartBarDrop->Initialize(m\_pSkinManager,SMART\_BAR\_DROP,window,entry.hInstance);  m\_listSmartBars.Add(m\_pSmartBarDrop,SMART\_BAR\_DROP);  }  m\_pSmartBarTopBanner = new CSmartBarTopBanner;  if(m\_pSmartBarTopBanner)  {  m\_pSmartBarTopBanner->Initialize(m\_pSkinManager,SMART\_BAR\_TOPBANNER,window,entry.hInstance);  m\_listSmartBars.Add(m\_pSmartBarTopBanner,SMART\_BAR\_TOPBANNER);  } |
| 初始化LED模块 | #if CVTE\_EN\_LED  //led:  #if (CUSTOMER\_ID == CUSTOMER\_DAIICHI\_FIAT\_DOBLO || CUSTOMER\_ID == CUSTOMER\_DS701P\_DAIICHI\_FIAT\_DOBLO)  m\_pLED = new CLED(255,255,255,0,FALSE,(m\_pConfig->GetCurParam()->iPowerStatusBeforeACCOff==1?FALSE:TRUE));  #else  m\_pLED = new CLED(m\_pConfig->GetCurParam()->iLEDRedCurrent,  m\_pConfig->GetCurParam()->iLEDGreenCurrent,  m\_pConfig->GetCurParam()->iLEDBlueCurrent,  m\_pConfig->GetCurParam()->iLEDBrightnessCurrent,  m\_pConfig->GetCurParam()->bLedAnimEffect,  (m\_pConfig->GetCurParam()->iPowerStatusBeforeACCOff==1?FALSE:TRUE));  #endif  #endif |
| 初始化MCU串口通信模块 | //com port:  m\_pCommander = new CCommanderEx;  if(m\_pCommander)  {  RETAILMSG(1, (TEXT("APP:MSG:CGUI\_Fantasy: Initialize: Initialize COM3 115200.\r\n")));  m\_bCommanderInitialized = m\_pCommander->Initialize(CommandProc,this,\_T("COM3:"),CBR\_115200,m\_hMainWnd);  } |
| 初始化音频模块 | //audio codec:  #if (CVTE\_DEF\_AUDIO\_TYPE==CVTE\_AUDIO\_TYPE\_FM1388\_AIC3254\_ST7418)  m\_pAudioCodec = new CAudioCodecST7418AIC3254(m\_pCommander,g\_hUIWnd,AspTypeFm1388);  #elif (CVTE\_DEF\_AUDIO\_TYPE == CVTE\_AUDIO\_TYPE\_AK7738)  m\_pAudioCodec = new CAudioCodecAK7738(m\_pCommander,g\_hUIWnd,AspTypeAK7738);  #else  #endif |
| 创建初始页面 | //open initial panels:  if(m\_pConfig->GetCurParam()->bMCUSoftwareUpdateFlag)  {  OpenPanels(MAKE\_PANEL\_ID(UI\_CLASS\_UPDATE,0));  RETAILMSG(DEBUG\_GUI,(\_T("MSG: [CGUI\_Fantasy::Initialize]:enter update panel,take %d ms............\r\n"),GetTickCount()-tc));  }  else  {  OpenPanels(MAKE\_PANEL\_ID(UI\_CLASS\_HOME,PAGE\_HOME\_WRAPPER));  OpenPanels(MAKE\_PANEL\_ID(UI\_CLASS\_MEDIA,PAGE\_MEDIA\_MAIN));  OpenPanels(MAKE\_PANEL\_ID(UI\_CLASS\_RADIO,PAGE\_RADIO\_MAIN));  OpenPanels(MAKE\_PANEL\_ID(UI\_CLASS\_SETTING,PAGE\_SETTINGS\_WRAPPER));  OpenPanels(MAKE\_PANEL\_ID(UI\_CLASS\_BT,PAGE\_BT\_MAIN));  #if CVTE\_EN\_CARPLAY  OpenPanels(MAKE\_PANEL\_ID(UI\_CLASS\_CARPLAY,PAGE\_CARPLAY\_WRAPPER));  #endif  #if CVTE\_EN\_ACCON\_WARNING  OpenPanels(MAKE\_PANEL\_ID(UI\_CLASS\_WARNING,PAGE\_ACCON\_WARNING\_MAIN));  #endif  OpenPanels(MAKE\_PANEL\_ID(UI\_CLASS\_IMAGE,PAGE\_IMAGE\_WRAPPER));  RETAILMSG(DEBUG\_GUI,(\_T("MSG: [CGUI\_Fantasy::Initialize]: load modules,take %d ms............\r\n"),GetTickCount()-tc));  } |
| 初始化GPS监控模块 | //GPS:  m\_hEvtGPSMonitor = CreateEvent(NULL,FALSE,FALSE,NULL);  EnableGPSMonitor(TRUE); |
| 发命令初始化MCU | //initialize MCU:  //mcu software update:  if(!m\_pConfig->GetCurParam()->bMCUSoftwareUpdateFlag)  {  if(m\_pCommander && m\_bCommanderInitialized)  {  m\_pCommander->IniMCU();  }  } |
| 显示UI主窗口 | //show windows:  if(!m\_bUIWindowShow)  {  ShowWindow(g\_hMainWnd, SW\_SHOW);  ShowWindow(g\_hUIWnd, SW\_SHOW);  m\_bUIWindowShow = TRUE;  } |

CGUI\_Fantasy::Initialize调用完后会继续执行CGUI\_Fantasy:: OnSystemReady函数，该函数会启用一个定时器TIMER\_AUTO\_OPEN\_SRC来执行CGUI\_Fantasy::OnAutoOpenSrc，该函数作用：

1. 验证跟MCU是否通信成功
2. 初始化音频模块参数
3. 加载关机前的模式
4. 启动一个定时器TIMER\_AUTO\_LOAD\_MODULES执行CGUI\_Fantasy::OnLoadModules来加载剩余模块