GRX Tips

GstarCAD 2022

GstarCAD

Table of Contents

1. Load GRX

1.1. The sequences of loading GRX

Phenomenon:
When using the GRX loader (essentially a GRX program) to load other GRX files. After the Startup group loading or manually entering the “appload” command to load the GRX loader, fail to load or the program crashes and the error report dialog box pops up.。

Possible Reason：
When GstarCAD loads GRX, the sequences are different from AutoCAD.
In ARX, after loading all the “kInitAppMsg” messages of ARX, send the corresponding "kLoadDwgMsg" message in turn to each ARX file.
However, in GRX, every time a "kInitAppMsg" message of GRX is loaded, the corresponding "kLoadDwgMsg" message will be sent to GRX immediately.
Which means, if some GRX initialization operations are placed in the “kLoadDwgMsg” message processing of the corresponding GRX, and called other GRX classes or commands. Then the GRX, which has been called, might has not been registered in the “kInitAppMsg” message. It will cause the calling command failed or crash.

Suggestion：
Put all GRX initialization operations in the kLoadDwgMsg message processing of the GRX loader, or manually adjust the GRX loading sequence according to the GRX initialization sequence.
For example, GRX A try to call GRX B, then GRX B must be loaded first.

1.2. Failed to load GRX

Phenomenon:
When loading the GRX by Startup group or manually entering the “appload” command, the command line prompts that the "GRX cannot be loaded".

Possible Reason:
1. GstarCAD2020 and above versions developed base on VS2017, and the corresponding platform working set is Visual Studio 2017 (v141). GRX programs developed with Visual Studio 2010 (v100) will not be able to load and run in GstarCAD2020 and above versions.
2. The SDK version is inconsistent with the GstarCAD version.
3. 32-bit GRX program cannot be loaded by 64-bit GstarCAD, and vice versa.

4. The local computer lacks necessary dependencies for running the program.

Suggestion:
a). Recompile the GRX code with the right SDK and Visual Studio version. And generate new GRX files.
b). Check the SDK version and GstarCAD version, keep the version number consistent.
c). Check the GstarCAD version (32-bit or 64-bit version), and recompile the GRX code in the corresponding environment.
d). Use the “depend” tool to check whether the dependency (dll) in the local computer is missing.

1.3. GRX is loaded successfully, but enter the "entry function" failed

Phenomenon:
Loading GRX files successfully, but when running the command, the command line prompts an “unknown command” error.

Possible Reason:
“grxport.lib” is the mapping file of the entry function. If it is not placed in the first position of the dependency, the program will not enter the entry function, and caused the command registration failed.

Suggestion:
Check the linker->input->additional dependencies in the project configuration, and ensure that put the grxport.lib at the first position.

2. Differences between GRX interface and ARX interface

2.1. “ACRX_DEFINE_MEMBERS” macro compilation error in GRX

Phenomenon:
When using the ACRX_DEFINE_MEMBERS macro in a class, if the constructor of the class has parameters, a compilation error will occur. The error is as follows:

error C2440: “<function-style-cast>”: Can’t transfer from “OdSmartPtr<T>” to “OdRxObjectPtr”.

Possible Reason:

The two macros ACRX_DECLARE_MEMBERS and ACRX_DEFINE_MEMBERS in GRX are different with those in ARX：

a). In GRX, if you use these two macros in a class, the class must inherit from AcRxObject

b). In GRX, due to the use of NEWOBJ_CONSTR within the macro, if a constructor with parameters is declared in the class, a default constructor without parameters must be declared for the call of the internal NEWOBJ_CONSTR.

Suggestion:

The custom class must inherit the AcRxObject class.

Declare a default constructor without parameters or try to avoid declaring a constructor with parameters.

For example：

```
class CBase
{
public:
    ACRX_DECLARE_MEMBERS(CBase);
    CBase(A* a);
    ~CBase();

};
```

Need to change to：

```
class CBase : public AcRxObject {
public:
ACRX_DECLARE_MEMBERS(CBase);
CBase();
CBase(A* a);
~CBase();
};
```

Or：

```
class CBase : public AcRxObject {
public:
ACRX_DECLARE_MEMBERS(CBase);
//CBase();
//CBase(A* a);
~CBase();
};
```

2.2. The return value of the intersectWith function is different with ARX

Phenomenon:
When judging whether two entities intersect according to the return value of the intersectWith function, an incorrect result is obtained.

Possible Reason:
When the intersectWith function has no intersection, the return value of GRX and ARX are different. When there is no intersection, ARX also returns eOk, and GRX returns an error value.

Suggestion:
Use GCAD_CORE to distinguish the different usages of GRX and ARX

For example：

```
bool bHasInters = false;
AcGePoint3dArray aryIntters;
Acad::ErrorStatus es = pCur->intersectWith(pBreak , AcDb::kOnBothOperands ,
aryIntters);
#ifdef GCAD_CORE
    if ( es != Acad::eOk ) {
    return bHasInters;
    }
#endif
    bHasInters = aryIntters.length() > 0 ;
```

2.3. The printing interface “beginDocument” function works different with ARX

Phenomenon:
After GRX calls the “beginDocument” function, it will automatically execute the corresponding message processing operations of the “beginPage” message and the “endPage” message in the print reactor.

Possible Reason:
In GRX, the “beginDocument” function will call the “beginPage” function automatically, and the “beginPage” function will call the “beginGenerateGraphics” function.

Suggestion:
When loading the printing interface, no need to call “beginPage”, “beginGenerateGraphics”, “endGenerateGraphics” and “endPage” functions.

For example：

```
AcPlPlotEngine* mPlotEngine;
    es = AcPlPlotFactory::createPublishEngine(mPlotEngine);
    es = mPlotEngine->beginPlot(NULL);
    AcPlPlotInfo plotInfo;
    AcPlPlotInfoValidator plotInfoValidator;
    plotInfo.setLayout(mPlotSettings->id());
    plotInfo.setOverrideSettings(mPlotSettings);
    plotInfoValidator.setMediaMatchingPolicy(AcPlPlotInfoValidator::kMatchEnabled);
    es = plotInfoValidator.validate(plotInfo);
    es = mPlotEngine->beginDocument(plotInfo, curDoc()->fileName(), NULL,1, true,
_T("c:\\test.eps"));
#ifndef GCAD_CORE
    AcPlPlotPageInfo pageInfo;
    es = mPlotEngine->beginPage(pageInfo, plotInfo, true);
    es = mPlotEngine->beginGenerateGraphics();
    es = mPlotEngine->endGenerateGraphics();
    es = mPlotEngine->endPage();
#endif
    es = mPlotEngine->endDocument();
    es = mPlotEngine->endPlot();
    mPlotEngine->destroy();
```

2.4. The implementation of “AcGsView::add()” is different

Phenomenon:
When “AcGsView::add()” is used in the code, a compilation error will be reported.
The error message is: the parameter list does not match.

Possible Reason:
The implementation of “AcGsView::add()” in GRX is different with ARX.

Suggestion:
Use GCAD_CORE to distinguish the different usages of GRX and ARX

For example：

```
Acad::ErrorStatus initDrawingCtrl(AcDbDatabase *pDb,const TCHAR *chSpace)
```

```
{
// have we got a valid drawing
if (NULL == pDb)
      {
  return Acad::eNullBlockName;
}
// initialize the preview control
mPreviewCtrl.init();
mPreviewCtrl.SetFocus();
AcDbBlockTableRecordPointer bptr(chSpace, pDb, AcDb::kForRead);
AcDbObjectId idCurrenVs;
// set the view to the Active AutoCAD view
idCurrenVs = SetViewTo(mPreviewCtrl.mpView, pDb, m_viewMatrix);
// tell the view to display this space
mPreviewCtrl.view()->add(bptr, mPreviewCtrl.model());
mPreviewCtrl.view()->setVisualStyle(idCurrenVs);
return Acad::eOk;
}
#ifdef GCAD_CORE
AcGePoint3d ptMin ( 1e20, 1e20, 1e20);
AcGePoint3d ptMax ( -1e20, -1e20, -1e20);
m_tempExt.set ( ptMin , ptMax ) ;
OdDbDatabasePtr pDb = acdbHostApplicationServices()->createDatabase();
initDrawingCtrl(pDb,L"*Model_Space");
#else
m_tempExt.minPoint().set(1e20, 1e20, 1e20);
m_tempExt.maxPoint().set(-1e20, -1e20, -1e20);
AcDbDatabase* pDb = new AcDbDatabase;
initDrawingCtrl(pDb,L"*Model_Space");
#endif
```

2.5. Release of the return value of the acedGetArgs() function

Phenomenon:
When using the “acedGetArgs” function to obtain the command line parameters of calling the LISP program or the GRX program, the program crashes.

Possible Reason:
In GRX, the release mechanism of “resbuf”, which returned by “acedGetArgs”, is different from that in ARX.

Suggestion:

In GRX, there is no need to call “acutRelRb()” to release the “resbuf” which is returned by “acedGetArgs()”.

2.6. The implementation of CAdUiPaletteSet class is different with ARX

Phenomenon:
An error was reported when compiling the code:
“The CFrameWnd::ShowControlBar parameter list does not match.”

Possible Reason:
The implementation of CAdUiPaletteSet class is different with ARX

Suggestion:
Use GCAD_CORE to distinguish the different usages of GRX and ARX

For example：

```
extern CPaletteSetWnd *glo_pMainWnd;

void SampleCmd()
{
    if (glo_pMainWnd == NULL)
    {
        glo_pMainWnd=new CPaletteSetWnd;
        if (glo_pMainWnd != 0)
        {
            CSize sizeDefault(200, 450);
CRect rcDefault(CPoint(150, 200), sizeDefault);
            DWORD dwStyle = WS_CHILD | WS_VISIBLE | CBRS_SIZE_DYNAMIC |
                                    CBRS_GRIPPER;
 dwStyle |= CBRS_FLOATING;

    glo_pMainWnd->Create(_T("Main ..."),dwStyle,rcDefault,acedGetAcadFrame());

    glo_pMainWnd->EnableDocking(CBRS_ALIGN_LEFT|CBRS_ALIGN_RIGHT);
            glo_pMainWnd->RestoreControlBar();
#ifdef _GRX_APP          // GstarCAD
            CMDIFrameWndEx *mdifwCurrent=acedGetAcadFrame();
            if (mdifwCurrent != 0)
                mdifwCurrent->DockPane(glo_pMainWnd);
#else                    // AutoCAD
            acedGetAcadFrame()->ShowControlBar(glo_pMainWnd, TRUE, FALSE);
```

```
#endif
            }
            else
            {
#if defined(_GRX_APP)     // GstarCAD
                glo_pMainWnd->ShowPane(TRUE, FALSE, TRUE);
#else                     // AutoCAD
                acedGetAcadFrame()->ShowControlBar(glo_pMainWnd, TRUE, FALSE);
#endif
            }
      }
}
```

CPaletteSetWnd is panel derived class

```
class CPaletteSetWnd : public CAdUiPaletteSet
{
      ......
}
```

Can also refer to the examples in SDK package. The path is: grxsdk\sample\ SimplePalette

3. Program Crash

3.1. When calling the “acedGetAcadFrame” function to close the application, the program crash

Phenomenon:
After loading GRX, when uninstalling or closing the application, GstarCAD crash.。

Possible Reason:
The “acedGetAcadFrame” function is used to get the pointer of the AutoCAD main frame window. If the “acedGetAcadFrame” function is called in the “On_kUnloadAppMsg” message, when the application program is closed or GRX is unloaded, and the program execution reaches “kUnloadAppMsg”, the pointer of the main window has been set to NULL. Then an error will occur if the pointer has been called again.

Suggestion:
Put the use of the “acedGetAcadFrame()” function before the “kUnloadAppMsg” message, and make a non-null judgment on the pointer before using it to avoid using a null pointer.

For example：

```
CMDIFrameWnd* p=acedGetAcadFrame();
        PWINDOWINFO pwi;
        p->GetWindowInfo(pwi);
```

Need to change to：

```
CMDIFrameWnd* p=acedGetAcadFrame();
if(p!=NULL)
{
    PWINDOWINFO pwi;
    p->GetWindowInfo(pwi);
}
```

GstarCAD 2022

https://www.gstarcad.net/

Gstarsoft