GRX Programming Guide

GstarCAD 2024

GstarCAD

Table of Contents

1. What is GRX?

GRX is the Runtime eXtension programming environment of GstarCAD, which is the latest application development SDK provided with GstarCAD. It provides Object-oriented development environment and APIs based on C++, which can be used to develop GstarCAD application programs, extension CAD classes and protocols, and create new commands in GstarCAD.

GRX is compatible with ARX application programs of AutoCAD® on the source-code level. By simply configuring projects, ARX programs running on the AutoCAD® can be easily ported to the GstarCAD.

GRX SDK allows direct access to GstarCAD database, graphics system and user-defined commands, etc. In addition, it can also be used with VLISP and other programming languages.

Developers only need few configurations to migrate ARX application for AutoCAD® to GRX application for GstarCAD.

The following works can be accomplished with GRX:

- Access the GstarCAD database
- Interact with the GstarCAD editor
- Create user interfaces using the Microsoft® Foundation Classes (MFC) (Windows only)
- Support the multiple document interface (MDI)
- Create custom classes
- Build complex applications
- Interact with other programming environments

2. Programming Environment

- Microsoft® Visual Studio Enterprise 2017 (Version 15.9.17)
- Windows SDK 10.0.17763.0
- CPU:

  Basic: 1.6 GHz CUP

  Recommended: 3.0 GHz CPU and above
- RAM:

  Basic: 2 GB

  Recommended: 8 GB and above
- Operation System (OS)

  Windows 11

  Windows 10 (version 1507 and above):

  Home, Professional, Education and Enterprise (not support LTSC and Windows 10 S)

  Windows 8.1 (with 2919355 update):

  Core, Professional and Enterprise

  Windows 7 SP1 (with the latest update):

  Home, Professional, Enterprise and Ultimate
- Monitor Resolution:

  1028x800 and above true color display, including 4K (3840x2160) display
- GstarCAD SDK 2024
- GstarCAD 2024
- .NET Framework 4.8 and above

3. Install GstarCAD SDK

Download GstarCAD SDK ('*GRXSDK.ZIP*' file) from GstarCAD website:

https://www.gstarcad.net/download/

Unzip GRXSDK.ZIP file to the local disk (e.g. '*C:\grxsdk*') and there will be 5 directories generated (in '*C:\grxsdk*') which are: **arx**, **inc**, **inc-x64**, **lib-x64** and **utils**.

arx contains the header files, library files and sample programs used for porting ARX programs to GRX programs. It contains the following directories:

- **Inc:** Header files used for porting from ARX to GRX
- **inc-x64**: Files used by COM and .NET (for 64-bit)
- **lib-x64**: GRX libraries (for 64-bit)
- **Samples**: Sample projects, including Dotnet, fact_dg, HelloADS, HelloA and SimplePalette.
  - **Dotnet**: .NET programming samples
    1) **Addline**: .NET programming sample of adding solid lines
    2) **Hello**: .NET programming sample of outputting prompt information
    3) **Vbhello**: Sample of .NET programming with VB .NET
  - **fact_dg**: Sample of LISP function definition
  - **HelloADS**: Sample of ADS programming
  - **HelloARX**: Sample of GRX programming
  - **SimplePalette**: Programming sample of how to create a set Palette windows
- **Utils:** Directory contains sub-directories of GRX extended applications, including APIs for extended function development, e.g. BREP for boundary representation.

Inc: Header files used for programming the GRX

inc-x64: Files used by COM and .NET (for 64-bit)

lib-x64: GRX libraries (for 64-bit)

Utils: Directory contains subdirectories of GRX extended applications, including APIs for extended function development, e.g. BREP for boundary representation.

4. Programming Instruction

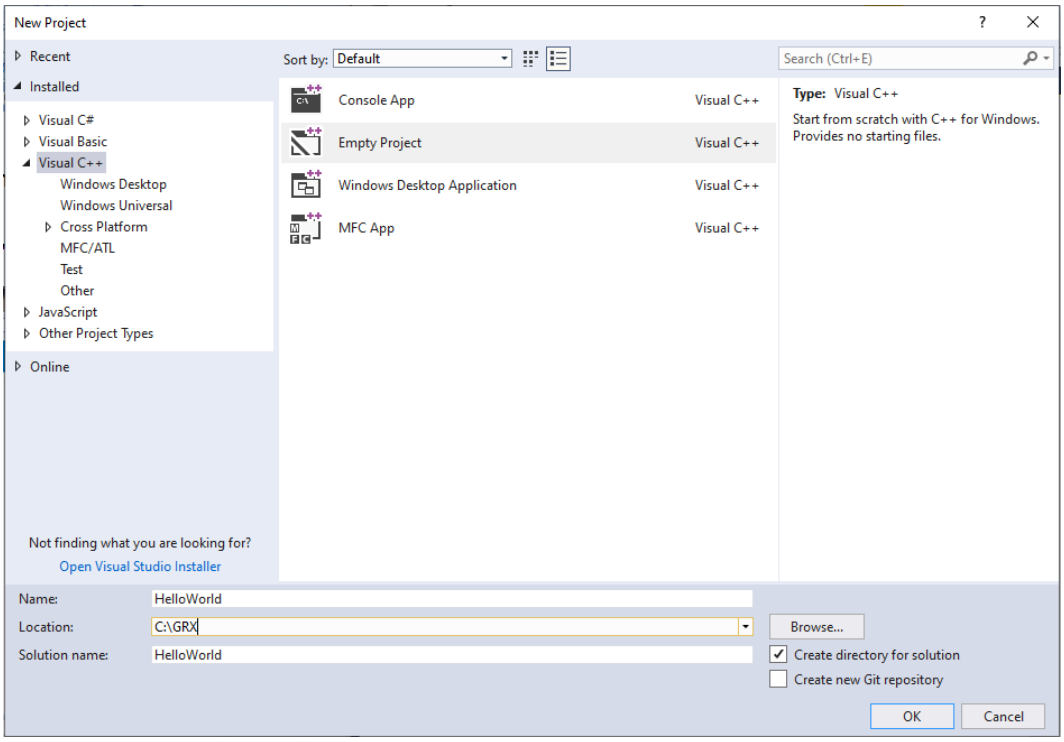
4.1. Create GRX Project under Win 64-Bit

4.1.1. Run Microsoft® Visual Studio 2017

Please click **File→New→Project** to launch the **New Project** dialog window. Select **Visual C++** in the **Installed** on the left side and click **Empty Project** in the middle window.

4.1.2. Input Project Saving Path and Project Name

Input '*HelloWorld*' at the name field in the **New Project** dialog window.

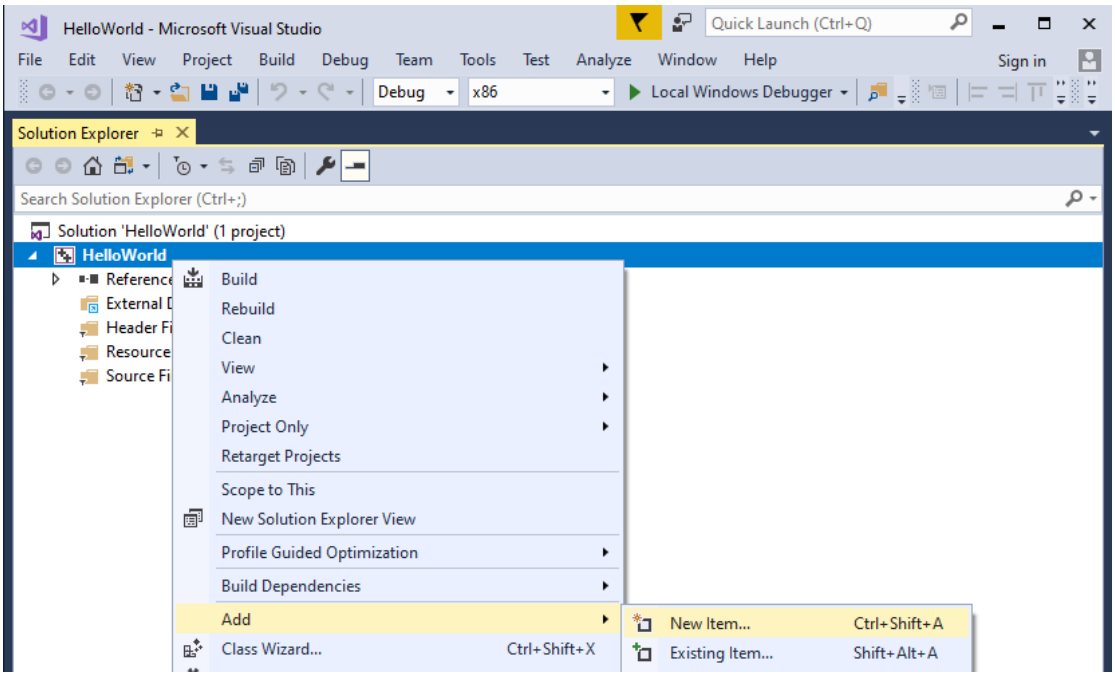


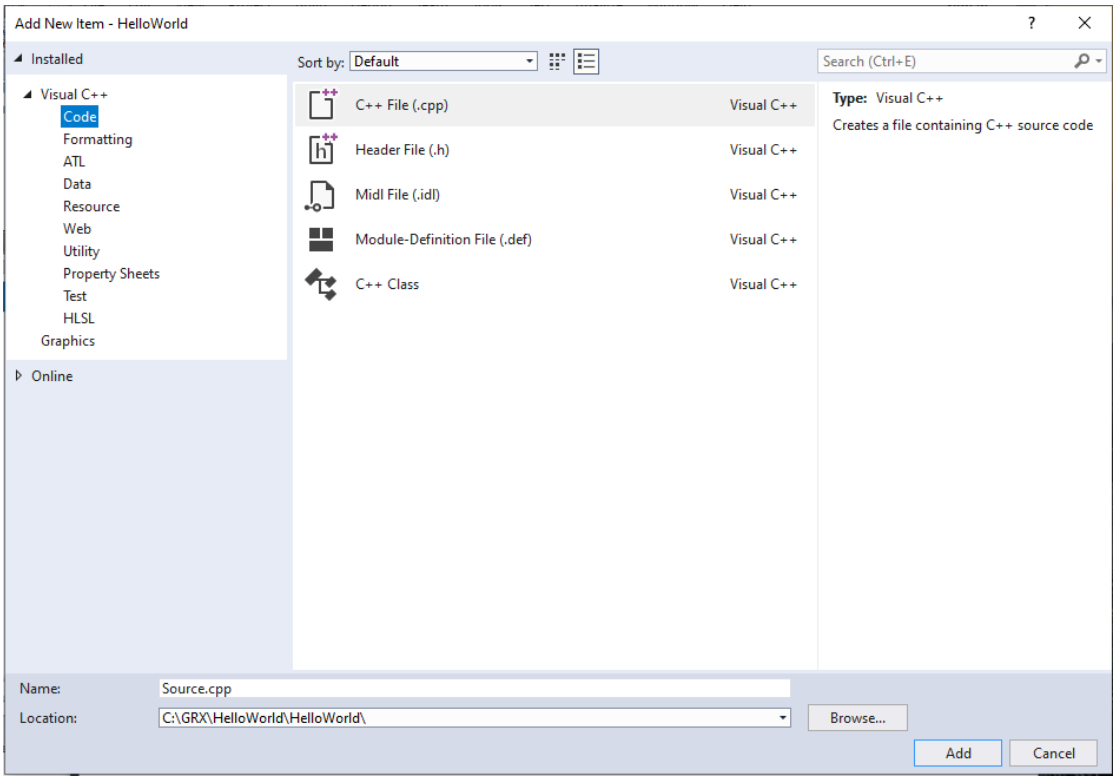
4.1.3. Finish Creating New Project

Click **OK** button to finish creating a new project.

4.1.4. Create New CPP File

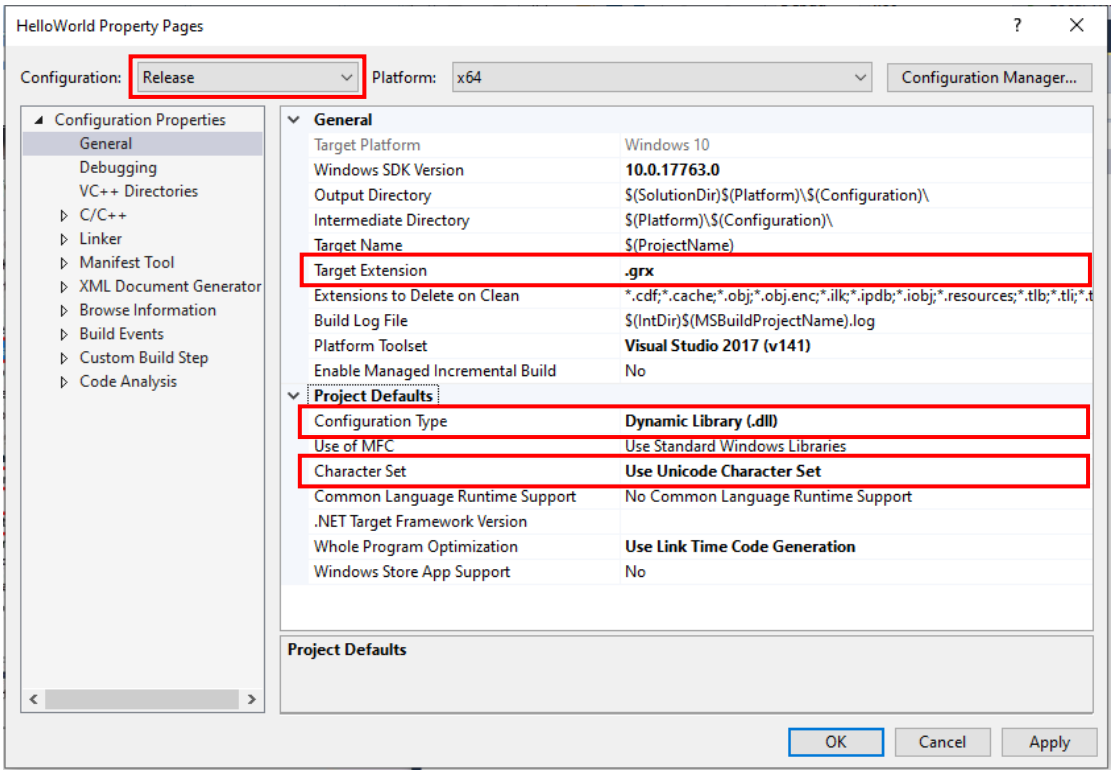
In Visual Studio 2017 dialog window, find the '*HelloWorld*' project you just created in the **Solution Explorer**. Select the '*HelloWorld*' project and right click on it, then select **Add→New Item**. After the **Add New Item - HelloWorld** dialog window pops out, select **Code** under **Visual C++** then select **C++ File (.cpp)** in the middle window.


HelloWorld - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Sign in
Debug
x86
Local Windows Debugger
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'HelloWorld' (1 project)
HelloWorld
Build
Rebuild
Clean
View
Analyze
Project Only
Retarget Projects
Scope to This
New Solution Explorer View
Profile Guided Optimization
Build Dependencies
Add
Class Wizard... Ctrl+Shift+X
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A


Add New Item - HelloWorld
Installed
Visual C++
Code
Formatting
ATL
Data
Resource
Web
Utility
Property Sheets
Test
HLSL
Graphics
Online
Sort by: Default
Search (Ctrl+E)
C++ File (.cpp) Visual C++
Header File (.h) Visual C++
Midl File (.idl) Visual C++
Module-Definition File (.def) Visual C++
C++ Class Visual C++
Type: Visual C++
Creates a file containing C++ source code
Name: Source.cpp
Location: C:\GRX\HelloWorld\HelloWorld\
Browse...
Add
Cancel

4.1.5. Set Compile and Link Configuration

Select the project '*HelloWorld*' in the **Solution Explorer** and select **Property** at the right click menu. After selecting the **Properties** option, the **HelloWorld Properties Pages** pops out.



1) Select **General** in **Configuration Properties** and set as below:

   Target Extension: *.grx*

   Configuration Type: *Dynamic Library (.dll)*

   Character Set: *Use Unicode Character Set*

2) Select **General** in **C/C++** and set as below:

   General/Additional Include Directories: *C:\grxsdk\arx\inc*

   NOTE: Under debug configuration, please delete _DEBUG macro definition and set **Code Generation/Runtime Library** as '*Multi-threaded DLL (/MD)*'.

   Language/Conformance mode: *No*

3) Select **Linker** and set as below:

   General/Additional Library Directories: *C:\grxsdk\arx\lib-x64*

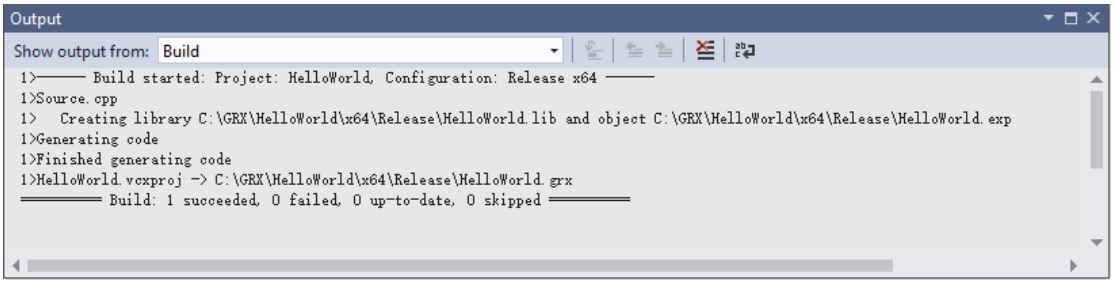
   Input/Additional Dependencies:

   AecModeler.lib;gcad.lib;gcax.lib;gcbase.lib;gcbr.lib;gccore.lib;gcdb.lib;GcDbConstraints.lib;GcDbPoi ntCloudObj.lib;gcdyn.lib;GcGeolocationObj.lib;gcgs.lib;GcImaging.lib;GcModelDocObj.lib;gplot.lib;

   Input/Module Definition File: *C:\grxsdk\arx\inc\AcRxDefault.def*

4) Click **Apply** and then click **OK** to finish the compiler and linker configuration.

5) Compile the project and make sure that the compilation is successful. Otherwise repeat the above steps to reconfigure the project settings.



4.1.6. Add Code

Add the following codes to '*HelloWorld.cpp*'.

```
#include "windows.h"
#include <tchar.h>
#include <arxHeaders.h>

void initApp();
void unloadApp();
void HelloWorld();

void initApp()
{
    //register a command with the GstarCAD command mechanism
    acedRegCmds->addCommand(_T("HELLOWORLD_CMDS"), _T("Hello"), _T("Hello"),
ACRX_CMD_TRANSPARENT, HelloWorld);
}
void unloadApp()
{
    acedRegCmds->removeGroup(L"HELLOWORLD_CMDS");
}
void HelloWorld()
{
    //print "Hello World" in GstarCAD command line
    acutPrintf(_T("\nHello World!"));
}
```

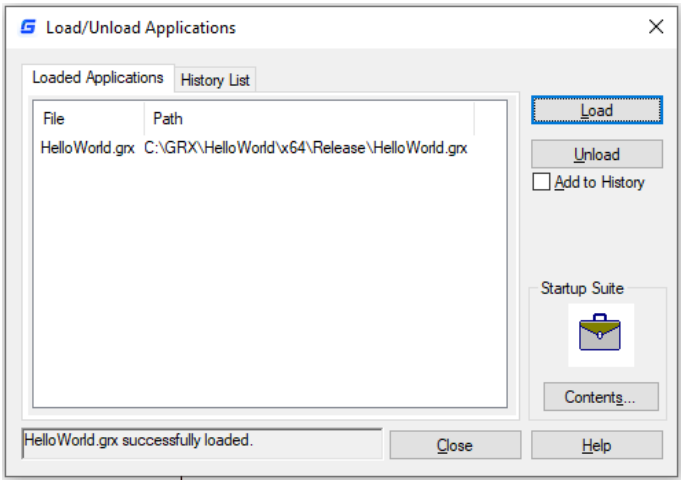
```
extern "C" AcRx::AppRetCode gcrxEntryPoint(AcRx::AppMsgCode msg, void *pkt)
{
	switch (msg)
	{
	case AcRx::kInitAppMsg:
		acrxDynamicLinker->unlockApplication(pkt);
		acrxDynamicLinker->registerAppMDIAware(pkt);
		initApp();
		break;
	case AcRx::kUnloadAppMsg:
		unloadApp();
		break;
	default:
		break;
	}
	return AcRx::kRetOK;
}
```

4.1.7. Compile Program

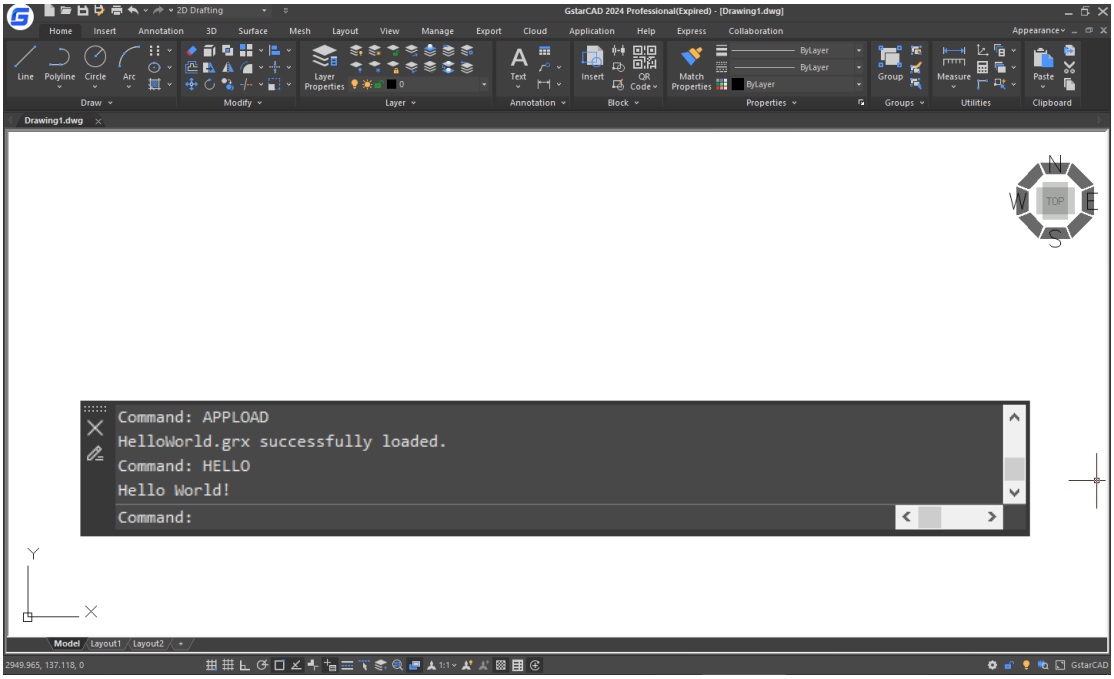
In Visual Studio 2017, click **Build→Build Solution** to create '*HelloWorld.grx*' file in '*C:\GRX\HelloWorld\x64\Release*' directory.

4.1.8. Run Program

Run GstarCAD and input '*appload*' at the command line, or select **Tools→Loading Applications** from menu to launch **Load/Unload Applications** dialog window. Select '*HelloWorld.grx*' and click **Load** to load it into GstarCAD.



Close the **Load/Unload Applications** dialog window and input '*hello*' at command line, '*Hello World!*' will be displayed at the command line as shown below.



4.2. GRX Programming with MFC Library

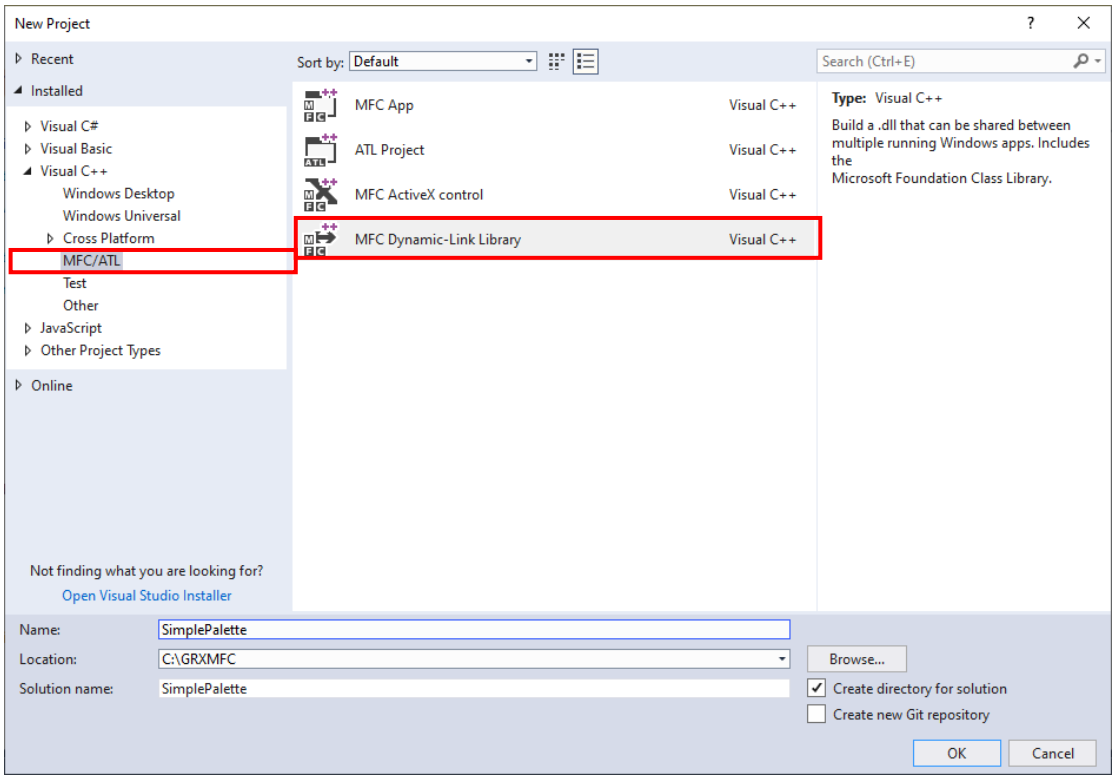
4.2.1. Run Microsoft® Visual Studio 2017

Please click **File→New→Project** to launch the **New Project** dialog window. Select **Visual C++** in the **Installed** on the left side and click **MFC/ATL** in the pull-down menu. Click **MFC Dynamic-Link Library (DLL)** in the middle window.

The following sample shows how to create a '*SimplePalette*' project.

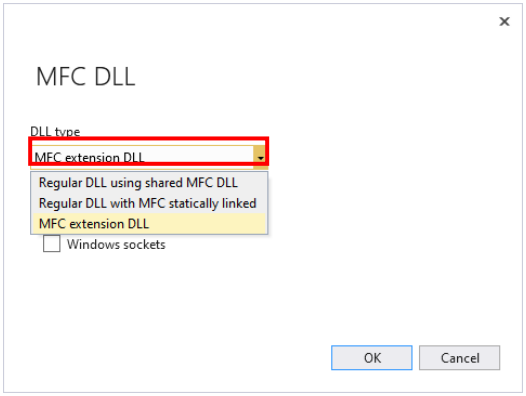
4.2.2. Input Project Saving Path and Project Name

Input '*SimplePalette*' at the name field in **New Project** dialog window, as shown below.



4.2.3. Choose DLL Type

Click **OK,** the **MFC DLL** dialog window pops out. Select **MFC Extension DLL** and click **OK**.

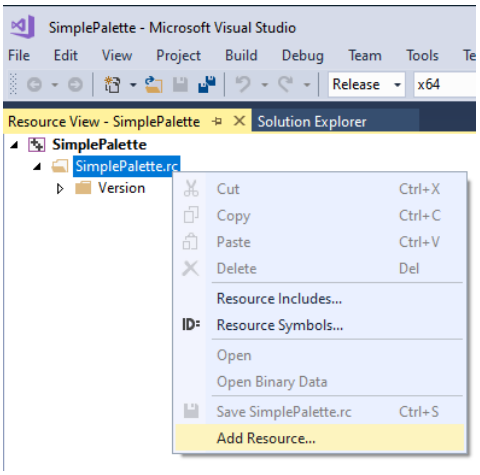


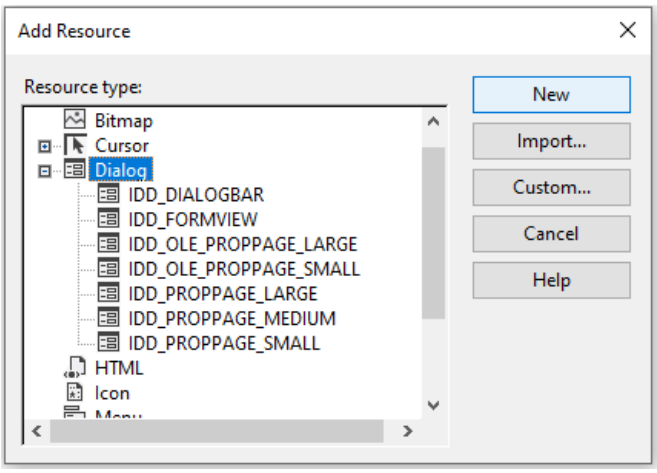
4.2.4. Finish Creating New Project

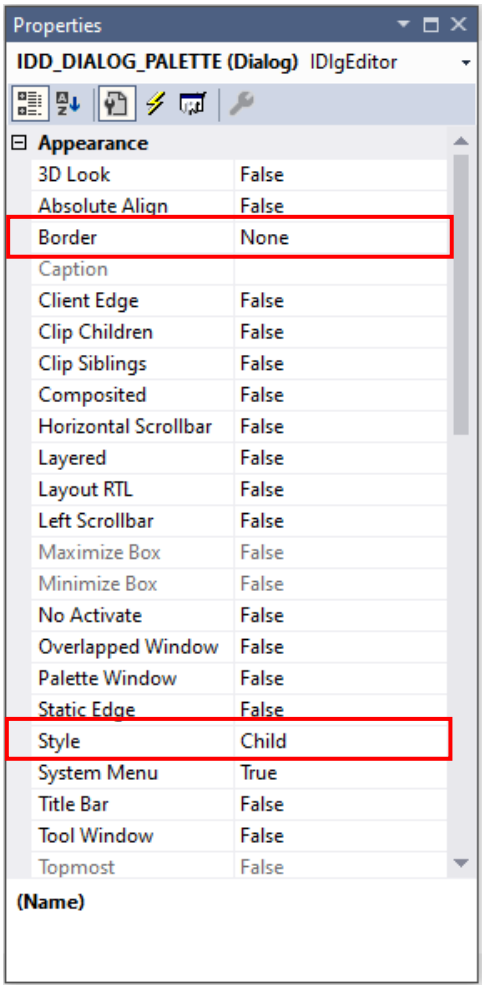
Click **OK** on **New Project** dialog window to finish creating '*SimplePalette*' project.

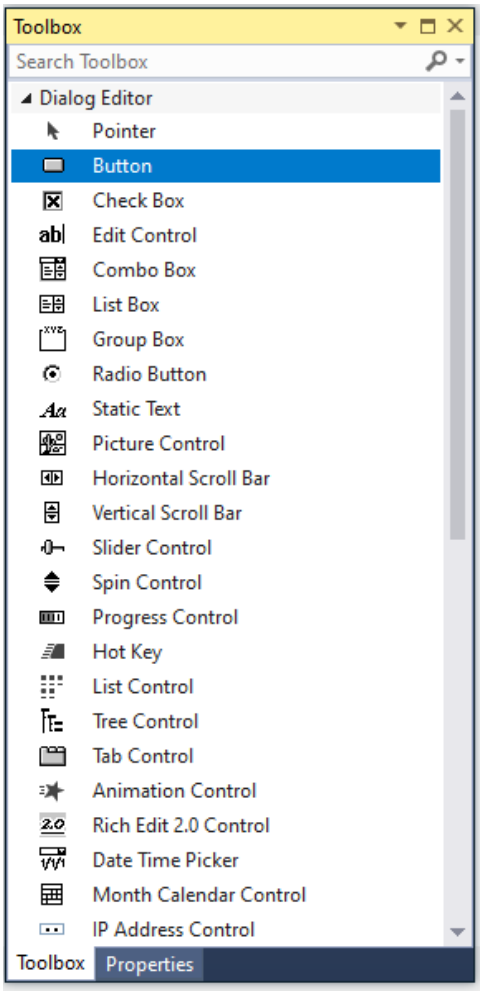
4.2.5. Create New Resource and Corresponding Class

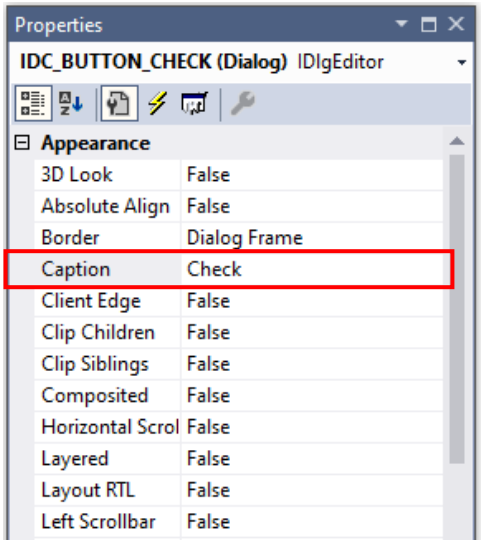
Launch the **Resource View** of Visual Studio 2017 (Ctrl+Shift+E), right click on '*SimplePalette.rc*' and select **Add Resource...** in the drop-down menu to pop out **Add Resource** dialog window. Select **Dialog** in the **Resource Type** window and click **New** to add a new dialog. Change the **ID** of newly created '*IDD_DIALOG1*' dialog to '*IDD_DIALOG_PALETTE*'. Right click on '*IDD_DIALOG_PALETTE*' and select **Properties** to set **Border** as '*None*', **Style** as '*Child*'. Then add an **Edit** control and a **Button** from **Toolbox** (Ctrl+Alt+X) and change the button **Caption** to '*Check*' and the **ID** to '*IDC_BUTTON_CHECK*', as shown below.

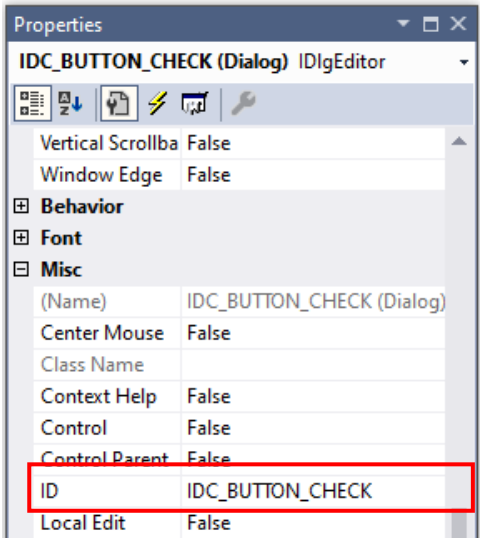




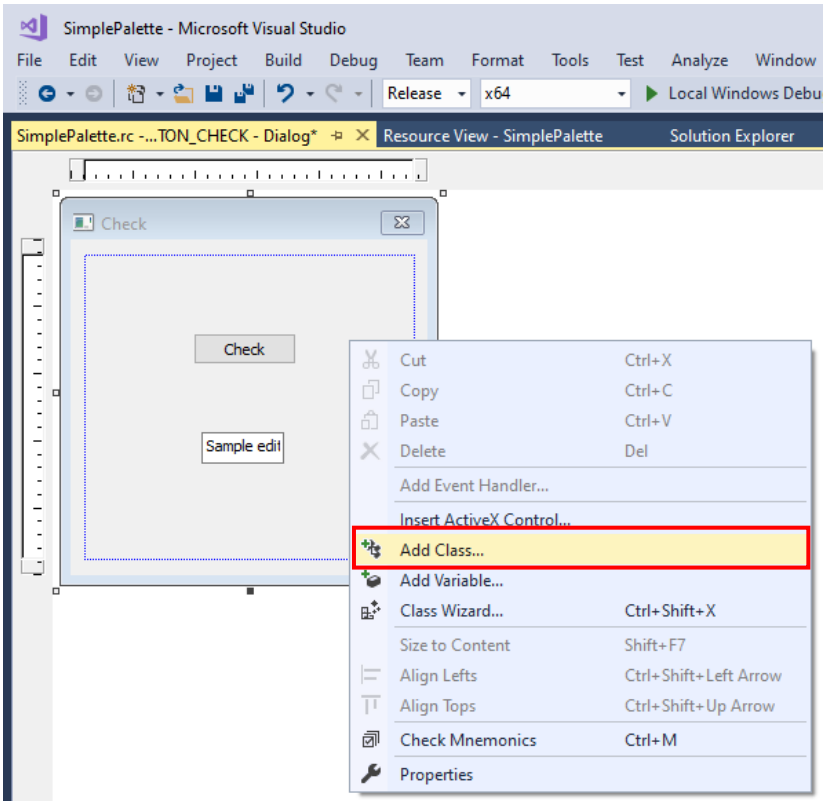

Properties
IDD_DIALOG_PALETTE (Dialog) IDlgEditor
Appearance
3D Look False
Absolute Align False
Border None
Caption
Client Edge False
Clip Children False
Clip Siblings False
Composited False
Horizontal Scrollbar False
Layered False
Layout RTL False
Left Scrollbar False
Maximize Box False
Minimize Box False
No Activate False
Overlapped Window False
Palette Window False
Static Edge False
Style Child
System Menu True
Title Bar False
Tool Window False
Topmost False
(Name)

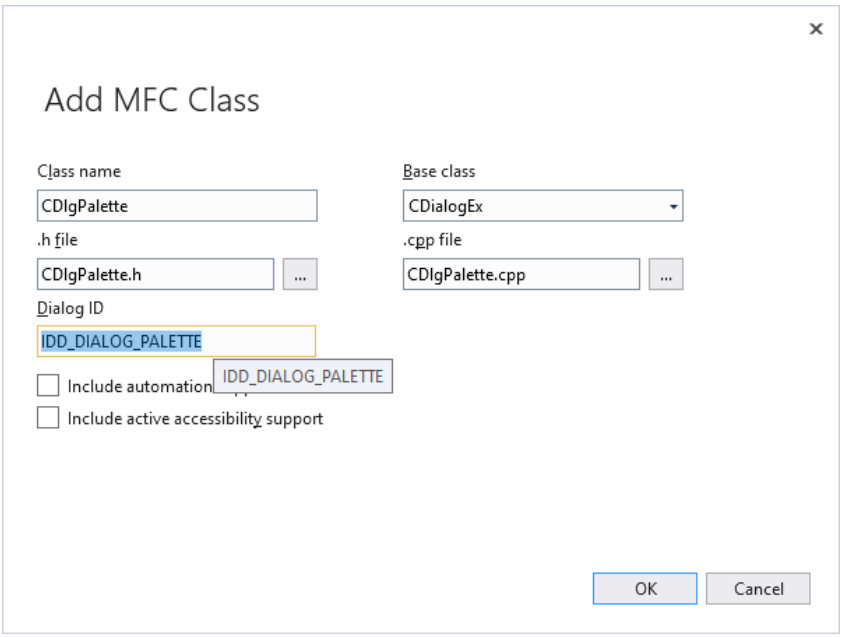

Toolbox
Search Toolbox
Dialog Editor
Pointer
Button
Check Box
Edit Control
Combo Box
List Box
Group Box
Radio Button
Static Text
Picture Control
Horizontal Scroll Bar
Vertical Scroll Bar
Slider Control
Spin Control
Progress Control
Hot Key
List Control
Tree Control
Tab Control
Animation Control
Rich Edit 2.0 Control
Date Time Picker
Month Calendar Control
IP Address Control
Toolbox
Properties


Properties
IDC_BUTTON_CHECK (Dialog) IDlgEditor
Appearance
3D Look False
Absolute Align False
Border Dialog Frame
Caption Check
Client Edge False
Clip Children False
Clip Siblings False
Composited False
Horizontal Scrol False
Layered False
Layout RTL False
Left Scrollbar False


Properties
IDC_BUTTON_CHECK (Dialog) IDlgEditor
Vertical Scrollba False
Window Edge False
Behavior
Font
Misc
(Name) IDC_BUTTON_CHECK (Dialog)
Center Mouse False
Class Name
Context Help False
Control False
Control Parent False
ID IDC_BUTTON_CHECK
Local Edit False

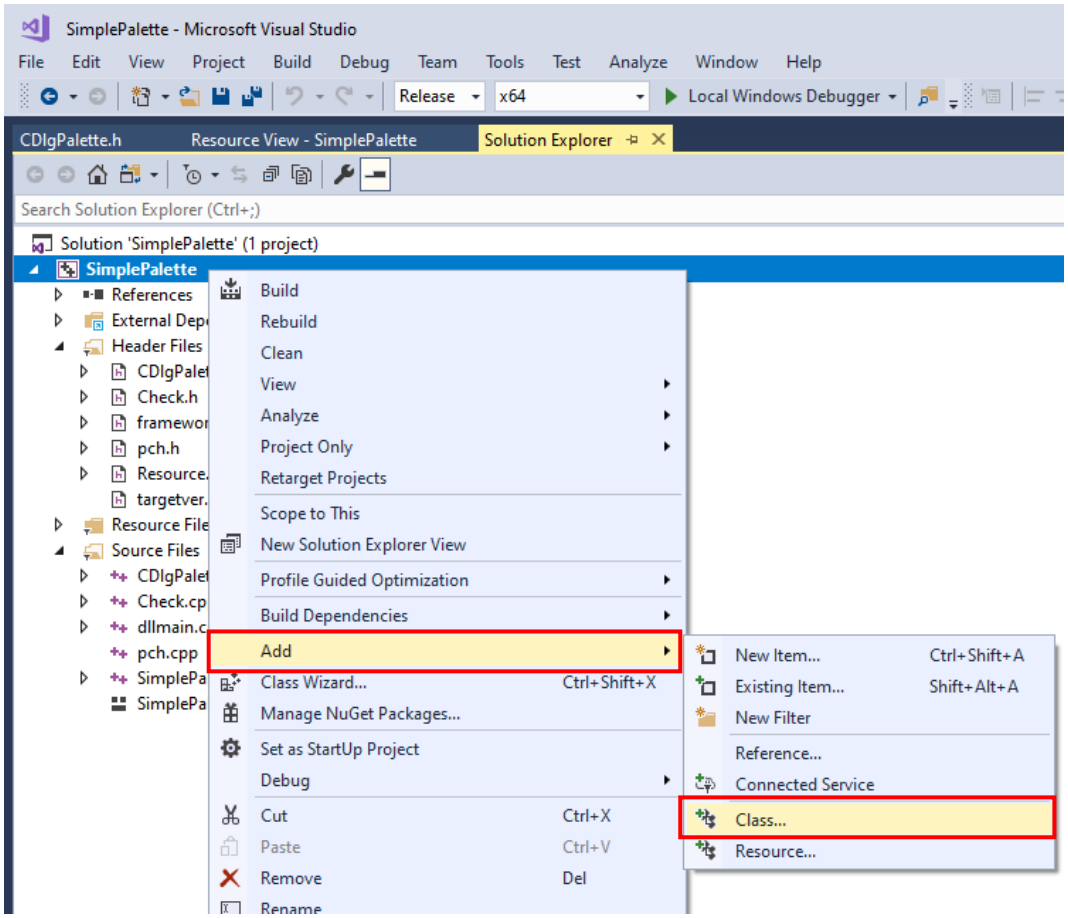
Right click on the newly created '*IDD_DIALOG_PALETTE*' dialog and select the **Add Class...** to pop out **Add MFC Class** dialog window. Set **Class Name** as '*CDlgPalette*' and click **OK**.

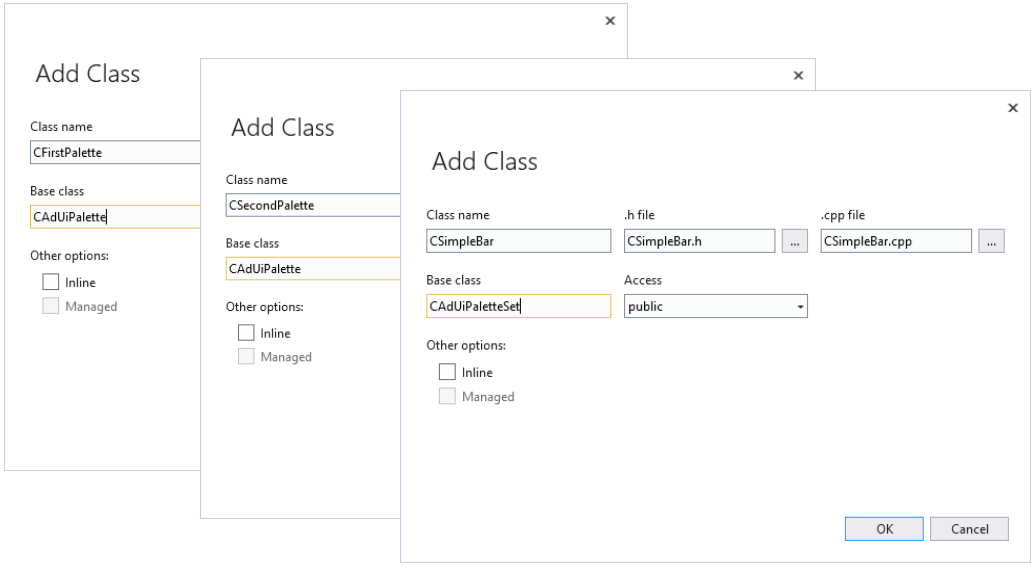




4.2.6. Create Three New C++ Class File

In the **Solution Explorer** of Visual Studio 2017, right click on the project '*SimplePalette*' and select **Add→Class...** to pop out **Add Class** dialog window, as shown below. Add 3 new classes named as '*CFirstPalette*', '*CSecondPalette*' and '*CSimpleBar*' and set the **Base class** as '*CAdUiPalette*', '*CAdUiPalette*' and '*CAdUiPaletteSet*' respectively.





4.2.7. Add Code

Add the following codes to the end of '*stadfx.h*' file (If the version of Visual Studio 2017 is higher than 15.9.17, it automatically generates a '*pch.h*' file which is equivalent to the '*stdafx.h*').

```
#include <arxHeaders.h>
```

(For the class codes of '*CDlgPalette*', '*CFirstPalette*', '*CSecondPalette*' and '*CSimpleBar*', please refer to samples (.cpp and .h files) of '*DlgPalette*', '*FirstPalette*', '*SecondPalette*' and '*SimpleBar*' in '*C:\grxsdk\arx\samples\SimplePalette*')

Add the following codes to '*SimplePalette.cpp*':

```
// SimplePalette.cpp: Define the initialization routine for the DLL.
//

#include "stdafx.h"
#include "CSimpleBar.h"

CSimpleBar* gpBar = NULL;

static void doIt()
{
	if (gpBar)
	{
		acedGetAcadFrame()->ShowControlBar(gpBar, TRUE, FALSE);
	}
	else
	{
		acedInitGet(0, L"Float Dock");
		ACHAR result[132] = { 0 };
		int rc = acedGetKword(L"\nFloat or Docking?", result);
		bool bFloat = true;
		if (rc == RTNONE)
		{
```

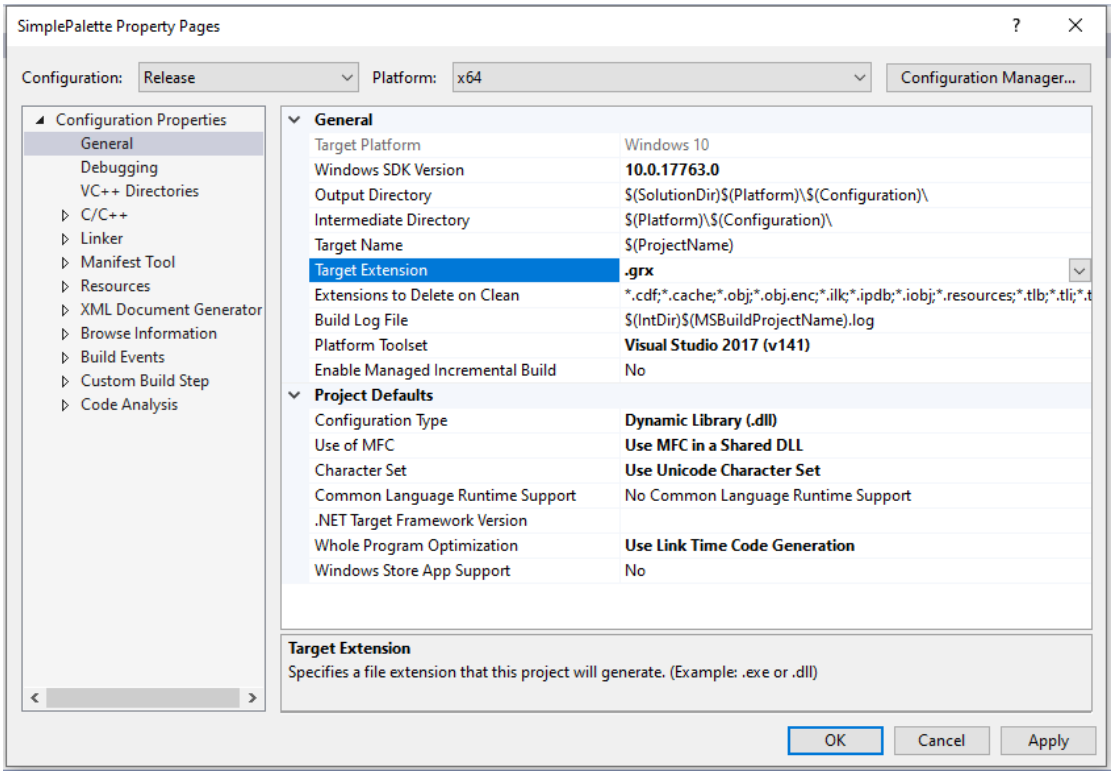
```
            bFloat = true;
        }
        else if(rc != RTNORM)
        {
            return;
        }
        else
        {
            bFloat = _wcsicmp(result, L"Float") == 0;
        }

        CSize sizeDefault(400, 450);
        CRect rcDefault(CPoint(150, 200), sizeDefault);

        gpBar = new CSimpleBar;

        DWORD dwStyle = WS_CHILD | WS_VISIBLE | CBRS_SIZE_DYNAMIC | CBRS_GRIPPER;
        if (bFloat)
        {
            dwStyle |= CBRS_FLOATING;
        }
        else
        {
            dwStyle |= CBRS_LEFT;
        }

        gpBar->Create(L"Simple", dwStyle, rcDefault, acedGetAcadFrame());
        gpBar->SetAutoRollup(FALSE);
        gpBar->EnableDocking(CBRS_ALIGN_LEFT | CBRS_ALIGN_RIGHT |
CBRS_ALIGN_BOTTOM);
        if (bFloat)
        {
            acedGetAcadFrame()->FloatControlBar(gpBar, CPoint(100, 100), CBRS_ALIGN_TOP);
        }
```

```
            else
            {
                acedGetAcadFrame()->DockControlBar(gpBar, AFX_IDW_DOCKBAR_RIGHT);
            }
        }
}

void initApp()
{
        acedRegCmds->addCommand(_T("ASDK_SAMPLES_SIMPLEPALETTE"),
            _T("ASDK_SIMPLEPALETTE"), _T("SIMPLEPALETTE"), ACRX_CMD_MODAL,
            doIt);
}



void unloadApp()
{
        acedRegCmds->removeGroup(_T("ASDK_SAMPLES_SIMPLEPALETTE"));

        if (gpBar)
        {
            gpBar->DestroyWindow();
            delete gpBar;
            gpBar = NULL;
        }
}



extern "C" AcRx::AppRetCode
acrxEntryPoint(AcRx::AppMsgCode msg, void* appId)
{
        switch (msg) {
        case AcRx::kInitAppMsg:
            acrxDynamicLinker->unlockApplication(appId);
```

```
            acrxDynamicLinker->registerAppMDIAware(appId);
            initApp();
            break;
        case AcRx::kUnloadAppMsg:
            unloadApp();
        }
        return AcRx::kRetOK;
}
```

4.2.8. Set Compile and Link Configuration

1) In the **Solution Explorer**, right click the project '*SimplePalette*' and select **Properties** to pop out the **SimplePalette Properties Pages** dialog window. The following settings in this section are completed in this dialog window.



2) Select **General** in **Configuration Properties** and set as below:

   Target Extension: *.grx*

   Configuration Type: *Dynamic Library (.dll)*

   Character Set: *Use Unicode Character Set*

3) Select **General** in **C/C++** and set as below:

   General-Additional Include Directories: *C:\grxsdk\arx\inc*

   NOTE: Under debug configuration, please delete _DEBUG macro definition and set **Code Generation/Runtime Library** as *'Multi-threaded DLL (/MD)'*.

   Language-Conformance mode: *No*

4) Select **Linker** and set as below:

   General-Additional Library Directories: *C:\grxsdk\arx\lib-x64*

   Input-Additional Dependencies:

   AecModeler.lib;gcad.lib;gcax.lib;gcbase.lib;gcbr.lib;gccore.lib;gcdb.lib;GcDbConstraints.lib;GcDbPointCloudObj.lib;gcdyn.lib;GcGeolocationObj.lib;gcgs.lib;GcImaging.lib;GcModelDocObj.lib;gplot.lib;

Input-Module Definition File: *C\grxsdk\arx\inc\AcRxDefault.def*

Advance-Target Machine: *MachineX64 (/MACHINE:X64)* (for 64-bit)

5) Click **OK** to finish the settings.

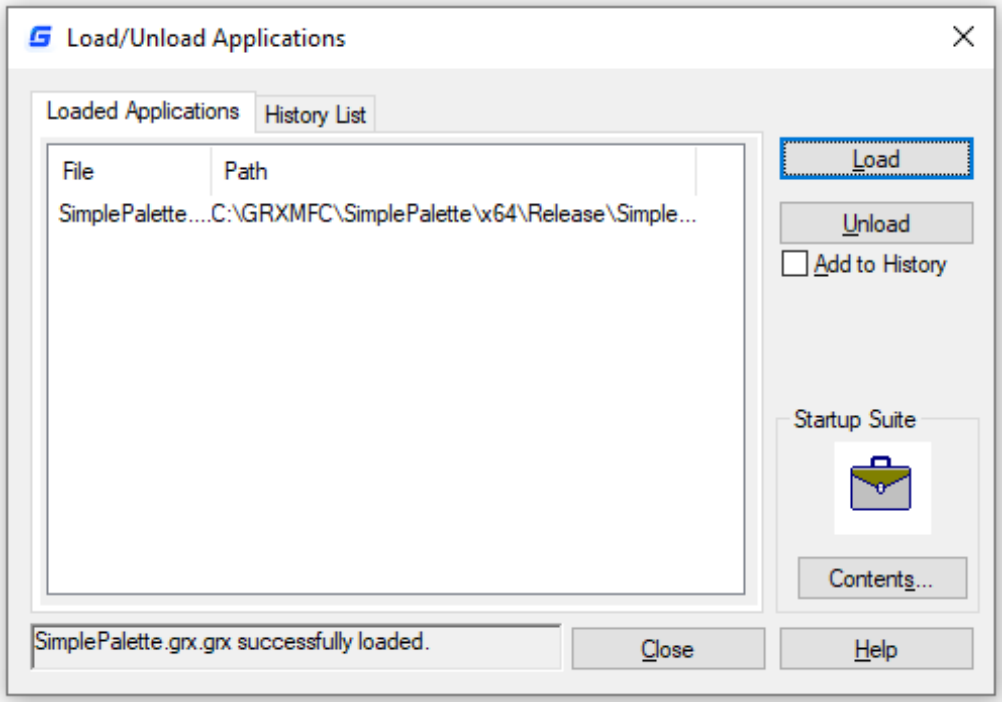
6) Compile the project and make sure that the compilation is successful. Otherwise repeat the above steps to reconfigure the project settings.

4.2.9. Compile Program

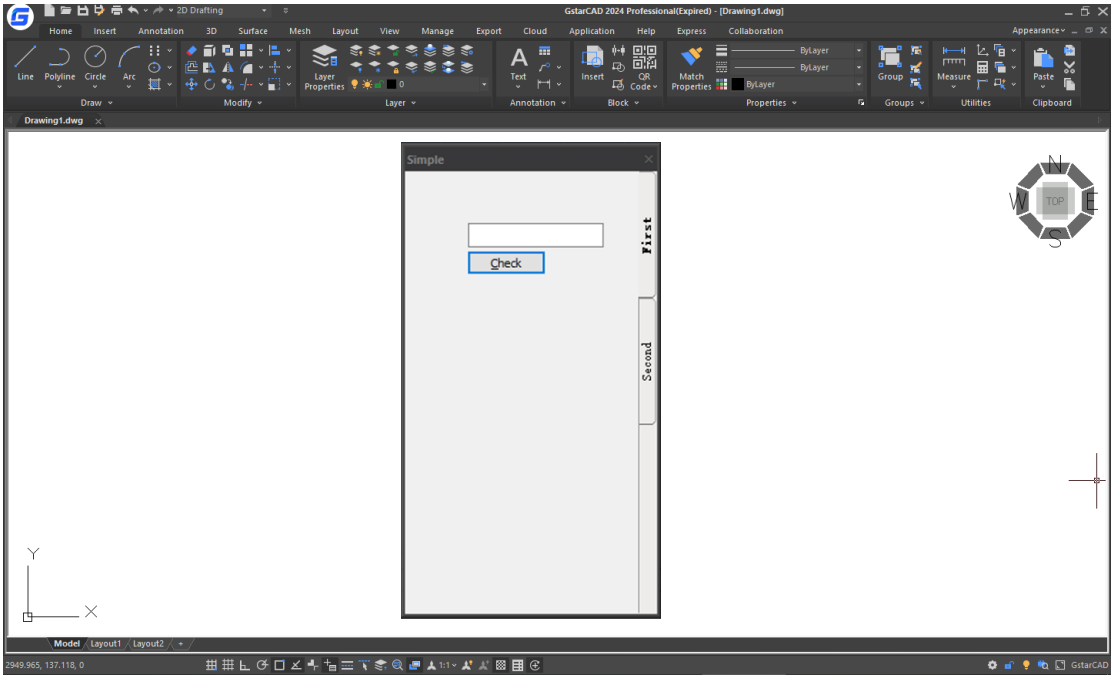
In Visual Studio 2017, click **Build→Build Solution** to create '*SimplePalette.grx*' file in the '*C:\GRXMFC\SimplePalette\x64\Release*' directory.

4.2.10. Run Program

Run GstarCAD and input '*appload*' at the command line, or select **Tools→Loading Applications** from menu to launch **Load/Unload Applications** dialog window. Select '*SimplePalette.grx*' and click **Load** to load it into GstarCAD.



Close the **Load/Unload Applications** dialog window and input '*simplepalette*' at command line, '*Float or Docking?*' will be displayed at the command line. Press **Enter** key and a dialog window will pop out as shown below.


GstarCAD 2024 Professional(Expired) - [Drawing1.dwg]
Simple
Check
First
Second
Model
Layout1
Layout2

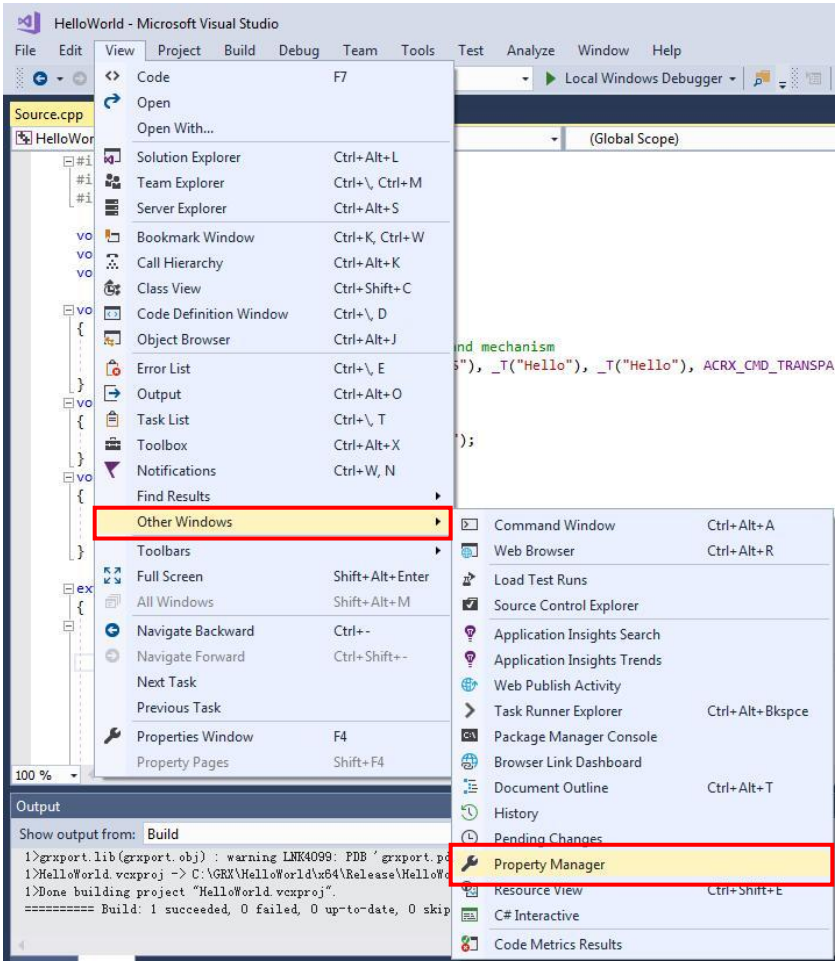
5. How to Use Project Property Sheet File

The **Project Property Sheet** is used to quickly set the project configurations of a new project. It also helps to reduce the possibility of misoperation on project settings.

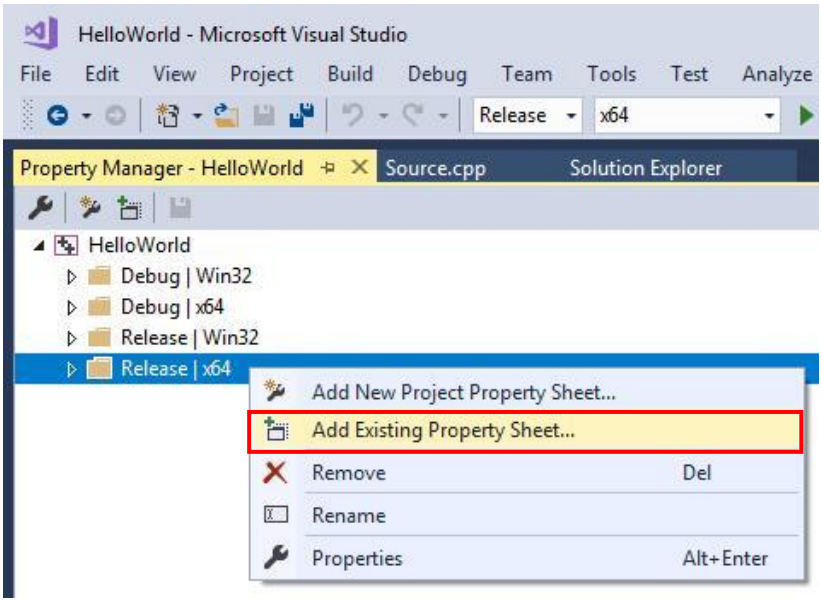
The project properties can be applied independently to any combination of build configuration (Debug or Release) and target GstarCAD platform (Win32, x64 or ARM).

GRX SDK provides the project property sheet for configurations of **Additional Dependencies**, **Additional Include Directories** and **Additional Library Directories**.

1) Take project '*HelloWorld*' as a sample to set configurations by using project property sheet file. Open project '*HelloWorld*', click **View→Other Windows→Properties Manager**, as shown below.



2) In **Properties Manager** dialog window, right click on **Release|x64** and then select **Add Existing Property Sheet...**.



3) Add property sheet file '*C:\grxsdk\arx\inc\arx.props*'. It contains property sheet file '*dbx.props*' which is used to add the following dependencies:
'*AecModeler.lib;gcad.lib;gcax.lib;gcbase.lib;gcbr.lib;gccore.lib;gcdb.lib;GcDbConstraints.lib;GcDbPointCloudObj.lib;gcdyn.lib;GcGeolocationObj.lib;gcgs.lib;GcImaging.lib;GcModelDocObj.lib;gplot.lib*;'
In the same way, add '*C:\grxsdk\arx\inc\rxsdk_releasecfg.props*'. It contains '*rxsdk_common.props*' which is used to add **Additional Include Directories** and **Additional Libray Directories**. In addition, '*C:\grxsdk\arx\inc\ rxsdk_debugcfg.props*' is necessary if it's a debug project.

4) In **Solution Explore** of Visual Studio 2017, right click on project '*HelloWorld*' and select **Properties**. Click **C/C++** and then click **General**, change the **Additional Include Directories** to '*C:\grxsdk\arx\inc*'. Click **Linker** and then click **Input**, change **Additional Dependencies** to '*grxport.lib*' and change **Module Definition File** to '*C:\grxsdk\arx\inc\AcRxDefault.def*'.

5) Refer to **4.1.7** and **4.1.8** to compile the project and run the program.

6. Description of GRX Class Library

The following libraries are frequently used in programming with GRX. These libraries have same functions with the corresponding ARX libraries.

- **GcRx**: same as **AcRX** of **ARX**, classes for binding an application and for runtime class registration and identification
- **GcEd**: same as **AcEd** of **ARX**, classes for registering commands and for event notification
- **GcDb**: same as **AcDb** of **ARX**, GstarCAD database class
- **GcGi**: same as **AcGi** of **ARX**, graphics classes for rendering entities
- **GcGe**: same as **AcGe** of **ARX**, utility classes for common linear algebra and geometric objects

6.1. GcRx

The **GcRx** class library provides system-level functionality such as DLL initialization and linking, and runtime class registration and identification as below:

- Object runtime class identification and inheritance analysis
- Runtime addition of new protocol to an existing class
- Object equality and comparison testing
- Object copy

6.2. GcEd

The **GcEd** class library is used to define and register new GstarCAD commands which operate in the same manner as the original ones.

6.3. GcDb

The **GcDb** classes are components of the GstarCAD database. This database stores all the information for the graphical objects that compose a drawing, and the non-graphical objects (such as layers, linetypes, and text styles) that are also part of a drawing.

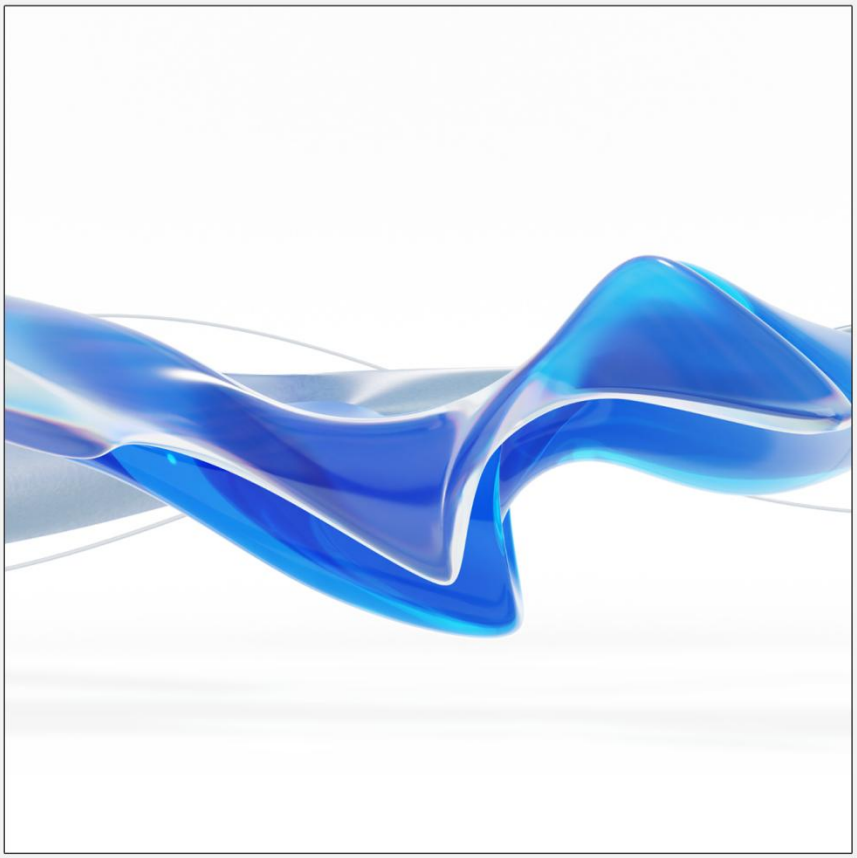
6.4. GcGi

The **GcGi** class library provides the graphic interface used for drawing CAD entities.

6.5. GcGe

The **GcGe** class library provides classes used for performing common 2D and 3D geometric operations, including utility classes such as vectors and matrices and basic geometric objects such as points, curves, and surfaces.

7. Copyright



https://www.gstarcad.net/