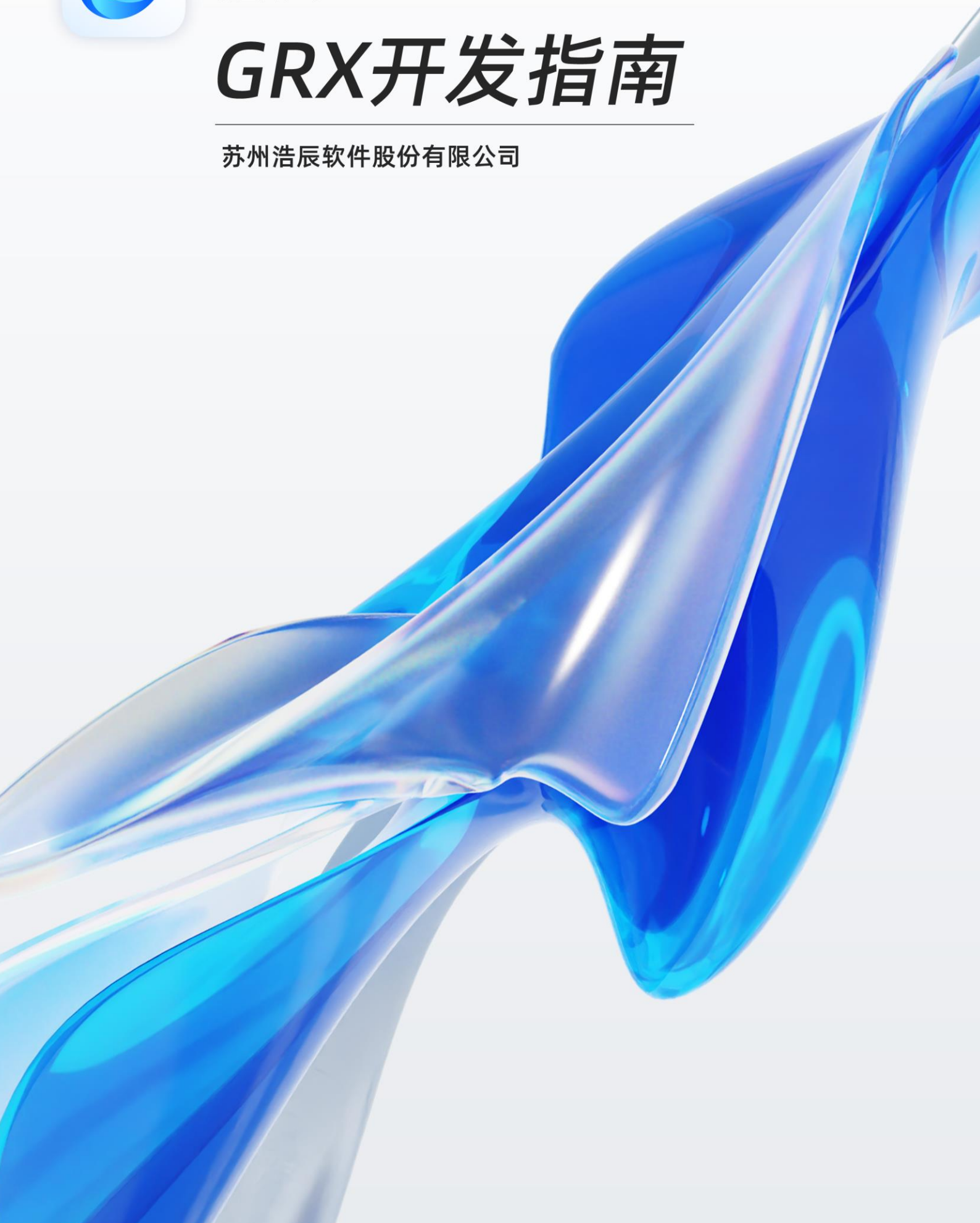




浩辰CAD 2025

GRX开发指南

苏州浩辰软件股份有限公司



目 录

1 什么是 GRX?	4
2 开发环境	5
3 安装 GRX SDK	6
4 开发指南	7
4.1 创建 win32 下 GRX 项目	7
4.1.1 运行 Microsoft Visual Studio 2017	7
4.1.2 输入项目的存放路径和项目名称	7
4.1.3 完成项目创建	8
4.1.4 创建一个新的 cpp 文件	8
4.1.5 设置编译和链接项	8
4.1.6 添加代码	10
4.1.7 编译程序	11
4.1.8 运行程序	11
4.2 带 MFC 库的 GRX 程序	13
4.2.1 运行 Microsoft Visual Studio 2017	13
4.2.2 输入项目的存放路径和项目名称	13
4.2.3 选择 DLL 类型	13
4.2.4 完成项目创建	14
4.2.5 创建一个新资源及对应的类	14
4.2.6 创建三个新的 C++ 类文件	15
4.2.7 添加代码	16
4.2.8 设置编译和链接项	19
4.2.9 编译程序	21
4.2.10 运行程序	21
5 如何使用项目属性表文件	23
6 GRX 类库说明	25
6.1 GcRx	25
6.2 GcEd	25

6.3 GcDb	25
6.4 GcGi	26
6.5 GcGe	26
7 版权声明	27

1 什么是 GRX?

GRX 是浩辰 CAD®的运行时扩展 (Runtime eXtension) 编程环境, 是浩辰 CAD®平台推出的最新二次开发软件包。它提供了以 C++为基础, 面向对象的开发环境及应用程序接口。这些接口可用来开发浩辰 CAD 应用程序、扩展 CAD 类和协议, 以及创建新的命令。

GRX 能实现源码级兼容二次开发商在 AutoCAD®平台上开发的 ARX 应用程序。开发者只需要进行简单的项目配置, 即可将运行在 AutoCAD 平台上的 ARX 程序移植到浩辰 CAD 平台上运行。

GRX 开发包可以用来直接访问浩辰 CAD 数据库、图形系统和用户自定义命令等。另外这些接口可以和 Visual LISP 以及其他编程语言一起使用。

用户可以使用 GRX 来完成下面的工作:

- 访问浩辰 CAD 数据库。
- 与浩辰 CAD 编辑器交互。
- 使用 MFC 创建用户界面。
- 支持多文档环境。
- 创建用户自定义类。
- 建立复杂应用程序。
- 与其他编程环境交互。

2 开发环境

- Microsoft Visual Studio 2017 （版本 15.9.17）
- Windows SDK 10.0.17763.0
- 处理器：
基本要求：1.6 GHz 处理器，建议：3.0 GHz 处理器及以上
- 内存：
基本要求：2 GB，建议：8 GB 及以上
- 操作系统
Windows11

Windows 10 版本 1507 或更高版本：家庭版、专业版、教育版和企业版(不支持 LTSC 和 Windows 10 S)

Windows 8.1 （带有更新 2919355）：核心版、专业版和企业版

Windows 7 SP1 （带有最新的 Windows 更新）：家庭普通版、家庭高级版、专业版、企业版和旗舰版

显示器分辨率：

传统显示器：1028 x 800 及以上真彩色显示器，包括 4K(3840 x 2160) 显示器
- GRX SDK 2025
- 浩辰 CAD 2025
- .NET Framework 4.8 及以上版本

3 安装 GRX SDK

将 GRXSDK.ZIP 解压到磁盘中，例如 C:\grxsdk。解压完成后会在 C:\grxsdk 路径下生成 5 个文件夹：arx、inc、inc-x64、lib-x64、utils。其中：

- arx 目录包含从 arx 程序移植到 GRX 程序使用的头文件，库文件及样例程序等。包含以下目录
 - inc 包含从 arx 程序移植到 GRX 程序使用的头文件
 - inc-x64 包含 64 位下 com 和 dotnet 使用的文件。
 - lib-x64 包含 64 位下引用的 GRX 库文件。
 - Samples 包含例子工程文件，分别是 dotnet、fact_dg、HelloADS、HelloARX、SimplePalette。

其中：

✧ Dotnet 文件夹包含 dotnet 使用实例，分别为 addline、hello、vbhello。其中：

- Addline 为 dotnet 增加实体线的实例。
- Hello 为 dotnet 输出提示信息的实例。
- Vbhello 为如何在 vb.net 下使用 dotnet 接口进行开发的实例。

✧ fact_dg 为 lisp 函数定义实例。

✧ HelloADS 为如何开发 ads 类型工程的实例。

✧ HelloARX 为如何使用 arx 接口进行开发的实例。

✧ SimplePalette 为如何使用 palette 类型模块的实例。

- utils 目录包含作为 GRX 扩展的应用程序的子目录，包括用于边界表示的 brep 等扩展功能开发接口。

- inc 包含开发 GRX 程序使用的头文件。
- inc-x64 包含 64 位下 com 和 dotnet 使用的文件。
- lib-x64 包含 64 位下引用的 GRX 库文件。

utils 目录包含作为 GRX 扩展的应用程序的子目录，包括用于边界表示的 brep 等扩展功能开发接口。

4 开发指南

4.1 创建 win32 下 GRX 项目

4.1.1 运行 Microsoft Visual Studio 2017

选择【文件】菜单，再选择【新建】子菜单，最后选择【新建】菜单的子菜单【项目】。点击【项目】菜单项，会弹出“新建项目”对话框中，在左边的【已安装】模板树中，选择【Visual C++】，右边列表中选择【空项目】。

下面以创建“HelloWorld”工程项目为例。

4.1.2 输入项目的存放路径和项目名称

在“新建项目”对话框中“名称”标签后填入“HelloWorld”。如图 4.1.1 所示。

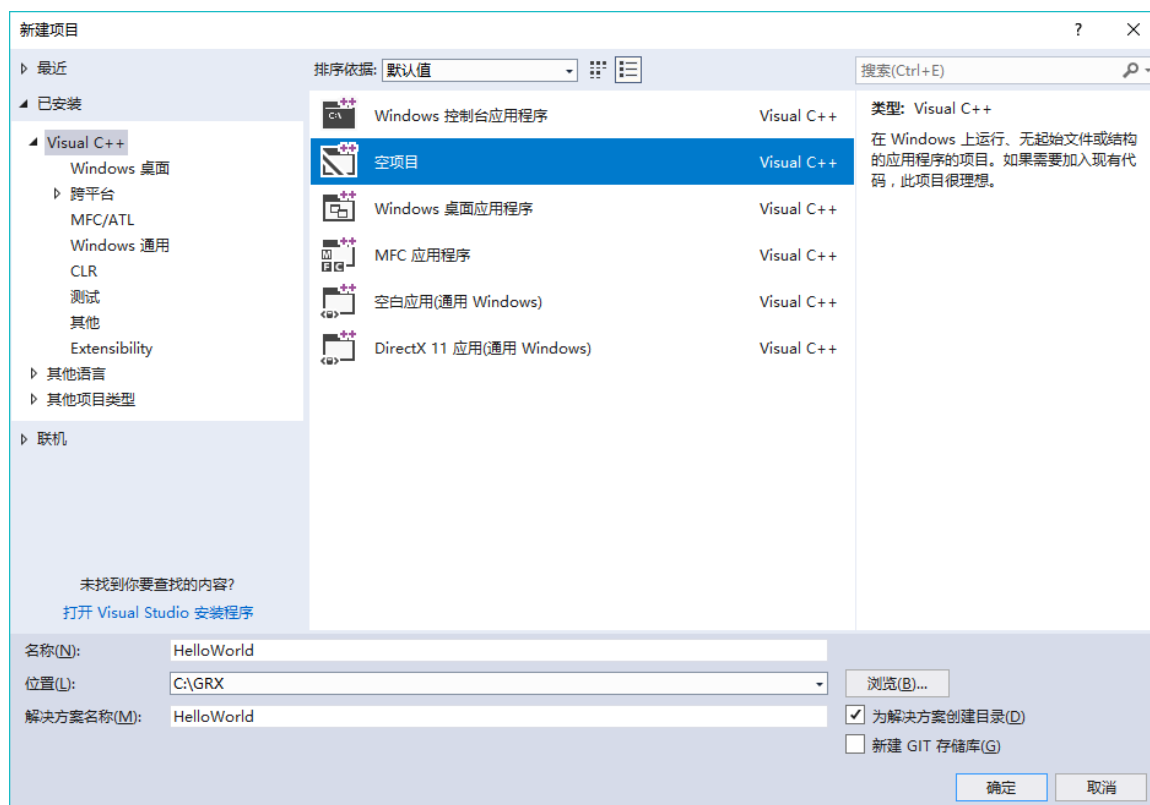


图 4.1.1

4.1.3 完成项目创建

点击上图对话框中的【确定】按钮，即完成了一个新项目的创建。

4.1.4 创建一个新的 cpp 文件

在 Visual Studio 2017 的“解决方案资源管理器”中，选中项目名称“HelloWorld”，点右键，点【添加】菜单项，选【新建项(W)】，出现【添加新项】对话框。在左边的类别树下选【代码】节点，右边的模板列表中选【C++文件(.cpp)】项。输入文件名称“HelloWorld”。如图 4.1.2 所示，点击【添加】按钮完成添加文件操作。

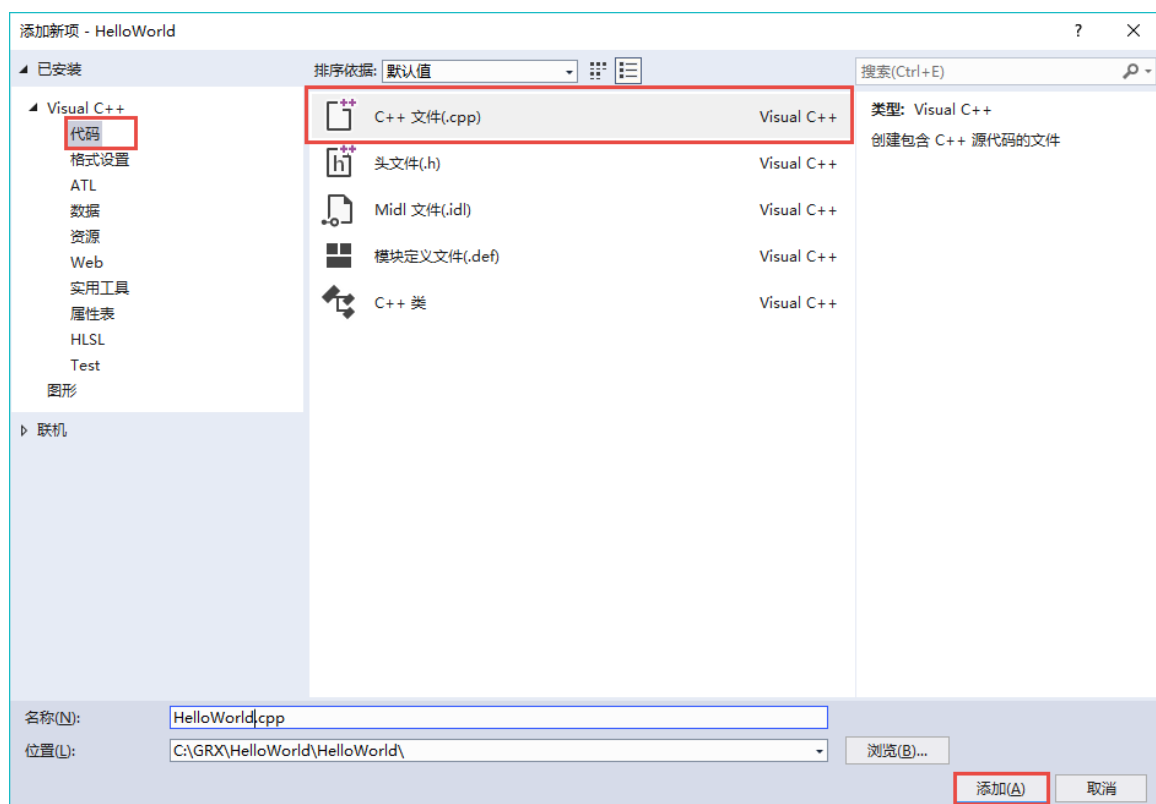


图 4.1.2

4.1.5 设置编译和链接项

- 1) 在 Visual Studio 2017 的“解决方案资源管理器”中，选中项目名称“HelloWorld”，点右键，点【属性】菜单项，会弹出“HelloWorld 属性页”对话框。如图 4.1.3 所示，本小节下面的设置都是在这个对话框中完成的。

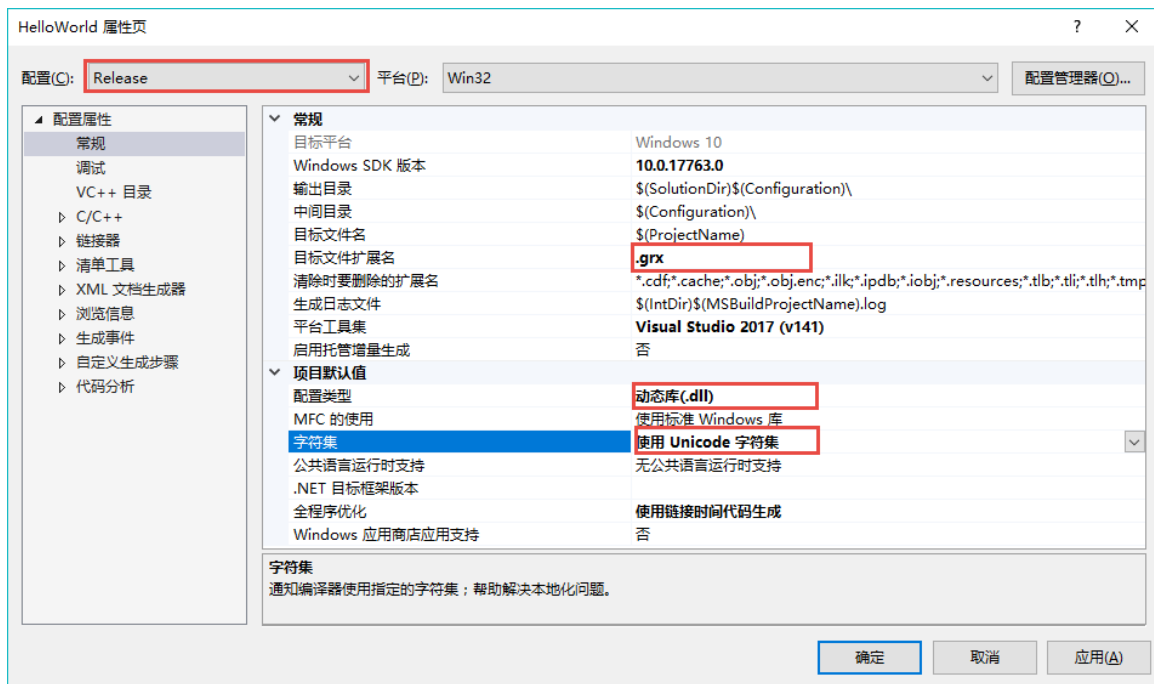


图 4.1.3

- 2) 选【常规】节点，进行如下设置。

【目标文件扩展名】设置为：.grx

【配置类型】设置为：动态库(.dll)

【字符集】设置为：使用 Unicode 字符集

- 3) 选择【C/C++】节点，进行如下设置：

【常规】/【附加包含目录】设置为：C:\grxsdk\arx\inc

如果是在 DEBUG 工程配置下则需要把_DEBUG 去掉，同时需要将【代码生成】/【运行库】修改为多线程 DLL (/MD)。

【语言】/【符合模式】设置为：否

- 4) 选择【链接器】节点，进行如下设置。

【常规】/【附加库目录】设为(64 位)：C:\grxsdk\arx\lib-x64

【输入】/【附加依赖项】设为：

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;

【输入】/【模块定义文件】设为：C:\grxsdk\arx\inc\AcRxDefault.def。

- 5) 点击【应用】，再点击【确定】按钮完成编译器的配置。
- 6) 编译，确保编译通过。否则重新配置。

4.1.6 添加代码

在 HelloWorld.cpp 里，添加如下代码：

```
#include "windows.h"

#include <tchar.h>

#include <arxHeaders.h>

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

void initApp()
{
    //register a command with the 浩辰 CAD 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 浩辰 CAD 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 编译程序

点击 Visual Studio 2017 的菜单项【生成】->【生成解决方案】，会在 C:\GRX\HelloWorld\Release 目录下生成 HelloWorld.grx 文件。

4.1.8 运行程序

启动浩辰 CAD，在命令行输入 appload，或者选择菜单项【工具】->【加载应用程序】，将会出现“加载应用程序”文件对话框，点【加载】按钮，选择我们生成的 HelloWorld.grx 文件，加载后如图 4.1.4 所示。

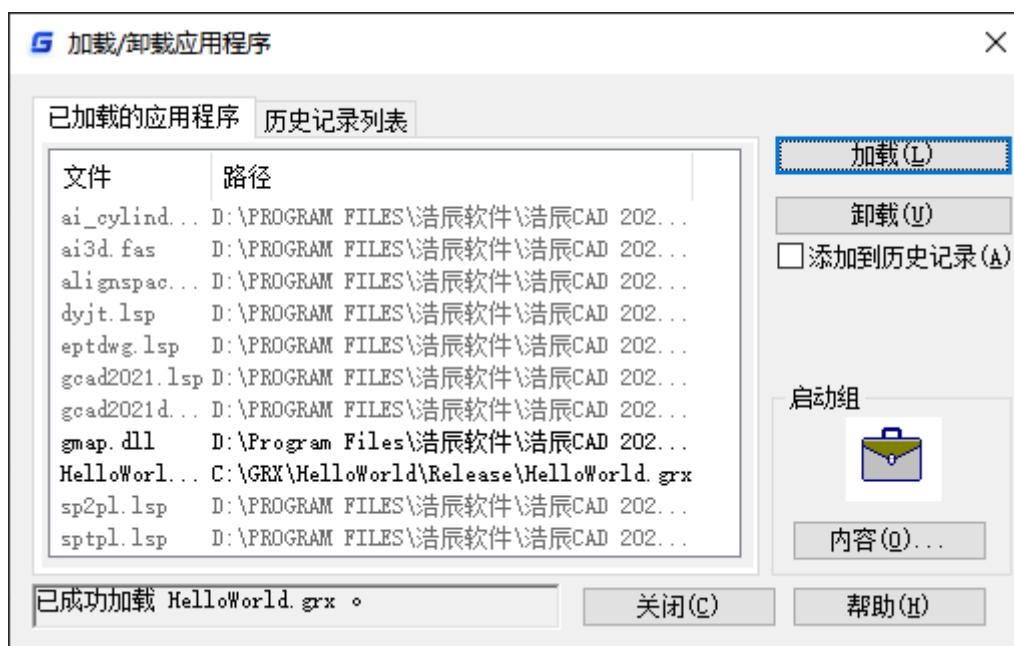


图 4.1.4

关闭“加载程序”对话框，在浩辰 CAD 命令行输入“helloworld”命令。将在命令行上打印输出：“Hello World!”。运行效果图如图 4.1.5 所示。

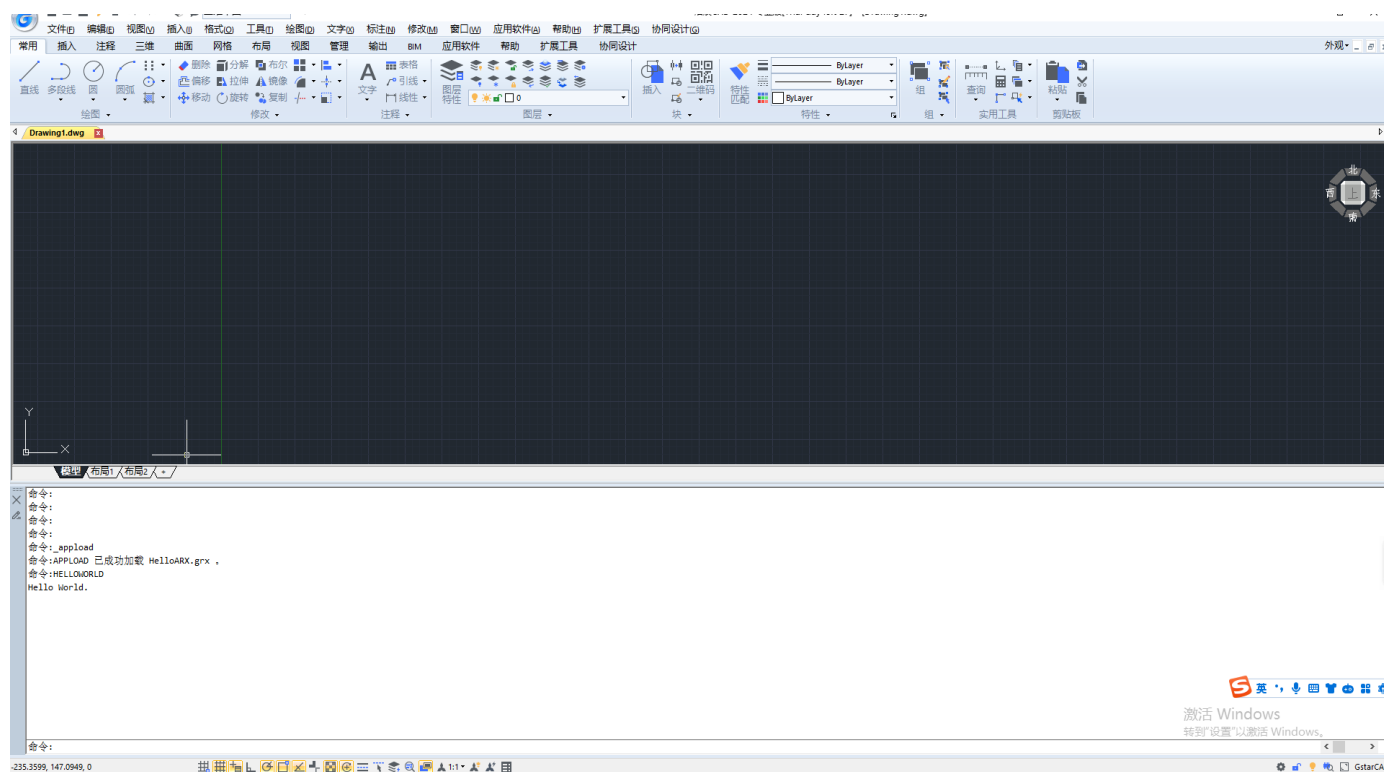


图 4.1.5

4.2 带 MFC 库的 GRX 程序

4.2.1 运行 Microsoft Visual Studio 2017

先选择【文件】菜单，再选择【新建】子菜单，最后选择【新建】菜单的子菜单【项目】。点击【项目】菜单项，会弹出“新建项目”对话框中，在左边的【已安装】模板树中，选择【Visual C++】，在下边的【MFC/ATL】列表中选择【MFC DLL】。

下面以创建“SimplePalette”项目工程为例。

4.2.2 输入项目的存放路径和项目名称

在“新建项目”对话框中“名称”标签后填入“SimplePalette”，如图 4.2.1 所示。

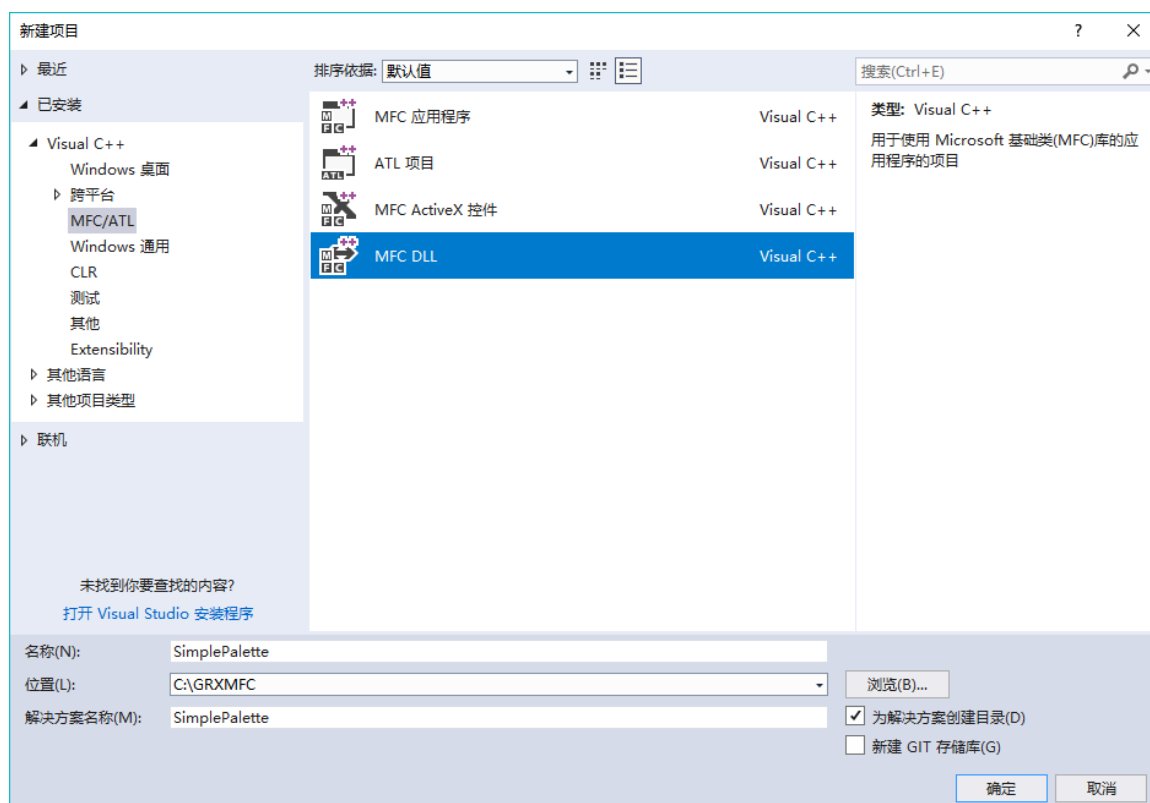


图 4.2.1

4.2.3 选择 DLL 类型

进行完第二步操作后，单击对话框中的【确定】按钮，出现【MFC DLL】对话框。在 DLL 类型下拉框中选择【MFC 扩展 DLL】。如图 4.2.2 所示。

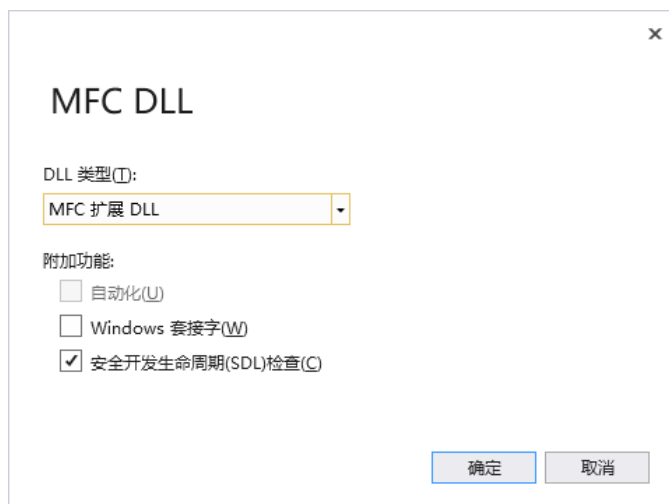


图 4.2.2

4.2.4 完成项目创建

点击上图对话框中的【确定】按钮，即完成了一个新项目的创建。

4.2.5 创建一个新资源及对应的类

在 Visual Studio 2017 的“资源视图”中，选中 SimplePalette.rc 右键添加资源，弹出“添加资源”对话框，选中【Dialog】，单击【新建】按钮，然后将对话框 ID 修改为“IDD_DIALOG_PALETTE”，Border 属性修改为 None, Style 属性修改为 Child。然后添加一个编辑框控件及一个 Button 按钮。按钮的 Caption 属性为&Check，ID 为“IDC_BUTTON_CHECK”。如图 4.2.3 所示。

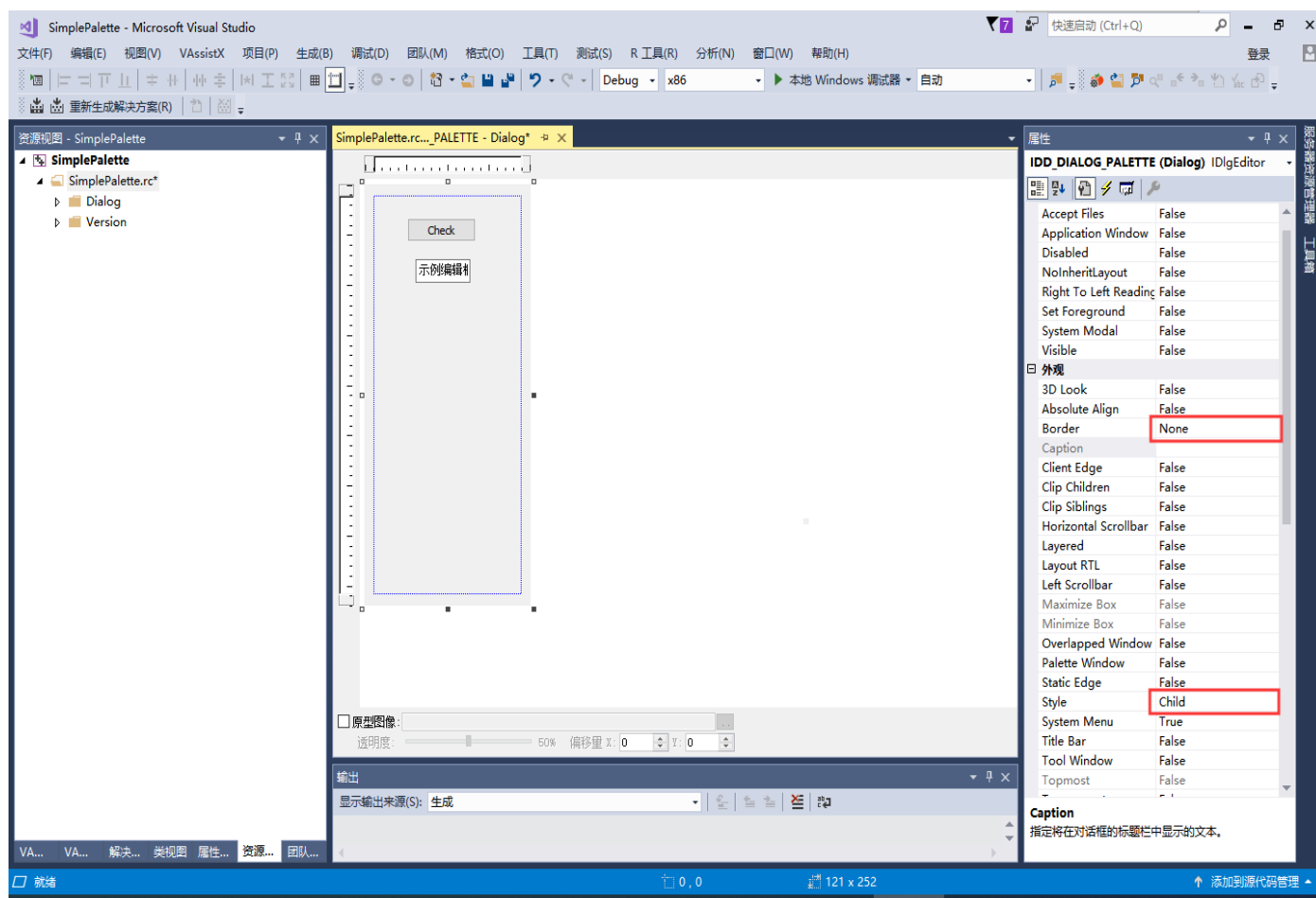


图 4.2.3

然后选中添加的对话框，单击右键，点【添加类】，会弹出【添加 MFC 类】对话框。类名设为“CDlgPalette”。点击【确定】按钮完成添加类操作。

4.2.6 创建三个新的 C++类文件

在 Visual Studio 2017 的“解决方案资源管理器”中，选中项目名称“SimplePalette”，点右键，点【添加】菜单项，选【类(C)...】，出现【添加类】对话框，如图 4.2.4 所示。

需要创建的类名分别为“CFirstPalette”、“CSecondPalette”、“CSimpleBar”，对应的基类名称分别为“CAduiPalette”、“CAduiPalette”、“CAduiPaletteSet”。

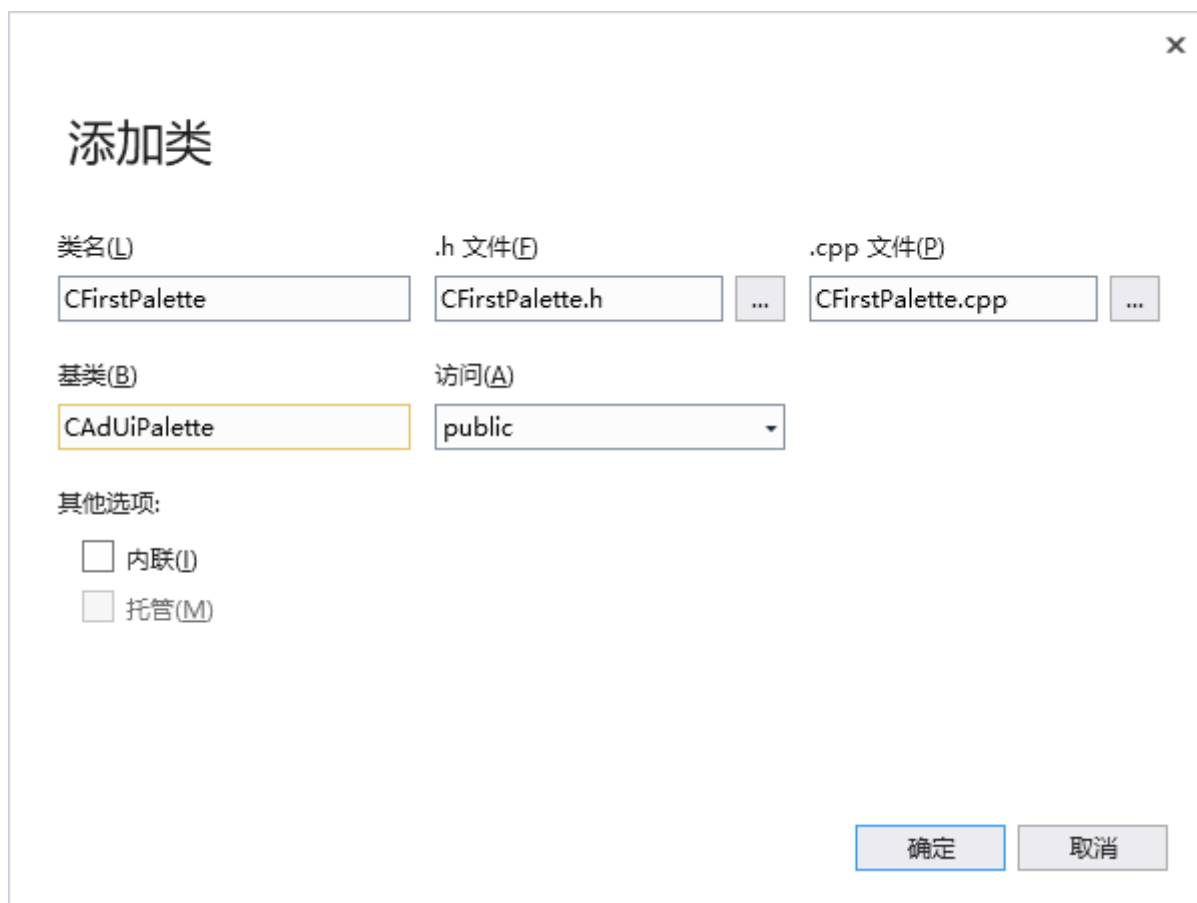


图 4.2.4

4.2.7 添加代码

在 stdafx.h 头文件（vs2017 高于 15.9.17，会自动生成 pch.h 头文件，相当于原来的 stdafx.h 头文件）里，最后处添加如下代码：

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

“CDlgPalette ”、“CFirstPalette”、“CSecondPalette”、“CSimpleBar” 类代码参见例子程序（C:\grxsdk\arx\samples\SimplePalette）中的“CDlgPalette ”、“CFirstPalette”、“CSecondPalette”、“CSimpleBar” 文件（.cpp 及.h）。

在 SimplePalette.cpp 里，添加如下代码：

```
// SimplePalette.cpp: 定义 DLL 的初始化例程。
//

#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
acrxEEntryPoint(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 设置编译和链接项

- 1) 在 Visual Studio 2017 的“解决方案资源管理器”中，选中项目名称“SimplePalette”，点右键，点【属性】菜单项，会弹出“SimplePalette 属性页”对话框。如图 4.2.5 所示，本节下面的设置都是在这个对话框中完成的。

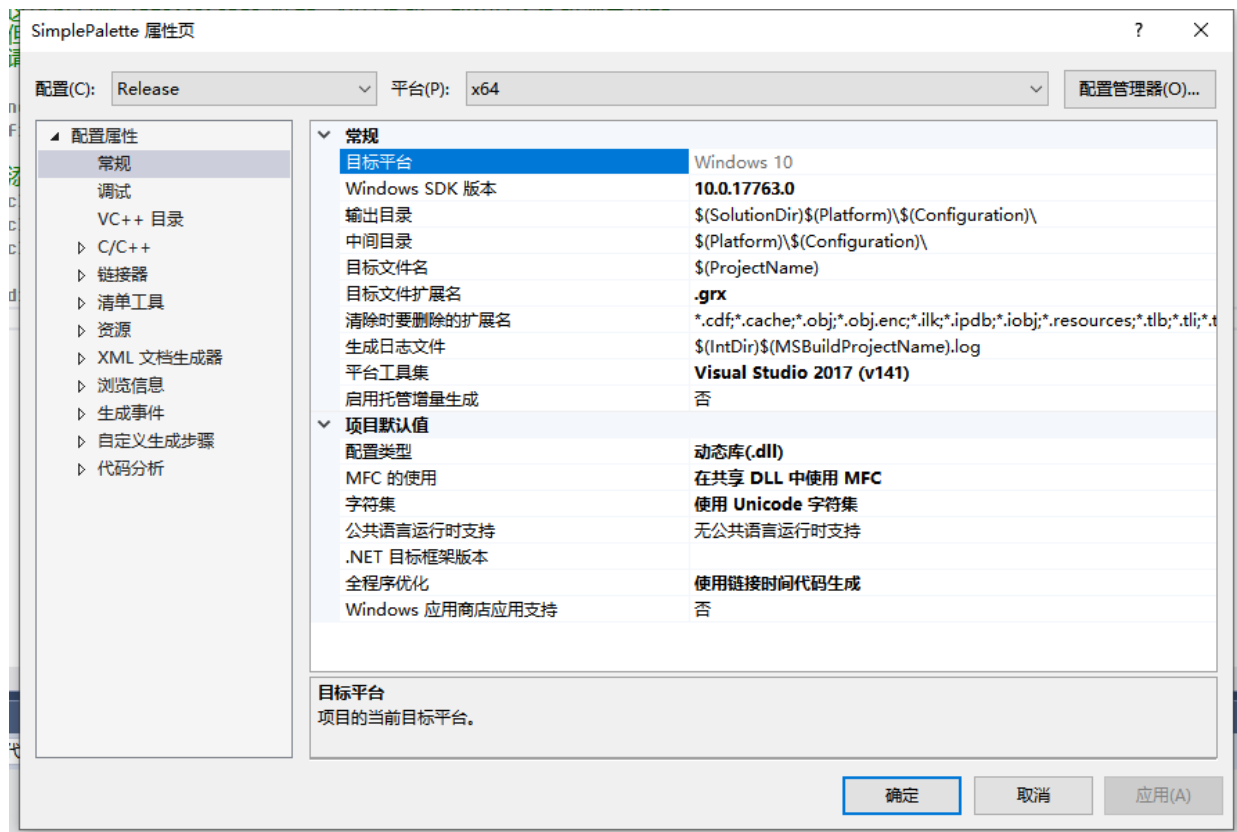


图 4.2.5

- 2) 选【常规】节点，进行如下设置。

【目标文件扩展名】设置为：.grx

【配置类型】设置为：动态库(.dll)

【字符集】设置为：使用 Unicode 字符集

- 3) 选择【C/C++】节点，进行如下设置：

【常规】/【附加包含目录】设置为：C:\grxsdk\arx\inc

如果是在 DEBUG 工程配置下，则需要把_DEBUG 去掉，同时需要将【代码生成】/【运行库】修改为多线程 DLL (/MD)。

【语言】/【符合模式】设置为：否

- 4) 选择【链接器】节点，进行如下设置。

【常规】/【附加库目录】设为(64 位)：C:\grxsdk\arx\lib-x64

【输入】/【附加依赖项】设为：

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;GcModelDoc0

bj.lib;gplot.lib;

【输入】/【模块定义文件】设为: C:\grxsdk\arx\inc\AcRxDefault.def

【高级】/【目标计算机】设为(64 位): MachineX64 (/MACHINE:X64)

5) 点击【确定】按钮完成编译器中链接器配置。

6) 编译, 确保编译通过, 否则重新配置。

4.2.9 编译程序

点击 Visual Studio 2017 的菜单项【生成】->【生成解决方案】, 会在 C:\GRXMFC\SimplePalette\Release 目录下生成 SimplePalette.grx 文件。

4.2.10 运行程序

启动浩辰 CAD, 在命令行输入 appload, 或者选择菜单项【工具】->【加载应用程序】, 将会出现“加载应用程序”对话框, 点【加载】按钮, 选择我们生成的 SimplePalette.grx 文件, 加载后如图 4.2.6 所示。

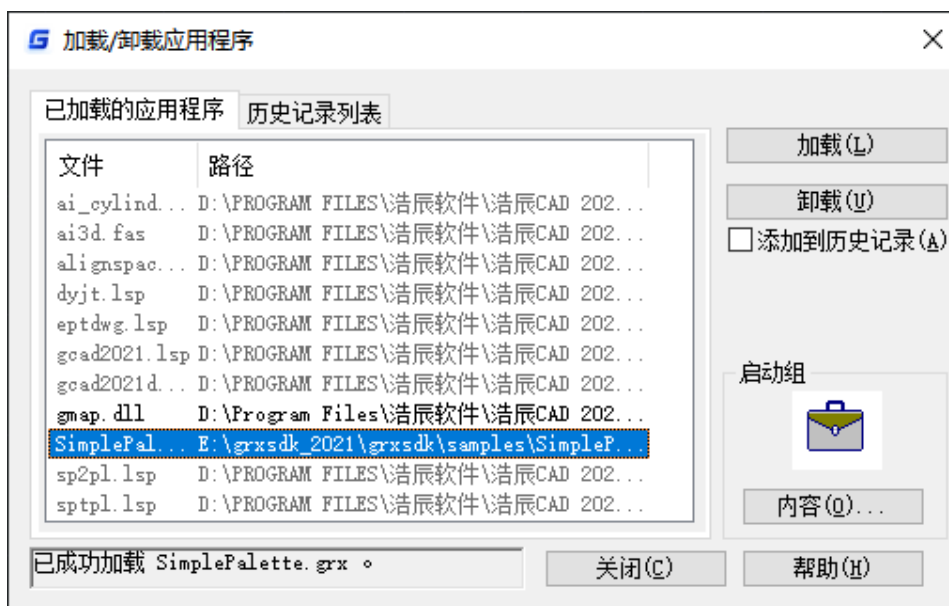


图 4.2.6

关闭“加载应用程序”对话框, 在浩辰 CAD 命令行输入“simplepalette”命令。会在命令行上打印输出: “Float or Docking? ”。直接按回车, 运行效果图如图 4.2.7 所

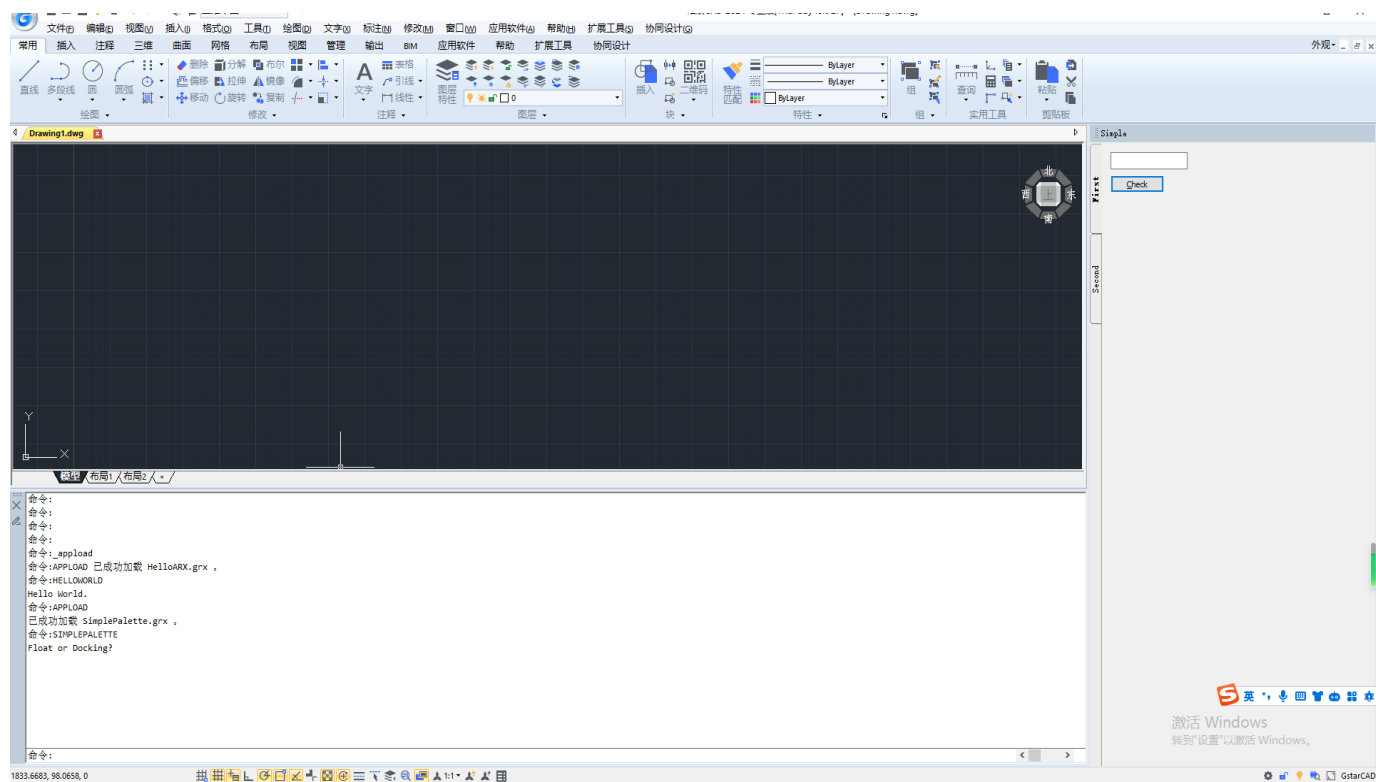


图 4.2.7

5 如何使用项目属性表文件

属性表是用来快速设置新建工程项目的属性配置。属性表还使项目的环境设置的误操作可能性变小。

可以将项目属性独立应用于生成配置（调试或发布）和目标平台（Win32、x64 或 ARM）的任意组合。

GRX 开发包中提供了配置附加依赖项、附加包含目录及附加库目录的项目属性表。

- 1) 以“HelloWorld”项目为例，使用项目属性表文件配置环境，打开“HelloWorld”项目。选择【视图】菜单，再选择【属性管理器】。如图 5.1 所示。

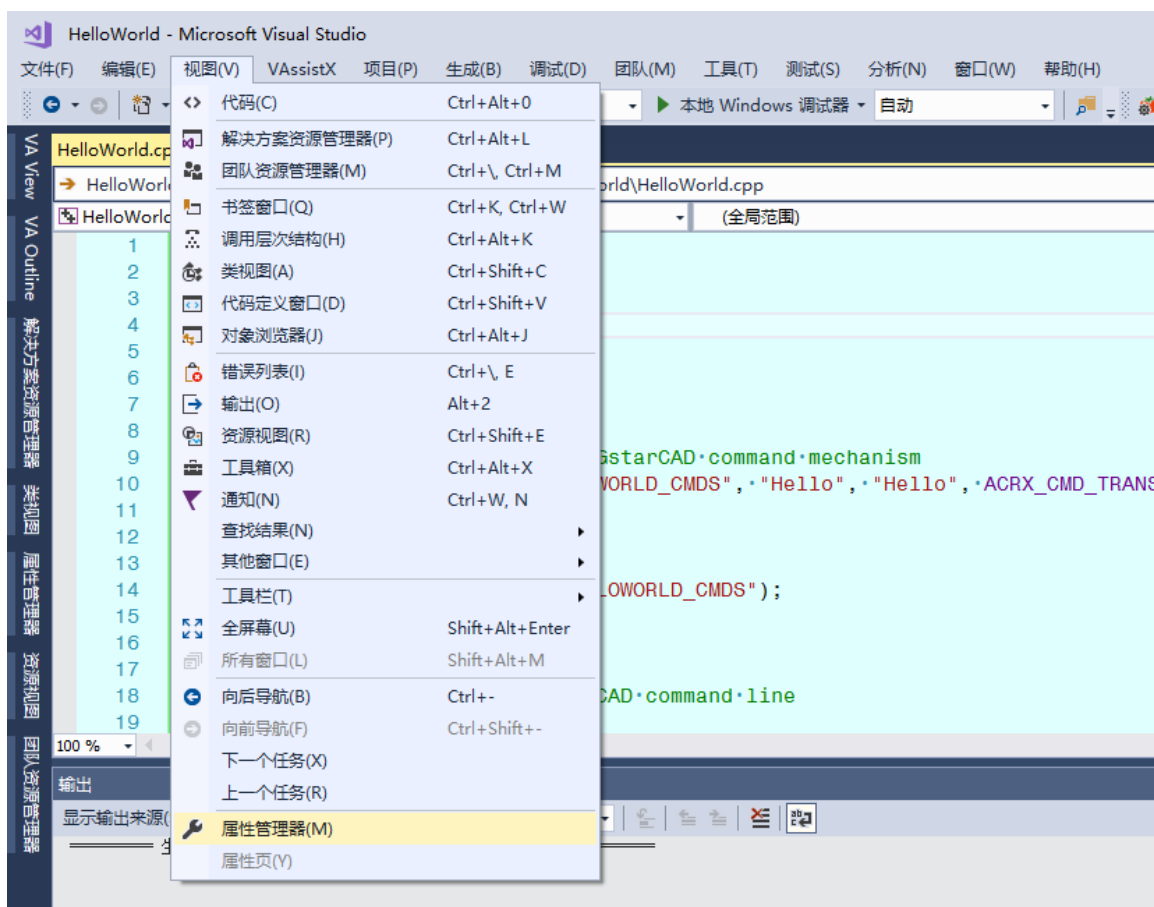


图 5.1

- 2) 点击【属性管理器】菜单项，会弹出“属性管理器”对话框，配置 Release | WIN32，右键->【添加现有属性表】，如下图 5.2 所示。

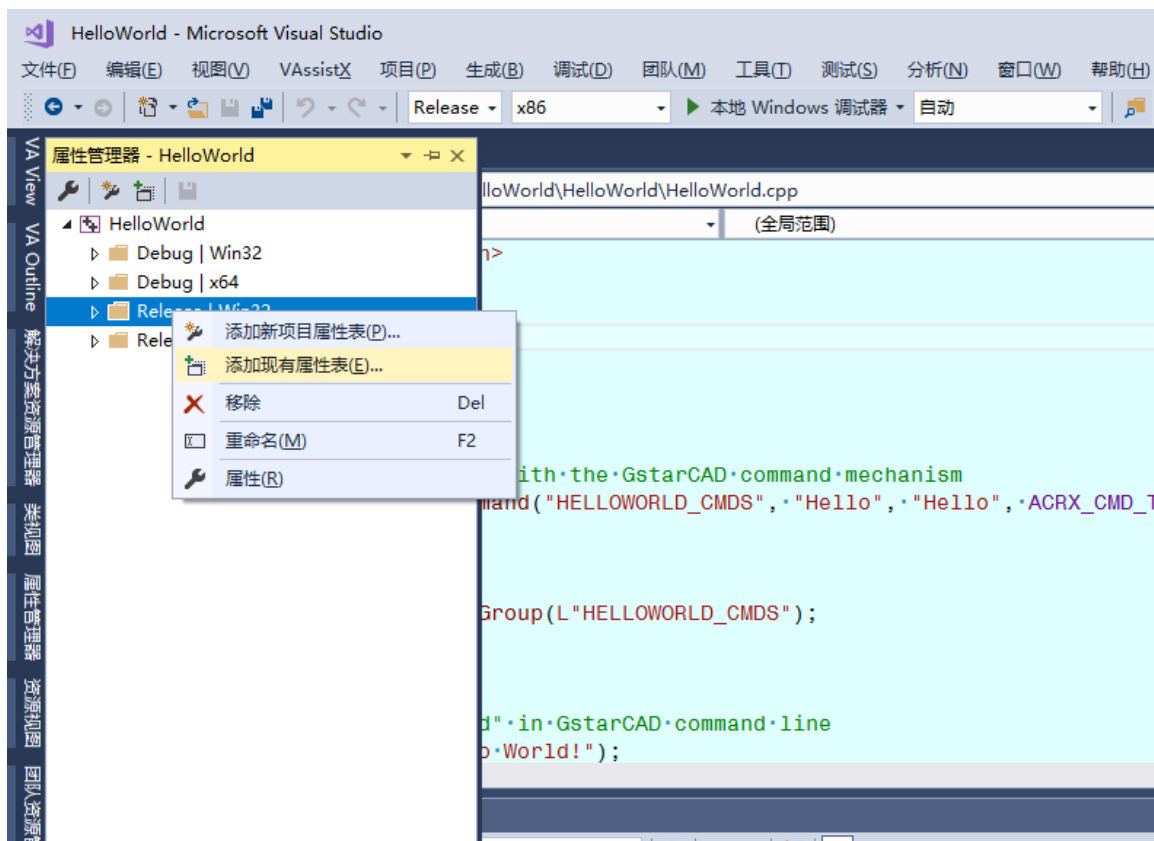


图 5.2

- 3) 然后添加 C:\grxsdk\arx\inc\arx.props 属性表文件（此属性表文件包含 dbx.props 属性表文件，作用是增加以下附加依赖项：

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;），按照同样的方法添加

C:\grxsdk\arx\inc\rxsdk_releasecfg.props 属性表文件（此属性表文件包含 rxsdk_common.props 属性表文件，作用是增加附加包含目录及附加库目录的路径）。如果配置的是 DEBUG 工程环境，需要添加 C:\grxsdk\arx\inc\rxsdk_debugcfg.props。

- 4) 在 Visual Studio 2017 的“解决方案资源管理器”中，选中项目名称“HelloWorld”，点右键，点【属性】菜单项，会弹出“HelloWorld”属性页对话框。选择【C/C++】节点，进行如下设置：
- 【常规】/【附加包含目录】设置为：C:\grxsdk\arx\inc。【链接器】节点【模块定义文件】设置为：C:\grxsdk\arx\inc\AcRxDefault.def。

- 5) 按照 4.1.7、4.1.8 小节步骤操作编译运行程序。

6 GRX 类库说明

下列库是使用 GRX 开发浩辰 CAD 过程中经常会使用的，这些库与 ARX 下对应的库所完成的功能相同，主要包括如下：

- GcRx-与 AcRx 库功能相同，用于绑定一个应用程序以及运行类的注册和识别。
- GcEd-与 AcEd 库功能相同，用于注册自定义命令和事件通告。
- GcDb-与 AcDb 库功能相同，浩辰 CAD 数据库类。
- GcGi-与 AcGi 库功能相同，用于浩辰 CAD 的图形类。
- GcGe-与 AcGe 库功能相同，用于通用的线性代数计算和几何对象的应用类。

6.1 GcRx

GcRx 类库用于 DLL 初始化以及运行类注册和识别时系统级别，提供下面的功能：

- 对象运行时类识别和继承分析。
- 在运行时向一个已经存在的类添加新的协议。
- 对象相等和组成判断。
- 对象拷贝。

6.2 GcEd

GcEd 类库用于定义注册新的浩辰 CAD 命令，这些命令和浩辰 CAD 内部命令完全一样。

6.3 GcDb

GcDb 类库用于操作浩辰 CAD 数据库。数据库中存储了所有图形对象的信息，这些图形对象叫做实体。实体和那些非图形实体(例如层、线型和文本样式)共同组成了浩辰 CAD 图形。

6.4 GcGi

GcGi 库提供了用来绘制造辰 CAD 实体的图形接口。

6.5 GcGe

GcGe 库提供了一些几何工具类(例如矢量和矩阵)来执行二维和二维几何操作，它也提供了基本的几何对象，例如点、曲线和曲面。

7 版权声明

版权所有：苏州浩辰软件股份有限公司

使用许可：允许复制、引用本文档的任何部分。未经许可，不得更改本文档的任何部分。在复制、引用时，请务必保留本声明，否则将追究法律责任。