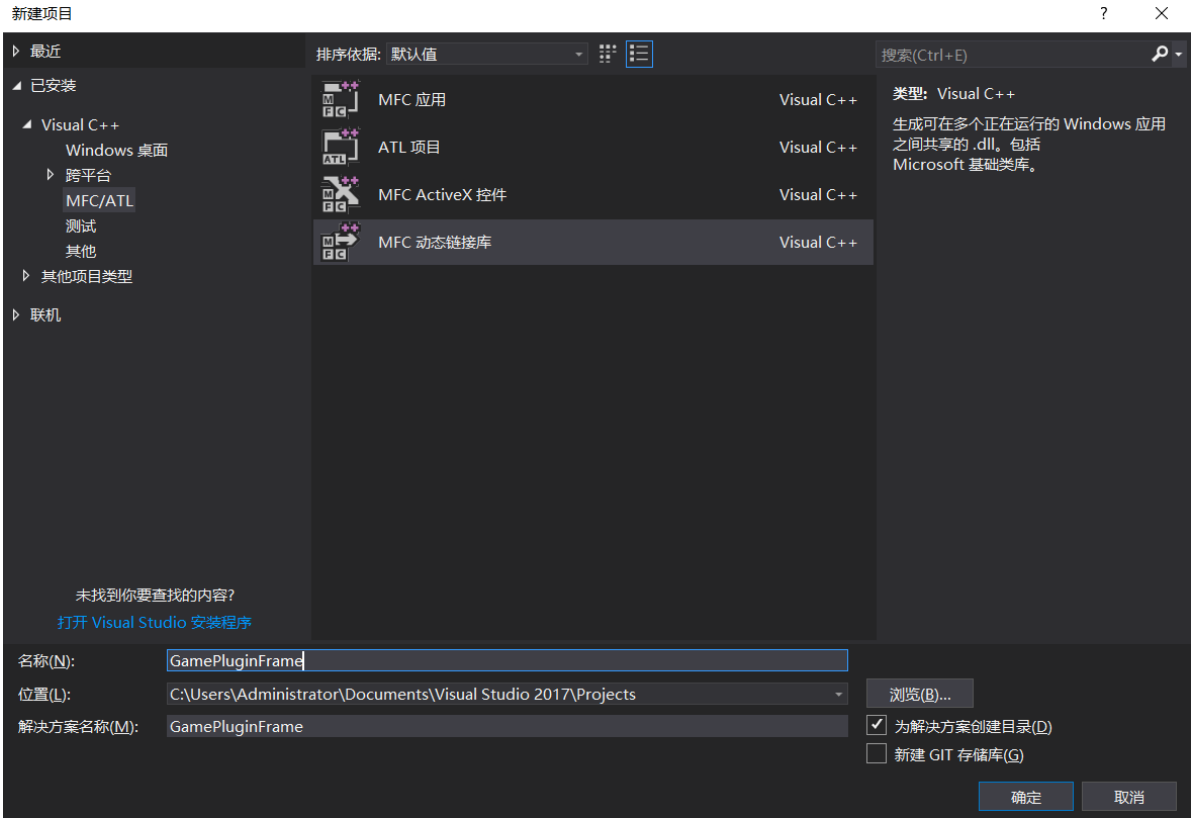
新建MFC项目
辅助框架设计
小结

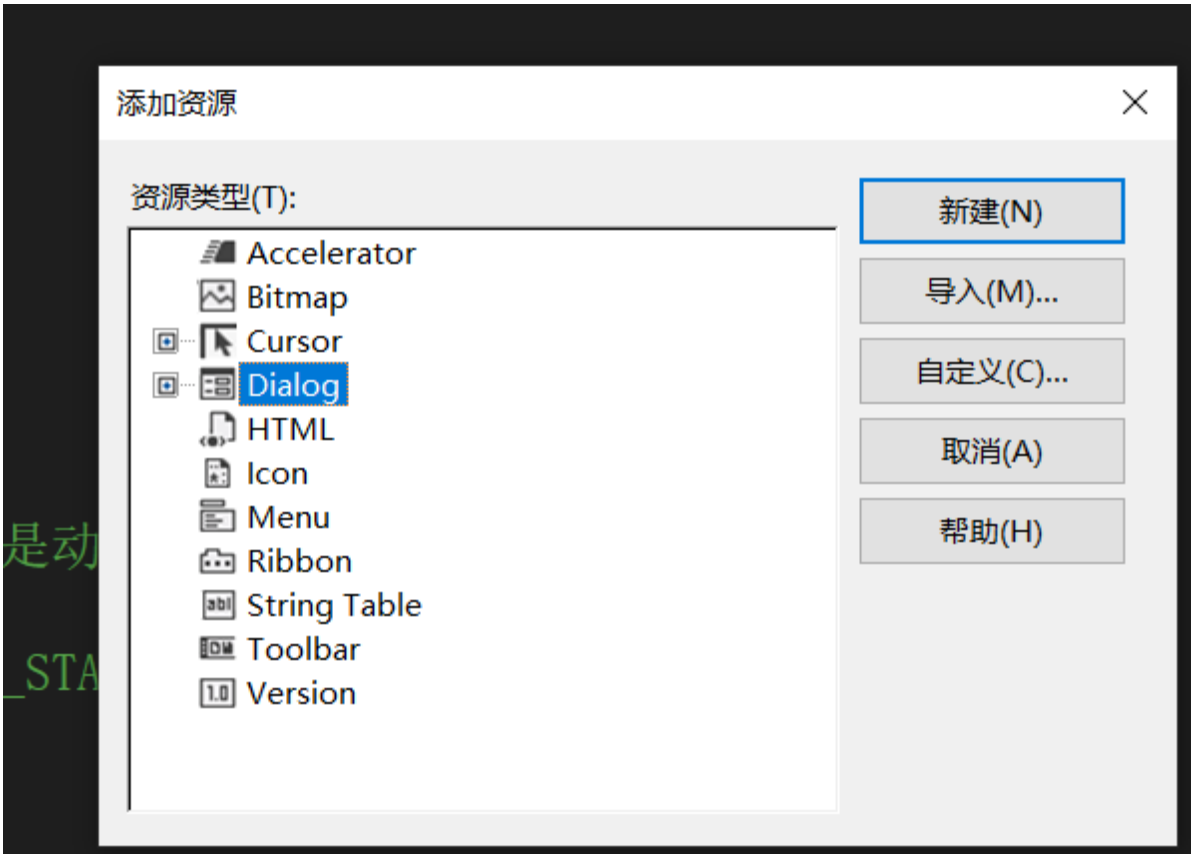
如果你大概了解过游戏引擎的概念，就会知道 游戏引擎是指一些已编写好的核心组件。为游戏设计者提供各种编写游戏所需的各种工具，其目的在于让游戏设计者能容易和快速地做出游戏程式而不用由零开始。

游戏引擎有一套通用的框架可以复用，那么游戏辅助当然也有。

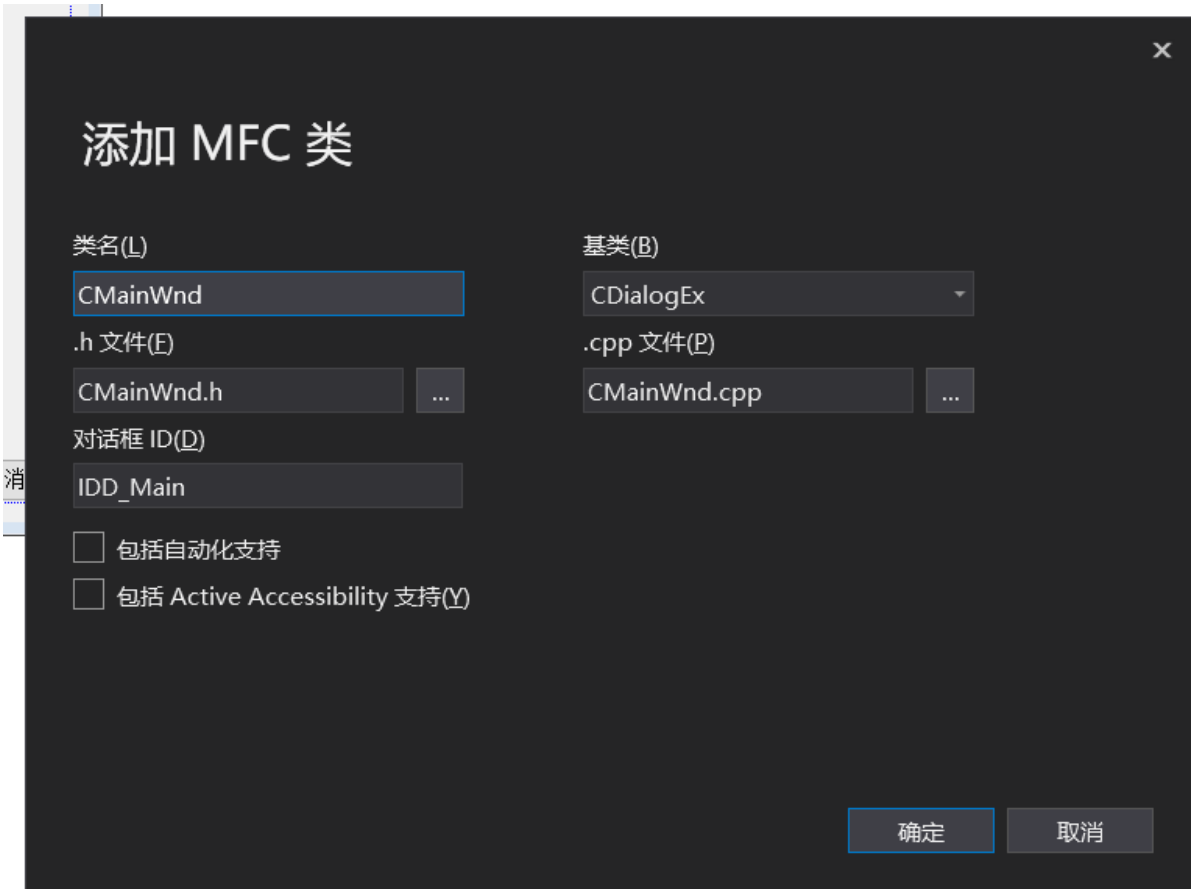
新建MFC项目



首先新建一个MFC的项目，命名为`GamePluginFrame`。



新建一个对话框资源



然后添加一个窗口类。

记得将项目属性设置为x64，因为传奇永恒是64位的游戏。

然后在MFC项目的InitInstance函数中创建一个线程

```
BOOL CGamePluginFrameApp::InitInstance()
{
    CWinApp::InitInstance();


    //创建线程 开启窗口
    HANDLE hThread = ::CreateThread(NULL, 0,
(LPTHREAD_START_ROUTINE)InitMainWnd, NULL, 0, NULL);
    CloseHandle(hThread);


    return TRUE;
}
```

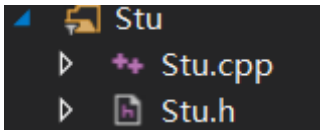
线程函数用于启动窗口，防止主线程卡死

```
void InitMainWnd()
{
    MainWnd main;
    main.DoModal();

    FreeLibraryAndExitThread(theApp.m_hInstance, -1);
}
```

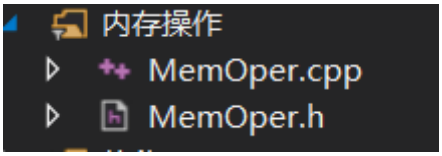
辅助框架设计

接着来设计整个游戏辅助的框架。按照模块管理的思想，将整过框架分为下面几个部分。



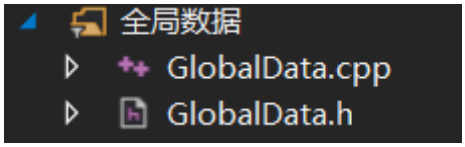
首先新建一个类，命名为Stu，里面保存游戏所需要用到的所有的结构体

然后是CPublic，里面保存整个项目需要用的通用的函数，例如字符串编码转换，输出调试信息等等。

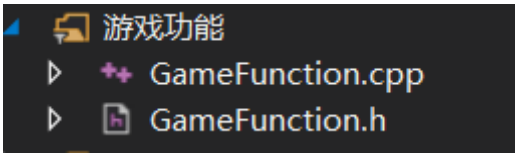


然后准备一个内存操作类，里面封装对所有数据类型的读取操作，避免写`*(DWROD*)EAX=Context`，这样的取内容代码，提升整体代码的可读性。

offset.h用于保存所有的游戏偏移，方便后续更新。



GlobalData.cpp用于保存游戏所需要用到的全局数据，例如游戏模块的基地址，我们只需要用一个全局变量存储，然后在dll注入的时候读取一次就可以了。



GameFunction.cpp用于保存找到的所有的游戏功能，例如走路call，选怪call，并且所有的函数都用`Fn_`开头，可以提升后续的代码编写效率。

GameData.cpp用于保存所有游戏的数据，例如人物属性，周围遍历属性等等，全部存在这一个类里面。

暂时就只需要用到这些类。

小结

将所有的代码分成不同的模块进行不同的管理，会大大提升后续的开发效率。

框架的东西并不属于开发知识，而是属于开发经验，可以让你少走弯路，后续进行快速开发。缩短时间成本。对于这一块的内容，我也是花了大量的时间进行整合优化。

下一Part我们来讲具体的框架代码实现。