火星时代教育

虚幻引擎中使用库

—虚幻引擎高级程序开发专业—

PART 1

虚幻中的库

库

虚幻引擎基于模块设计功能，拆分模块在C++中即使用库的方案来解决，在虚幻引擎中使用库是非常方便的！我们可以通过添加虚幻引擎模块的方式来完成库的引入，但与模块又稍有不同，设置上会有一些出别。首先需要在项目的Sources文件夹下添加新的文件夹，命名为ThirdParty，注意目录名称不能修改。然后在目录下，添加文件夹，名称为模块名称，此模块用于引入库文件。结构如下

Sources

|-ThirdParty

|-模块名称

|-Include（头文件）

|-Lib（库文件）

|-模块名称.build.cs

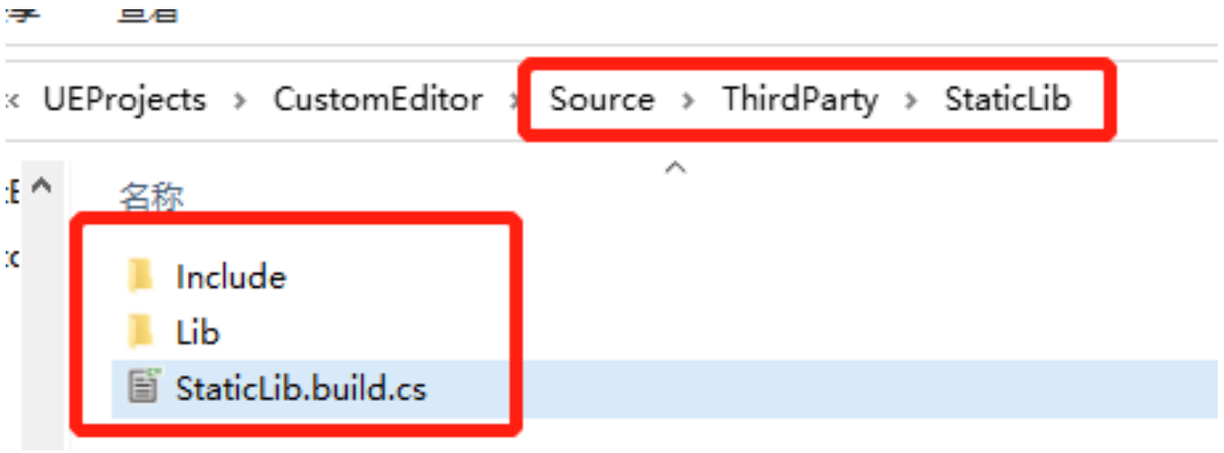
注意：虚幻引擎使用的是64位编码，所以我们编译库需要使用64位编码，并且注意不要胡乱引入其他头文件，虚幻引擎有可能不支持！

PART 2

静态库

创建模块资产

创建路径关系如下



配置CS文件

Cs文件配置如下

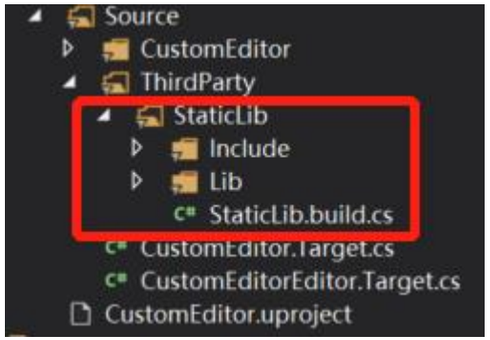
```
using System.IO;
using UnrealBuildTool;

public class StaticLib : ModuleRules
{
    public StaticLib(ReadOnlyTargetRules Target) : base(Target)
    {
        //设置模块类型为外部库
        Type = ModuleType.External;

        if (Target.Platform == UnrealTargetPlatform.Win64)//win平台配置
        {
            // 设置库路径
            PublicLibraryPaths.Add(Path.Combine(ModuleDirectory, "Lib"));
            // 设置依赖库名称
            PublicAdditionalLibraries.Add("StaticLib.lib");
            // 设置头文件路径
            PublicIncludePaths.Add(Path.Combine(ModuleDirectory, "Include"));
        }
    }
}
```

编译

完成添加后在VS中重新生成整个项目，编译完成，刷新solution（右键uproject文件，生成vs文件），重新整理路径关系。刷新后打开vs即可看到插件路径内容。



使用

在项目的cs文件中添加库模块，重新编译工程，即可使用库中的内容。

```
public class CustomEditor : ModuleRules
{
    public CustomEditor(ReadOnlyTargetRules Target) : base(Target)
    {
        PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;

        PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject", "Engine", "InputCore", "Slate",
            "SlateCore", "StaticLib" });

        PrivateDependencyModuleNames.AddRange(new string[] { "Slate", "SlateCore" });

        // Uncomment if you are using Slate UI
        // PrivateDependencyModuleNames.AddRange(new string[] { "Slate", "SlateCore" });

        // Uncomment if you are using online features
        // PrivateDependencyModuleNames.Add("OnlineSubsystem");

        // To include OnlineSubsystemSteam, add it to the plugins section in your uproject file with the Enabled attribu
    }
}
```

正常引入头文件使用即可

CustomEditorGameModeBase.h | CustomEditorGameModeBase.cpp | StaticLib.build.cs | Thric

CustomEditorGameMode | E:\UEProjects\CustomEditor\Source\CustomEditor\CustomEditorGameMode

```
1     // Fill out your copyright notice in the Description page of Pr
2
3   #include "CustomEditorGameModeBase.h"
4   #include "ThirdParty/StaticLib/Include/StaticLib.h"
5
6   void ACustomEditorGameModeBase::TestFun()
7   {
8       StaticLib st;
9       int num = st.Max(50, 60);
10      UE_LOG(LogTemp, Log, TEXT("== %d"), num);
11  }
12
```

PART 2

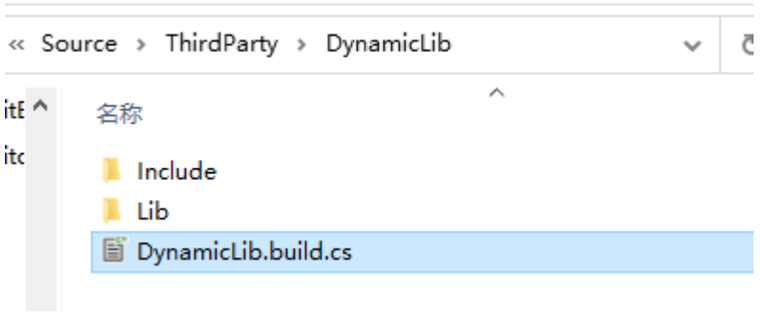
动态库

说明

与静态库使用几乎一致，需要做的就是先导出库文件，注意库文件导出x64平台，然后在引擎下创建模块文件。注意动态库需要将lib库与dll库一同放置在项目模块中。

创建模块资产

创建路径关系如下



配置CS文件

Cs文件配置如下，然后编译项目，刷新项目，然后在工程中**引入模块**即可

```
using System.IO;
using UnrealBuildTool;

public class DynamicLib : ModuleRules
{
    public DynamicLib(ReadOnlyTargetRules Target) : base(Target)
    {
        //设置模块类型为外部库
        Type = ModuleType.External;

        if (Target.Platform == UnrealTargetPlatform.Win64)//win平台配置
        {
            // 设置库路径
            PublicLibraryPaths.Add(Path.Combine(ModuleDirectory, "Lib"));
            // 设置依赖库名称
            PublicAdditionalLibraries.Add("DynamicLib.lib");
            // 设置头文件路径
            PublicIncludePaths.Add(Path.Combine(ModuleDirectory, "Include"));
            // 设置延迟加载动态库
            PublicDelayLoadDLLs.Add("DynamicLib.dll");
        }
    }
}
```

使用

与静态库不同，动态库需要进行显示调用，方可应用，通过以下代码可以完成显示调用逻辑。

由于配置了静态库，所以无需再通过名称查找调用函数句柄，相比较更加简单。

```cpp
#include "CustomEditorGameModeBase.h"
#include "ThirdParty/StaticLib/Include/StaticLib.h"
#include "ThirdParty/DynamicLib/Include/Human.h"
#include <Paths.h>

void ACustomEditorGameModeBase::TestFun()
{
    //加载动态库
    DynamicHandle = FPlatformProcess::GetDllHandle(*FPaths::Combine(FPaths::ProjectDir(), TEXT("Source\\ThirdParty\\DynamicLib\\Lib\\DynamicLib.dll")));
    if (DynamicHandle)
    {
        //库内类
        Human h;
        int num = h.Max(50, 100);
        UE_LOG(LogTemp, Log, TEXT("==%d"), num);
        //卸载库
        FPlatformProcess::FreeDllHandle(DynamicHandle);
    }
}
```

THANK YOU

虚幻4高级程序开发专业