【Unity基础教程】重点知识汇总（十四）

Unity C#编写DLL文件

DLL是什么（概念）

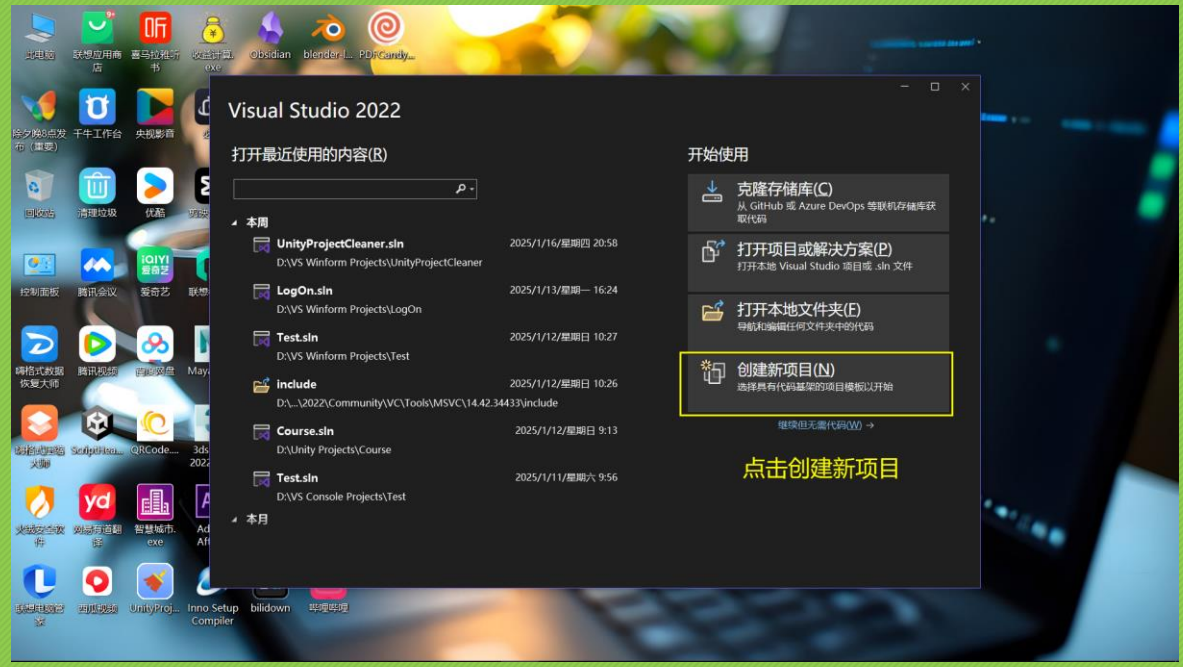
概念：DLL（Dynamic Link Library，**动态链接库**）是一种**包含代码和数据的文件**，程序在运行时可以**动态加载并调用**它的内容，而不是在编译时将其嵌入到可执行文件中。它是一种**模块化编程**的实现方式，在Windows操作系统中被广泛使用。在Windows系统下，其扩展名为**.dll**，但在不同的操作系统中也有类似的概念（如Linux下的.so文件或macOS下的.dylib 文件）。

用途及作用：

- 代码复用：将通用功能封装（**隐藏细节**）到一个DLL中，不同的程序可以**共享**该代码，**避免重复编写**。
- 模块化开发：将应用程序**拆分成多个模块**，每个模块都可以用DLL实现，方便独立开发和测试。
- 按需加载：应用程序只在**需要时加载**DLL，从而减少初始内存使用。
- 版本升级：可以**独立更新**DLL，而无需重新编译整个应用程序，降低维护成本。
- 多语言互操作性：通过DLL，可以实现**不同编程语言之间的交互**，比如用C#调用C++编写的 DLL。

编写DLL（准备工作）

工作内容：使用Visual Studio，运用C#语言编写DLL文件，并导入Unity3D引擎中进行测试和使用。首先，我们需要通过Visual Studio Installer安装好**.NET 桌面开发**工作负荷。（操作流程可参考【Unity基础教程】重点知识汇总（六））

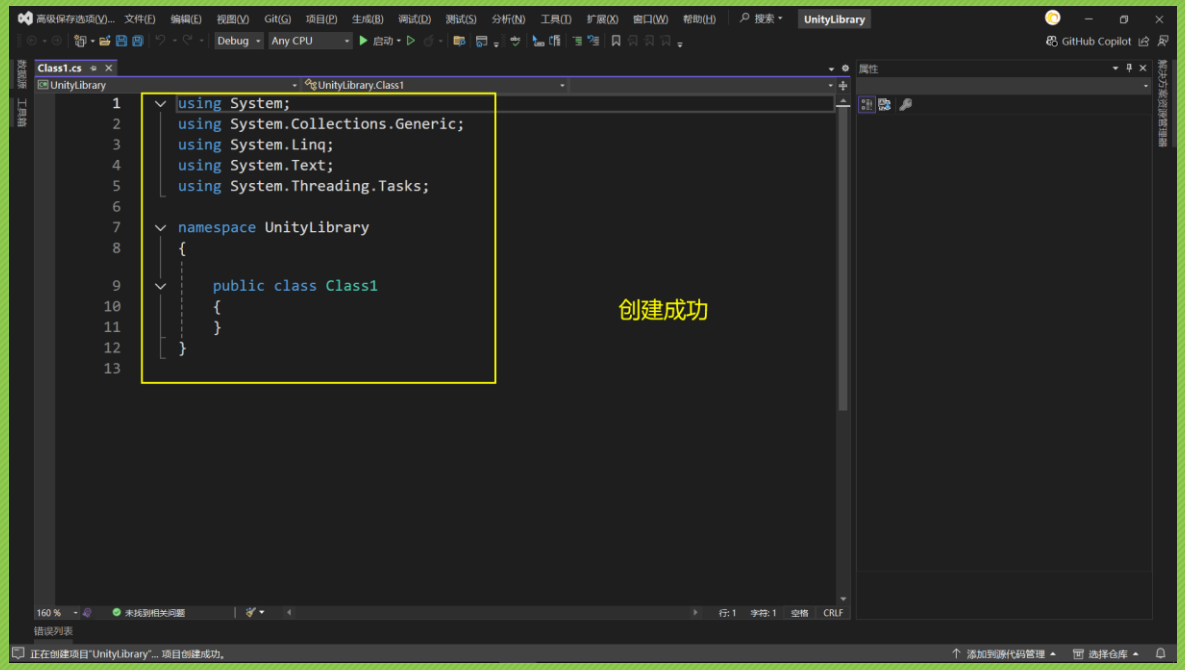


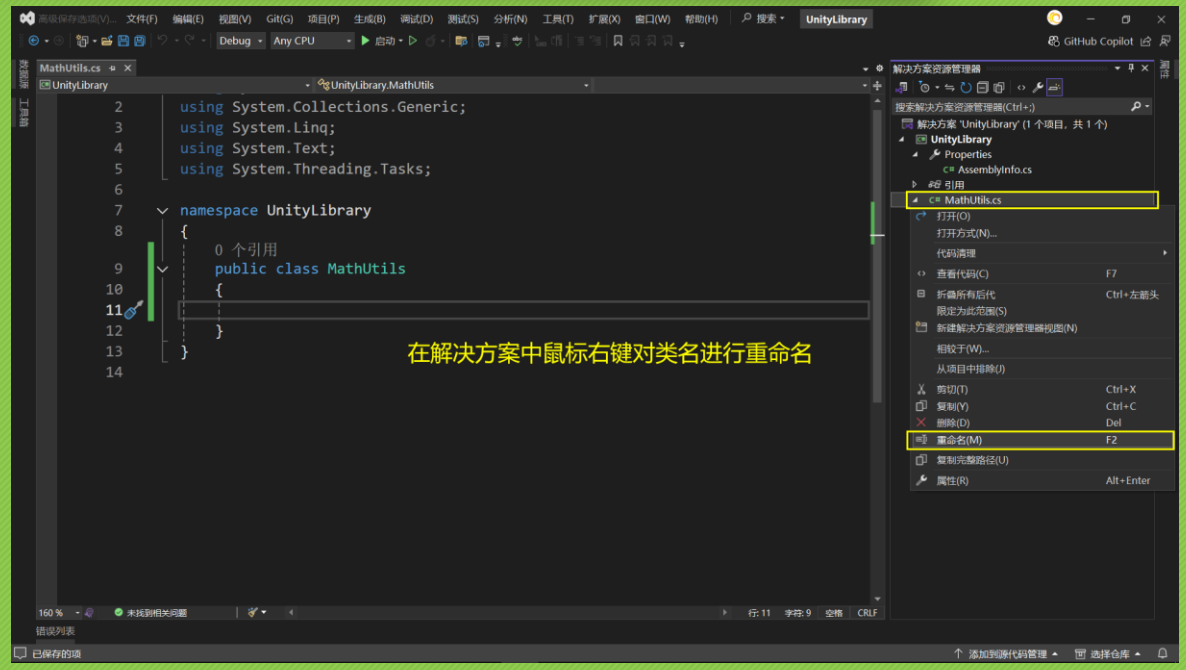
编写DLL（创建项目）


创建新项目
最近使用的项目模板(R)
Windows 窗体应用(.NET Framework)
控制台应用
WPF 应用(.NET Framework)
类库
全部清除(C)
C#
Windows
所有项目类型(T)
类库
用于创建面向 .NET 或 .NET Standard 的类库的项目
C# Android Linux macOS Windows 库
WPF 类库
用于创建目标为 .NET WPF 应用程序的类库的项目
C# Windows 桌面 库
这两个都可以
类库(.NET Framework)
用于创建 C# 类库(.dll)的项目
C# Windows 库
以它作为演示
Windows 窗体类库
用于创建面向 .NET Windows 窗体(WinForms)的类库的项目模板。
C# Windows 桌面 库
WPF 用户控件库
用于为 .NET WPF 应用程序创建用户控件库的项目
C# Windows 桌面 库
WPF 自定义控件库
上一步(B)
下一步(N)

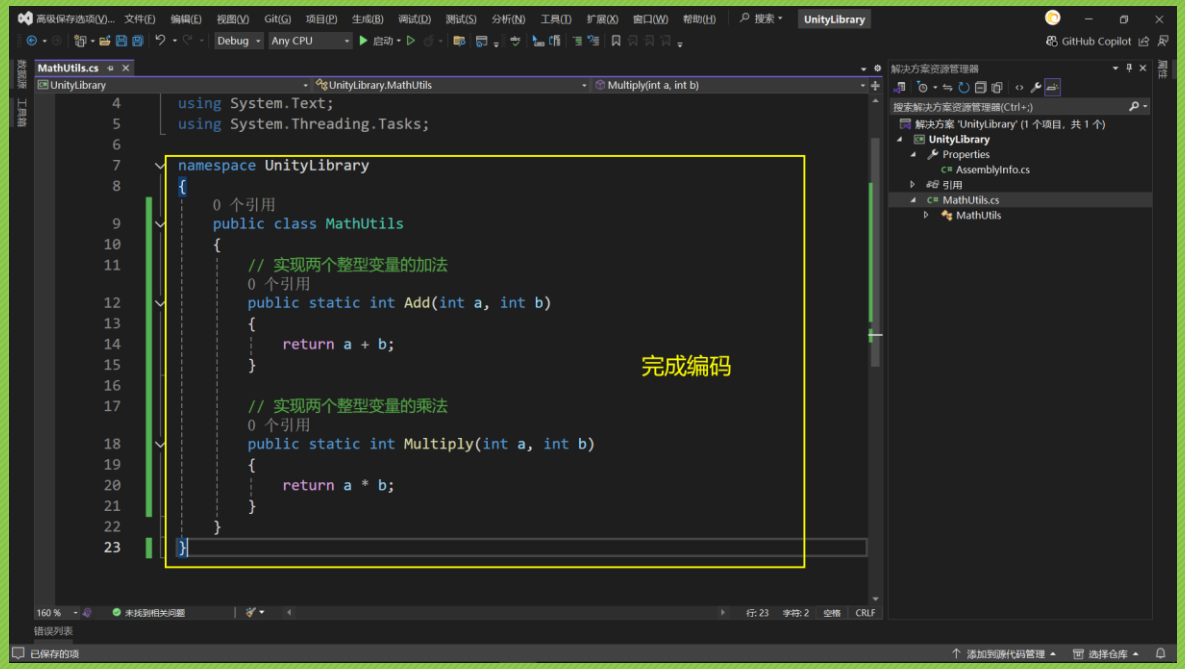

配置新项目
类库(.NET Framework)
C# Windows 库
项目名称(J)
UnityLibrary
位置(L)
D:\VS DLL Projects\
解决方案名称(M)
UnityLibrary
将解决方案和项目放在同一目录中(D)
框架(F)
.NET Framework 4.8
项目 将在"D:\VS DLL Projects\UnityLibrary\UnityLibrary\"中创建
重命名项目名称
项目位置
解决方案名称通常与项目名称一样
（自动关联）
框架默认即可
（后续还可以调整）
上一步(B)
创建(C)

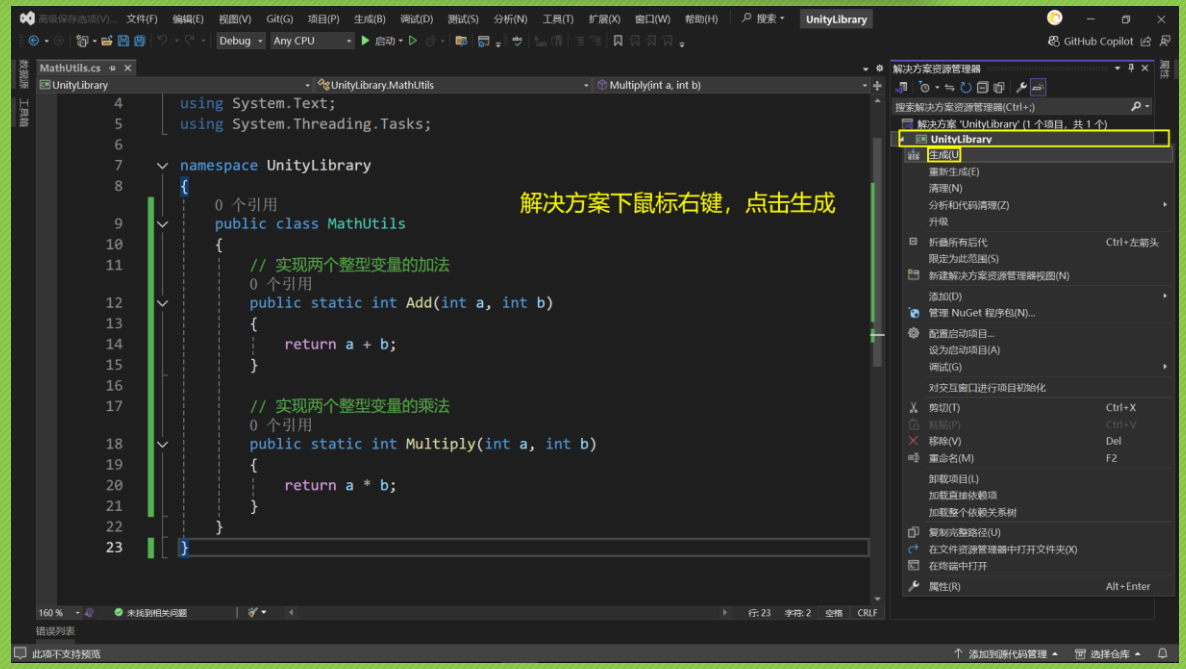
编写DLL（项目基本设置）



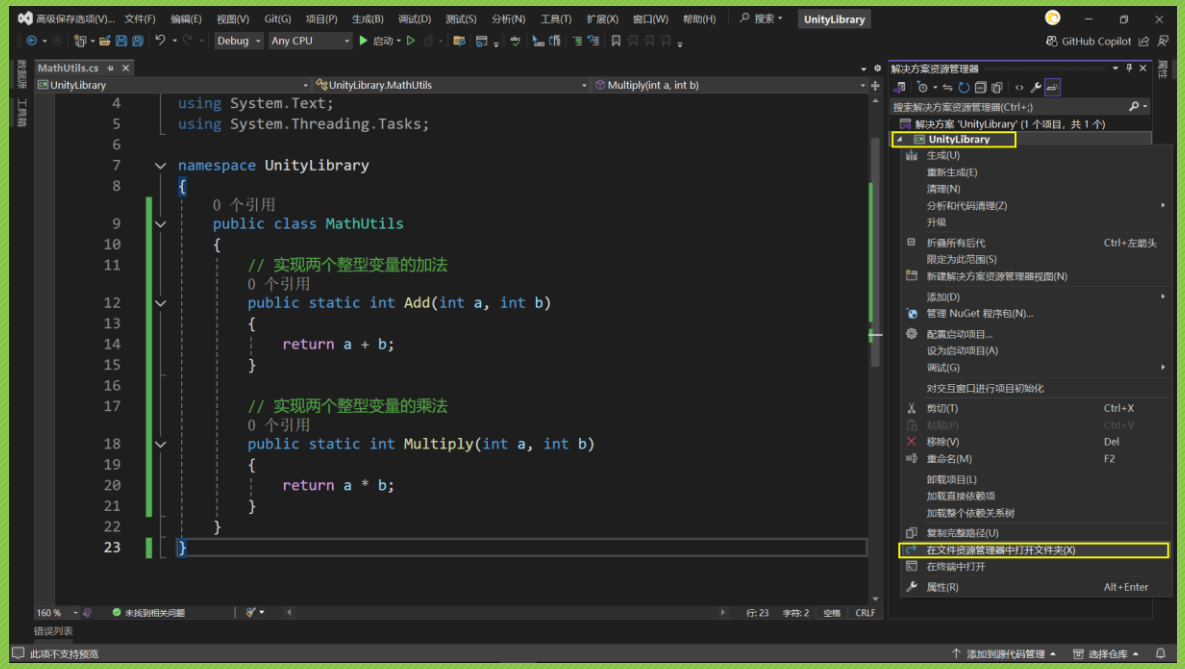


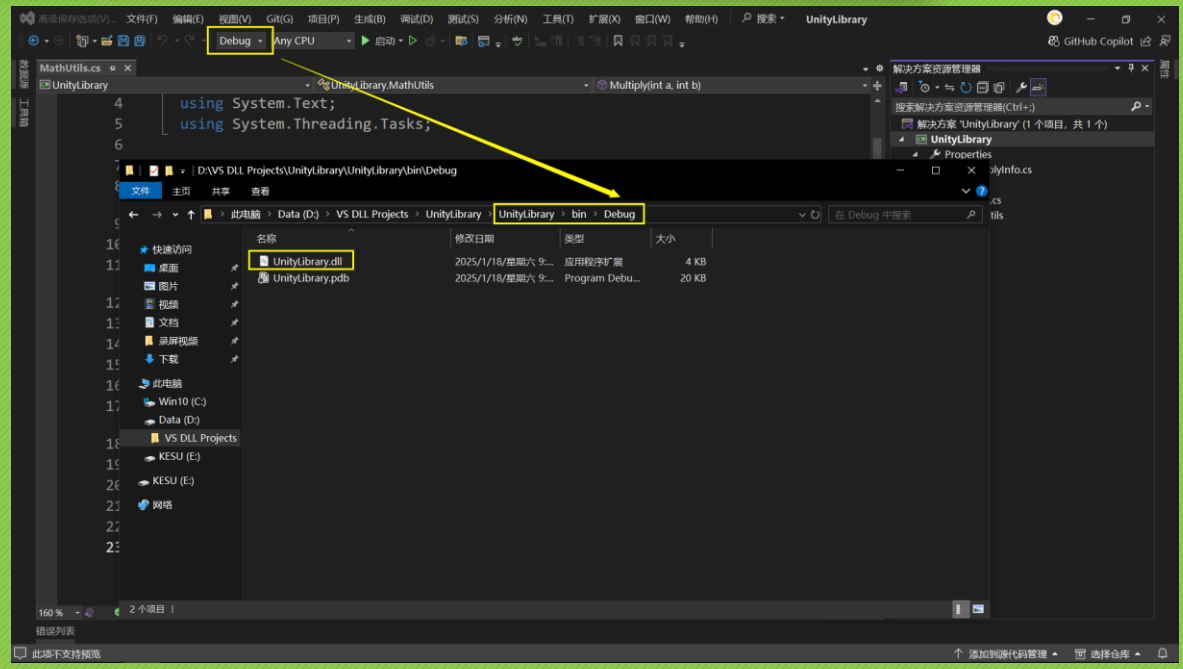
编写DLL（编写代码并编译生成）





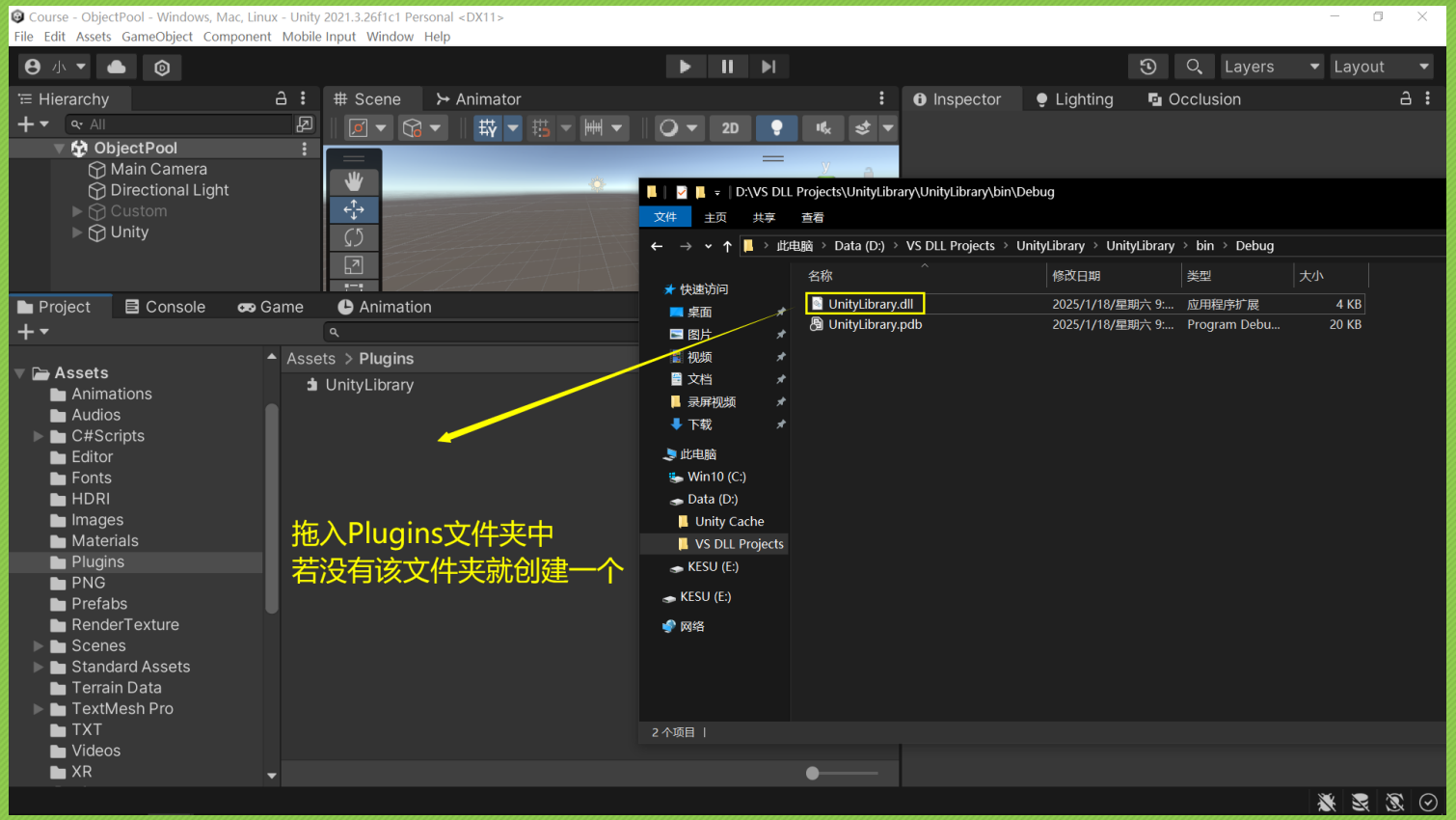
编写DLL（编写代码并编译生成）





编译成功后，生成的DLL文件通常位于项目目录下的**bin/Debug**或**bin/Release**文件夹中。Release模式它会把一些注释或帮助调试的部分删除，使程序保留最基础的代码，会更加精简。而Debug模式则会相应保留一些内容。这里我们以Debug模式为例。

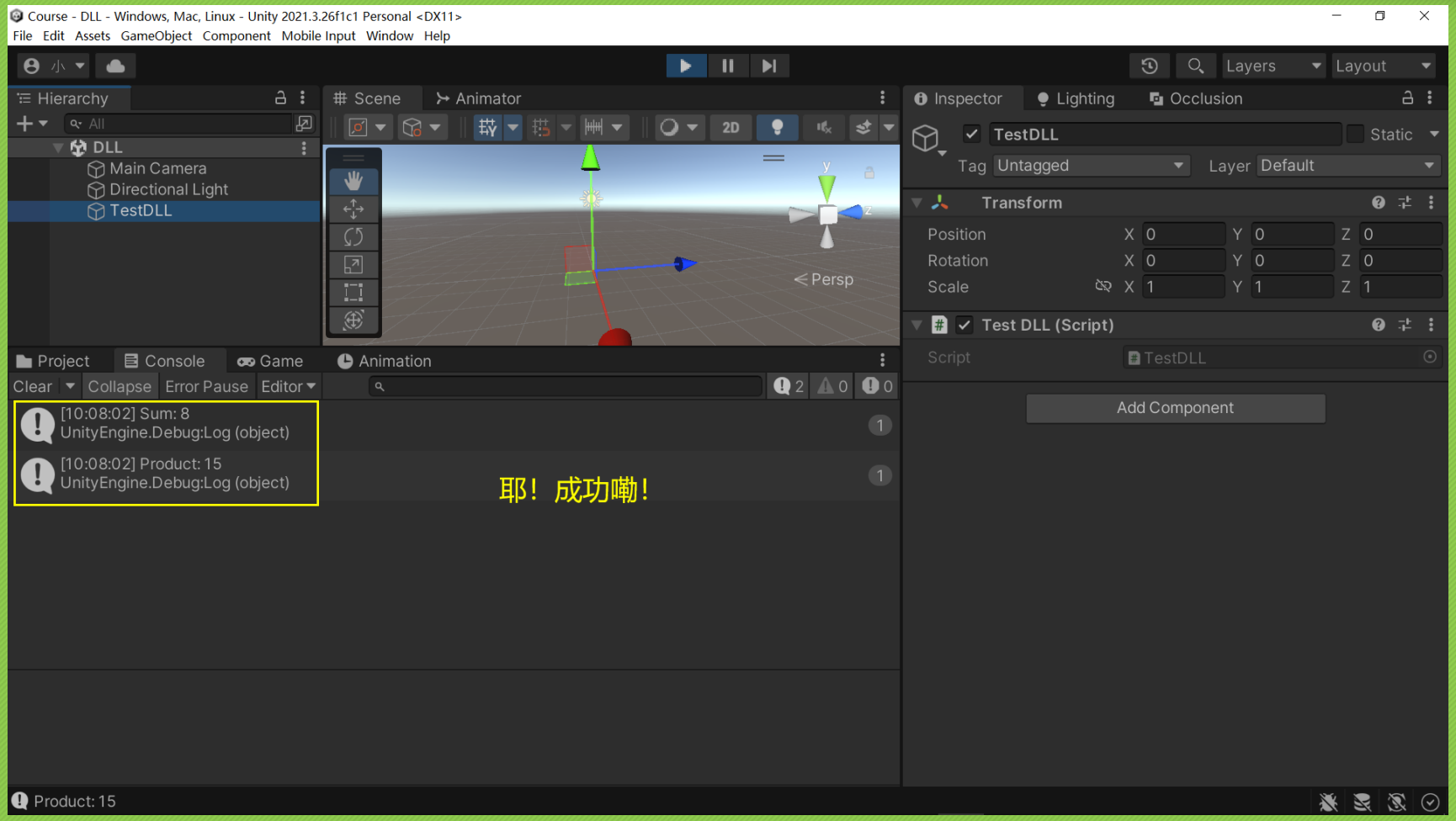
编写DLL（导入Unity并测试）




```cs
using UnityEngine;
using UnityLibrary;

// ================================================
// 实现功能：测试UnityLibrary.dll
// 编码作者：小贺儿
// 修改时间：2025年01月18日 10:05:54
// 修改内容：新建脚本
// 备注说明：该脚本挂载在TestDLL同名物体对象上
// ================================================

public class TestDLL : MonoBehaviour
{
    void Start()
    {
        int sum = MathUtils.Add(5, 3);
        Debug.Log($"Sum: {sum}");

        int product = MathUtils.Multiply(5, 3);
        Debug.Log($"Product: {product}");
    }
}
```

***注意引入UnityLibrary命名空间。**

编写DLL（播放运行）



注意事项：

- 确保DLL的目标框架与Unity**兼容**。Unity2020及以上版本默认使用.NET Standard 2.1。2019及以下版本使用.NET Framework 4.x。
- 避免命名空间和类名与Unity或其他插件**冲突**。
- 可以使用Visual Studio的**调试功能**来调试DLL。

使用DLL的优缺点（总结）

优点：

- **高效性。**DLL文件可以由多个程序**共享**，减少磁盘和内存占用。
- **易维护。**修改或升级功能时，只需**替换**DLL文件，不需要重新部署整个应用程序。
- **模块化开发。**大型项目可以被**拆分**为多个模块，由不同团队并行开发。
- **跨语言支持。**可以用**不同**语言编写DLL，然后供其他语言调用。
- **动态加载。**程序运行时根据需要加载DLL，提升启动速度。

缺点：

- **版本控制问题（“DLL地狱”）。**不同版本的DLL文件可能导致兼容性问题（新版的DLL修改了接口，而旧版程序还在调用旧接口）
- **调试复杂。**在调用DLL时，错误的**定位和修复**可能比较困难，尤其是跨语言调用时。
- **依赖问题。**如果程序运行时找不到所需的DLL，会导致**崩溃或功能失效**。同时，如果DLL被篡改，可能会带来**安全隐患**。
- **平台依赖。**DLL通常是平台相关的，比如Windows的.dll无法直接在Linux上使用（需改为.so）。

DLL的应用场景（总结）

- **游戏开发**
  - 将复杂的物理引擎、AI模块、音频引擎等功能打包成DLL，方便**独立维护**。
- **图像处理**
  - 提供**图像编辑、特效渲染**等功能的库。
- **网络通信**
  - 封装**网络协议、数据加密**等功能，供应用程序调用。
- **数据库访问**
  - 通过DLL封装数据库访问功能，提供**统一**的接口。
- **跨平台库**
  - 一些跨平台框架（如OpenCV）通过动态库提供**跨平台**功能。

【Unity基础教程】重点知识汇总（十四）

Unity C#编写DLL文件