# Unity - CustomPackages

## 官方文档链接

* 1. <https://docs.unity.cn/cn/current/Manual/CustomPackages.html>

## 包命名

* 1. 正式名称：注册包时使用

1. 正式名称必须符合 Unity Package Manager 命名约定，该命名约定使用反向域名表示法
2. 以 <domain-name-extension>.<company-name>（例如，或）开头，即使您的公司或网站名称以数字开头
3. 如果希望它在编辑器中可见，则不得超过 50 个字符。如果包名称不需要显示在编辑器中，则 Unity 包管理器会限制为 214 个字符或更少。
4. 仅包含小写字母、数字、连字符 （-）、下划线 （\_） 和句点 （.）
5. 若要指示嵌套命名空间，请在命名空间后加上一个句点。例如，“com.unity.2d.animation”和“com.unity.2d.ik”。
   1. 显示名称：用户可以在编辑器中看到的名称
6. 显示名称应简短，但应在一定程度上表明包中的内容。除此以外，Unity Package Manager 对显示名称没有任何限制。

## 包布局

<package-root>

├── package.json // 包清单，定义了包的依赖项和其他元数据。

├── README.md // 开发者包文档。

├── CHANGELOG.md // 对于包更改的描述，按照相反的时间顺序列示

├── LICENSE.md // 包含包许可证文本。

├── Third Party Notices.md // 含满足法律要求所需的信息。

├── Editor // 特定于 Editor 平台的 Assets 文件夹。与 Assets 下的 Editor 文件夹不同，这只是一个约定，不会影响资源导入管线

│ ├── <company-name>.<package-name>.Editor.asmdef

│ └── EditorExample.cs

├── Runtime // 特定于运行时平台的 Assets 文件夹。这只是一个约定，不会影响资源导入管线。

│ ├── <company-name>.<package-name>.asmdef

│ └── RuntimeExample.cs

├── Tests // 用于存储包中包含的任何测试的文件夹。

│ ├── Editor // 编辑器平台特定的测试文件夹。

│ │ ├── <company-name>.<package-name>.Editor.Tests.asmdef

│ │ └── EditorExampleTest.cs

│ └── Runtime // 特定于运行时平台的测试

│ ├── <company-name>.<package-name>.Tests.asmdef

│ └── RuntimeExampleTest.cs

├── Samples~ // 用于存储包中包含的任何示例的文件夹。

│ ├── SampleFolder1

│ ├── SampleFolder2

│ └── ...

└── Documentation~ // 用于存储包中包含的任何文档的文件夹。

└── <package-name>.md

* Unity 会忽略以 ~ 字符结尾的所有文件夹名称的内容（unity编辑器不会显示该文件内容），并且不会使用 .meta 文件跟踪它们
* CHANGELOG.md - 建议格式

# Changelog

All notable changes to this project will be documented in this file.

The format is based on [Keep a Changelog](https://keepachangelog.com/en/1.1.0/),

and this project adheres to [Semantic Versioning](https://semver.org/spec/v2.0.0.html).

## [Unreleased] // 提供 Unreleased 区块以记录即将发布的更新内容

### Added // 新添加的功能

### Changed // 对现有功能的变更

### Deprecated // 已经不建议使用，即将移除的功能

### Removed // 已经移除的功能

### Fixed // 对 bug 的修复

### Security // 对安全性的改进

## [1.1.1] - 2023-03-05

### Added

### Fixed

### Changed

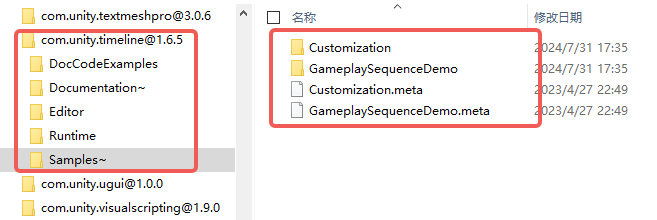
### Removed

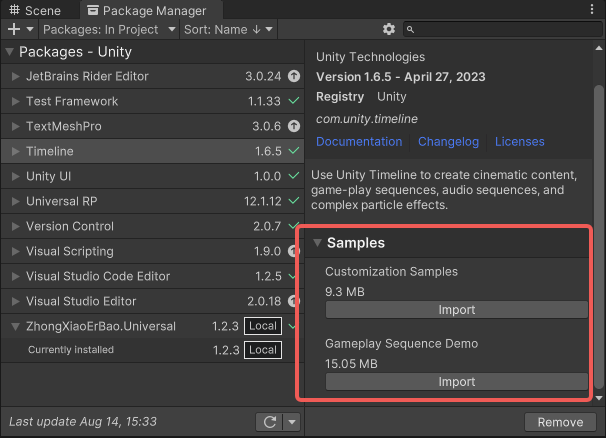
## 将测试添加到包中

（暂时用不到，等需要时再学习和添加笔记）

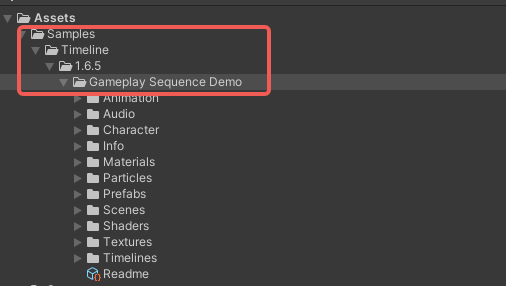
## 创建示例包

* 创建示例包后，可以在Unity Package Manager里，通过Samples将示例包添加到工程文件的Assets文件夹中





* 导入到Assets文件夹后，文件位置如下所示



1. 说明：当打开“Package Manager”窗口并选择包含样本的包时，包的详细信息面板中会针对包中的每个样本显示一个“导入”按钮。选择“导入”时，包管理器会将该示例的整个子文件夹结构复制到项目Assets文件夹下
2. 添加示例步骤：
3. 将资源文件或 C#示例代码文件放置 Samples~ 文件夹下
4. 在 package.json 清单文件中的 samples 数组下为每个示例添加一个 JSON 对象，即：将 JSON 数组添加到名为 samples 的 package.json 文件

{

"samples": [

{

"displayName": "HDRP Shaders",

"description": "Contains sample shaders for the High Definition render pipeline",

"path": "Samples~/SamplesHDRP"

},

{

"displayName": "URP Shaders",

"description": "Contains sample shaders for the Universal render pipeline",

"path": "Samples~/SamplesUniversalRP"

},

{

"displayName": "Standard RP Shaders",

"description": "Contains sample shaders for the Standard render pipeline",

"path": "Samples~/SamplesStandard"

}

]

}

* 说明：
* displayName：在 Package Manager 窗口中，将出现在资源包详细信息中的示例的名称。
* description：对样本演示或包含的内容的简要说明，这只是为了软件包清单。描述不会显示在界面中，甚至不会作为工具提示显示。
* path：从 Samples~ 文件夹到该示例的根文件夹的路径。

## 包清单

{

(\*必需\*)// 符合 Unity Package Manager 命名约定的唯一标识符，该约定使用反向域名表示法

  "name": "com.[company-name].[package-name]",

(\*必需\*)// 包版本号，使用格式

  "version": "1.2.3",

（推荐）// 显示在 Unity 编辑器中的用户友好名称（例如，在 Project 窗口、Package Manager 窗口等中）

  "displayName": "Package Example",

（推荐）// 包的简要说明。这是显示在“包管理器”窗口的详细信息面板中的文本

  "description": "This is an example package",

（推荐）// 指示包与的最低 Unity 版本兼容。如果省略，包管理器会认为包与所有 Unity 版本兼容

  "unity": "2019.1",

（可选）// 默认情况下，“项目”窗口会隐藏包的资源，并在“检查器”窗口中使用“对象选取器”时从结果中省略它们。设置此属性"false"可确保此包的资产始终可见

"hideInEditor":"",

（可选） // Unity 版本的一部分，指示包兼容的特定 Unity 版本。当更新的包需要在 Unity alpha/beta 开发周期内进行更改时，可以使用此属性。如果包需要新引入的 API，或者使用在没有 API 更新程序规则的情况下以非向后兼容方式更改的现有 API，则可能会出现这种情况

"unityRelease": "0b5",

（可选）// 此包文档的自定义位置，指定为 URL

注意：当包管理器无法访问 URL 位置时（例如，如果存在网络问题），它会执行以下操作：

- 如果已安装包，包管理器将打开一个文件浏览器，显示包缓存中的文件夹。

- 如果未安装包，包管理器会显示一条警告，指出脱机文档不可用。

"documentationUrl": "https://example.com/",

（可选）// 此包的更改日志的自定义位置，指定为 URL

  "changelogUrl": "https://example.com/changelog.html",

（可选）// 使用 SPDX 标识符格式的 OSS 许可证的标识符，或诸如“引用 LICENSE.md 文件”之类的字符串。如果在包清单中省略此属性，则包必须包含一个文件LICENSE.md

"license":"",

（可选）// 此包的许可证信息的自定义位置，指定为 URL

  "licensesUrl": "https://example.com/licensing.html",

（可选）// 包依赖项的映射。键是包名称，值是特定版本。它们标识此包所依赖的其他包

  "dependencies": {

    "com.[company-name].some-package": "1.0.0",

    "com.[company-name].other-package": "2.0.0"

 },

（可选）// 包装中包含的样品清单。每个示例都有一个显示名称、一个说明以及从文件夹开始的示例文件夹的路径

"samples":"",

（可选）// 为 Package Manager 提供其他信息的常量。保留仅供内部使用。

"type":"",

（可选）// Package Manager 搜索 API 使用的关键字数组

 "keywords": [

    "keyword1",

    "keyword2",

    "keyword3"

  ],

（可选）// 包的作者。此属性仅支持一个作者。此属性有一个必填字段 name 和两个可选字段 email 和 url

  "author": {

    "name": "Unity",

    "email": "unity@example.com",

    "url": "https://www.unity3d.com"

  }

}

* 说明：可选项可以忽略（不使用）。但是，如果它们存在，则它们必须具有有效的值。

## 程序集定义和包

* 程序集定义：

1. 将特定于 Editor 的代码存储在根 Editor 程序集定义文件中：
2. Editor/[YourCompany].[YourPackageName].Editor.asmdef
3. 将特定于运行时的代码存储在根运行时程序集定义文件中：
4. Runtime/[YourCompany].[YourPackageName].Runtime.asmdef
5. 为 Editor 和运行时脚本配置相关的测试程序集：
6. Tests/Editor/[YourCompany].[YourPackageName].Editor.Tests.asmdef
7. Tests/Runtime/[YourCompany].[YourPackageName].Runtime.Tests.asmdef

* 包依赖项

"versionDefines": [

{

"name": "com.unity.render-pipelines.high-definition",

"expression": "7.1.0",

"define": "HDRP\_7\_1\_0\_OR\_NEWER"

},

{

"name": "com.unity.modules.particlesystem",

"expression": "1.0.0",

"define": "USING\_PARTICLE\_SYSTEM"

}

]

## 版本控制

（暂未用到，后面需要时进行学习和补充）

## 创建CustomPackages

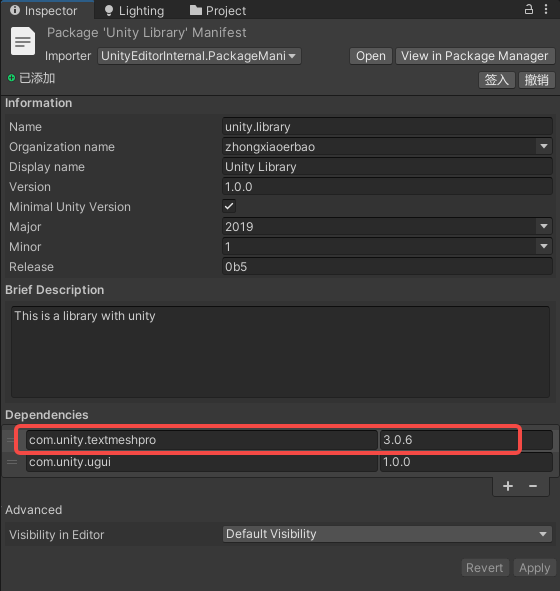
* 1. 在文件管理器中，打开Unity项目的Package文件夹
  2. 在Package文件夹中，为包创建一个文件夹
  3. 在子文件夹中，加入描述包及其依赖的JSON文件
  4. 将文件按布局约定进行归纳整理
  5. 建立好包的内容后，需要编辑包的Mainfest JSON文件，来描述包及其依赖
  6. 如果资源包包含有脚本，脚本必须指定到一个类型为asmdef的程序集定义文件
  7. Unity>Create>Assembly Defination，根据约定命名asmdef：公司名+功能名+引擎版本

## 分发资源包

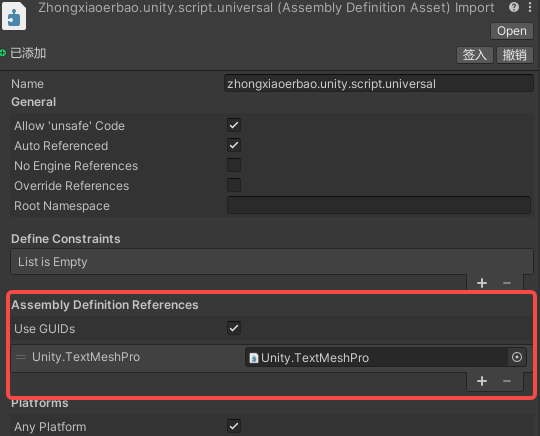
* 1. 添加许可证MD文件+第三方声明MD文件
  2. 分发1
     1. 压缩为.zip文件
     2. Unity>Window>Package Manager>Add Package from Disk，将.zip文件添加到包管理器中
  3. 分发2
     1. Git代码库
     2. 使用者电脑安装Git工具
     3. 复制URL
     4. Unity>Window>Package Manager>Add Package from Git URL

## Q：unity自定义包中，缺少某些命名空间

* 此处以”TextMeshPro”为例进行说明；
* 在package.json中，添加该命名空间的应用包



* 在\*.asmdef中，也需要将其添加为依赖项



## Q：在自定义包中，有多个嵌套文件夹，编辑器显示包时会怎么显示？

如：folder1 > folder2 > custom package，custom package是遵循自定义包规则的文件夹

* Unity只会从package.json的父级文件夹开始显示
* 如下图，虽然CSV文件夹在com.zhongxiaoerbao.library > unity里，但是unity编辑器中，只从package.json所在的文件夹开始显示内容

