# 第8章 资源管理

## 8.1 运行时加载资源

很多时候，我们不需要一开始就把某个资源放在场景中，而是在用到它的时候动态地将它加载到场景中。例如，有一个物体有可能出现在很多场景中，但是它出现的频率极低（比方说它是一个隐藏NPC，或者是一个错误提示框，又或者是一个打破记录时才会出现的提示框）。进一步说，某些游戏可以在运行中通过读取外部文件或者URL来加载资源，以减少初始的资源大小，或者是下载更新内容。

Unity专门设置了Resources Folders（资源目录）来存放需要运行时加载的资源，这些资源可以一开始不出现在场景中，但是在需要时可以随时被加载。另外还可以创建Asset Bundles（资源包）。这些资源可以认为是与游戏的主体流程完全分离的，可以通过读取文件或者访问URL的方式来访问，也可以称之为外部资源。

### 8.1.1 资源包概述（Asset Bundles）

一个资源包（Asset Bundle）是一组外部资源的集合。可以创建许多资源包，换句话说是许多组外部资源的集合。这些文件可以不直接被包含到Unity程序里面，而可能被放在服务器上，最终用户通过URL或者其它方式访问它们。

要创建一个资源包，可以在编辑器脚本中调用BuildPipeline.BuildAssetBundles方法来实现。这个方法的参数是一组物体的数组，这些物体会被打包到一个资源包中，另外还有一些其它参数选项。打包成功后会得到一个文件，之后可以通过AssetBundle.LoadAsset方法来在运行时加载这个文件。

### 8.1.2 资源目录（Resource Folders）

资源目录可以存放许多资源，Unity发布时它们也会被拷贝到发布版本中。但是它们可以不被场景中任何一个GameObject引用。

只要在Unity工程的Assets目录中创建一个名称为“Resources”的目录，即可作为资源目录。在Resources目录中还可以新建多个目录以管理所有资源。在需要用到这些资源的时候，调用Resourcs.Load方法即可。

**注意：**

在打包之后，所有资源目录中的资源以及它们的依赖关系会被保存到resources.assets文件中。If an asset is already used by another level it is stored in the .sharedAssets file for that level. The Edit -> PlayerSettings First Streamed Level setting determines the level at which the resources.assets will be collected and included in the build.

If a level prior to “First streamed Level” is including an asset in a Resource folder, the asset will be stored in assets for that level. If it is included afterwards, the level will reference the asset from the “resources.assets” file.

Only assets that are in the Resources folder can be accessed through [Resources.Load()](https://docs.unity3d.com/ScriptReference/Resources.Load.html). However many more assets might end up in the “resources.assets” file since they are dependencies. (For example a Material in the Resources folder might reference a Texture outside of the Resources folder)

只有存在于资源目录中的资源，才能使用Resources.Load方法访问。但是其实大部分资源出现在resources.assets文件中，因为它们被引用到了。例如资源目录中有一个材质，它引用了一个贴图文件，而贴图文件不在资源目录中，那么贴图最终也需要出现在resources.assets文件中。

### 8.1.3 资源卸载

可以通过调用AssetBundle.Unload方法卸载资源，如果将unloadAllLoadedObjects参数指定为true，那么所有资源包内部引用的资源与用AssetBundle.LoadAsset方法加载的资源都会被销毁，同时它们占用的资源也会被释放。

从资源包中加载资源并创建GameObject之后，有时更希望卸载资源包并释放内存，但是仍然保留已经创建的GameObject。这样做的好处是可以尽可能多地释放内存用来加载其它资源。这种情况下，将上述参数指定为false即可。当资源包被卸载之后，就不能再读取其中的资源了。

如果需要销毁场景中用Resource.Load加载的物体，可以先用Object.Destroy方法销毁它们。然后调用Resource.UnloadUnsusedAssets方法来释放相关的资源。

## 8.2 资源包（AssetBundles）

资源包是一种打包的文件，包含了特定平台的资源（包含模型、贴图、预制体、声音甚至可以是整个场景），并可以在运行时被加载。资源包包含了物体之间的依赖关系。例如：资源包A中的一个材质，可以引用资源包B中的一个贴图。而且为了加快资源包在网络中的传输，还可以利用内置的压缩算法将资源包压缩，根据不同的情况可以选择不同的压缩算法（LZMA或LZ4）。

资源包用于实现游戏扩展包（DLC）非常合适。因为可以缩小原始包的体积，让用户稍后下载根据特定平台定制优化的内容，以减少运行时的内存压力。

### 8.2.1 资源包里有什么？

其实，“资源包”包含两种不同但是又有关联的含义。

首先，它是磁盘中实实在在的文件。我们可以把资源包看作一种压缩文件，这种压缩文件可以看作是一个容器，就像是一个文件夹，可以存放很多种类型的文件。而这些文件又包含两种类型：序列化的文件与资源文件。序列化的文件包含了游戏中的元素，分别写入到这个资源包中。而资源文件会以纯粹的二进制数据块的方式保存，以便于我们能够从磁盘中快速加载。

资源包的另一个含义是它是一种对象，可以用脚本来操作它，比如从中提取具体的资源。资源包包含了一个资源表，里面有每个资源的路径，这样你就可以随时查找到某个资源并加载。

## 8.3 资源包使用流程

下面介绍使用资源包的使用步骤。每个步骤在本章后面还会有详细讨论。

### 8.3.1 选择打包的资源

要将某些资源打入一个资源包中，步骤如下：

1. 从工程视图窗口中选择要打包的资源。
2. 在inspector中查看。
3. 在inspector窗口底部，可以看到一个设置资源包和变体的栏位。