AssetBundle

1，AssetBundle是什么？

AssetBundle可以指两种不同但相关的东西。

首先是磁盘上的实际文件，我们称之为 AssetBundle 存档。包含两种类型文件：序列化文件和资源文件。

序列化文件包含分解为各个对象并写入此单个文件的资源。

资源文件只是为某些资源（纹理和音频）单独存储的二进制数据块，允许我们有效地在另一个线程上从磁盘加载它们。

其次是代码中的AssetBundle对象。此对象包含一个映射，即从已添加到此存档的资源的所有文件路径到按需加载的资源所包含的对象之间的映射。

2，AssetBundle工作流程？

一，为 AssetBundle 分配资源

1，通过Inspector右下角设置AssetBundle；2，通过代码（待续）

二，构建 AssetBundle

创建一个名为Editor的文件夹，并将包含以下内容的脚本放在该文件夹中：

BuildPipeline.BuildAssetBundles(assetBundleDirectory, BuildAssetBundleOptions.None, BuildTarget.StandaloneWindows);

三，将 AssetBundle 上传到场外存储

如果计划将AssetBundle上传到第三方托管站点，请在此步中执行该操作。

四，加载AssetBundle和资源

3，为AssetBundle准备资源？

逻辑实体分组

逻辑实体分组是指根据资源所代表的项目功能部分将资源分配给 AssetBundle。

类型分组

将相似类型的资源（例如音频轨道或语言本地化文件）分配到单个 AssetBundle。要构建供多个平台使用的 AssetBundle，类型分组是最佳策略之一。

并发内容分组

将需要同时加载和使用的资源捆绑在一起。

准备资源的一些原则：

将可能同时加载的对象分到一组，例如模型及其纹理和动画。

如果多个AssetBundle依赖的其他AssetBundle中的资源，拉入一个共享 AssetBundle。

如果一个AssetBundle中只有不到50%的资源经常同时加载，请考虑拆分该捆绑包。

将多个小型的（少于5到10个资源）但经常同时加载内容的 AssetBundle 组合在一起。

如果一组对象只是同一对象的不同版本，请考虑使用AssetBundle变体。

项目实战（待续）

4，构建 AssetBundle？

BuildAssetBundleOptions

有三个BuildAssetBundleOptions可以处理AssetBundle压缩：

BuildAssetBundleOptions.None: 使用LZMA格式压缩, 整个包全部解压之后才能使用.

BuildAssetBundleOptions.UncompressedAssetBundle：未压缩。

BuildAssetBundleOptions.ChunkBasedCompression：此捆绑包选项使用称为LZ4的压缩方法，因此压缩文件大小比 LZMA 更大，但不像 LZMA 那样需要解压缩整个包才能使用捆绑包。LZ4使用基于块的算法，允许按段或“块”加载 AssetBundle。解压缩单个块即可使用包含的资源，即使 AssetBundle 的其他块未解压缩也不影响。

BuildTarget

如果不想在构建目标中进行硬编码，可以使用EditorUserBuildSettings.activeBuildTarget，它将自动找到当前设置的目标构建平台，并根据该目标构建AssetBundle。

成功构建了AssetBundle之后，您可能会注意到AssetBundles目录包含的文件数量超出了最初的预期。确切地说，是多出了 2\*(n+1) 个文件。

对于在编辑器中指定的每个 AssetBundle，可以看到一个具有 AssetBundle 名称+“.manifest”的文件。

5，加载AssetBundle？

可以使用四种不同的 API 来加载 AssetBundle。

AssetBundle.LoadFromMemoryAsync

AssetBundle.LoadFromFile

WWW.LoadfromCacheOrDownload（即将弃用，使用 UnityWebRequest）

UnityWebRequest 的 DownloadHandlerAssetBundle （Unity 5.3 或更高版本）

IEnumerator InstantiateObject(){

string uri = "file:///" + Application.dataPath + "/AssetBundles/" + assetBundleName;

UnityEngine.Networking.UnityWebRequest request = UnityEngine.Networking.UnityWebRequest.GetAssetBundle(uri, 0);

yield return request.Send();

AssetBundle bundle = DownloadHandlerAssetBundle.GetContent(request);

GameObject cube = bundle.LoadAsset<GameObject>("Cube");

GameObject sprite = bundle.LoadAsset<GameObject>("Sprite");

Instantiate(cube);

Instantiate(sprite);

}

6，从 AssetBundle 加载资源？

决定如何加载资源时有几个选项。我们有LoadAsset和LoadAllAssets及其各自的异步对应选项LoadAssetAsync和LoadAllAssetsAsync。

GameObject gameObject = loadedAssetBundle.LoadAsset<GameObject>(assetName);

Unity.Object[] objectArray = loadedAssetBundle.LoadAllAssets();

AssetBundleRequest request = loadedAssetBundleObject.LoadAssetAsync<GameObject>(assetName);

yield return request;

var loadedAsset = request.asset;

AssetBundleRequest request = loadedAssetBundle.LoadAllAssetsAsync();

yield return request;

var loadedAssets = request.allAssets;

7，加载AssetBundle清单？

如果 AssetBundle 中包含依赖项，则在加载尝试实例化的对象之前，务必加载包含这些依赖项的捆绑包。Unity 不会尝试自动加载依赖项。加载AssetBundle清单可能非常有用。特别是在处理 AssetBundle 依赖关系时。

AssetBundle assetBundle = AssetBundle.LoadFromFile(manifestFilePath);

AssetBundleManifest manifest = assetBundle.LoadAsset<AssetBundleManifest>("AssetBundleManifest");

string[] dependencies = manifest.GetAllDependencies("assetBundle");

foreach(string dependency in dependencies)

{

AssetBundle.LoadFromFile(Path.Combine(assetBundlePath, dependency));

}

8，管理已加载的AssetBundle？

Unload是一个非静态函数，可用于卸载 AssetBundle。参数指示是否还要卸载通过此 AssetBundle实例化的所有对象。

AssetBundle.Unload(true) 卸载从 AssetBundle 加载的所有游戏对象（及其依赖项）。这不包括复制的游戏对象（例如实例化的游戏对象），因为它们不再属于 AssetBundle。

发生这种情况时，从该 AssetBundle 加载的纹理（并且仍然属于它）会从场景中的游戏对象消失，因此 Unity 将它们视为缺少纹理。

通常，使用AssetBundle.Unload(false)不会带来理想情况。大多数项目应该使用 AssetBundle.Unload(true) 来防止在内存中复制对象。

大多数项目应该使用 AssetBundle.Unload(true)。有两种常用方法：

1，在应用程序生命周期中具有明确定义的卸载瞬态 AssetBundle 的时间点，例如在关卡之间或在加载屏幕期间。

2，维护单个对象的引用计数，仅当未使用所有组成对象时才卸载 AssetBundle。

如果使用 AssetBundle.Unload(false)，销毁对象之后调用 Resources.UnloadUnusedAssets。

实践（待续）

9，Unity编辑器下开发时候使用的模拟模式？

模拟模式（待续）

10，Unity Asset Bundle Browser工具（了解）