

AssetBundle

游戏中的版本更新一般分为大版本更新和小版本更新 Version 6.1.2

小版本更新 一般主要更新的是资源和配置文件

大版本 一般就是更新代码

Unity中的游戏物体 一般会包括很多的依赖关系 比如身上的材质球 图片 和 其他组件

AssetBundle就是为了做资源的更新而存在的

好处：AB包可以将资源(prefab 二进制文件 图片 音乐等)进行压缩打包

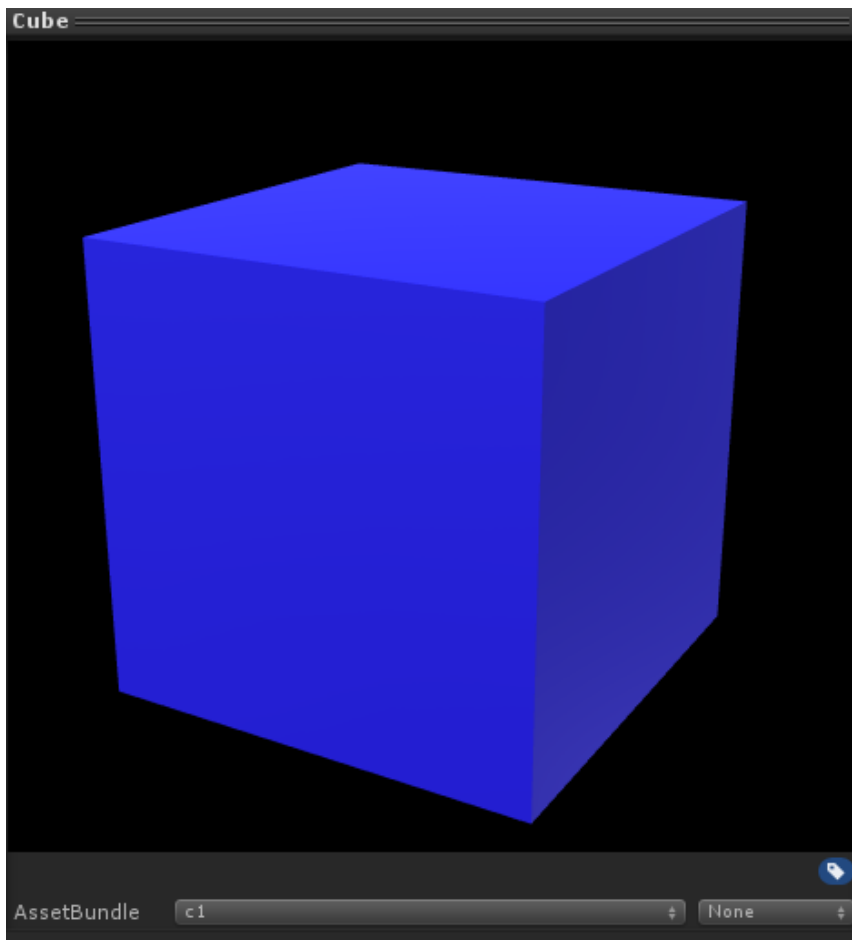
对应的资源可以包含全部的依赖关系

AssetBundle使用流程

1 打包 2 下载 3 卸载

打包

在Inspector窗口下方设置打包名字(AB包名)



打包模式包括单独打包和整体打包两种

开发中打包 至少打包 两个平台的AB包 安卓和苹果

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEditor;
5
6 public class Create : MonoBehaviour {
7
8     [MenuItem("AssetBundle/Build(single)")]
9     static void Build_AssetBundle()
10    {
11        //本地测试: 把打包后的AB包放在StreamingAssets文件夹下
12        //移动平台在读取的时候只能读取这个路径
13        //服务器下载 放在服务器上客户端通过WWW进行下载
14
15        BuildPipeline.BuildAssetBundles(Application.dataPath +
16        "/StreamingAssets/",
17        BuildAssetBundleOptions.DeterministicAssetBundle |
18        BuildAssetBundleOptions.ChunkBasedCompression,
```

```

17         BuildTarget.StandaloneWindows);
18     AssetDatabase.Refresh();
19 }
20 [MenuItem("AssetBundle/Build(Collection)")]
21 static void Build_AssetBundle_Collection()
22 {
23     AssetBundleBuild[] buildMap = new AssetBundleBuild[1];
24     buildMap[0].assetBundleName = "AllBundle";
25
26     //在Project视图中 选择要打包的对象
27     Object[] selects= Selection.GetFiltered(typeof(Object),
SelectionMode.DeepAssets);
28     //存储每一个资源在项目中的路径字符串
29     string[] allAssets = new string[selects.Length];
30
31     for (int i = 0; i < selects.Length; i++)
32     {
33         allAssets[i]=AssetDatabase.GetAssetPath(selects[i]);
34     }
35     buildMap[0].assetNames = allAssets;
36
37     BuildPipeline.BuildAssetBundles(Application.dataPath +
"/StreamingAssets/", buildMap,
38         BuildAssetBundleOptions.DeterministicAssetBundle |
BuildAssetBundleOptions.ChunkBasedCompression,
39         BuildTarget.StandaloneWindows);
40     AssetDatabase.Refresh();
41
42 }
43 }
44

```

打包选项

```

public class Test : MonoBehaviour
{
    [MenuItem ("Custom Editor/Build AssetBundles")]
    static void CreateAssetBundlesMain()
    {
        // 0 BuildAssetBundleOptions.None          一构建AssetBundle没有任何特殊的选项

        // 1 BuildAssetBundleOptions.UncompressedAssetBundle 一不进行数据压缩。如果使用该项，因为没有压缩\解压缩的过程，
        //                                           AssetBundle的发布和加载会很快，但是AssetBundle也会更大，下载变慢

        // 2 BuildAssetBundleOptions.CollectDependencies      一包含所有依赖关系

        // 4 BuildAssetBundleOptions.CompleteAssets         一强制包括整个资源

        // 8 BuildAssetBundleOptions.DisableWriteTypeTree    一在AssetBundle中不包含类型信息。发布web平台时，不能使用该项

        // 16 BuildAssetBundleOptions.DeterministicAssetBundle 一使每个Object具有唯一不变的hash ID，可用于增量式发布AssetBundle

        // 32 BuildAssetBundleOptions.ForceRebuildAssetBundle 一强制重新Build所有的AssetBundle

        // 64 BuildAssetBundleOptions.IgnoreTypeTreeChanges  一忽略TypeTree的变化，不能与DisableTypeTree同时使用

        //128 BuildAssetBundleOptions.AppendHashToAssetBundleName 一附加hash到AssetBundle名称中

        //256 BuildAssetBundleOptions.ChunkBasedCompression 一Assetbundle的压缩格式为lz4。默认的是lzma格式，下载Assetbundle后立即解压。
        //                                           而lz4格式的Assetbundle会在加载资源的时候才进行解压，只是解压资源的时机不一样













        BuildPipeline.BuildAssetBundles("Assets/StreamingAssets", BuildAssetBundleOptions.None, BuildTarget.Android);
    }
}

```

打包后的目标文件夹 StreamingAssets

整体打包比单独打包节省很多的空间

manifest文件查看对应的bundle记录的信息

 allbundle	2018/7/8 20:22	文件	49 KB
 allbundle.manifest	2018/7/8 20:22	MANIFEST 文件	1 KB
 allbundle.manifest.meta	2018/7/8 20:22	META 文件	1 KB
 allbundle.meta	2018/7/8 20:22	META 文件	1 KB
 c1	2018/7/8 20:08	文件	49 KB
 c1.manifest	2018/7/8 20:08	MANIFEST 文件	1 KB
 c1.manifest.meta	2018/7/8 20:08	META 文件	1 KB
 c1.meta	2018/7/8 20:08	META 文件	1 KB
 c2	2018/7/8 20:45	文件	39 KB
 c2.manifest	2018/7/8 20:45	MANIFEST 文件	1 KB
 c2.manifest.meta	2018/7/8 20:08	META 文件	1 KB
 c2.meta	2018/7/8 20:08	META 文件	1 KB

加载

使用WWW类进行加载

网络资源和StreamingAssets文件夹都必须使用WWW进行加载

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4

```

```

5 public class Load : MonoBehaviour {
6
7     public static string m_PathURL;
8     void Awake () {
9         //根据平台去判断 每个平台的路径是不一样的
10        m_PathURL =
11        #if UNITY_ANDROID
12            "jar:file://" + Application.dataPath + "!/assets/";
13        #elif UNITY_IPHONE
14            Application.dataPath + "/Raw/";
15        #elif UNITY_STANDALONE_WIN || UNITY_EDITOR
16            "file://" + Application.dataPath + "/StreamingAssets/";
17        #else
18            string.Empty;
19        #endif
20
21    }
22
23    void OnGUI()
24    {
25        if (GUILayout.Button("加载Cube1"))
26        {
27            StartCoroutine(LoadAssetBundleSingle("c1"));
28        }
29        if (GUILayout.Button("加载Sphere1"))
30        {
31            StartCoroutine(LoadAssetBundleSingle("c2"));
32        }
33        if (GUILayout.Button("加载所有的AssetBundle"))
34        {
35            StartCoroutine(LoadAssetBundleAll());
36        }
37        if (GUILayout.Button("LoadByCache"))
38        {
39            StartCoroutine(LoadMainCacheObject("s1"));
40        }
41    }
42
43    IEnumerator LoadAssetBundleSingle(string assetName)
44    {
45        WWW www = new WWW(m_PathURL + assetName);
46        yield return www;

```

```

47         string[] name = www.assetBundle.GetAllAssetNames();
48
49         Instantiate(www.assetBundle.LoadAsset(name[0]));
50
51     }
52
53     IEnumerator LoadAssetBundleAll()
54     {
55         WWW www = new WWW(m_PathURL + "allbundle");
56         yield return www;
57
58         string[] names = www.assetBundle.GetAllAssetNames();
59
60         foreach (string name in names)
61             Instantiate(www.assetBundle.LoadAsset(name));
62     }
63
64     IEnumerator LoadMainCacheObject(string path)
65     {
66         //清除缓存
67         //Caching.CleanCache();
68         WWW bundle = WWW.LoadFromCacheOrDownload(m_PathURL + path, 2);
69         yield return bundle;
70         string[] name = bundle.assetBundle.GetAllAssetNames();
71         Instantiate(bundle.assetBundle.LoadAsset(name[0]));
72     }
73 }

```

卸载

