**Unity3D教程：关于LightMapping和NavMesh烘焙的动态加载**

Posted on 2013年07月18日 by U3d / [Unity3D 基础教程](http://www.unitymanual.com/category/manual/unity3d-%e5%9f%ba%e7%a1%80%e6%95%99%e7%a8%8b) /被围观 6 次

熟悉Unity的朋友都应该知道，unity有内部LightMapping烘焙和NavMesh寻路的功能。但这些非常好用的功能，都是基于对某个已经保存的关卡（scene）进行烘焙（Bake）的操作，我一直以为，一般来说这样很难做到外部加载或者动态加载的。

但通过最近一段时间的研究，发现其实这些基于scene的功能，也是可以动态加载的，虽然在使用上是有一定的限制，不过合理利用还是能做到不少想要的效果的。

假设我们限制有这样的项目要求，游戏项目是网页游戏，由于容量的关系，我们所有的场景都是外部动态加载的，但为了效果好，我们使用了LightMapping烘焙了场景，而为了有寻路功能，我们对场景进行了NavMesh的烘焙。我们希望把这个烘焙了的场景像一般的assetBundle那样从外部加载进来。

现在我们需要把这个烘焙好的场景导出成assetBundle。

使用的方法是：BuildPipeline.BuildPlayer

BuildPipeline.BuildPlayer 编译播放器

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | | **static** function BuildPlayer (levels : **string**[], locationPathName : **string**, target : BuildTarget, options : BuildOptions) : **string**<!--DVFMTSC--> | | |
|  | |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | | 用法举例： | | |
|  | |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | | <!--wp\_fromhtmlpreview\_devfmt--> | | |
|  | |  |  |

|  |  |
| --- | --- |
| 1 | @MenuItem ("Build/BuildWebplayerStreamed") |
| 2 | **static** function MyBuild(){ |
| 3 | **var** levels : **String**[] = ["Assets/StreamedScene1.unity", "Assets/StreamedScene2.unity", |
| 4 | "Assets/StreamedScene3.unity"]; |
| 5 | BuildPipeline.BuildPlayer( levels, "StreamedWebplayer.unity3d", |
| 6 | BuildTarget.WebPlayer, BuildOptions.BuildAdditionalStreamedScenes); |
| 7 | } |

例如我们把上面烘焙好的场景保存成scene001，那么我们就可以写成：

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 1 | **var** levels : **String**[] = ["Assets/scenes/scene001.unity"]; |
| 2 | BuildPipeline.BuildPlayer( levels, "scene001.unity3d", |
| 3 | BuildTarget.FlashPlayer , BuildOptions.BuildAdditionalStreamedScenes); |

这样我们可以把这个scene001的场景导出成了scene001.unity3d的assetBundle。当我们想从外部加载这个烘焙好的场景时，可以使用WWW来下载他，但用法和一般的www加载assetBundle有点不同的用法。

假设我们把这个导出的asssetBundle放在了一个路径（url），那么我们可以这样加载它：

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 1 | WWW www = new WWW(Url); |
| 2 | **yield** **return** www; |
| 3 | **if** (www.error == **null**) |
| 4 | { |
| 5 | AssetBundle level = www.assetBundle; |
| 6 | Application.LoadLevelAdditive(“”); |
| 7 | } |

可以看出，在前两步，我们的做法和普通的www加载assetBundle是一样的，都是获取一个新的WWW资源，但下面的一步就有点不一样，由于他是一个场景，而不是一个GameObject，所以我们不能通过Instantiate 来实例化他，而是用Application.LoadLevel 或者Application.LoadLevelAdditive 之类的加载关卡的方法来读取他的。在读取之前，我们只需要随便的把获取回来的www .assetBundle赋予给一个变量，然后，我们就能按照关卡的名称来LoadLevel了。Unity3D教程手册：www.unitymanual.com

现在，关卡已经能加载进一个新的项目了，我们可以看到，lightmapping的效果，和NavMesh寻路的网格，依然存在。

不过我们需要注意以下几个问题：

1、和一般的assetBundle一样，假如我们导出的关卡场景里面包含了某些新项目里面没有的脚本，那么这些脚本在导入新项目的时候会全部丢失。

2、NavMesh的外部加载只能使用Application.LoadLevel ，如果使用Application.LoadLevelAdditive 的话，NavMesh会全部丢失。也就意味着，NavMesh的场景一次只能加载一个，而不能同时加载多个。所以用NavMesh做动态大地图是不可行的。但LightMapping可以使用Application.LoadLevelAdditive ，可以多个场景同时叠加。