Asset Graph 使用手册

dionysoslai

概要

AssetGraph 是一个图形化工具，主要是为了解决资产导入、构建资产包方面的工作量。使用这个工具，我们可以构建图形化工作流--创建、修改和更改资产设置，然后实现自动化打包功能。

安装

这块跟其他包安装方式差不多，但有一点需要主要。再Package Manager窗口，需要勾选**Advanced->Show preview packages**才能看到AssetGraph 包。

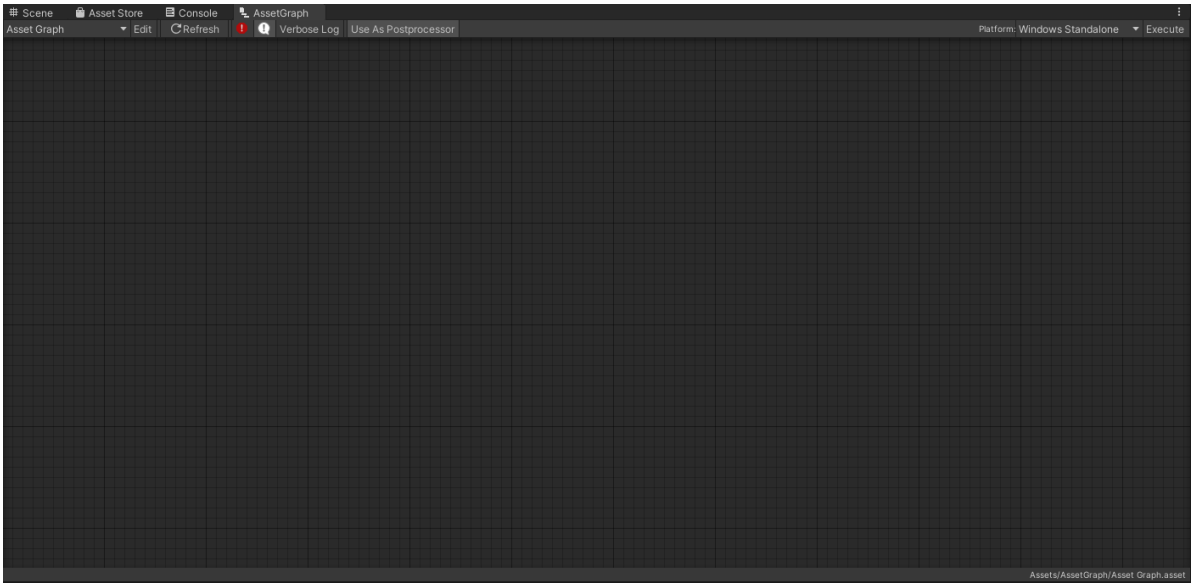
基础用法

安装AssetGraph之后，通过菜单**Window->AssetGraph**即可看到具体菜单内容。其中有2个比较重要：

1. Open Graph Editor：编辑单个图表工作流；
2. Open Batch Build Window：批处理多个图标工作流；

1. 创建一个Graph

创建一个Graph，主要有2种方式：**Window -> AssetGraph -> Open Graph Editor，然后点击"Create"按钮**或者在Project窗口下**右键->Create->Asset Graph，然后双击打开即可**。结构如下所示：

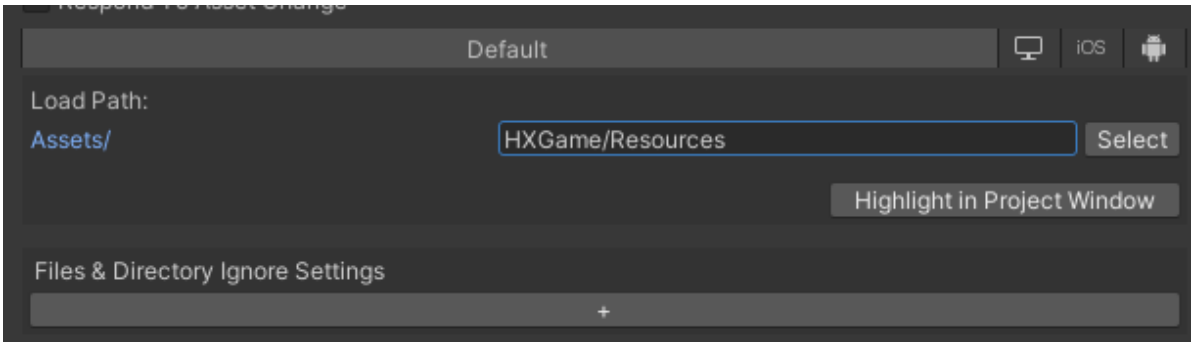


其中，通过左上角**Asset Graph->Create New..**，同样可以创建一个新的Graph。

2. 加载Assets

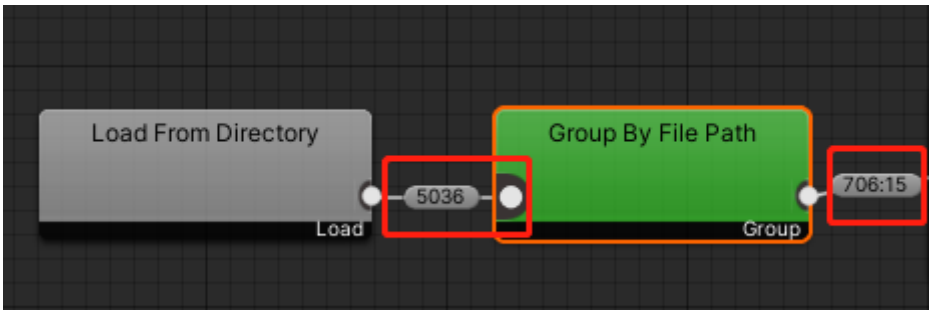
工作流第一步是添加加载Assets节点，这里我们可以通过在面板**右键->Load Assets->Load From Directory**方式添加；或者直接将要加载的文件夹拖到对应面包中即可。

点击该节点，就可以在Inspector界面中选择或者更改需要加载的文件夹，这里我们选择“HXGame/Resources"目录。如下所示：



3. 资产分组

对资产我们一般都需要进行分组打包（如果将一个文件夹内容都打进同一个Asset Bundle中，可能会出现包过大问题），AssetGraph 提供分组有：文件夹路径、文件大小、文件名三种方式。这里我们可以选择路径方式：**右键->Group Assets->Group by File Size**，并在Inspector界面设置中Grouping keyword设置为：Prefab/*/。表示我们只对Prefab下文件进行分组。

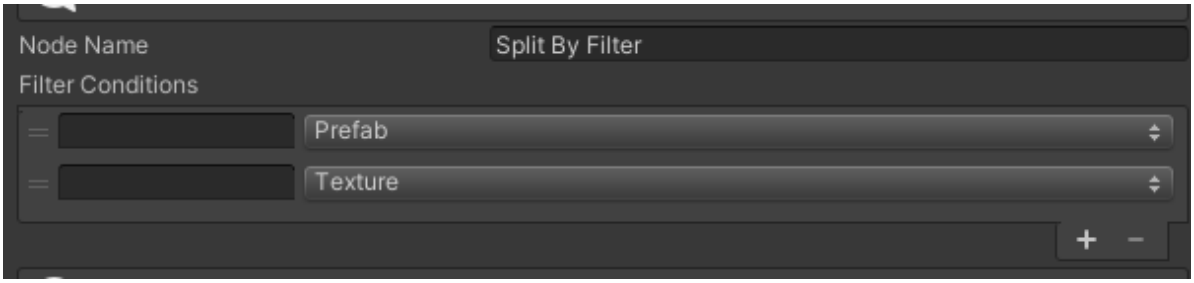


如图所示，从刚开始5036个文件缩减到706，其中15表示15个组（正好对应Prefab路径下的15个文件夹）。

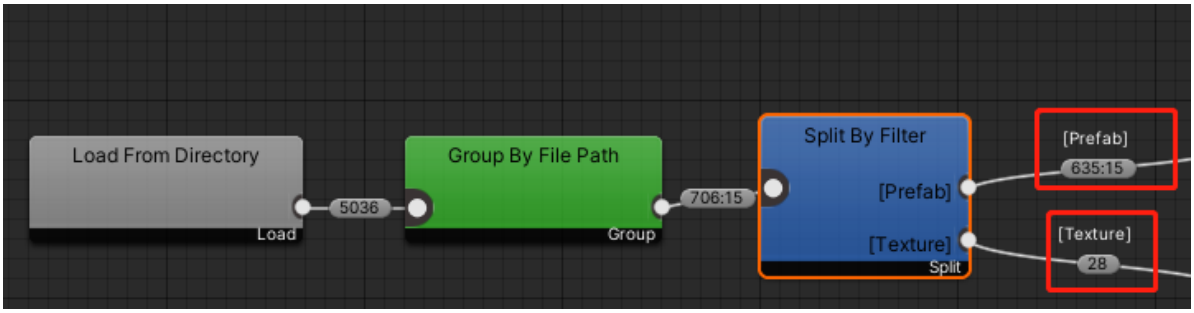
4. 资产分类/过滤

在使用目录方式进行资产分组时，给定的资产类型闭并不总是相同的。有时候我们希望能将同一类资产打包在一个包里，比方将材质、贴图、预设分开。这里我们就可以用到**Split by Filter**功能。

在面板**右键->Split Assets->Split by Filter**添加一个Filter节点。此时，该节点只有入口没有出口，因此我们需要在Inspector界面添加配置：



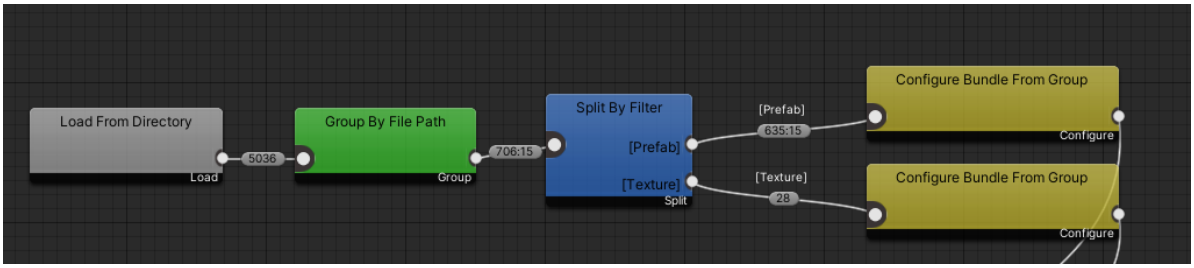
这里，我们选择了prefab 和 texture 2种过滤方式。在面板中，我们可以看到：Prefab有635（总共15组，即在15个文件夹中），Texture只有28个，并且都是在同一个组中（全部在UIAssets组中）。



5. 设置Bundle属性

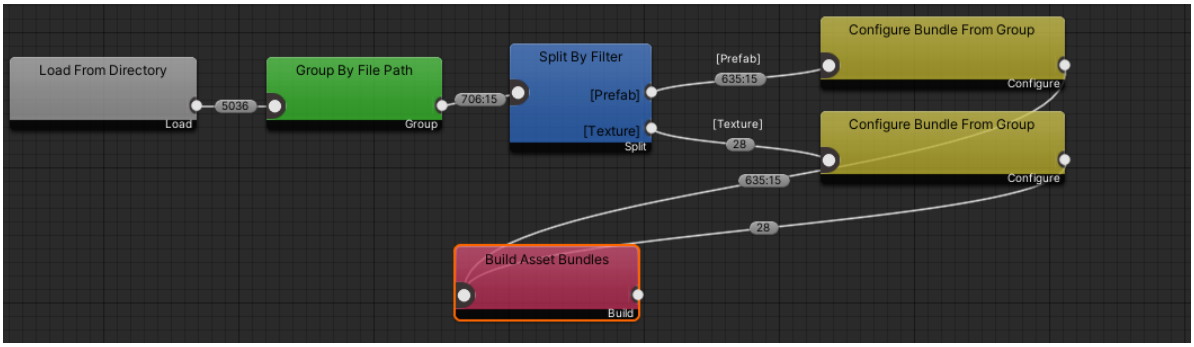
当我们进行了以上资产分组后，接下来需要将其转换为Asset Bundle。因此我们配置一些AB的信息：Bundle Name 和 Varints（其中，variants 并不是必须）。在面板中**右键->Configure Bundle->Configure Bundle From Group**，添加Configur 节点。

这里，我们添加2个节点，并分别在Inspector 中，将Bundle Name Template设置成bundle_texture*和*bundle_texture*。

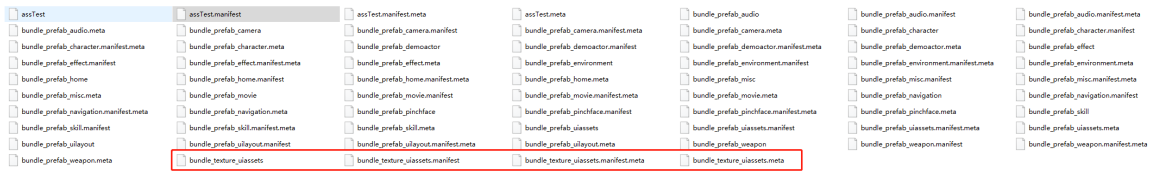


6. 打包

到这一步，我们已经成功的设置了bundle。现在进行最后一步构建Asset Bundle。在面板**右键->Build->Build Asset Bundles**，添加一个Build 节点。然后将Configure 节点连接到Build 节点。如果，我们要更改构建设置，选择Build 节点，在Inspector面板中，将列出我们可以更改的配置。这里，我们勾选Disable Write TypeTree选项。最后，全部打包图标如下所示：



现在，我们就可以打包了，点击面板右上角**Execute**按钮即可。执行结果如下：



可以看到，最终我们打出来了16个包，其中15个Prefab包，1个Texture 包。

一些高级特性

1. 批量打包

菜单**Window->Asset Graph -> Open Batch Build Window**，提供批量打包功能。

2. 创建新资产

通过Asset Graph 可以从Group 中创建Prefab，**右键->Create Assets->Create Prefab From Group**即可添加此节点。默认提供了2中创建方式：Replace GameObject by Name 和 Replace With incoming GameObject 2种。这两个都是通过脚本配置的，我们可以自定义修改一些内容

其他内容

Asset Graph 还提供了诸如编辑资产、提取公共资产等内容，暂时还需要研究下。

一些思考

先对于Asset Bundle Browser 和 Addressable 2个插件。Asset Bundle Browser 插件功能比较单纯，简单提供可视化AB打包工具；Addressab 插件则是在基于AB模式，提出一整套打包、加载、更新方案。Asset Graph 则侧重与AB打包整体方案解决。主要有几个亮点：

1. Grouph打包模式：一个文件夹下包，可以根据大小、文件名、路径方式进行分组，甚至我们可以提供根据文件修改频率进行分包，统一将高频率包打包一起，减少更新包大小；
2. 资产分类：目前Asset Graph可以根据不同类型资产分开打包，这样就可以进行一些几种话管理。比方我们可以统一将一些ui资产打在一个包里，例如加载管理；
3. 统一构建/批处理构建：可以加快构建速度。
4. 自动创建Prefab：通过提供create功能，我们应该可以实现一些复杂的Prefab一键完成功能（这个在最近“《黑暗之潮》中的应用经验及技术分享”中提到，不过还需要验证下）。

主要一些不足：

1. Asset Graph 的基础是文件夹管理，因此，我们首重工作还是要进行合理项目文件管理，否则会出现包分散的问题（在多约束条件下，一个包里面就只有几个资源）--因此创作者需要将一些文件合理的放到特定文件夹中（可以做成工具）；
2. 在实际使用最终的AB时，我们加载具体一个资产，还是需要提前知道这个资产在哪个AB包中（除非生成一些配置文件）；