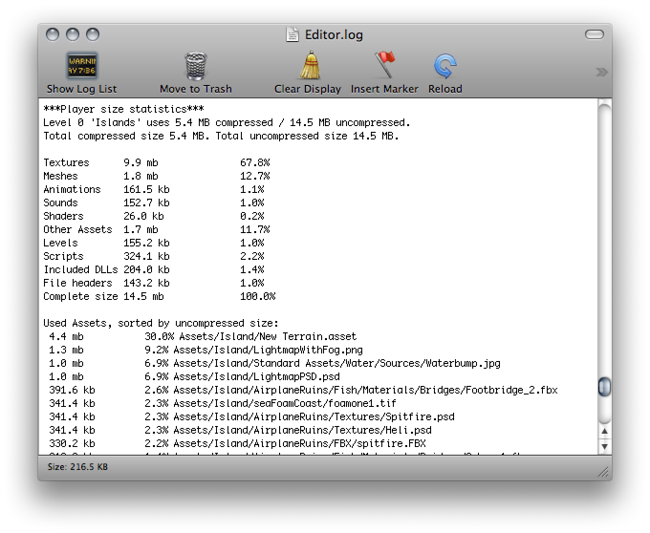
**Reducing the File Size of the Build**

**降低生成文件的大小**

Keeping the file size of the built app to a minimum is often important, especially for mobile devices or for app stores that impose a size limit. The first step in reducing the size is to determine which assets contribute most to it, since these assets are the most likely candidates for optimisation. You can find this information out from the Editor Log just after you have performed the build (select Open Editor Log from the small panel menu in the top right of the Console window).

尽可能小的保持生成出的app文件大小也是非常重要的。尤其是对于移动设备或者对大小有限制的应用商店。减少大小的第一步就是先确定哪一部分的资源占了最大的比重，因为这些资源是最优可能的优化对象。你可以在执行发布之后再编辑器的log中发现这部分信息（在Console窗口右上方的小面板菜单中选择Open Editor Log）



The editor log just after a build

发布之后的editor log

The log provides a summary of assets broken down by type and then lists all the individual assets in order of size contribution. Typically, things like textures, music and videos will take up the most storage while scripts, levels and shaders are often negligible. Note that the *File Headers* mentioned in the summary are not assets in themselves. The headers are actually the extra data that is added to “raw” asset files to store references and settings. The headers normally make very little difference to asset size but if the value may be large if you have numerous large assets in the Resources folder.

The log helps you identify assets that you might want to remove or optimise but you should consider the following before setting to work:

日志提供了根据类型划分并按照占用大小排列的资源的摘要信息。通常情况下，像纹理，音频和视频会占用最大的存储空间，脚本，关卡和着色器经常可以忽略不计。注意摘要中提到的文件头本身不是资源。文件头实际上是添加到原始资源文件中用于挤在引用和设置的额外数据。文件头通常情况下不会对资源大小产生很大的影响，但是如果你Resources文件目录下的资源数量很大的话，文件头也会很大。日志帮助你明确那些你可以希望移除或者优化的资源，但是在执行之前你还需要考虑以下几点：

\* Unity re-codes imported assets into its own internal formats, so the choice of source asset type is not relevant. For example, if you have a multi-layer Photoshop texture in the project then it will be flattened and compressed before building. Exporting the texture as a PNG file will not make any difference to build size, so you should stick to the format that is most convenient for you during development.

\* Unity将导入的资源以它自己的格式重新编码，所以原始资源的格式是不相干的。比如说，如果你在项目中有一张多层级的Photoshop纹理那么它在生成之前会被扁平化然后进行压缩处理。将纹理导出成一张png文件并不会对生成包大小产生影响，所以你应该在开发过程中选择对你来说最方便的格式。

\* Unity strips most unused assets during the build, so you don’t gain anything by manually removing assets from the project. The only assets that are not removed are scripts (which are generally very small anyway) and assets in the Resources folder (since Unity can’t determine which of these will be needed and which won’t). With this in mind, you should make sure that the assets in the Resources folder are only the ones you really need for the game. You might be able to replace assets in Resources with AssetBundles to load assets dynamically and thereby reduce the player size.

\* Unity在生成过程中会剥离大部分没有使用的资源，所以你不用在项目中手动移除资源。唯一不会被移除的资源就是脚本（仅占用很小的空间）和Resources目录下的资源（因为Unity不能确定其中哪一些是会被使用的）。因此，你需要确保在Resources目录下的资源都是游戏真正需要的。你可以将Resources下的资源替换成AssetBundles的形式动态加载来降低终端的大小。

**Suggestions for Reducing Build Size**

**降低生成包大小的建议**

**Textures**

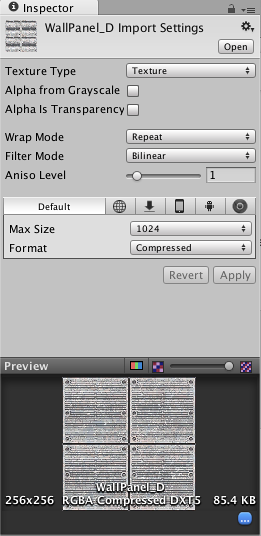
**纹理**

Often textures take up most space in the build. The first to do is to use compressed texture formats (DXT(Desktop platforms) or PVRTC) where you can.

通常情况下纹理在生成中占用了大部分的空间。首先要做的事就是尽可能使用压缩的纹理格式（DXT（桌面平台）或者PVRTC） 。

If that doesn’t get the size down, try to reduce the size of the textures. The trick here is that you don’t need to modify the actual source content. Simply select the texture in the Project view and set Max Size in Import Settings. It is a good idea to zoom in on an object that uses the texture, then adjust the Max Size until it starts looking worse in the Scene View.

如果这不能降低大小，那就尝试降低纹理的尺寸。这里有个技巧，你不需要真的去修改原文件内容。只需要简单的在Project视图中选中纹理然后在Import Settings中修改Max Size.这是一个很好的压缩纹理的方式，调整Max Size直至它在场景视图中看起来变得糟糕。



Changing the Maximum Texture Size will not affect your texture asset, just its resolution in the game

修改Maximum Texture Size将不会影响你的纹理资源，只会影响它在游戏中的分辨率

The following table shows how much storage the image formats take up in bytes per pixel:

下面的表格展示了不同的图像格式每个像素所占用的存储空间大小，以byte为单位：

|  |  |
| --- | --- |
| **Compression** | **Memory consumption (bytes/pixel)** |
| **Standalone & WebGL** |  |
| RGB Crunched DXT1 | *variable* |
| RGBA Crunched DXT5 | *variable* |
| RGB Compressed DXT1 | 0.5 bpp |
| RGBA Compressed DXT5 | 1 bpp |
| RGB 16bit | 2 bpp |
| RGB 24bit | 3 bpp |
| Alpha 8bit | 1 bpp |
| RGBA 16bit | 2 bpp |
| RGBA 32bit | 4 bpp |
| **iOS** |  |
| RGB Compressed PVRTC 2 bits | 0.25 bpp (bytes/pixel) |
| RGBA Compressed PVRTC 2 bits | 0.25 bpp |
| RGB Compressed PVRTC 4 bits | 0.5 bpp |
| RGBA Compressed PVRTC 4 bits | 0.5 bpp |
| RGB 16bit | 2 bpp |
| RGB 24bit | 3 bpp |
| Alpha 8bit | 1 bpp |
| RGBA 16bit | 2 bpp |
| RGBA 32bit | 4 bpp |
| **Android** |  |
| RGB Compressed DXT1 | 0.5 bpp (bytes/pixel) |
| RGBA Compressed DXT5 | 1 bpp |
| RGB Compressed ETC1 | 0.5 bpp |
| RGB Compressed PVRTC 2 bits | 0.25 bpp (bytes/pixel) |
| RGBA Compressed PVRTC 2 bits | 0.25 bpp |
| RGB Compressed PVRTC 4 bits | 0.5 bpp |
| RGBA Compressed PVRTC 4 bits | 0.5 bpp |
| RGB 16bit | 2 bpp |
| RGB 24bit | 3 bpp |
| Alpha 8bit | 1 bpp |
| RGBA 16bit | 2 bpp |
| RGBA 32bit | 4 bpp |

The formula for the total image storage size is *width \* height \* bpp*. If you are using mipmaps then the storage will be about a third larger than the single image.

图像整体存储大小计算公式为 *width \* height \* bpp(宽度 \* 高度 \* 每个像素占用的bytes数).* 如果你使用了mipmaps，那么相对于单独的图像还会占用额外的1/3存储空间。

By default Unity compresses all textures when importing. For faster workflow in the editor, you can turn compression off from the Preferences but all textures will be compressed in the build regardless of this setting.

Unity会在导入时对所有纹理进行压缩。为了在编辑器中更快的工作，你可以在偏好设置（Preferences）中关闭压缩，但是无论如何在生成时所有的纹理都会进行压缩。

**Meshes and Animations**

**网格和动画**

[Meshes](http://docs.unity3d.com/540/Documentation/Manual/class-Mesh.html) and imported Animation Clips can be compressed so they take up less space in your game file. Compression can be turned on in the Mesh Import Settings.

网格和导入的动画片段可以通过压缩的方式降低他们在游戏中占用的空间。在Mesh Import Setting可以开启压缩。

Mesh and Animation compression uses quantization, which means it takes less space but the compression can introduce some inaccuracies. Experiment with what level of compression is acceptable for your models.

网格和动画的压缩使用了量化的方式，这意味着会占用更少的空间，但是压缩会导致一些不确定性。 尝试出你的模型可接受的压缩级别。

Note that mesh compression only produces smaller data files and does not use less memory at run time. Animation keyframe reduction produces smaller data files *and* uses less memory at run time and generally you should always have it enabled.

注意网格压缩只会减少数据文件的大小，但是不会减少运行时的内存占用。动画关键帧的减少会降低数据文件的大小以及运行时的内存通常情况你应该总是将它设置成可用。

**DLLs**

By default, Unity includes only the following DLLs in the built player:

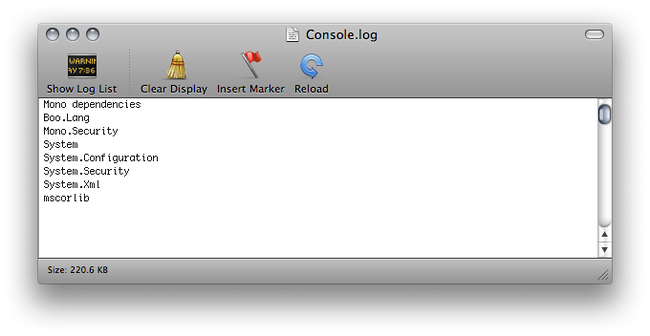
默认情况下，Unity仅在生成终端时引入了以下的DLL。

* \* mscorlib.dll
* \* Boo.Lang.dll
* \* UnityScript.Lang.dll

\* UnityEngine.dll

When building a player, you should avoid any dependencies on *System.dll* or *System.Xml.dll*. Unity does not include these in the built player by default but if you use their classes then they will get included. These DLLs will add about a megabyte to the player’s storage size. If you need to parse XML in your game, you can use a library like [Mono.Xml.zip](http://docs.unity3d.com/540/Documentation/uploads/Examples/Mono.Xml.zip) as a smaller alternative to the system libraries. While most Generic containers are contained in mscorlib, Stack<> and few others are in System.dll, so you should avoid these if possible.

当你生成终端时，你应该避免对*System.dll* 或者 *System.Xml.dll* 的依赖。Unity默认生成终端时不会引入这些但如果你在代码中使用了它们的类就会引入它们。这些DLL会增加终端大小大约1mb。如果你希望在游戏中解析XML，你可以使用类似[Mono.Xml.zip](http://docs.unity3d.com/540/Documentation/uploads/Examples/Mono.Xml.zip)的轻量级类库而不是System中的库。Mscorlib中已经包含了最通用的容器，栈和其他则在System.dll中，你应该尽可能的避免它们。



As you can see, Unity is including System.Xml.dll and System.dll, when building a player

你可以看到，Unity在生成终端时导入了System.Xml.dll 和 System.dll

**Reducing Mobile .NET Library Size**

**降低移动端.NET类库的大小**

Unity supports two .NET API compatibility levels for some mobile devices: .NET 2.0 and a subset of .NET 2.0. You can select the appropriate level for your build in the [Player Settings](http://docs.unity3d.com/540/Documentation/Manual/class-PlayerSettings.html).

Unity为了一些移动设备支持了两种.NET API兼容级别：.NET 2.0和 .NET 2.0子集，你可以在[Player Settings](http://docs.unity3d.com/540/Documentation/Manual/class-PlayerSettings.html)中选择生成的合适级别。

The .NET 2.0 API profile is similar to the full .NET 2.0 API. Most library routines are fully implemented and so this option offers the best compatibility with pre-existing .NET code. However, for many games, the full library is not needed and the superfluous code takes up valuable memory space.

.NET 2.0 API 简介和完整的.NET 2.0 API类似。大部分库程序是完全实现了的，所以这个选项为现有的.NET代码提供了最好的兼容性。然而，对于很多游戏来说，完整的库并不是必须的，多余的代码占用了宝贵的内存空间。

To avoid wasted memory, Unity also supports the .NET 2.0 Subset API profile. This is very similar to the Mono “monotouch” profile, so many limitations of the “monotouch” profile also apply to Unity’s .NET 2.0 Subset profile. (More information on the limitations of the “monotouch” profile can be found [here](http://www.mono-project.com/archived/monotouchlimitations/)). Many library routines that are not commonly needed in games are left out of this profile in order to save memory. However, this also means that code with dependencies on those routines will not work correctly. This option can be a useful optimization but you should check that existing code still works after it is applied.

为了避免内存的浪费，Unity也支持.NET 2.0 Subset API 。这和Mono的“monotouch”类似，所以对于“monotouch”的一些限制在Unity’s .NET 2.0 Subset中也有。（更多关于“monotouch” 的信息在这里查看）。许多在游戏中并不需要的库程序在摘要中被剔除以节省内存。然而，这也意味着依赖于这些程序的代码无法正常的执行。这个选项是一个有效的优化项但是你应该对现有的代码进行检查确保他们在执行优化后依然能够正确执行。