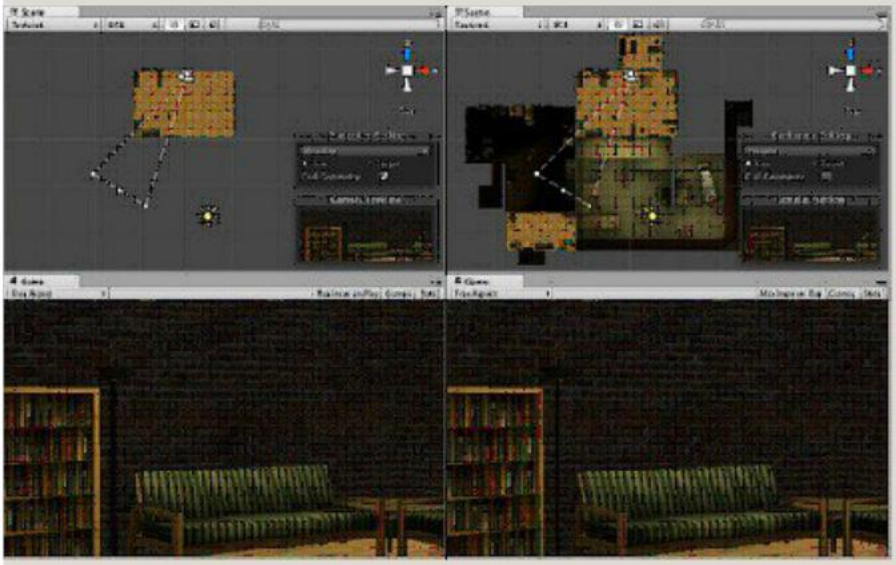
首先介绍下 draw call（这个东西越少你的游戏跑的越快）：
在游戏中每一个被展示的独立的部分都被放在了一个特别的包中，我们称之为“描绘指令”（draw call），然后这个包传递到 3D 部分在屏幕上呈现出来。这就和你希望你的亲友收到准备好的圣诞礼物需要包装好然后穿过城市准时放在他应该出现的地方一样没什么不同。你的 CPU 来完成包装和传递他们的活，同时会消耗很多的带宽，所以最终分配好这些关键性资源很重要。目前，真正可怕的事情是从描绘指令消耗远景开始，每一个独立的飞溅到地板上的血迹和一个角色或者一具死尸消耗的字节是一样的多的：他们都消耗同样的描绘指令。除此之外，没有什么更多的差别。
那么如何降低 draw call 呢？？那么我们就用到 Culling(剔除）技术。如果不应用这个技术，电脑是不管 3721 把场景里所有的东西都送去渲染的。看得见的也渲染，看不见得照样也送去渲染。很傻是吧，那咋办呢。得告诉电脑，那个你
看得见的渲染，看不见的就算了。于是就有了
1.视锥体剔除（Frustum Culling）这个 unity 系统自带了好像，就不用操心了。
2.遮挡剔除（Occlusion Culling）

Unity 3 专业版内置了一个强大的 Occlusion Culling 插件 Umbra 免费的
遮挡剔除（Occlusion Culling） 遮挡剔除是一种什么样的特性呢， 当一个物体被其他物体遮挡住而不在摄像机的可视范围内时不对其进行渲染。. 遮挡剔除在 3D 图形计算中并不是自动进行的。因为在绝大多数情况下离 camera 最远的物体首先被渲染，靠近摄像机的物体后渲染并覆盖先前渲染的物体(这被称为重复渲染,无效渲染"overdraw"). 遮挡剔除不同于视锥体剔除. 视锥体剔除只是不渲染摄像机视角范围外的物体而对于被其他物体遮挡但依然在视角范围内的物体则不包括在内. 注意当你使用遮挡剔除时你依然受益于视锥体剔除（Frustum Culling）.



左边的场景使用了遮挡剔除. 右边的场景未使用遮挡剔除.

遮挡剔除的运行将通过在场景中使用一个虚拟的摄像机来创建一个物体潜在可视性状态（set）的层级. 这些数据可以让每个运行时间内的摄像机来确定什么能看见什么看不见。通过这些数据, Unity 将确定只把可以看见的物体送去渲染. 这将降低 draw calls 的数量并提供游戏的运行效率.

occlusion culling 的数据由单元（cells）组成. 每个单元是整个场景范围数值的一部分. 更多特定的单元来自一个二叉树（ binary tree）. Occlusion Culling 使用两个叉, 一个给 View Cells (静态物体) 另一个给

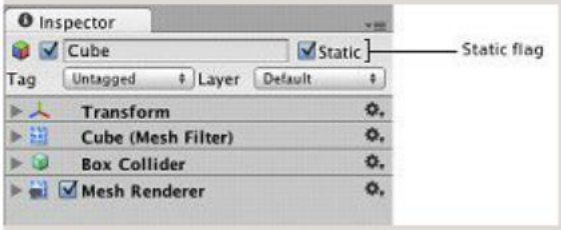
Target Cells (移动物体). View Cells map 给出了一个定义了静态可视物体的索引列表 （精确剔除后的静态物体）.

非常重要的一点是在创建你的物体时要随时注意，因为你需要在物体的尺寸和单元的尺寸间取得一个好的平衡. 理想情况下, you shouldn't have cells that are too small in comparison with your objects but equally you shouldn't have objects that cover many cells.有时你可以通过将大的物体拆成几个部分来改进遮挡剔除效果. 无论如何你仍然能够将小的物体合并为一体来降低 draw calls, 在它们都属于一些小的组件的时候, occlusion culling 将不起作用.确定组件中那个是可视的组件的选集和可视信息被认为是 PVS (潜在可视状态 Potentially Visible Set).

Occlusion Culling 设置

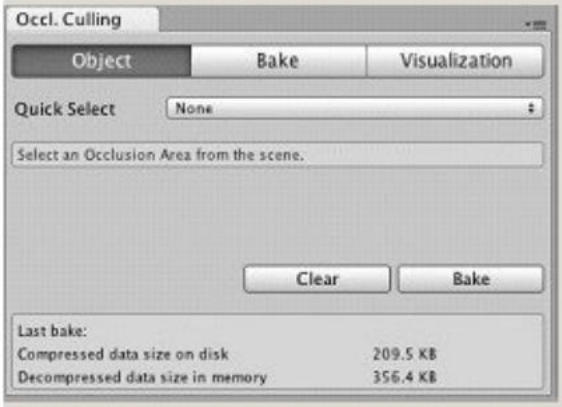
为了使用遮挡剔除 需要进行相关的手动设置.首先 你关卡中的几何体必须被分割成明显的不同尺寸的块. 这也有助于布置关卡中小块的容易定义的区域 被其他大物体遮挡（例如墙,建筑物） 这意味着每个单独的 mesh 被确定是否渲染取决于 occlusion 数据. 所以如果你有一个物体包含了房间里的所有家具那么所有的家具要么全渲染，要么全不渲染. 是否渲染是基于组件而不是基于每个单独物体自身的。

在检视面板（Inspector） 你需要标识（tag） 所有需要应用遮挡剔除的场景物体。最快的方法是是将你需要标示为 Static 的物体作为一个 Empty GameObject 的子物体并设置这个 Empty GameObject 为 Static, 当 option 出现的时候选择 affect children. 当子物体被 tagged as Static 你可以取消子物体和 Empty GameObject 的父子关系.



检视面板中的 Static checkbox
下一步点击 Tools->Occlusion Culling. 打开 Occlusion Culling 检视面板，在面板中你会发现几个数值被调整过了。这些数值前面描述过:-

Occlusion Culling - Object



遮挡剔除检视面板的 object 标签.

Object 标签可以让你创建 Occlusion Areas GameObjects. 这些区域让你指定什么地方你会使用遮挡剔除.

注意: 默认情况下如果你不创建任何遮挡剔除区域, 整个场景都会进行遮挡剔除.

Properties

Quick Select
快速选择和创建一个 Occlusion Area 并且快速编辑它.

None
这个是默认值.

Create New
创建一个新的 Occlusion Area GameObject.

当你点击 Create New in the Quick Select drop down, Unity 将自动创建一个 Occlusion Area 并且删除默认区域(整个场景). 现在你可以根据需要开始调整 Occlusion Area(s).

Occlusion Area 检视面板.

Properties

Size
定义 Occlusion Area 的尺寸.

Center
选择 Occlusion Area 的中心. B 默认为 0,0,0 并在 box 的中心位置.

Is View Volume
定义摄像机的活动范围区域. 点选这个选项才能应用遮挡剔除在 Occlusion Area 内的静态物体.

Is Target Volume
如果你要遮挡剔除运动物体，打开这个选项.

Target Resolution
确定区域内的 Occlusion Culling 精度. 即一个 Occlusion Area 的单元尺寸. 注意: 这个选项只对 Target Areas（移动物体）起作用.

Low
减少计算时间但同时精度降低.

Medium
计算时间和精度中等，比较平均.

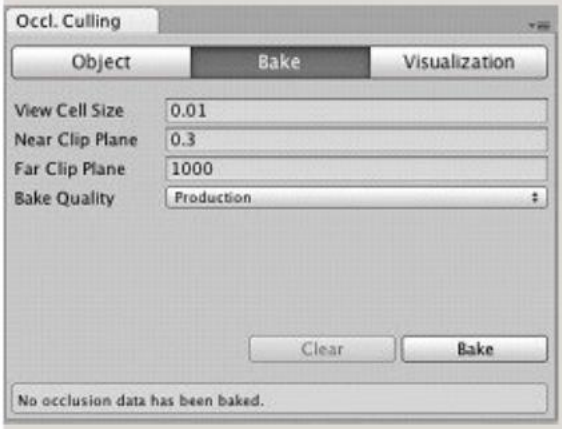
High
计算时间长但精度高.

Very High
精度很高，计算时间更长.

Extremely High
最高精度.注意: 计算时间令人发指。

如果摄像机处于遮挡区域之外或者任何物体超出区域, 这些物体将不会被遮挡剔除.

Occlusion Culling - Bake



Occlusion culling 检视面板 bake tab.

Properties

View Cell Size
每个 view area 单元的尺寸, 尺寸越小遮挡剔除越精确. 这个数值用来平衡遮挡剔除的精度和存储容量

Near Clip Plane
Near clip plane 如果设置最小数值 那么游戏中所有摄像机都可看到.

Far Clip Plane
Far Clip Plane 用于选择 objects. 任何物体的距离大于这个设定数值都会被自动遮挡.(Should be set to the largest far clip planed that will be used in the game of all the cameras)

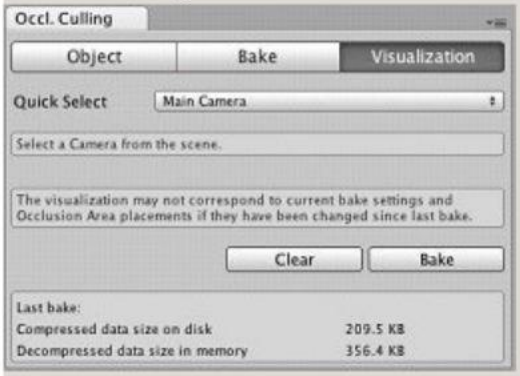
Quality
品质级别

Preview
开发阶段使用这个 (ie,不是很准确但可以让你御览在游戏中的大致表现)

Production
如果开发基本结束准备发布那就该选择这个(会耗费更多的时间但更精确)

当你调整完这些数值后可以点击 Bake 按钮开始处理 Occlusion Culling 数据. 如果你对结果不太满意,你可以点击 Clear 按钮删除以前的计算数据.

Occlusion Culling - Visualization



Occlusion culling 检视面板 visualization 可视 标签

Properties

Quick Select
让你快速选择场景中的任何摄像机来观看遮挡剔除的效果

The near and far planes 定义了一个虚拟摄像机来计算遮挡剔除数据. 如果你有几个摄像机 near 或 far planes 不同, 你应该设置 near plane 和 largest far plane distance 适配所有摄像机来调整物体的包括范围.

所有场景里的物体只在数值范围内起作用所有请确定你的所有物体都处于可视范围.

当你准备好生成 occlusion data, 点击 Bake 按钮. 记住在 Bake 标签的 Quality selection box 中事先选择 Preview 和 Production. Preview 可以快速生成数据并可以快速检查结果. Production 用于产品发布前的生成和检测.

请记住遮挡剔除的数据计算生成速度取决于事先设置的 cell levels（单元级别）, 数据尺寸大小和精度. Unity 会在窗口底部显示 PVS 运算状态.

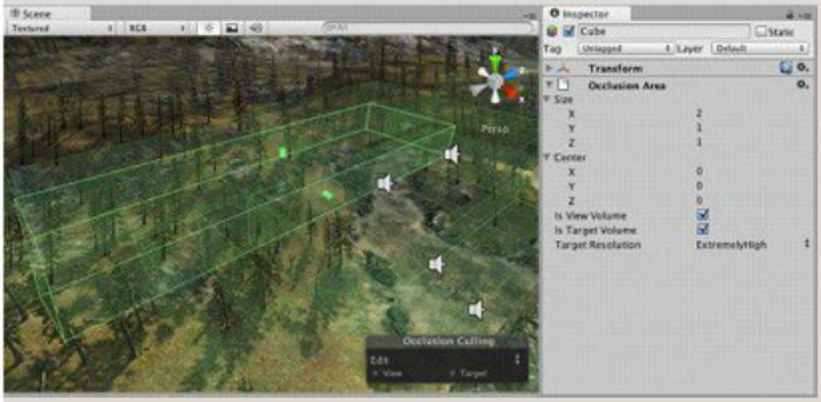
运算处理结束后, 你会在 View Area 看到一些不同颜色的 cube. 颜色相同的区域共享遮挡剔除数据.

点击 Clear 如果你想删除所有遮挡剔除的事先计算好的数据（预计算数据）.

Moving Objects

如果想应用遮挡剔除到一个运动物体你必须创建一个 Occlusion Area 然后设定其尺寸来适配运动物体的活动空间(注意：运动物体不能被标示为 static).

创建 Occlusion Area 后, 检查 Is Target Volume checkbox 来遮挡剔除运动物体.



移动物体的 Occlusion Area properties

Size

设定 Occlusion Area 尺寸.

Center

设定 Occlusion Area 的中心. 默认 0,0,0 并位于 box 的中心.

Is View Volume

定义摄像机能到哪里. 检查这个数值来遮挡剔除 Occlusion Area 中的 static objects .

Is Target Volume

遮挡剔除运动物体时必选

Target Resolution

确定区域内的遮挡剔除精度。这将决定 Occlusion Area 的单元尺寸. 注意: 只对 Target Areas 起作用.

Low

减少计算时间但同时精度降低.

Medium

计算时间和精度中等，比较平均.

High

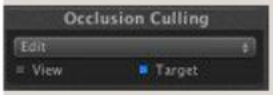
计算时间长但精度高.

Very High

精度很高，计算时间更长.

Extremely High
最高精度.注意: 计算时间令人发指。

添加完 Occlusion Area 后, 你需要了解它是如何划分单元中的 box. 如果想了解 occlusion area 如何计算运动物体,你必须在 Scene View 中点选 Target 按钮 并且 关闭 View 按钮同时 Occlusion Culling 检视面板是打开的.



Selecting Target (moving objects) or View (static objects) 让你御览 calculated data 的运作.

Testing the generated occlusion

occlusion 设置完毕后, 打开 Cull Geometry option (in the Occlusion Culling window) 并在场景视窗移动 Main Camera.

iScene View 中的 Occlusion View mode

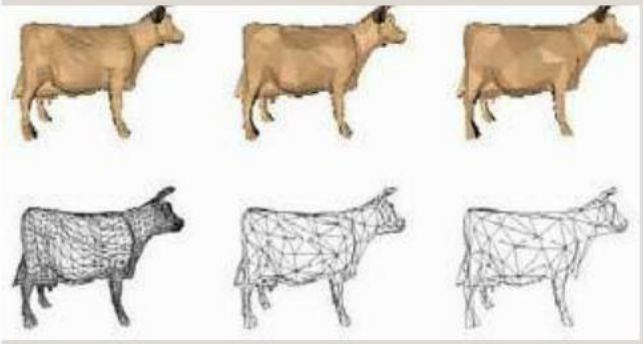
当你在周围移动 Main Camera (无论是否在 Play mode 下), 你将会看到不同的物体 disable. 你在这里需要找出的是 occlusion data 中的任何错误. 当你移动摄像机时你可能会发现会有物体突然出现在视野当中. 如果这种情况发生, your options for fixing the error are either to change the resolution (if you are playing with target volumes), or to move objects around to cover up the error. To debug problems with occlusion, you can move the Main Camera to the problematic position for spot-checking.

运算处理结束后, 你会在 View Area 看到一些不同颜色的 cube. 蓝色 cubes 表现的是 Target Volumes 的单元划分. 白色 cubes 表现的是 View Volumes 的单元划分.如果参数设置正确你会看到一些物体不被渲染. 这表示要么这些物体不在摄像机视角范围内要么被其他物体遮挡住了.

如果 occlusion 完成后, 场景内任何物体也没有被遮挡, 拆分物体至更小的 pieces 以便它们能被完整地包含在一个单元中.
要提醒朋友们的是，如果场景复杂，品质设置高的话，烘焙过程将漫长的令人发指。赶紧升级电脑吧！！！

对于没有买专业版并无法使用 umbra 插件的朋友也不必担心，还有另外一个 Culling 剔除插件可以选择，是完全免费共享的。具体名字忘记了，不过去 unity 网站的讨论版搜一下 Culling 关键词可以很容易找见。就到这里 .就到这里.转自：http://blog.sina.com.cn/s/blog_409cc4b00100oivo.html

LOD(Level-of-detail)是最常用的游戏优化技术。如果你的程序可以定制开发应用 LOD 的模块，当然是很美好的事情。不过如果没有也没关系，大家可以使用 UniLOD 这个第三方的 LOD 插件。免费共享的哦（向 UniLOD 开发者致敬,赞美伟大的共享精神！！！）



以下是简介：

功能众多，涉及到场景管理，模型优化，资源管理员，及显示效果变化，且不需要脚本编写，全部通过编辑器实现！目前此 unity3d 扩展为开源的，你可以下载到源代码来研究学习。

特色：

·1.自动简化网格体(Windows + Unity Pro only)

根据用户的设定自动减面。

· 2.Level-of-detail 编辑器

方便快捷的创建 LOD 组件来切换不同的品质级别或通过设定距离数值来显示或隐藏物体。

创建你的 LOD 场景

方便快捷的用 LOD 组件来替换现有组件将现有的场景转换为拥有 LOD 管理的场景。

将单个的 mesh 和贴图成组来降低 draw calls

将场景存储为 标准 assets 或者 asset 包

· 3.场景管理

Streaming 支持

Stream your scene as the player moves through the world

资源自动 loaded/unloaded

Stream from resource folder, or asset bundles

Minimal performance impact with resource buffering

可以根据需要调整, 速度或者内存

Takes care of switching quality levels as the player moves though the world

充分优化, 使用最少的资源

· 4.资源管理

所有 asset bundles 和普通资源都以标准方式载入。

Keeps reference counts on your resources and 自动释放

· 5.完整植入 Unity 并只需要通过 UnityEditor API 来操作。

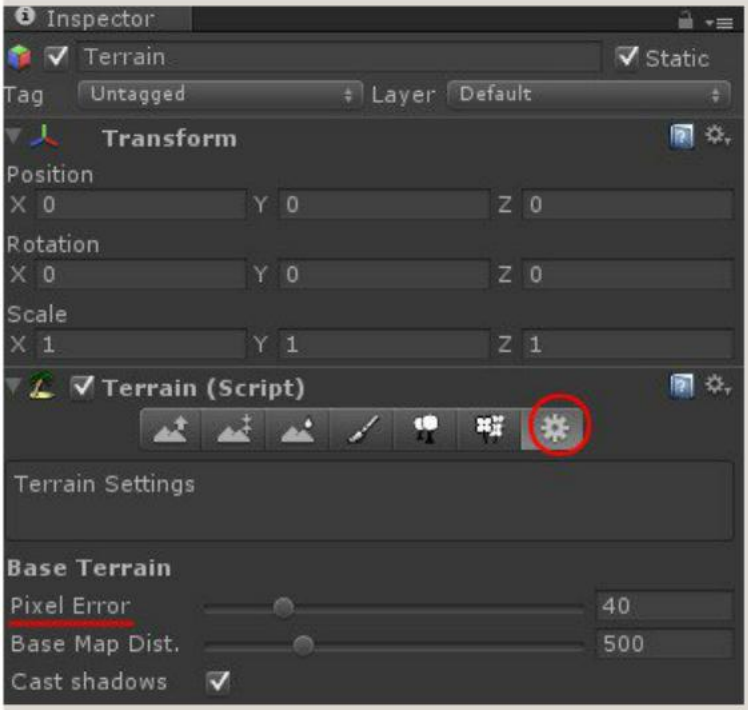
无需输入代码

Sorry,场景管理和资源管理的有些内容没搞懂就不乱翻译了。

链接：

http://unity3d8.com/content/lod 扩展 beta 版本发布 unilod-beta-levelofdetail-and-streaming-support

当然，Unity3d 自己的地形是自带 LOD 功能的，当你刷好你的地形后，你只需要调整 Pixel Error 这个参数即可。Unity3d 会自动计算生成地形的 LOD，无需你做其他的任何设置。



要提醒朋友们的是，如果你使用了 Lightmap,那么同时使用 LOD 的时候会有一些麻烦，我们的办法

是制作模型的时候事先做好第二套 uv(不使用 Unity 的自动计算 lightmapUV 功能），而且所有 LOD 的第二套 UV 的分布位置都一致，很费工。不知道有没有更好的方法！！！

转自：http://blog.sina.com.cn/s/blog_409cc4b00100o6qu.html

动态实时灯光相比静态灯光，非常耗费资源。所以除了能动的角色和物体（比如可以被打的到处乱飞的油桶）静态的地形和建筑，通通使用 Lightmap。

强大的 Unity 内置了一个强大的光照图烘焙工具 Beast，这个东东是 Autodesk 公司的产品（可怕的垄断，感觉和 3d 沾边的软件丫都要插一手）。据说用来制作过杀戮地带和镜之边缘。

镜之边缘建筑场景漂亮干净的光影，Lightmap 的效果。

在 Unity 中制作 Lightmap 很方便，调节几个参数后直接烘焙即可。支持 GI，Skylight，

效果一流！！！当然你需要一台好点的机器，不然漫长的烘焙过程你就有的等了。

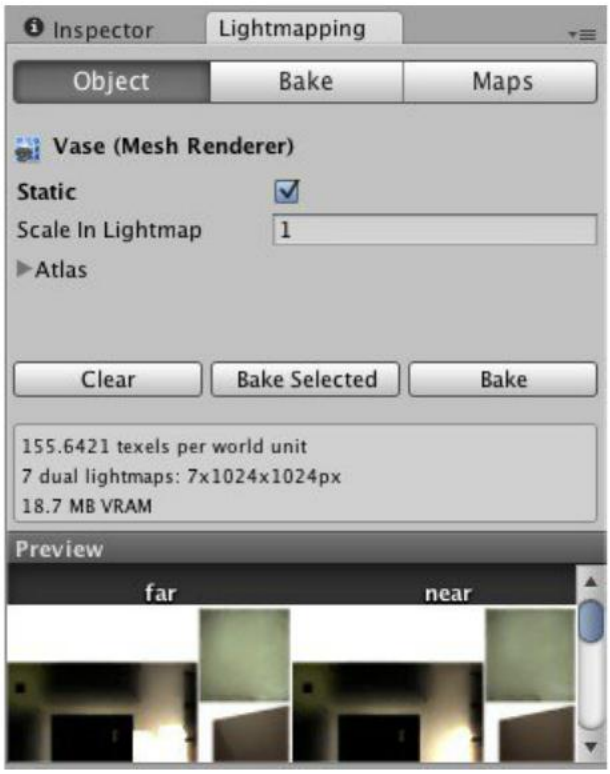
内置的光照图烘焙工具 Beast

P 场景准备和光照图烘焙

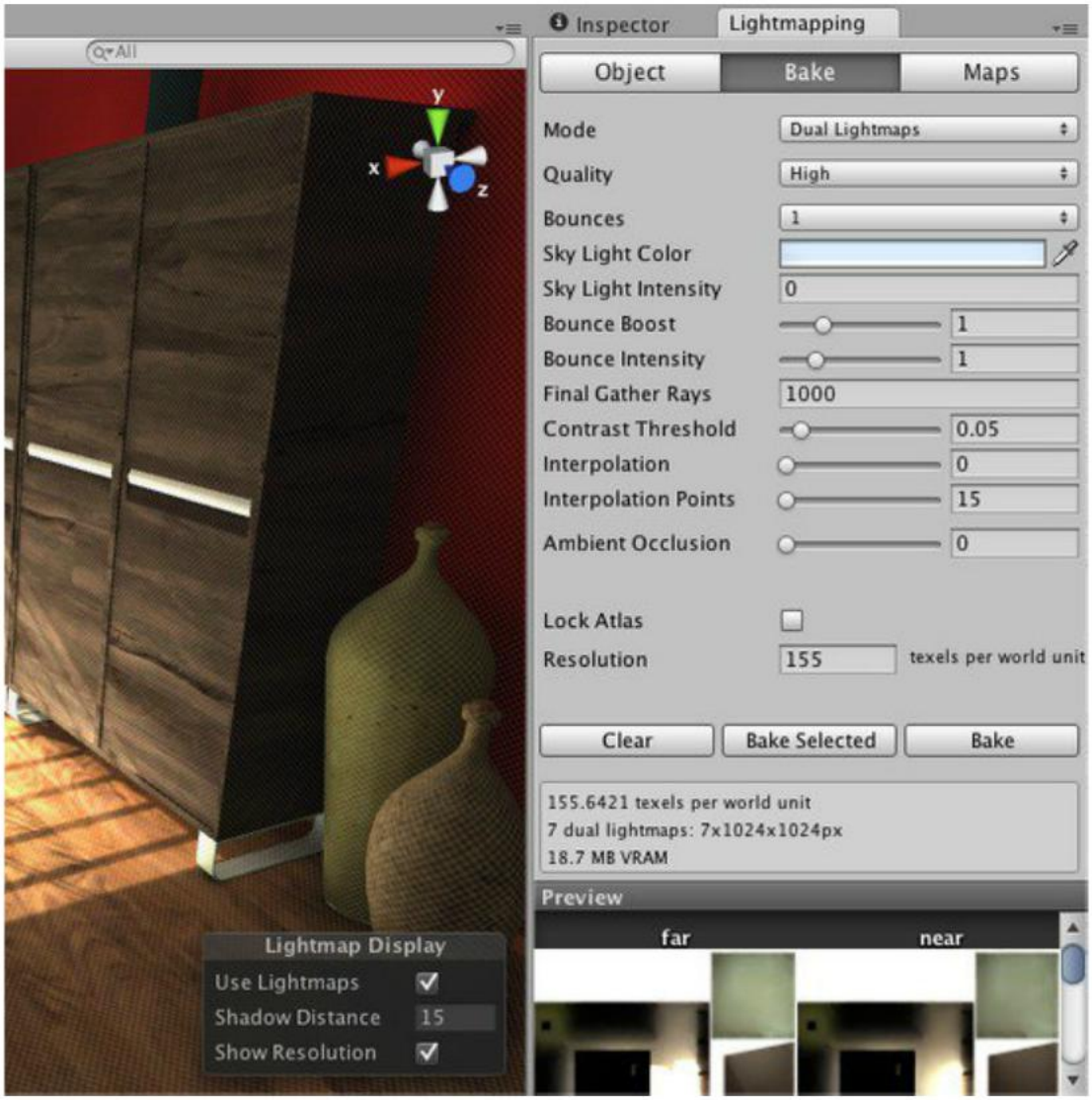
点选 **Window --> Lightmapping** 打开光照图烘焙面板:

1. 确认所有将要被用来烘焙光照贴图的网格体 UVs 正确无误. 最简单的办法是在 mesh import settings 中选择 Generate Lightmap UVs 选项（由 Beast 自动分 uv）

2.在 Object 面板中将所有网格体或地形标注为 static – 这将告诉 Unity, 这些物体将不会被移动和改变并且可以被赋予光照贴图。



3.为了控制光照贴图的精度, 进入 **Bake** 面板 并调整 **Resolution** 的值. (为了更好的了解你的 lightmap texels 使用情况, 在 **Scene** 视窗中找到 **Lightmap Display** 小窗口并且选择 **Show Resolution**).



1. 点击 Bake 按钮。
2. Unity Editor's 会出现一个进度条,位置处于右下角.
3. 当烘焙结束, Lightmap Editor 窗口会显示已经烘焙好的光照图.

Scene 和 game 视图会同时自动更新 - 现在你的场景已经有了光照图的效果!

Unity Lightmap 的设置还有更详细和更高端的内容，请参考自带的文档，那才是王道啊！！！！！

最近制作的一个野外场景快完工了。阿弥托福！！！！希望能跑的流畅，千万别返工啊！！！

为了避免出现杯具，提前研究了一些资料，涉及到Lod技术 (Levels of Detail，多细节层次)，

选择剔除(Culling)，光照贴图（Lightmap)当然还有强大而又脆弱的 Unity3d 引擎.

所有资料都来自互联网和 Unity 自带的文档。有些文档我自己业余时间翻译了下，错误的地方请朋友们指正。