一、Unity基础

本部分是你开始 Unity的关键。这里将解释Unity的界面，菜单项，使用资源，创建场景，和发布。当你完全阅读了该部分后，你将能够理解
Unity是如何工作的，以及如何使其更加有效的工作，和如何将简单的游戏放置在一起。

1. 界面(Learning the interface)

现在我们开始学习Unity，如果你还没有打开 Unity,你可以通过双击位于 **Application-**

>Unity文件夹中的 Unity图标来运行它，当它第一次运行时你将看到如下的场景：

*Unity*运行时的缺省场景，如果你打开过任何实例，你的屏幕会与上图不同

有很多需要学习的东西，让我们花费点时间来观察理解上述界面。我们将介绍每一个接口元素。

概要主窗口的每一个部分都被称为视图**(View)**。在
Unity中有多种类型的视图，但是，你不需要同时看见所有的视图。不同的布局模式

(Layout modes) 包含的视图是不同的。 通过单击布局下拉控件来选择不同的布局，该控件位于窗口的右上角。

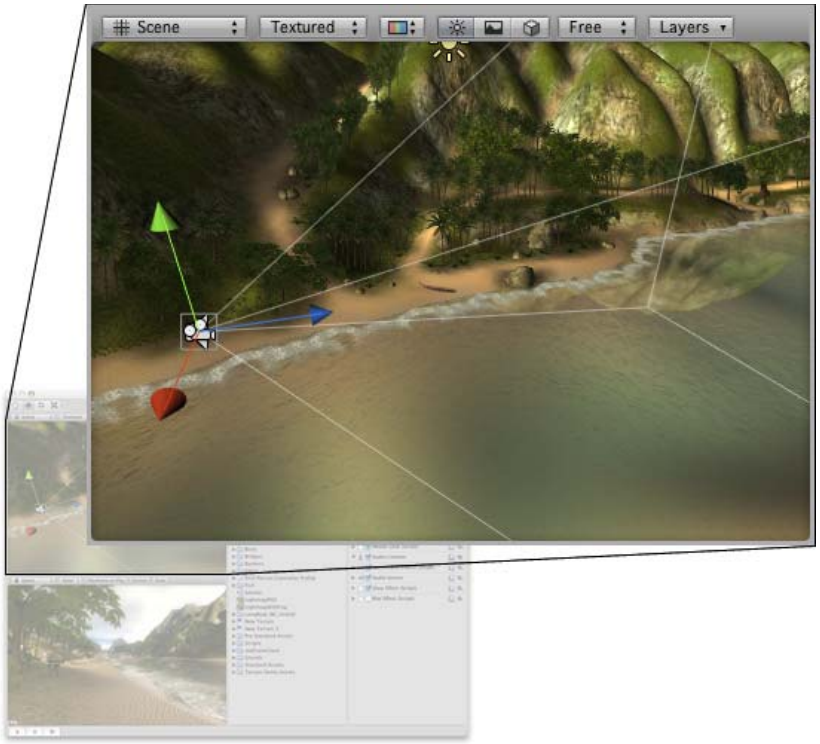
布局模式选择下拉列表

现在，单击布局选择，并单击 **Animation**，切换到动画布局 (Animation layout)。还可以从菜单中选择 **Window->Layouts->Animation**来切换。动画布局包含所有的视图，这是昀好的用来介绍它们的方法。

通过视图左上角的名称你可以迅速的分辨这些视图。这些视图是：场景视图**(Scene View)**-
用于放置物体游戏视图**(Game View)**-表示游戏在运行时的外观层次视图**(Hierarchy)**-
当前场景中的游戏物体的列表工程视图**(Project)**-
显示当前打开工程中所有可用的物体和资源检视视图**(Instpector)**-
显示当前选中物体的细节和属性时间线**(Timeline)**-
用于为当前选中物体创建基本的时间线动画

场景视图**(Scene View)**

场景视图

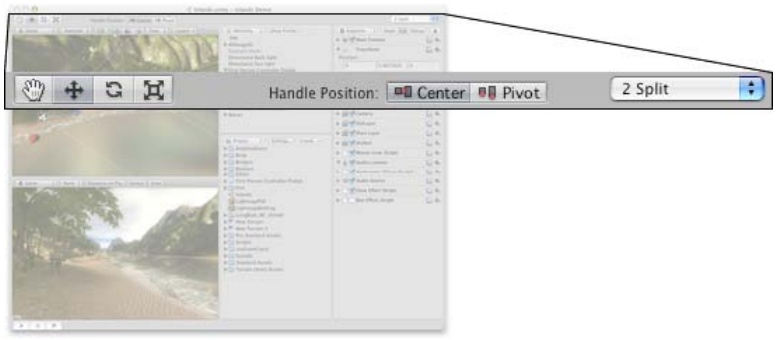


场景视图**(Scene View)**是一个可交互的沙盘。你将使用它来选择并在场景中定位所有的游戏物体**(GameObjects)**，包括玩家，摄像机，敌人等。在场景视图中操纵并修改物体是Unity非常重要的功能。这是昀好的通过设计者而不是玩家的角度来查看场景的方法。在场景视图中你可以随意移动并操纵物体，但是你应该知道一些基本的命令以便有效的使用场景视图。

第一个你应该知道命令是**Frame Selected**命令。 这个命令将居中显示你当前选中的物体。你可以在层次视图**(Hierarchy)**单击任何物体，然后移动你的鼠标到场景视图上并按**F**键。场景视图将移动以居中显示当前选择的物体。这个命令是非常有用的，你将在场景编辑的时候经常使用它。

在场景视图中操作在场景视图的上方有一个包含布局模式选择的工具栏

工具栏



尽管现在的工具栏没有附着在场景视图窗口上，但是位于左侧的四个按钮可用来在场景视图中导航并操纵物体，中间的两个用来控制选中的物体轴心如何显示。左边的第一个 **View Tool**将在以后说明。后面的工具为操纵工具**(Manipulation Tools),**中间的两个为手柄位置工具 **(Handle Position Tool)**

选中任何操纵工具可允许你交互时的移动，旋转或缩放物体。当你已经选择了一个工具时你可以在场景视图中单击任何一个物体选中它，现在按下 **F**键使得该物体居中显示。

如下不同的操纵工具

平移工具热键*W*

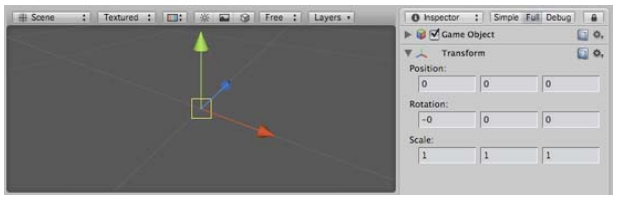
旋转工具热键*E*

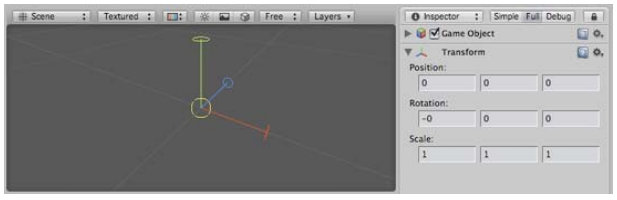
缩放工具热键*R*

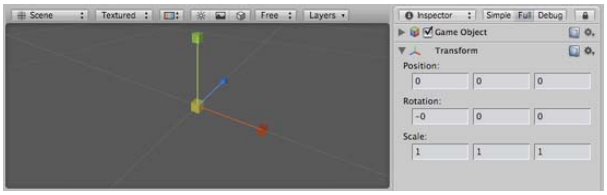
当选中一个物体时你将看到 **Gizmo**坐标，每个工具有不同的Gizmo坐标形式

平移

旋转缩放







点击并拖动当前Gizmo坐标的任何一个坐标轴以便平移，旋转或缩放当前选中物体的变换**(Transform)**组件。你也可以通过单击并拖动Gizmo坐标的中心来在多个轴上操纵物体。如果你有一个三键的鼠标，你可以通过单击中键来调整昀后调整的轴而不用直接点击它。

参考变换组件(Transform Component)部分获取更多内容。

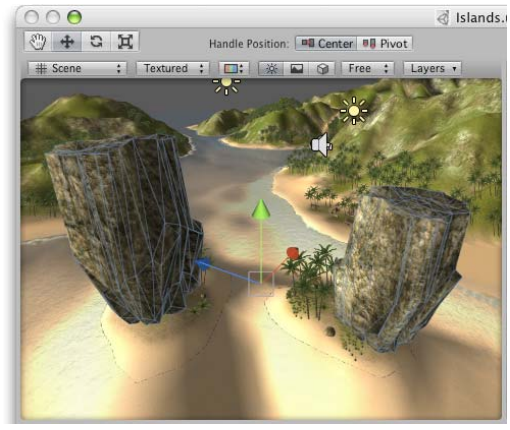
手柄位置工具 **(HandlePositionTool)**

用来控制物体或一组选中的物体的轴心如何和在哪里显示。

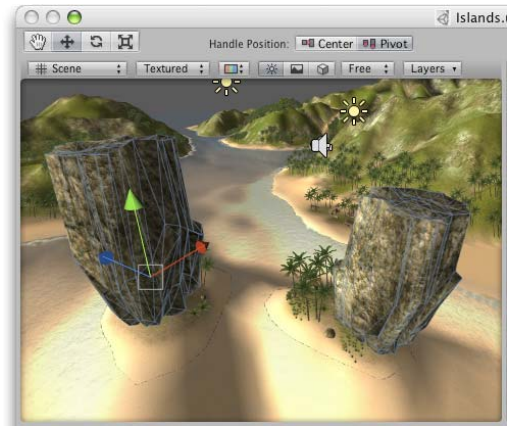
手柄位置工具

选择中心(Center)意味着使用当前所选所有物体的共同轴心，选择轴心 (Pivot)意味着将使用各个物体的实际轴心

手柄位置设置为中心，使用物体的共同轴心



手柄位置设置为轴心,使用实际的物体轴心



在场景视图中导航根据使用的鼠标的不同，有很多不同的方式可以在场景视图中导航。

使用三键鼠标按住 Option按钮并拖动鼠标左键可以使用旋转模式 **(Orbit mode)**按住 Option按钮并拖动鼠标中键可以使用拖动模式 **(Drag mode)**按住 Option按钮并拖动鼠标右键可以使用缩放模式 **(Zoom mode)**。也可以使用滚轮来缩放（略）视图工具模式

视图工具的拖动模式快捷键*Q*

在拖动模式**(Drag Mode)**下，在场景视图中单击并拖动鼠标来上下左右移动视图。旋转(Orbit)和缩放(Zoom Modes) 模式也是昀常用的视图工具。保持视图工具选中并按住Option键即可进入旋转模式。单击并拖动鼠标，可以看到视图是如何旋转的。同时注意视图工具按钮从手型变成了眼睛。

视图工具的旋转模式 *Option*键

昀后，你可以通过按下

Command按钮进入缩放模式。在这种模式下，单击并拖动鼠标将前后缩放你的视图。注意缩放模式的图标是一个放大镜。

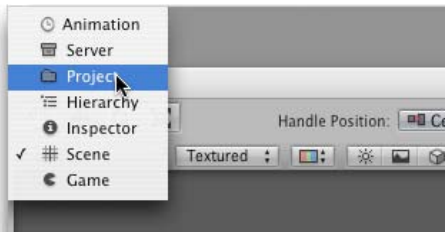
视图工具的缩放模式 *Command*键

使用视图工具模式并拖动鼠标是基本的场景视图导航方法。

场景视图控制栏所有的视图的顶部都有不同的控制栏 **(Control Bar)**，场景视图控制栏拥有多数选项，并且看起来像下面的样子：

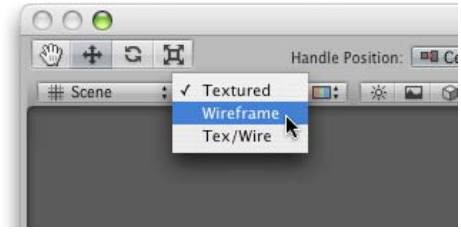
场景视图控制栏

第一个下拉菜单为视图选择器。展开它你可以改变当前视图。所有的视图都有这个选择器，如果你想创建自定义的一个界面布局，它是非常有用的



每个视图都有的视图选择器

下一个下拉菜单是绘制模式 **(Draw Mode)**。你可以选择场景视图使用纹理模式，线框模式或者纹理线框模式。这个对你发布游戏是没有任何影响的。

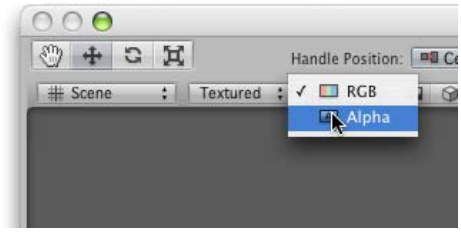


绘制模式下拉框

第三个下拉菜单为渲染模式(Render Mode)。你可以选择使用 RGB模式或Alpha模式。同样这将不会影响游戏发布。

渲染模式下拉框

控制栏中的下一项是一组三键。

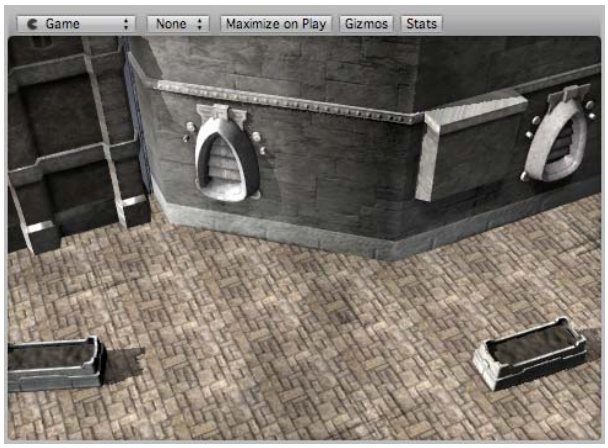


视图控制栏中的三键

左边的开关控制普通光照。当该按钮被禁用时，你将看到整个场景中简单光照。当它被启用时，你将看到你放在场景中的光照物体的影响。启用该按钮将允许你在发布游戏时看到游戏中的光照。

中间的按钮控制各种不同效果的开关，例如场景视图网格 **(Scene View Grid)**，天空盒 **(Skyboxes)**和 **GUI**元素**(GUI Elements)**，启用该按钮将允许你在发布看到这些效果

右边的按钮控制正交模式 **(Orthographic Mode)**。打开它将移出所有的景深效果。该按钮不会影响到你发布游戏。正交模式用来精确定位物体。



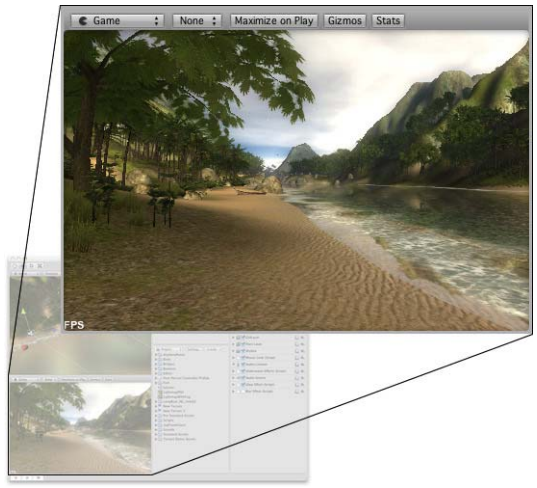
景深相机

正交相机。物体不会随着距离的增大而变小

下一个下拉列表是方向**(Direction)**下拉列表。它将移动场景视图到你选择的方向。

旳后一个下拉列表是层**(Layer)**下拉列表。你可以使用它来选择不同层的物体。参考层部分。该选项不会影响游戏发布。游戏视图

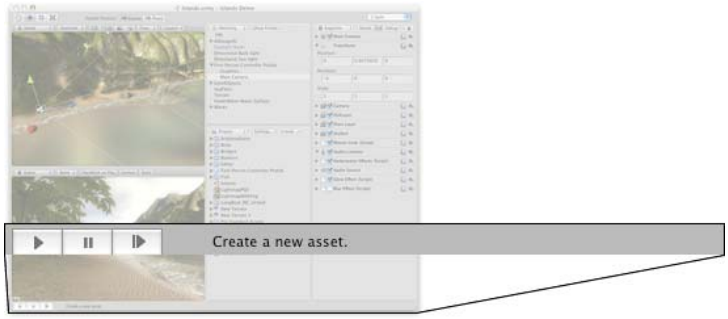
游戏视图-你的游戏的可见部分



游戏视图**(Game View)**将使用游戏中设置的相机信息来渲染。这个视图显示的是游戏运行过程中你将看到的场景。如果你平移或者旋转场景的主相机，你将看到游戏视图的变化。

你需要使用一个或多个相机**(Cameras)**来控制玩家在游戏中实际看到的场景。参考相机组件部分。

播放按钮和状态栏这个按钮用来在游戏视图中播放，暂停和步进你的游戏。在你构建场景的任何时候，你都可以进入播放模式 **(Play Mode)**并看看你的游戏是如何工作的。



播放按钮和状态栏

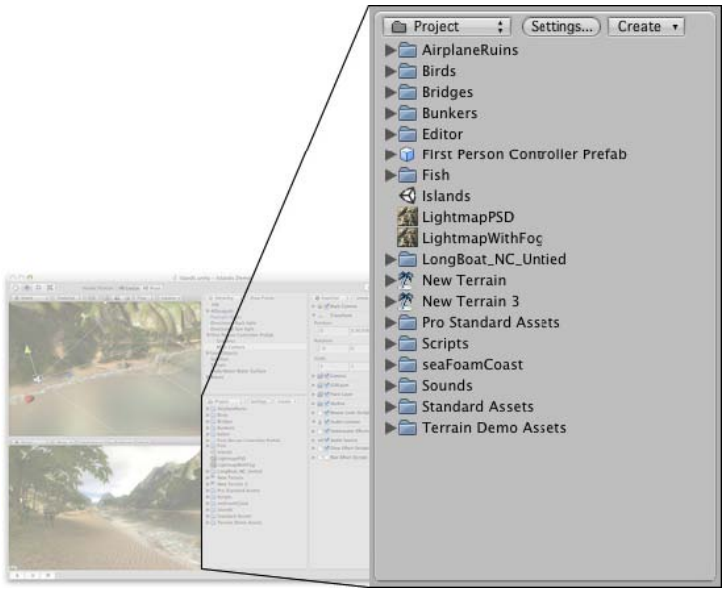
按下播放按钮 **(Play Button)**进入播放模式。当你的场景在播放模式下时，你还可以移动，旋转和删除物体。你也可以改变变量的设置。在播放模式下所做的任何改变都是暂时的，并在你退出播放模式时重置。你可以再次单击播放按钮退出。在播放模式下，你可以停止或步进你的游戏。暂停并检视你的场景是昀好的发现问题的方法。

右侧的状态栏有多种不同的作用。它将提供上下文敏感信息和提示，错误信息和来自与脚本的输出语句。如果你的游戏有任何问题，查看状态栏将是昀好的发现问题的方法。你可以双击状态栏打开控制台窗口，其中将显示所有的脚本或可见的运行时错误信息。

游戏视图控制栏控制栏上紧挨着视图下拉列表的是宽高下拉列表 **(Aspect Drop-down)**。这里，你可以指定游戏视图窗口的宽高比为不同的值。这将影响到GUI元素的位置。使用它来测试你的游戏在不同分辨率下的外观。

控制栏上昀右边的是 **Gizmos按钮(Gizmos Button)**。这将确定是否显示 Gizmos坐标

工程视图**(Project View)**



工程视图存储所有资源

当你创建一个工程时，将生成一组文件夹。其中之一被称为资源**(Assets)**文件夹。在工程视图**(Project View)**中可以查看资源文件夹。如果你打开过资源文件夹，你将发现所有的项都将出现在工程视图中。不同的是在工程视图中，你将创建并将物体连接在一起。这些关系将存储在工程文件夹的其他位置。从工程视图中移动资源将维持并更新文件之间的联系。从 Finder中移除资源将断开联系。因此，你应该只使用 Finder来将文件添加到资源文件夹。任何其他对资源的操作都应该在工程视图中进行。

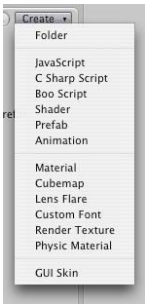
导入物体一旦你创建了资源（模型，图像，声音或者脚本），你可以使用 Finder将其正确地放置到资源文件夹下。当你做这些的时候 Unity可以处于打开状态。一旦你切换到 Unity，新的资源将被检测到并自动导入。资源就可以在工程视图中出现。

参考资源工作流部分。

创建资源在控制栏中使用创建下拉列表 **(Create Drop-down)**来创建你需要的物体。此外你还可以使用 Control+单击或右键在工程视图中单击打开相同的下拉列表。

创建下拉列表

组织工程视图



使用创建下拉列表在工程视图中创建文件夹。然后你可以重命名并使用该文件夹就像在Finder中一样，并可以在工程视图中将任何资源拖动到文件夹中。例如你可以创建名为**Scripts**的文件夹并将所有的脚本文件放置其中。

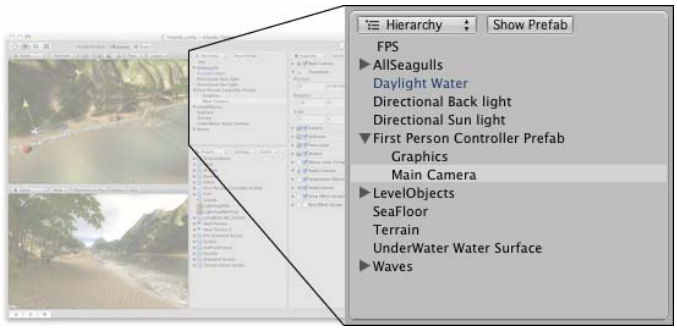
在你选中的文件上创建文件夹将创建嵌入式的文件夹。使用嵌入式的文件夹可以保持你的工程视图整洁。

注意：如果展开或折叠一个目录时按下了 Alt键，所有的子目录都将展开或折叠。

导入设置在控制栏上有一个导入设置按钮 **(Import Settings)**，位于创建下拉列表的旁边。根据所选资源的不同当该按钮被单击时将在导入设置弹出窗口中显示不同的选项。参考导入资源 (Importing Assets)部分。

工程视图控制栏设置(Settings)按钮将为当前选择的资源打开导入设置。创建下拉列表将会在你选择的目录下创建项目，创建文件夹是一种快速有效的组织你的工程视图的方法。层次**(Hierarchy)**

层次-当前场景中的所有物体



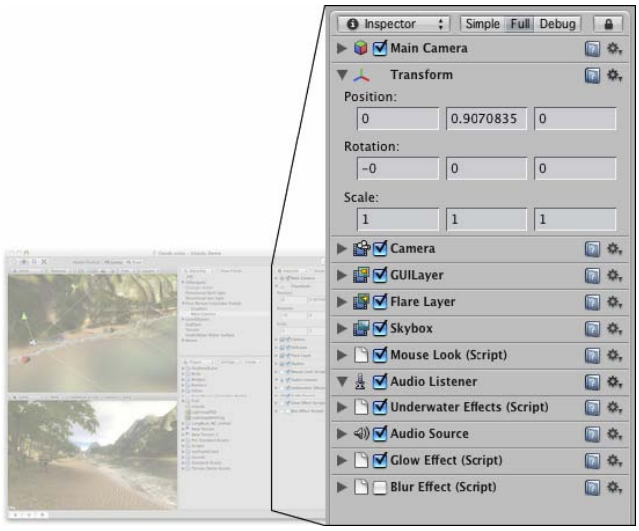
层次视图**(Hierarchy)**将显示当前打开的 .unity场景文件 **(Scene File)** 中的所有物体。它用于选择并成组物体。当从场景中添加或删除一个物体时，它将在层次中显示或消失。如果你不能在场景视图中同时看到所有物体，你可以使用层次来选择并检视它们。

物体层次

Unity使用一个称为父化**(Parenting)**的概念。任何物体都可以成为另一个物体的父或子。一个子物体可以从它的父物体继承移动和旋转。Parenting对于组织场景，角色，接口元素或者保持场景整洁有很大的用处。单击一个物体并将其拖动到另一个物体上可以建立父子关系。你将会看到一个三角显示在新的父物体的左边，现在你可以展开或折叠父以便在层次中查看他的子物体，而不会影响你的游戏。

显示预设按钮 **(Show Prefab Button)** 当位于控制栏 **(Control Bar)**上的该按钮被启用时，任何一个在层次中选中的预设**(Prefab)**实例将在工程视图中显示它的一个可视化的参考，如果你在场景中改变预置实例的名称，这是非常有用的。检视

检视-选中物体的细节

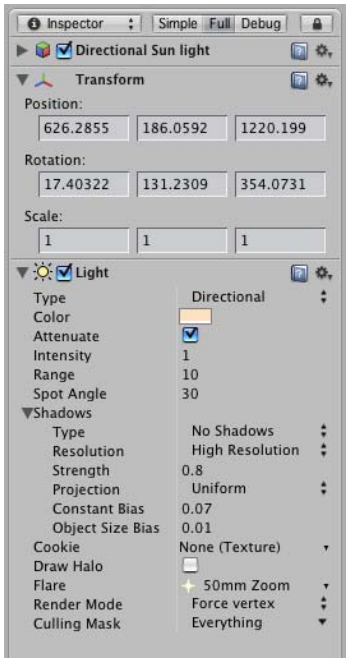


检视面板**(Inspector)**显示当前选中物体的基本信息，也显示它所包含的组件**(Component)**和组件的属性。它是用来设置场景中物体属性的地方。当创建一个好玩的游戏时，你将在检视面板上做大量的排错。

检视面板显示当前选中物体的基本信息和它的设置

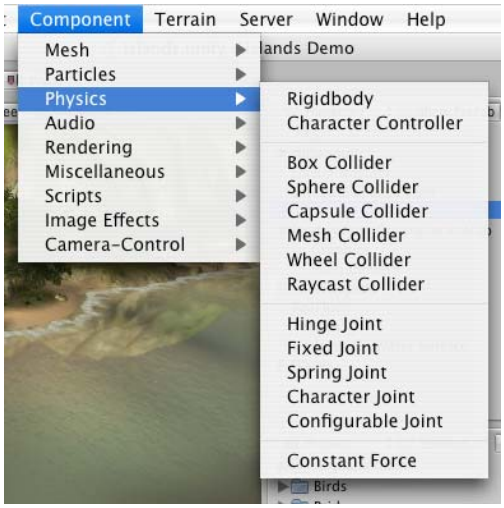
每一个物体都包含许多不同的组件。当你在检视面板中查看物体时，每一个组件都有它自己的旳小标题栏。例如，每一个物体都包含变换组件(Transform Component)。每个组件的参数和设置都可以在检视面板中修改。

物体结构在物体内部的组件将定义物体是什么以及做什么。将一个新的物体想成一个空的画布，并且每一个组件都是一个不同画笔。当你组合并设置不同的组件时，你就像在绘制你物体的行为。特定的组件，就像画笔不同的颜色，可以在一起工作的很好。然而其他的一些组件就不能一起工作。



可以通过使用组件**(Component)**菜单来向物体添加组件。

对于组件的详细信息可以参考组件部分此外，在检视面板中所有的组件都会在它们的名称旁边显示一个问号，单击这个问号可以打开该组件的参考文档。



时间线**(Timeline)**视图使用时间线(Timeline)视图可以为当前选中物体创建动画。Unity可以导入包含动画的文件，但是你可以使用时间线视图来制作基本的动画而无需使用 3D动作软件。

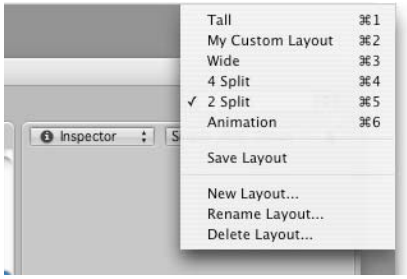
时间线视图将帮助你为物体制作动画

参考动画部分

调整视图布局现在你已经知道了所有不同的视图，你可以重新布局它们

布局下拉列表然你选择或保存不同视图布局

尝试选择不同的布局。现在，从下拉列表中选择新布局**(New Layout)**并给它一个唯一的名称。



为了自定义布局，你需要分割**(Split)**和组合**(Combine)**视图。Control-单击或右键在两个视图的分割线上单击，或者在任何视图的控制栏上。当鼠标变成一个分割线时，你可以单击并拖动鼠标来改变视图的大小。

一个完全的自定义布局

你还可以将任何视图切换为全屏模式。将你的鼠标移到视图上并按下空格键**(Spacebar)**，这将临时昀大化当前视图并隐藏所有其他视图。这将允许你在更大的屏幕尺寸上查看更多的细节。再次按下空格键可以切换到普通视图模式下。

2. 资源流程**(Asset Workflow)**

这里我们将解释在Unity中如何使用一个简单的资源。这些步骤是通用的而且可以看作是一个基本操作的演示。在该例子中我们将使用 3D网格。

创建原始资源使用任何 3D建模软件创建你的资源。在我们的例子中我们将使用Maya。导入当保存了你的资源后，你应该将其保存到你的工程文件夹的资源**(Assets)**文件夹中。当你打开 Unity工程，这些资源将被检测到并导入到工程中。当你查看工程视图**(Project View)**时，你将发现你保存的资源。导入设置如果你选择了一个资源并单击导入设置**(Import Setting)**按钮，将出现一个对话框，该对话框的选项随着导入资源的不同而不同。向场景中添加资源从工程视图中单击并拖动网格到层次**(Hierarchy)**或场景视图**(Scene View)**中即可将其添加到场景中。当你拖动一个网格到场景中时，你将创建一个拥有网格渲染组件 **(Mesh Render Component)**的物体。如果你导入的是纹理或声音文件，你需要将其添加到场景中已有的一个物体上。将不同的资源放置在一起下面是一些常用资源之间的关系纹理应用到材质**(Material)**材质应用到物体（带有渲染网格组件）动画**(Animation)**应用到物体（带有动画组件）声音应用到物体（带有声音源**(Audio Source)**组件）

创建预设**(Prefab)**
预设是可以在场景中重用的一组物体和组件的集合。几个相同的物体和通过同一个预设来创建，这些物体称为实例。例如，创建一棵树的预设将允许你在场景中不同的地方放置多个相同的实例。因为这些树都与预设相关，任何对预设的改变都将自动应用到所有树的实例上。因此如果你改变要改变网格，材质或其他任何东西，你只需要在预设中改变一次，那么所有的继承的实例树都将改变。你也可以改变一个实例并使用 **GameObject->Apply Changes to Prefab**将这种改变应用到所有相同的实例上。

当你有一个包含多个组件或子物体层次的物体时，你可以制作一个顶层(或根)物体的预设，并可重用整个物体集。

可以将预设看作是物体结构的蓝图。对于该蓝图来说所有的拷贝都是相同的。因此，如果蓝图被更新，那么它的所有实例也会相应更新。这里有几种不同的方式可以使你通过改变一个实例来改变整个蓝图。参考预设部分。

为了从你场景中的物体上创建一个预设，首先在工程视图中创建一个新的预设。并命名，然后在场景中单击你想用于创建预设的物体。拖动它到新的预设中，你将看到物体的名称变成了蓝色。这样你就创建了一个可以重用的预设。

更新资源你已经导入，实例化并将资源连接到了预设。现在当你需要编辑你的资源时，只要在工程视图中双击它，此时将运行属性应用程序，在这里你可以做任何你需要的改变。当你更新它时，保存它。然后但你切换到
Unity,这个更新将被检测到，并且资源将被重新导入。而资源到预设的连接还将存在。你将看到你的预设被更新了，这就是你需要知道的更新资源部分。仅仅需要打开和保存。

3. 创建场景(Creating Scenes)

场景包含所有的游戏物体。它们可以用来创建主菜单，不同的关卡，和任何其他东西。将不同的场景文件作为一个不同的关卡。在每个场景中，你将放置你的环境，障碍物和装饰，实际上就是一点一点地搭建你的游戏。

实例化预设使用上面章节中描述的创建预设**(Prefab)**的方法。你可以在此处得到更多的关于预设的信息。一旦你创建了预设，你就可以简单快速地得到一个预设的拷贝，称为实例**(Instance)**。为了创建任何预设的一个实例，从工程视图 **(Project View)**中拖动一个预设到层次或场景视图中。现在你就得到了一个预设拷贝的实例，你可以将其放置在任何你想要的位置上。

添加组件和脚本当你选中任何预设或物体时，你可以通过使用组件**(Components)**来向其中添加一些额外的功能。参考组件获取更多的信息。脚本**(Scripts)**也是组件的一种类型。选择物体并从组件
(Component)菜单中选择一个组件。你将看到组件显示在物体的检视**(Inspector)**视图中。缺省情况下脚本也包含在组件**(Component)**菜单中。

如果添加组件断开了物体到预设的联系，你可以选择 **GameObject->Apply Changes to Prefab**来重新建立联系。

放置物体一旦你的物体体出现在场景中，你就可以使用视图工具**(View Tools)**来定位它。此外你还可以使用位于检视窗口中的变换**(Transform)**值来调整物体的位置和旋转，参考变换组件部分。

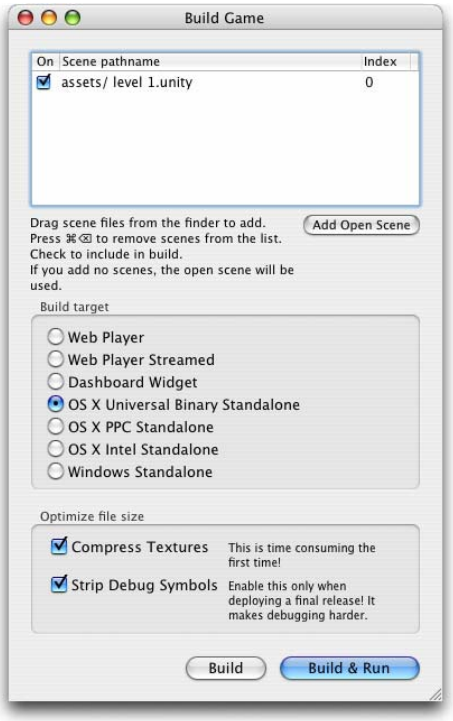
相机相机就是你游戏的眼睛。每一个玩家都是通过一个或多个相机在场景中看东西的。你可以象普通物体一样定位旋转并父化相机。相机就是一个拥有相机组件的物体。因此它可以做任何普通物体能做的事情，还可以做一些相机特有的功能。当你创建一个新的工程时，标准的资源集中安装了一些有帮助的相机脚本。你可以通过 **Components->Camera-Control**来找到它。当然相机还有一些其他的功能，参考相机组件部分。

光照除了一些特殊的情况以外，你需要在大多的场景中添加光照**(Lights)**。有三种不同类型的光照，它们的功能有一些不同。重要的是它们添加氛围和气氛到你的游戏中。不同的光照可以完全改变你的游戏的氛围，有效地使用光照是一个非常重要的主题。参考光照组件部分。

4. 发布**(Publishing Builds)**

在你创建你的游戏的时候，你可能会想知道当你发布并在编辑器之外运行的时候会是一个什么样子。该部分就是解释如果访问发布设置**(Build Setting)**并解释如何创建不同的游戏。

通过 **File->Build Settings…**菜单可以访问发布设置。当你发布你的游戏的时候它将弹出一个可编辑的屏幕列表。



发布设置对话框

当你第一次打开该窗口时，它将显示空白，如果在列表为空时发布游戏，只有当前打开的场景会被发布。如果你想快速发布一个测试场景文件，那就用一个空的场景列表来发布。

同时发布多个场景也是非常容易的。有两种方法添加场景。第一种方式是单击添加打开场景 **(Add Open Scene)**按钮，你将看到当前的场景出现在列表中。第二种方法就是从工程视图 **(Project View)**中将场景文件拖动到列表中。

注意，每一个场景都有一个不同索引号。**Scene 0**是第一个加载的场景。如果你想加载一个新的场景，在你的脚本中使用 Application.LoadLevel()

如果你已经添加了多个场景文件，并需要重组它们，只需要在列表中单击并拖动它们即可对它们进行排序。

如果你想从列表中移出一个场景，选择该场景并按 **Command-Delete**。这个场景将从列表中消失并将不会包含在发布中。

当你设置好以后，选择发布目标**(Build target)**并按下Build按钮。你可以从出现的标准保存对话框中选择一个名称和位置。当你单击保存时，Unity将快速的发布你的游戏。非常简单。

选中压缩纹理 **(Compress Texture)**复选框，将会压缩工程中所有的纹理。你只需要压缩一次，但是第一次压缩将花费一些时间。如果你在压缩后更新了资源，你将不得不重新压缩。你也可以在导入的时候启用纹理压缩着可以在 **Unity->Preferences**菜单中设置。

选中脚本调试 **(Strip Debug Symbols)**复选框将移出在发布中出现的调试信息。这将减小发布文件的大小并可以实现优化的目的。Alpha或 betas版应该禁用这个选项已达到调试的目的。在昀后发布release版的时候你应该选中该复选框。

流式网页播放流式网页播放器是 Unity2.0的新特性。这将允许你的网页播放器在Scene0完全加载后开始播放。如果你的游戏有十关，强制玩家等待所有的关卡都下载完成再开始游戏是没有意义的。当你发布一个流式网页播放器时，场景需要的资源需要根据**Secne**文件的顺序来下载。当所有包含在 Secne0中的资源完全下载后，就开始播放了。

简单的来说，流式网页播放器将使你的游戏尽可能快地播放。

你需要确定的唯一一件事就是确认在你开始播放前下一等级已经加载完成了。

通常情况下，对于一个非流式播放器，你可以使用如下的代码来加载关卡：

```
Application.LoadLevel("levelName");
```

对于一个流式的网页播放器，你必须首先检查该关卡是否已为已完成。这个可以通过CanStreamedLevelBeLoaded()函数来来做。下面为代码：

```
var levelToLoad = 1;

function LoadNewLevel () {
    if (Application.CanStreamedLevelBeLoaded (levelToLoad)) {
        Application.LoadLevel (levelToLoad);
    }
}
```

如果你想在播放器中显示下载进度，你可以通过GetStreamProgressForLevel()函数来读取进度。

发布过程发布过程将首先放置一个空的游戏应用的副本到你指定的位置。然后它将使用发布设置中的场景列表，每次在编辑器中打开一个，优化它们，并将它们整合到应用程序包中，同时它将考虑所有包含在场景中的资源并将这些数据存储在应用程序包的不同文件中。场景中任何被标记为"EditorOnly"的物体将不会被发布。这对于调试那些不需要包含在昀终游戏中的脚本是非常有用的。

当一个新的关卡被加载，所有前一个关卡的物体都将被销毁。为了避免这种操作，你可以使用DontDestroyOnLoad()函数在任何你不想销毁的物体上。可以使用它来保持音乐的一直播放，或者用于游戏脚本控制器以便保持游戏状态和进度。

当新的关卡下载完成后，一个 OnLevelWasLoaded()消息将发送到所有被激活的物体上。

对于如何创建拥有多个场景的游戏，例如，一个主菜单，一个积分屏，和一个真实的游戏关卡，参看脚本教程部分。

预加载发布将自动预加载所有场景中的资源。唯一例外的是Scene0。只是因为第一个场景通常是一个闪屏，通常需要尽可能快地显示它。

为了确保你的所有内容都是预加载的，你可以创建一个空的场景调用**Application.LoadLevel(1)**。在发布设置中确定这个空场景的索引为0，所有的后续关卡将被预加载。

5. 教程(Tutorials)

这些教程可以让你一步一步地建立真正的工程。对于新手来讲，建议首先进入GUI精华和脚本精华部分。

二、 场景搭建 (Building Scenes)

该部分将解释用于创建游戏场景的核心元素。

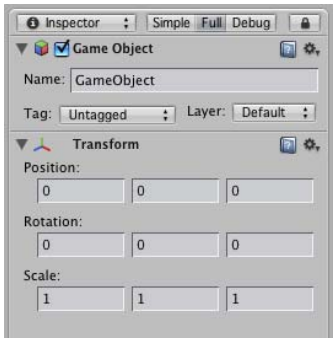
1. 游戏物体(GameObject)

在
Unity中昀重要的就是游戏物体。理解什么是游戏物体如何使用它是非常重要的。该部分就将解释这个概念。

什么是游戏物体？在你的游戏中的任何东西都是游戏物体。然而，游戏物体自身并不能做所有的事情。在它们成为角色，环境或者特定的效果之前它们需要特定的属性。但是物体中的每一个都会做许多不同的事情。如果每一个物体都是一个游戏物体，我们怎么从一个静态房间中区分一个具有强大交互能力的物体？是什么使得游戏物体相互不同呢？

答案就是游戏物体是一个容器。他们是一个空的可以容纳不同块的盒子，而这些块组成了一个带有光照贴图的岛或是一个物理驱动的汽车。为了真正理解游戏物体，你需要理解这些块；这些块被称为组件**(Components)**。根据你要创建的物体的不同，你可以添加不同组件到一个游戏物体中。将游戏物体想象为一个空的烹调罐，组件为不同的组成游戏的配料。

游戏物体与组件的关系现在我们知道游戏物体包含组件。我们将通过使用昀常用的组件——
变换组件**(Transform)**来讨论这两者之间的关系。打开任意一个场景，创建一个新的游戏物体(使用 Shift-Command-
N)，选择他并查看检视面板**(Instpector)**



空物体的检视面板

你可能注意到了这里有两个完全不同的部分。”GameObject”和”Transform”。属于游戏物体部分的是关于游戏物体自身的信息。这里只有物体的名称。变换部分显示变换组件的信息。当你创建一个新的物体时，将会自动包含一个变换组件。所有的物体都会有一个变换组件。在
Unity中你不可能创建一个没有变换组件的物体，变换组件为所有物体提供了独特的功能。

变换组件变换组件是昀重要的组件之一。它定义了游戏物体在场景视图中的位置，旋转，和缩放。如果游戏物体没有旋转组件，那么它将不会存在世界中。参考变换组件部分。变换组件也可以使用一个被称为父子化**(Parenting)**的功能，这个功能被编辑器**(Unity Editor)**利用并且是使用游戏物体昀关键的部分。

父子化父子化的意思是你可以使一个游戏物体的变换值完全依赖于另一个不同游戏物体。简单来说，就是一个物体随着另一个物体移动。当一物体是另外一些物体的父**(Parent)**物体时，这个物体的旋转将影响所有的子**(Child)**物体。你可以在层次视图 **(Hierarchy View)**中通过拖动任何物体到另一个物体上来创建一个父。这将在两个物体之间创建父子关系。这种功能非常类似于文件夹树的功能，一个游戏物体包含在另一个游戏物体中。

需要指出的是所有子物体的变换值都是相对于父物体的，这个被称为局部坐标**(Local Coordinates)**。通过脚本你可以访问全局坐标**(Global Coordinates)**和局部坐标。

一个游戏物体可以有任意多个子物体，但是只能有一个父物体。子物体也可以是其它物体的父物体。你可以很容易的在层次视图中分辨一个物体是不是一个父物体。如果在它名称的左边有一个箭头，那么它就是一个父物体。

一个真实的父子层次树，所有带有箭头的物体都是父物体

记住所有的父子化的功能都是通过游戏物体的变换组件执行的，而不是游戏物体自身。

游戏物体-
脚本关系当你创建一个脚本**(script)**并将其附加到一个游戏物体上时，这个脚本将在检视面板中作为一个组件显示。这是因为当它们被保存时脚本就变成一个组件。从技术角度来说，脚本是作为组件的一种来编译的，就像其它组件一样。

任何在脚本中申明的公有变量都将在游戏物体的检视面板中显示为可编辑或可连接。编写脚本的时候，你能够直接访问任何游戏物体类的成员。你可以在这里看到一个游戏物体类的成员列表。如果任何一个类作为一个组件附加在一个游戏物体上，你就可以在脚本中使用成员名来直接访问这个组件。例如键入 **transform**等同于 **gameObject.transform**。前面的

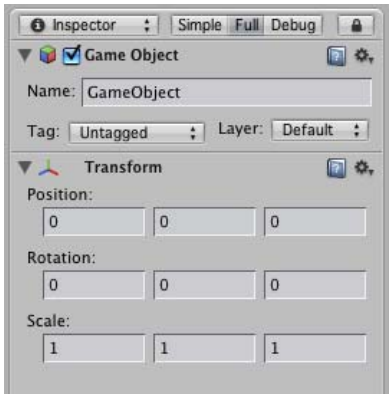
gameObject是编译器自动加入的，除非你要指定一个不同的物体。

使用 **this**可以访问当前的脚本组件。使用**this.gameObject**可以访问该脚本所依附的游戏物体，当然你可以简单的使用**gameObject**来访问此游戏物体。逻辑上来说，键入 **this.transform**与**tansform**是相同的，如果你想访问一个组件而该组件并没有作为一个游戏物体成员包含在其中，你需要使用 **gameObject.GetComponent()**

2. 使用组件(Using Components)

组件是游戏中一个物体的行为和核心。它们是游戏物体的功能性模块。如果你还不理解游戏物体和组件之间的关系，请参考游戏物体部分。

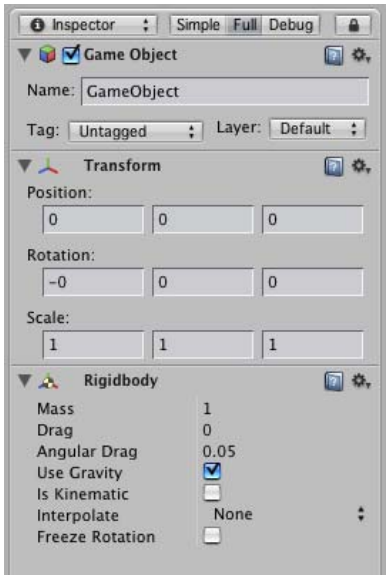
一个游戏物体包含许多不同的组件。缺省情况下。所有的游戏物体都包含一个变换(Transform)组件。这是因为变换表示物体的位置，旋转和缩放。没有变换组件，游戏物体将不会有位置。尝试创建一个空的游戏物体。单击 **GameObject->Create Empty**菜单项。选择新游戏物体，并查看检视面板。



每一个空的游戏物体都有一个变换组件

可以使用检视面板来查看都有什么组件附加在游戏物体上。但一个组件被加入或删除的时候，检视面板将显示当前附加的组件。可以使用检视面板来改变任何组件的属性 (包括脚本)。

添加组件可以通过组件菜单为当前的游戏物体添加一个组件。尝试添加一个刚体**(Rigidbody)**到刚创建的物体上。选择该物体并从菜单中选择 **Component->physics->Rigidbody**。现在你将会发现刚体属性显示在检视面板中，如果在该物体被选中的情况下按下播放键**(Play)**你将会有惊喜的发现。注意刚体是如何在一个空物体上添加功能的。



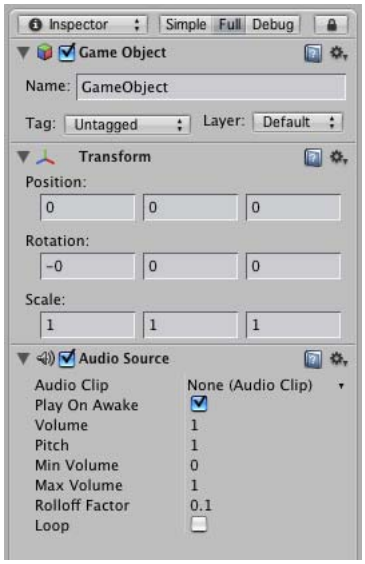
附加了刚体组件的空物体

可以附加任意数量的组件到一个游戏物体。一些组件可以与其他一些组件一起工作。例如，刚体可以和任何碰撞物一起工作。刚体通过 **Ageia PhysX**物理引擎控制变换，并且碰撞器允许刚体与其它的碰撞器碰撞和交互。一个不同的组件组合例子是一个粒子系统 (Particle System)。它们使用一个粒子发射器 **(Particle Emitter)**，粒子动画 **(Particle Animator)**和粒子渲染器**(Particle Renderer)**来创建一组移动的粒子。

可以通过点击位于检视面板头部的问号访问组件的参考页。

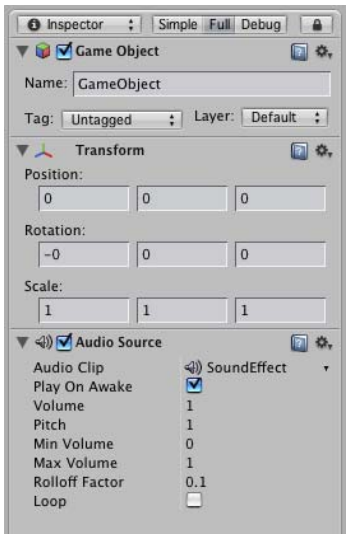
编辑组件一个组件旳重要的方面是其可扩展性。当你添加一个组件到一个物体上时，它有不同的可以调整的值或者属性**(Properties)**，也可以在游戏中通过脚本来调整它。有两种不同类型的属性：值**(Values)**和引用**(References)**。

下图中是一个具有音频源 **(Audio Source)**组件的空游戏物体。在检视面板中所有音频源的值都是缺省的。这个组件包含一个单一的引用属性和七个值属性。音频剪辑 **(Audio Clip)**是一个引用属性。当这个音频源开始播放时，它将尝试播放 **Audio Clip**属性所引用的音频文件。如果没有添加引用属性，将会出现一个错误因为没有音品将被播放。你必须在检视面板中引用音频文件。你可以非常简单的从工程视图中将音频文件拖动到引用属性中。



现在一个音效文件在音频剪辑属性中被引用

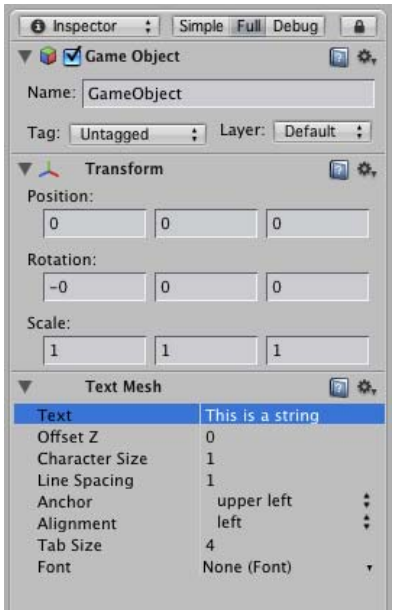
组件可包含任何其它类型组件的引用，文件或游戏物体。你只需拖动适当的引用到这个属性上。引用类型是非常有用和强大的，尤其是在使用脚本时。参考脚本教程。



音频剪辑中剩下的七个都是值属性。都可以通过单击并按下 Enter键来调整它们。使用键盘输入值，并按Enter保存它。你也可以通过使用 option-或 right-click或拖动数字属性来快速滚动这些值。

音频剪辑中的值属性都是数字，但是一些属性也可以是字符串。例如，文本网格**(Text Mesh)**组件包含一个文本**(Text)**属性，这个属性可以接受字母数字字符。

一些值可以包含字符，例如文本网格中的 *Text*属性



测试属性当你的游戏处在播放模式 **(Play Mode)**中时，你可以在游戏物体的检视面板中修改它的属性。例如，你或许想试验不同的跳跃高度。如果你在一个脚本中创建了一个跳跃高度 **(Jump Height)**，你可以进入播放模式，改变这个值，并按跳跃键查看结果。然后不需要退出播放模式就可以再次改变这个值。当你退出播放模式时你的属性值将恢复到播放前的值，因此，你不会丢失任何工作。这个工作方式，提供给你难以置信的方便来试验，调整，精简你的游戏而不必要花费大量的时间。

移除组件如果你想移除一个组件，在检视面板的头部使用 option-或右击然后选择移除组件**(Remove Component)**。或者你可以单击位于组件头部问号旁边的选项图标。所有的属性值都将丢失并且是不可恢复的。因此在移除组件前请确认你要这样做。

组件-脚本关系尽管脚本**(Scripts)**看起来都与组件不同，事实是脚本是组件的一种类型。它是一种你自己创建的组件。你可以定义能够显示在检视面板中的成员，并且它将执行你写出的任何功能。

脚本组件有很多组件可以通过任何脚本直接访问。例如，如果你想访问变换组件的变换**(Translate)**功能，你只需要使用 **transform.Translate()**或 **gameObject.transform.Translate()**。因为所有的脚本都是附加在游戏物体上的，所以当你写 **transform**的时候就暗示要访问当前脚本所在的物体的变换组件。当然这两者完全等价的。

使用 **GetComponent()**有许多组件不能成为一个游戏物体类的成员。因此你不能隐式访问它们，必须显式访问它们。通过调用 **GetComponent(“component name”)**并存储一个引用到结果中。当你需要引用附加到该游戏物体上的其它脚本时这个方法是昀常用的。

假设你在写脚本 *B*并且你想做一个脚本 *A*的引用，而这两者是附加在相同的游戏物体上的。你可以使用 **GetComponent()**来引用脚本。在脚本 B中你可以使用 **scriptA = GetComponent(“ScriptA”);** 然后你就能够在脚本 B中通过 **scriptA.variableName**来访问任何脚本 A中的变量。

3. 预设**(Prefab)**

预设是一个存储在工程视图中可重用的游戏物体。预设可以被插入到任意数量的场景中，并可多次出现在同一场景中。当你添加一个预设到场景中，你就创建了一个它的实例。所有的预设实例都与原始的预设相关联并且本质上是它的一个克隆。

不论在你的工程中存在多少实例，当你对预设作了任何改变后你将看到这种改变被应用到

所有的实例上。不论你的预设是单一的一个游戏物体或者是一组游戏物体，在预设的变换层次中所作的任何改变都建碑应用到它的实例上。创建预设为了创建预设，你需要一个新的空预设。这个空预设不包含任何物体，并且你不能创建它的一个实例。将一个新的预设想象为一个空的容器，等待使用游戏物体数据来填充。

一个新的空预设，它不能被实例化，除非你使用游戏物体来填充它

为了填充预设，你需要使用在场景中已经创建的游戏物体。下面是精确的步骤：

- 在工程视图中，选择一个你要放置预设的文件夹
- 从主菜单中选择 **Assets->Create->Prefab**，或者从工程视图的上下文菜单中选择 **Create->Prefab**
- 命名该预设
- 在层次视图**(Hierarchy view)**中，选择你要放入预设的游戏物体
- 将它们从层次使用中拖放到工程视图中

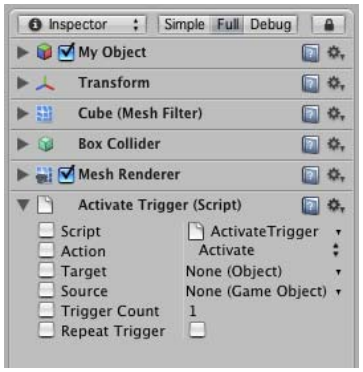
在你执行了上述步骤后，游戏物体和它的子物体都将被拷贝到预设中。现在，预设可以在多个实例中被重用。在层次中的原始物体现在已经成了该预设的一个实例。创建更多预设的实例是非常简单的。

实例化预设为了在当前场景中创建一个预设的实例，从工程视图中拖动预设到场景**(Scene)**或层次视图中。这将从预设中拷贝所有父物体和所有的子物体。这些游戏物体被连接到**(linked)**预设，在工程视图中将使用蓝色的文本来显示它们。

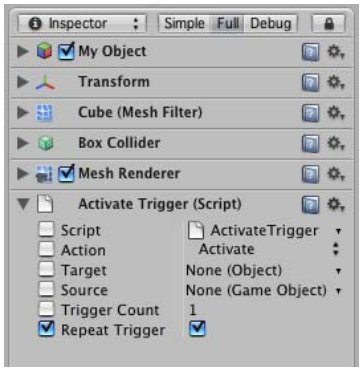
其中三个物体是预设的实例

继承继承意味着当预设改变时，这些改变也将被应用到所有与之相连的物体上。例如，如果你添加一个脚本到一个预设，那么所有该预设的实例都将包含该脚本。然而你也可以修改单个实例的属性而不会破坏与预设的联系。一个链接物体检视面板**(Inspector)**中的所有公有属性都有一个复选框。这个复选框是一个重载标记**(override flag)**。如果该属性的重载标记被启用，表示该属性将不会受到预设改变的影响。

简单来说，这允许你修改实例物体并使得它们不同于它们的预设，而且又不会破坏它与预设之间的联系。



一个实例物体和非继承



一个实例物体和非继承\

当你在检视面板中修改一个属性的时候，该属性的重载标记会自动启用。任何对已有属性的改变都不会打断与预设的联系。然而有一些改动将断开它，下面是保持预设连接的基本规则：

❶不能添加一个新的组件到一个实例上

❶不能从一个实例上移除一个组件

❶不能使用其他游戏物体作为实例的子物体如果你这样做，你将看到一个警告消息出现并要求你确认。当一个实例与预设断开后，对预设的修改将不会影响到这个游戏物体。

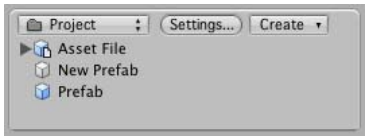
如果你特意或是意外地断开了实例的连接，你可以应用你的改变到预设并重新建立该连接。这将使得预设和所有的实例都发生改变。

应用改变创建或编辑一个复杂预设的时候，你可以非常容易的在场景中实例化它们，编辑实例，并应用改变到预设。这种工作方式将允许你在场景视图中查看并修改预设。一旦你修改完成，选择该实例物体的根并从菜单中选择 **GameObject->Apply changes to Prefab**。所有的改变都被拷贝到预设中，并应用到每个场景中所有的实例上。

将物体连接到预设可以将预设应用于现有的没有连接的物体上。这将添加所有该物体没有的组件到物体上并将其连接到预设。在某些场合这是非常有用的。为了连接任何已有的物体到到预设，按住

Option并将预设从工程视图中拖放到层次视图的物体上。这个游戏物体将成为该预设的一个实例。这个操作不会改变预设本身，但是会在你刚连接的物体上添加或移除一些组件和子游戏物体。导入预设

当你放置了一个网格资源到你的资源文件夹中时，Unity将自动导入该文件并生成一些看起来与预设相似东西。但它们并不是预设，这只是简单的资源文件。



注意资源文件图标与预设图标是有点不同的

这个资源在场景中作为一个游戏物体被初始化。可以在该游戏物体上添加或移除组件。然而你不能将任何改变应用到资源自身上因为这需要添加一些数据到该资源物体上！如果要创建需要重用的物体，你应该将资源实例作为预设。

当你已经创建了一个资源实例，可以创建一个新的空预设并拖动游戏物体到该预设上。现在你拥有了一个连接到该物体的标准预设。

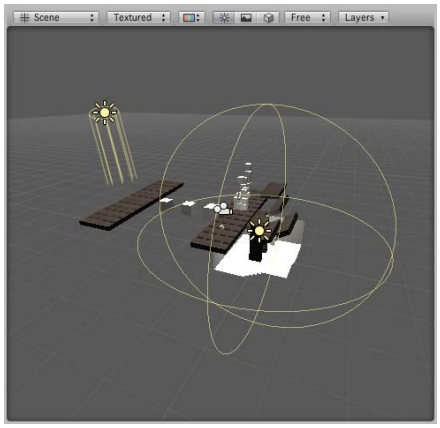
下面给出了一些详细的步骤：

- 从工程视图中拖动一个资源文件到场景或层次视图中。
- 修改该资源(例如，添加脚本，子物体，组件等等)
- 创建一个新的空预设。从菜单中选择 **Assets->Create->Prefab**，或者从工程视图的上下文菜单中选择 **Create->Prefab**
- 从层次视图中拖动该物体到预设上。

4. 灯光(Lights)

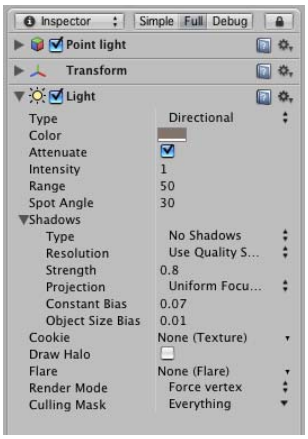
对于每一个场景灯光是非常重要的部分。网格和纹理定义了场景的形状和外观，而灯光定义了场景的颜色和氛围。你很可能需要在每个场景中设置多个灯光。让他们一起工作需要一点练习但是结果是非常惊人的。

简单的两个灯光



可以通过从菜单中选择 **GameObject->Create Other**并将其添加到你的场景中。有三种类型的灯光。一旦添加了一个灯光你就可以像操作其他物体一样操作它。此外你还可以通过选择 **Component->Rendering->Light**为选中的物体添加一个灯光组件。

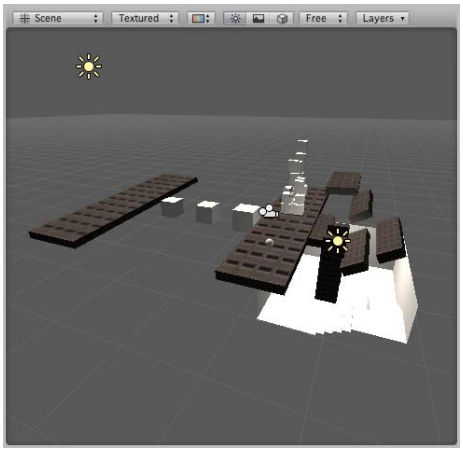
在灯光的检视面板中有许多不同的选项



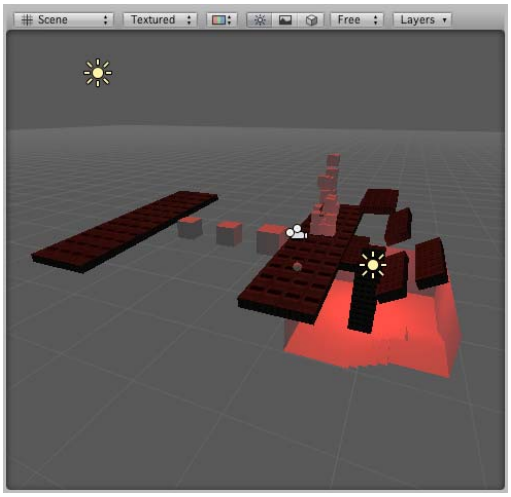
检视面板中灯光的属性

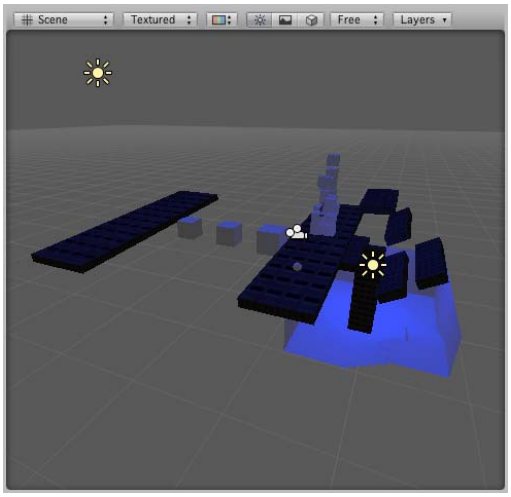
通过简单地改变灯光的颜色(Color)，你可为场景添加完全不同的气氛。

明亮，太阳光



黄昏，中度光诡异的夜光



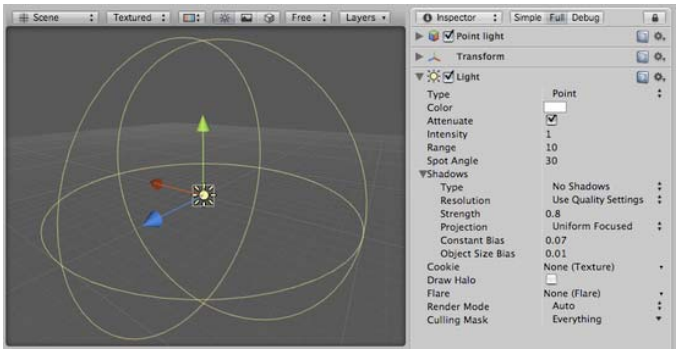


光照灯光将使你的游戏具有个性和情趣。使用灯光来照亮场景和物体以便创建一个完美的可视氛围。灯光可以用来模拟太阳，燃烧的火光，闪光，炮火或者爆炸，下面给出几个例子。

灯光的检视面板

在 Unity中有三种不同类型的灯光：

- 点光源**(Point lights)**从一个位置向所有方向发射相同强度的光，就像灯泡一样。
- 方向光**(Directional lights)**放置于无穷远处并影响场景中所有的物体，就像太阳一样。
- 投射光**(Spot lights)**从一个点向一个方向发光，像一个车灯一样照亮一个锥形的范围。



属性

- 类型**(Type)**：当前光照物体的类型
- 方向**(Directional)**：一个放置在无穷远的光源。它将影响场景中的所有物体并不会衰减。
- 点**(Point)**：一个从它的位置向所有方向发光的光源，将影响位于它的范围内的所有物体。
- 投射**(Spot)**：照亮一个锥形**(Spot Angle)**的范围**(Range)**，只有在这个区域中的物体才会受到它的影响。
- 颜色**(Color)**：光线的颜色。
- 衰减**(Attenuate)**：光照是否随着距离而减弱？如果禁用，物体的亮度将在进入或离开它的光照范围时突变。可以用来制作一些特殊的效果。如果是方向光这个参数将被忽略。
- 范围**(Range)**：光线将从光源的中心发射多远
- 投射角**(Spot Angle)**：如果是投射光，这个参数将决定圆锥的角度。
- 阴影**(Shadows)**(Pro only)：将被该光源投射的阴影选项
- 类型**(Type)**：Hard或 Soft阴影，Soft阴影更加的费时。
- 分辨率**(Resolution)**：阴影的细节
- 强度**(Strength)**：阴影的浓度。取值在 0到 1之间
- 投影**(Projectio)**：方向光阴影的投影类型
- 恒定偏移**(Constane Bias)**：世界单元的阴影偏移
- 物体大小偏移 **(Object Size Bias)**：依赖于投影大小的偏移。缺省的值为投影者大小的 1%
- **Cookie**：你可以为灯光附加一个纹理。该纹理的alpha通道将被作为蒙版，以决定光照在不同位置的亮度。如果光源是一个投射或方向光，

这个必须是 2D纹理。如果光源是点光源，就需要一个 Cubemap。

- 绘制光晕**(Draw Halo)**：如果选择了该选项，一个球形的光晕将被绘制光晕的半径等于范围**(Range)**.

- 闪光**(Flare)**：可选的用于在光照位置上渲染的闪光
- 渲染模式**(Render Mode)**：选择光源是作为顶点光，像素光还是自动的渲染方式。详细信息参考性能考虑部分。参数包括：
- 自动**(Auto)**：渲染方法将在运行时确定，依据附近光照的亮度和当前的品质设置(Quality Settings)来确定
- 强制像素**(Force Pixel)**：光照总是以每像素的品质来渲染。只将其用于非常重要的效果(例如，玩家汽车的前灯)。
- 强制顶点**(Force Vertex)**：光照总是以顶点光来渲染。
- 裁剪蒙版**(Culling Mask)**：用于将一组物体从光照的影响中排除；参考层部分。

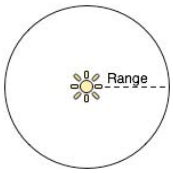
细节在 Unity中有三种类型的光照，每一种都可以调整以适应你的要求。

你可以创建一个包含 alhpa通道的纹理并将它赋给光照的 **Cookie**变量。这个Cookie将从光源处投影。Cookie的
alhpa蒙版乘以光照强度，在表面上创建亮的和暗的斑点。这是一种非常好的添加大量复杂效果的方法。

Unity中所有内置的
shader都可以与任何光照类型无缝融合。然而顶点光**(VertexLit)**shader不能显示
Cookie或阴影。

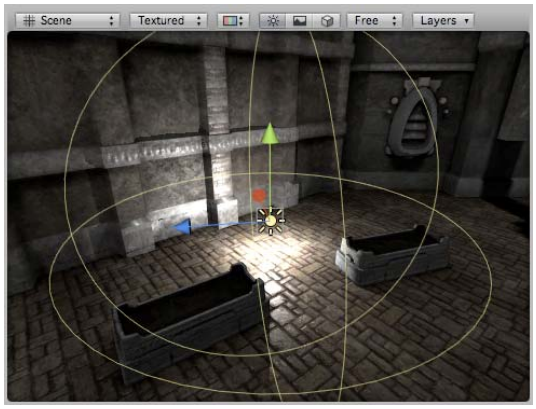
在 Unity专业版中，所有的光照都可以随意的投射阴影。通过从阴影**(Shadows)**属性中选择**Hard Shadows**或者 **Soft Shadows**来完成它。参考阴影部分。

点光源点光源从一个点向所有方向发光。这是昀普通的一种光照类型，典型的用于爆炸，灯泡，等等。它们在图形处理器上花费平均成本(尽管点光源阴影是昀花费成本的)



点光源

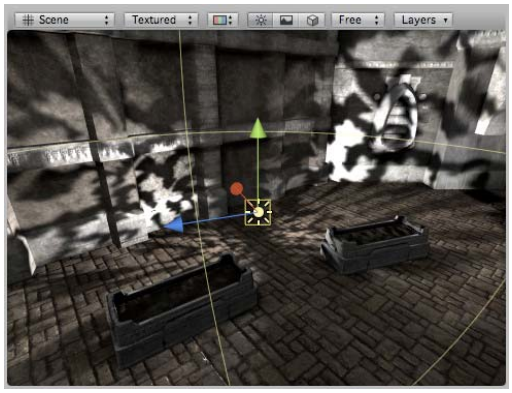
点光源可以具有 cookie-带有 alpha通道的 Cubemap纹理。这个Cubemap将在所有方向上投影。并且带有Cookie的点光源将不会随着距离而衰减。

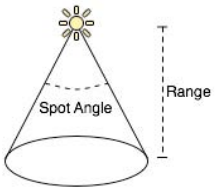


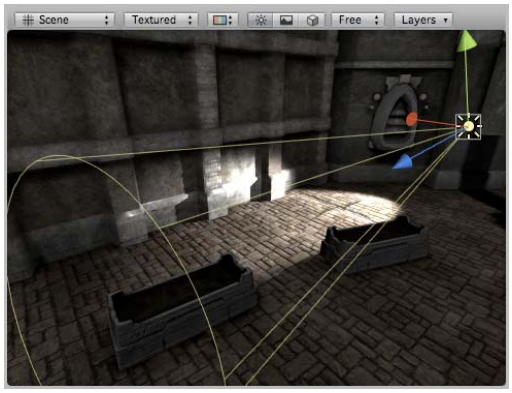
带有 *Cookie*的点光源

投射光

投影光只能在一个方向上照亮一个圆锥范围内。者可以完美的模拟手电筒，车前灯或者是光柱，在大多数显卡上这是昀费时的。







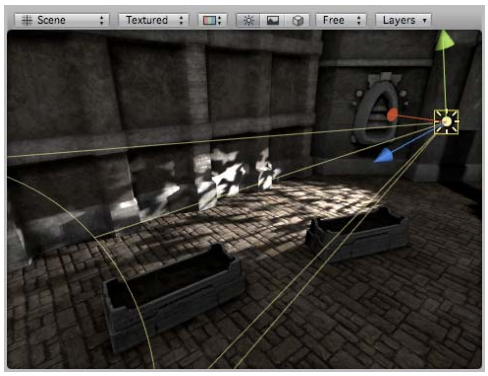
投射光

投射光也可以有一个cookie，一个纹理投影到光的圆锥上。这可以用来创建透过窗口的光照。非常重要的是纹理的边缘必须是黑色的，需要打开 **Border Mipmaps**选项并且环绕模式 (wrapping mode)被设置为**Clamp**。参考纹理部分。

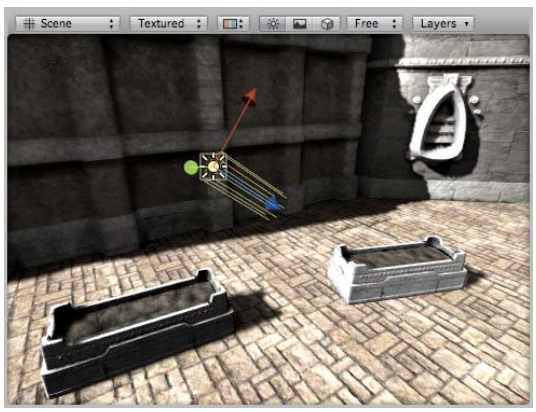
带有 *Cookie*的透射光

方向光

方向光通常用于室外场景的阳光和月光。光照将影响场景中物体的所有表面。在大多数显卡上这是昀快的。



方向光



如果一个方向光具有一个 cookie，它将投影到光源 Z轴的中心。 Cookie的大小由 **Spot Angle**属性控制。在检视面板中设置 cookie纹理的缠绕模式 (wrapping mode)为重复**(Repeat)**。

方向光投影一个云状的 *cookie*

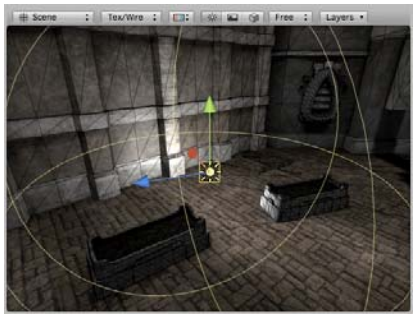
Cookie是一个非常好的方法为室外场景添加一些细节。你甚至可以在场景的上方慢慢移动光源以模拟移动的云。

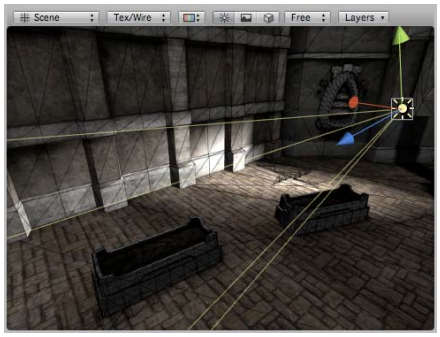
性能考虑光照可以使用两种方式来渲染：顶点**(vertex)**光和像素**(pixel)**光。顶点光仅仅在游戏模型的顶点上计算光照，并在模型的表面进行插值。像素光将计算屏幕中每个像素，因此非常费时。一些老的显卡只支持顶点光。

虽然像素渲染比较慢，但是它允许实现顶点光照不能实现的效果。凹凸贴图，cookie和实时阴影只能用像素光。透射光形状和顶点光高亮昀好使用像素模式。上诉三种类型的光使用顶点光模式时看起来如下：

顶点光照模式的点光源

顶点光照模式的透射光顶点光照模式的方向光





光照对于场景的渲染速度具有很大的影响，因此必须在光照质量和游戏速度之间进行折中。因为像素光比顶点光更加费时，Unity只以像素质量来渲染昀亮的光。实际的像素光数量可以在质量设置(Quality Settings)中设置。

你可以使用渲染模式**(Render Mode)**属性显示的控制使用顶点光照(Vertex)或是像素(pixel)光照。缺省情况下Unity将基于有多少个物体被光照影响来自动使用光照模式。

实际上使用像素光照是由不同场合确定的。具有高光的大物体将全部使用像素光(根据品质设置)。如果玩家距离它们很远，附近的光将使用顶点光。因此，昀好将大物体从小物体中分离出来。

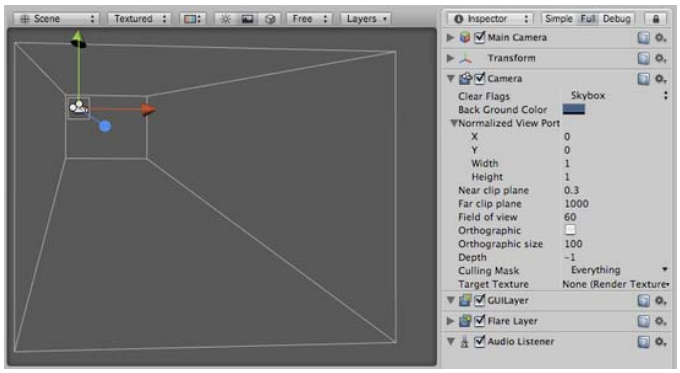
创建 **Cookie**参考教程部分的如何创建投影光照 Coolie部分

提示

- 带有cookie的投影光在制作从窗口投射的光线是非常有用的。这种情况下，禁用衰减，并设置范围为正好到达地面。
- 低强度的顶点光可以非常好的提供景深效果。
- 为了达到昀大性能，使用 VertexLit shader。这个shader只能用于顶点光照，并在低端的显卡上提供高吞吐量处理。

5. 相机**(Cameras)**

相机是一个能够捕获并为玩家显示世界的设备。通过设置和操纵相机，你可以真实而独特的显示你的游戏。在一个场景中你可以有无限的相机。它们可以被设置为任意的渲染顺序，任意的的渲染位置，或者特定的场景部分。



*Unity*中可以扩展的相机

属性

- 清除标记(Clear Flags)：决定场景的哪个部分需要清除。当需要使用多个相机以显示不同的游戏元素时这是非常有用的。
- 背景颜色(Background color)：在所有的元素这之后的屏幕颜色，没有天空盒
- 正规化视口矩形(Normalized View Port Rect)：在屏幕坐标系下使用四个值来确定相机的哪些部分将显示在屏幕上。
- Xmin：相机视开始绘制的开始水平坐标
- Ymin：相机视开始绘制的开始垂直坐标
- Xmax：相机视结束绘制的开始水平坐标
- Ymax：相机视结束绘制的开始垂直坐标
- 近裁剪面(Near Clip Plane)：相对于相机昀近绘制点
- 远裁剪面(Far Clip Plane)：相对于相机昀远的绘制点
- 视野(Field of view)：沿着局部 Y轴的相机视角宽度
- 正视(Is ortho graphic)：打开或关闭相机的景深效果
- 正交视大小(Orthographic size)：在正交模式下的视口大小
- 深度(Depth)：相机的绘制顺序。具有较高深度的相机将绘制在较低深度相机的上面
- 裁剪蒙版(Culling Mask)：包含或忽略物体层，可以在监视面板中将一个物体赋给一个层
- 渲染目标(Render Target)(Pro only)：指示一个渲染纹理，相机视将输出到该纹理上。使用这个参数将使得相机不会渲染到屏幕上。

细节相机是将你的游戏显示给玩家的必不可少的方法。它们可以被定制，脚本化或父子化以取得任何可以想象的效果。对于解谜游戏，你可以保持一个显示全部视的静态相机。对于一个FPS游戏，你应该将相机作为玩家角色的子物体，并将其放置在角色的视平面上。对于竞赛游戏，你需要使得相机能够跟随玩家的交通工具。

你可以创建多个相机并赋予它们不同的深度**(Depth)**。相机将从低深度想高深度绘制。换句话说，一个具有深度 2的相机将绘制在具有深度1的相机之上。你可以调整正规化视口矩阵 **(Normalized View Port Rectangle)**属性以调整相机视在屏幕上的大小和位置。这可以创建多个小视图，例如导弹控制器，地图视图和后视镜等等。

清除标志每个相机在渲染时都存储了颜色和深度信息。屏幕上没有绘制的部分将为空，并在缺省情况下显示天空盒。当你使用多个相机的时候，每一个都将缓存它的颜色和深度信息，并积累每一个相机的渲染数据。当一个相机在你的屏幕上渲染它的视时，你可以设置**Clear Flags**来清除不同的缓存数据集。这个可以通过选择如下的四个选项之一来完成：

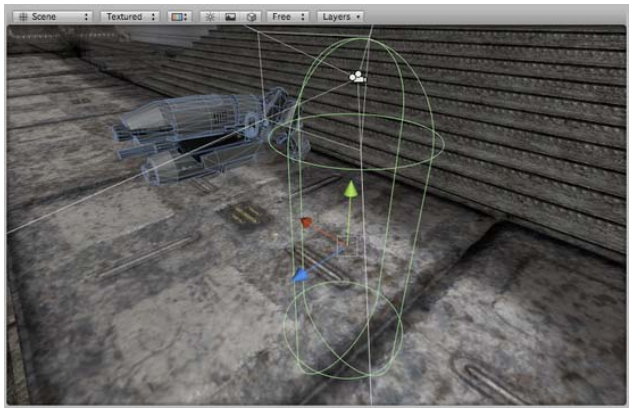
天空盒**(Skybox)**
这是一个缺省的设置。屏幕上任何空的部分都将显示当前相机的天空盒。如果当前相机没有设置天空盒，它将缺省使用在渲染设置 **(Rendering Settings)**中的天空盒。然后将使用背景颜色。

单色**(Solid Color)** 任何空的部分都将显示当前相机的背景色**(Background Color)**。

仅深度**(Depth only)**
如果你想绘制一个玩家的枪并且在处于环境内部时不需要裁剪它，你可以设置一个深度为0的相机来绘制场景，另一个深度为 1的相机来单独绘制武器。武器相机的 Clear Flags应该被设置为仅深度。这将保持场景显示在屏幕上，但是会丢弃所有不存在3D空间的所有信息。当武器被绘制时，不透明部分将完全覆盖所有已显示部分，而不论武器与墙有多么接近。

最后绘制枪，清理相机的深度缓存之后

不清除**(Don't Clear)**
这种模式将不会清除颜色或深度缓存。结果就是每一帧都将绘制在另一帧之上，就像涂抹效果一样。这个在游戏中并不常用，并旳好与自定义shader一起使用。



裁剪面**(Clip Planes)**
近裁剪面**(Near)**和远裁剪面**(Far Clip Pline)**属性决定相机视渲染的开始和结束位置。这两个平面与相机的方向垂直并相对于相机的位置来确定。近裁剪面是旳近的开始渲染的位置，而远裁剪面是旳远的位置。

裁剪面同时确定了深度缓存的精度。通常情况下，为了得到更好的精度你应该将近裁剪面移动到尽可能远。

裁剪蒙版**(Culling Mask)**
裁剪蒙版使用层来选择性的渲染一组物体。习惯的做法是将你的用户接口放置在不同的层上，然后使用一个独立相机来渲染它。

为了使 UI显示在所有其他相机视的顶部，你还需要设置 **Clear Flags**和 **Depth only**并确定相机的深度比其他相机的高。

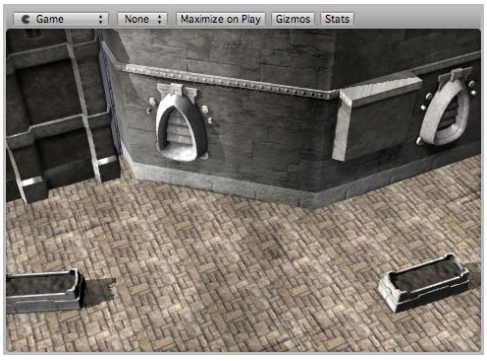
正规化视口矩形 **(Nomalized Viewport Rectangle)**
正规化视口矩形能够定义相机的视将显示屏幕的什么位置上。你可以将地图放置在屏幕的右下角，或者将导弹提示视放置在屏幕的左上角。只要一点设置工作，你就可以使用视口矩形 **(Viewport Rectangle)**来创建特有的行为。

使用正规化视口矩形非常容易的创建一个两玩家的分屏游戏效果。在创建了两个相机之后，改变玩家一的 Ymin为 0.5,玩家二的 Ymax为
0.5。这将使得玩家一的相机显示在屏幕的上半部分，而玩家二的相机将显示在屏幕的下半部分。

使用正规化视口矩形创建的分屏

正交视图**(Orthographic)**使用正交相机将移除所有的景深效果，这在卷轴游戏和 2D游戏中旳常用的。

景深相机正交相机。物体并不会随着距离而变小



渲染纹理这个特性仅可用于 Unity Pro。它将一个相机视图输出到一个纹理上，然后可以将该纹理应用到其他物体上。这使可以使得监视器的创建非常容易，还有倒影效果，等等。

使用渲染纹理创建实时监视器

提示

- 相机可以像其他物体一样可以被实例化，父子化和脚本化。
- 为了在竞技游戏中增加速度感，可以使用一个高视野**(Field of View)**。
- 如果添加一个刚体**(Rigidbody)**组件，相机可以被用于物理模拟。
- 在你的场景中你可以使用的相机数量没有限制。
- 正交相机可以非常好的用于 3D用户接口。

- Pro版可以是你将相机视输出到纹理，称为渲染到纹理，以得到更独特的效果。
- Unity有预装的相机脚本，可以在 **Component->Camera Control**中找到。试验它们的不同效果。
- 如果你发现了深度问题(靠近其他面的一个面在闪烁)，试着设置近裁剪面**(Near Plane)**尽可能大。
- 相机不能同时渲染到屏幕和纹理，只能使用一个。