

Unity课件

第一章

1 Unity介绍

Unity是由Unity Technologies开发的一个能够轻松创建三维视频游戏、建筑可视化及实时三维动画等互动内容的、多平台的综合型开发工具，也是一个全面整合的专业游戏引擎。通过Unity简单的用户界面，用户可以完成几乎所有工作。

其编辑器运行在Windows和Mac OS X下，可发布游戏到Windows、Mac、Android、iPhone、Xbox、VR设备等多个平台。而且Unity还支持网页浏览器。其跨平台的优越性不言而喻。

说了这么多还不知道Unity到底是什么？

Unity是移动游戏领域较为优秀的游戏引擎，起初是专为制作3D游戏而出现的，后来随着Unity的发展，慢慢加入了2D的功能，从Unity发展到如今，Unity已经成长为一个非常成熟的游戏引擎，说到Unity的发展史，我们来看看Unity的光辉历程。

2005年6月，Unity1.0发布。Unity1.0是一个轻量级、可扩展的依赖注入容器，有助于创建松散耦合的系统。它支持构建子注入、属性/设置方法注入和方法调用注入。

2009年3月，Unity2.5加入了对Windows的支持。Unity2.5完全支持Windows Vista与Windows XP的全部功能和互操作性，而且Mac OS X中的Unity编辑器也已经重建，在外观和功能上都形成了统一。Unity2.5的优点就是Unity几乎可以在所有平台上建立任何游戏，实现了真正的跨平台。

2009年10月，Unity2.6独立版开始免费。Unity2.6支持了许多的外部版本控制系统，如Subversions、Perforce、Bazaar以及其他VSC系统

等。除此之外，Unity2.6与Visual Studio完整的一体化也增加了Unity自动同步Visual Studio项目的源代码，实现了所有脚本的解决方案和智能配置。

2010年9月，支持多平台的Unity3.0发布。其新增加的功能有：方便编辑桌面左侧的快速启动栏、增加支持Ubuntu12.04、更改桌面主题和在dash中隐藏“可下载的软件”类别等。

2012年2月，Unity发布了3.5版本。

2012年11月，Unity又发布了一个非常重要的版本，4.0版本，相对于之前的版本，该版本新加入了DirectX 11的支持和全新的Mecanim动画工具，支持移动平台的动态阴影等。


2013年11月，Unity4.3发布，同时支持了2D游戏的开发。


2014年11月，Unity4.6发布，加入了新的UI系统，Unity开发者可以使用基于UI框架和视觉工具的Unity强大的新组件来设计游戏或应用程序。

2015年3月，UnityTechnologies在GDC2015上正式发布了Unity5.0，Unity首席执行官John Riccitiello表示，Unity5.0是Unity发展史上的重要里程碑。我们使用的便是Unity5.0之后发布的一个新版本Unity5.2.2f1。

2015年6月，Unity5.1发布，加入了VR和AR设备优化的渲染管道，可直接插入OculusRift开发机进行测试。头部追踪和景深FOV都会自动地应用到摄像头。

2 Unity3d环境搭建

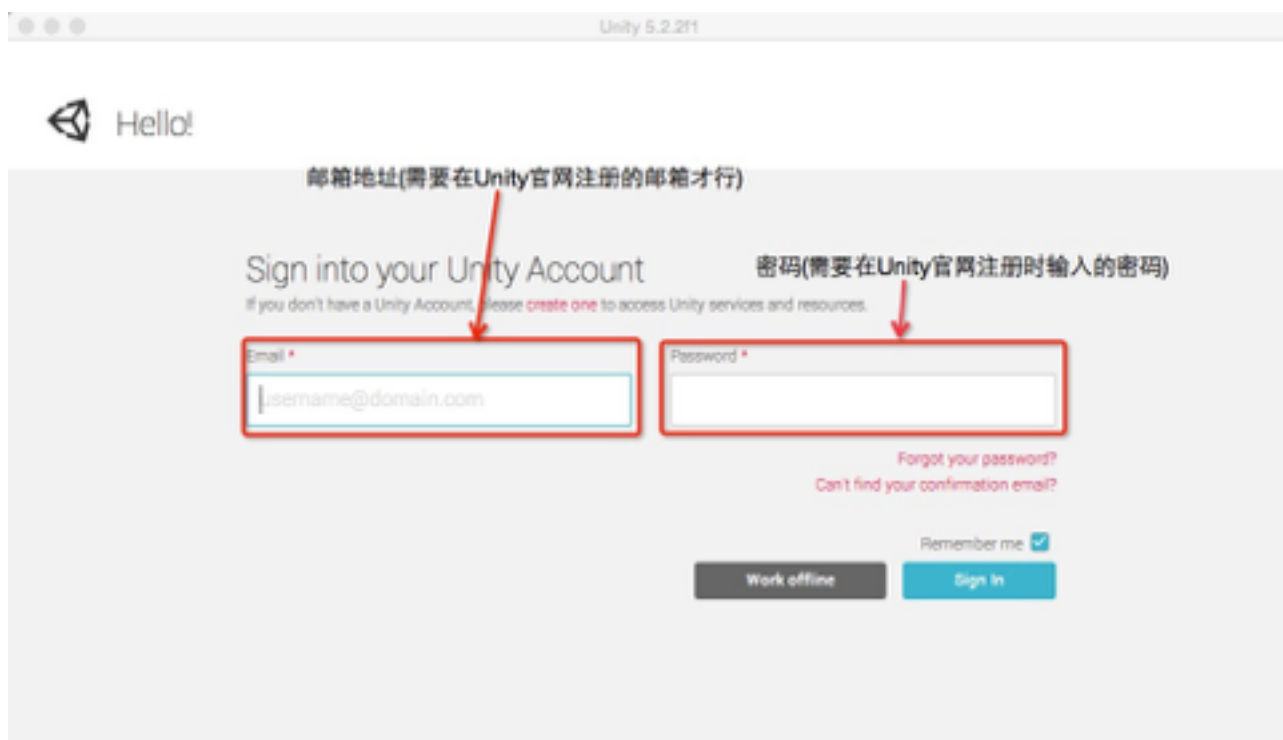
Windows平台下的安装与Mac平台下的安装其实都差不多，只要下载对应平台的版本就可以，我们以Mac平台为例，Mac下对应的Unity下载链接为：<http://unity3d.com/cn/get-unity/>，我们教程下载的Unity5.2.2f1的版本，如图所示， Unity-5.2.2f1.pkg。我们双击安装即可。安装成功，如

图所示，，打开安装成功后的Unity应用程序，相对于其他游戏引擎，Unity引擎的安装非常方便，只要会在Windows或者Mac下安装软件就行，那么下节我们将介绍Unity编辑器基础，以及如何使用Unity创建一个Unity工程。

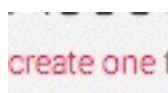
第二章

1 启动Unity及创建Unity账号

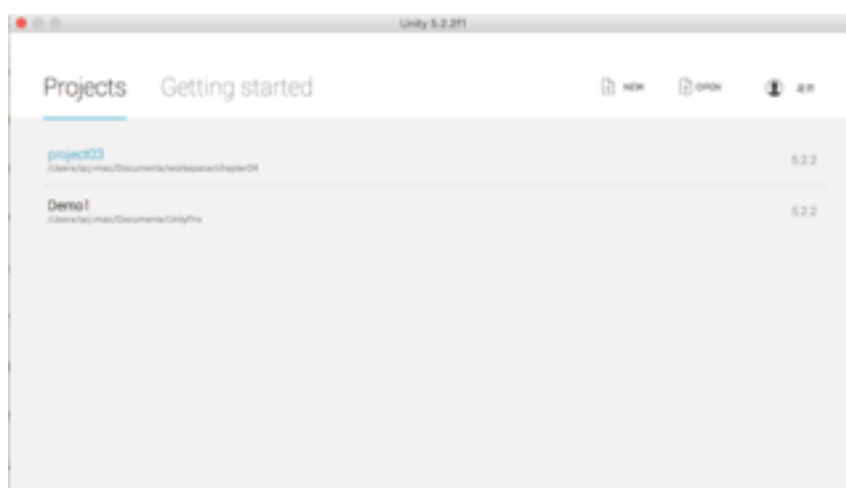
当我们第一次启动Unity应用程序时会弹出以下窗口，如下图所示，



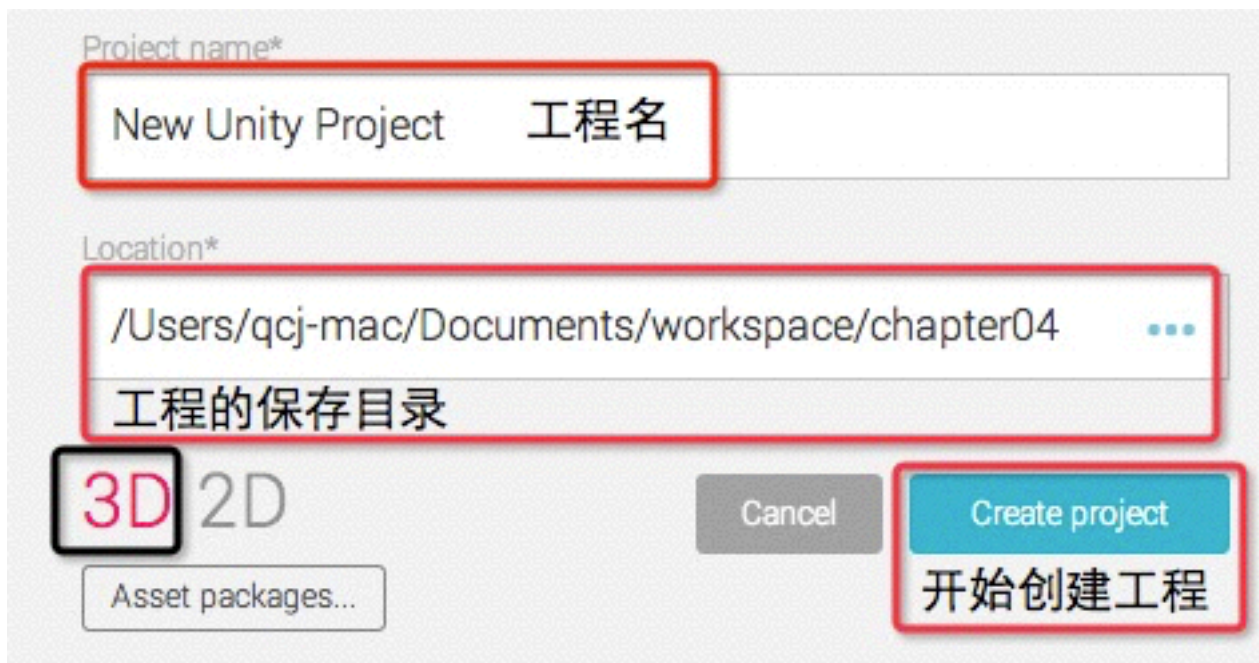
如果之前没有账号，可以点击上图的create one按钮，注册一个，如下图



关于如何注册在这里就不再详细的介绍了，只要输入邮箱及密码登录后即可创建Unity工程。我们来看看登录成功后的Unity界面，



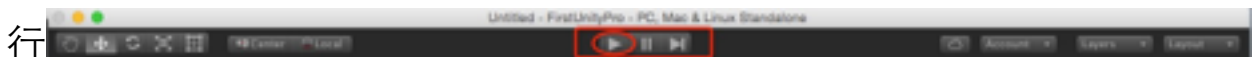
点击上图右上角的NEW  可以创建一个Unity工程，



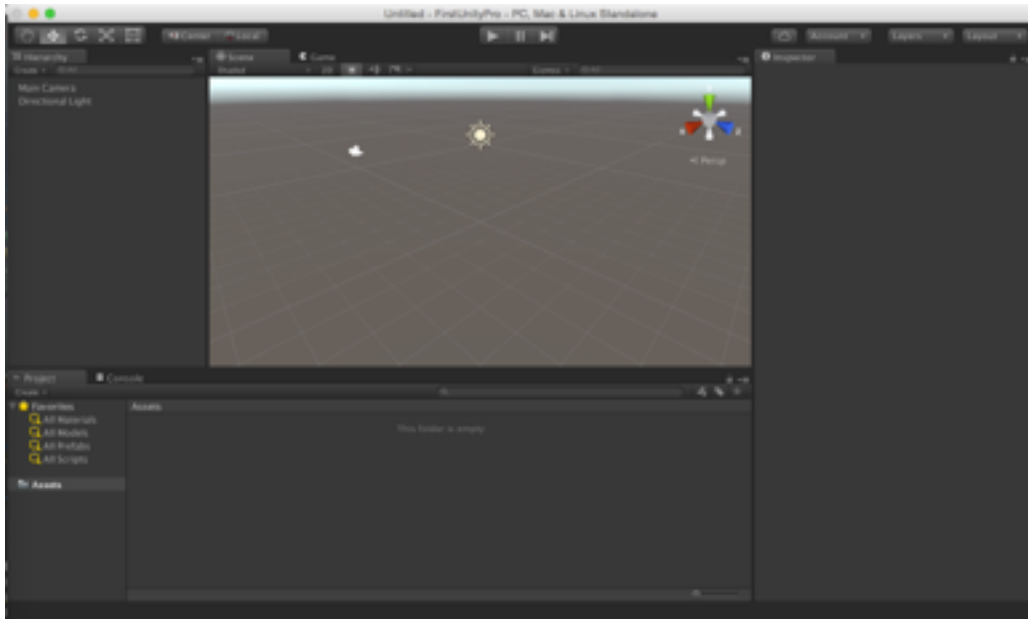
选择3D创建一个Unity3D工程。

2 第一个Unity程序

上节我们介绍了如何使用Unity编辑器创建Unity工程，在我们点击Create project之后，Unity就会小退一会儿，不要着急，这是Unity在后台创建工程当中，依据电脑性能，一般大概20秒左右，我们新建的工程就能创建完毕，并且此时Unity会显示我们创建的工程，如下图所示，点击运行



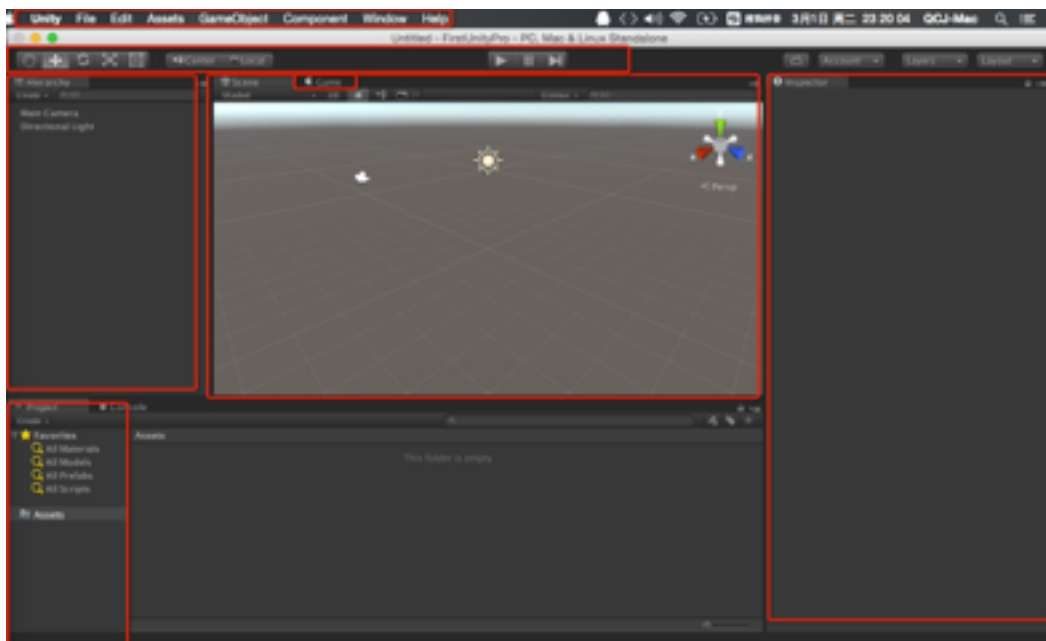
，就能看到工程运行后的效果。



以上就是我们创建工程到运行工程的一个过程，接下来将学习Unity编辑器。

3 Unity编辑器基础

Unity编辑器从直观上看，有几个模块是非常明显的，Hierarchy(层级视图)、Project(工程视图)、Scene(场景视图)、Inspector(检视视图)、Game(游戏视图)。其实Unity编辑器还可以分为菜单栏、工具栏、以及上面所讲的五大视图，大体可以被分为三个大模块，如下图所示，



如上所示就是Unity编辑器的功能分布图。我们将从这几个大模块中一一讲解每个模块具体都做些什么？

4 菜单栏

菜单栏上主要的菜单项有File、Edit、Assets、GameObject、Component、Window、Help。

菜单项File的主要功能是创建/保存/打开场景、创建/保存/打开工程以及发布到各平台时的一些设置入口。

菜单项Edit主要是一些拷贝、删除操作，其中比较重要的子功能有Project Settings，这个子菜单我们后续要用的时候一起去学习。

菜单项Assets主要是对资源文件的管理，可以用此菜单项的子功能去外部导入一些我们需要的模型、纹理、脚本等资源，还可以导出我们项目中使用到的资源。


菜单项GameObject可以为场景创建对象，也可以为对象创建父子关系等，具体操作我们需要带入到实际开发中去讲解。


菜单项Component是Unity中最重要的一个东西，又名组件，可以说，Unity中显示在Game中的都是组件，因为即使是一个对象，它也是组件拼凑而成的，是组件的集合体。而脚本只有挂载到对象上成为组件后才会生效。一般组件有网格、特效、音频、灯光、相机、导航寻路、渲染等，Transform组件是所有对象必有的一个组件，即使是一个空对象。


菜单项Window可以设置一下烘焙(后面专门会讲这个知识)、天空盒、动画、切换视图等操作。具体功能也需要在实际开发中去讲解。Help菜单项不是我们的重点，暂且能看懂就行。


下一节我们来学习一下工具栏的使用。


5 工具栏

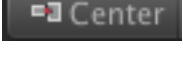
工具栏中的作用是移动整个场景，工具栏中的工具一般用的比较频繁，因此适当的记住一些快捷键也能为开发增加不少效率，而移动整个场景的工具的快捷键是Q。


图标为的作用是移动场景中的某个对象，快捷键是W，此工具可以使对象在某个方向上进行平移操作，用于调节对象在场景中显示位置。

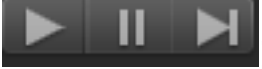
图标为的工具可以让场景中的某个对象沿着某个轴进行旋转操作，快捷键为E，注意:不限于x，y，z方向的轴。


图标为的工具是对对象进行缩放操作，快捷键为R。

图标为的工具是可以圈出对象的几何中心点到各个边界的中点所组成的四边形。具体需要在场景中去观察总结。

图标为的工具能够改变对象的轴心点(当选择Center的时候是以物体包围盒的中心；而选择Pivot的时候，则是以物体本身的轴心)。

图标为的工具能够改变物体的坐标系系统(当选择Global的时候是以世界坐标为基准；而选择Local的时候是自身坐标)。

图标为的工具分别是运行、暂停、逐帧播放，在运行期间，对对象的修改很多时候还是会被还原的。

图标为的工具为分层下拉列表，用来控制Scene视图中对象的显示的。

图标为的工具为布局下拉列表，用来切换视图的布局。

6 Hierarchy(层级)视图

Hierarchy视图(以下就叫做层级视图)是显示当前场景中真正用到的对象。层级视图中对象的排列是按照一定顺序来的, 用户如果随意命名场景中的对象, 那么就非常容易重名, 当要用查找所需的对象时就难以辨别了, 所以良好的命名规范在项目中就很重要。

在层级视图中提供了一种快捷方式将相似的对象组织在一起, 既为对象建立父子化关系, 这样的好处是便于大量对象的移动和编辑, 当然也可以为每一个独立的子对象进行编辑操作。具体的操作我们有相关的案例进行练习。

7 Project(项目)视图

Project (项目)视图是整个项目工程的资源汇总,包含了游戏场景中用到的脚本、材质、字体、贴图、外部导入的网格模型等所有的资源文件。Project视图由Create菜单、search by Type(按类型搜索)菜单、search by Label(按标签搜索)菜单、搜索栏和资源显示框等部分组成。

在Project视图会显示项目所包含的全部资源, 每个Unity项目文件夹都会包含一个Assets文件夹, Assets文件夹是用来存放用户所创建的对象和导入的资源, 并且这些资源是以目录的方式来组织的, 用户可以直接将资源直接拖入Project视图中或是依次打开菜单栏的Assets->Import New Asset项来将资源导入到当前的项目中。


用户使用第三方软件例如Photoshop对资源文件夹中的资源进行修改并保存时, Unity会自动更新资源以保持同步更新, 用户可立即在Unity编辑器中看到修改后的结果。

当创建一个工程时, 会生成一组文件夹。其中之一就被称为资源(Assets)文件夹。在工程视图(一般在项目视图的右边)中可以查看资源文件夹。如果打开过资源文件夹, 将会发现所有的项都将出现在工程视图中。具体操作需要在实际开发中去学习。

8 Scene(场景)视图

Scene视图是用于显示场景中用到的所有模型、光照、相机、材质等，Scene视图中的对象都可以进行位移、旋转操作。

9 Game(游戏)视图

Game视图是用于预览游戏运行后的效果，便于在开发中进行调试。在Game视图中有几个功能需要知道，，当选中Maximize on Play的时候，游戏运行后会全屏显示，选中Mute audio后，游戏运行后，场景处于静音状态，Stats会显示游戏运行之后的设备状态信息，Gizmo用于显示游戏对象的图标。

10 Inspector(检视)视图

检视视图用于显示在游戏场景中当前所选择对象的详细信息，以及游戏整体的属性设置。包括对象的名称、标签、位置坐标、旋转角度、缩放、组件等信息。

通过以上的学习，大概对Unity编辑器有了一个初步的认识，但是还是不能通过上面所学的东西做出一些有趣的东西。

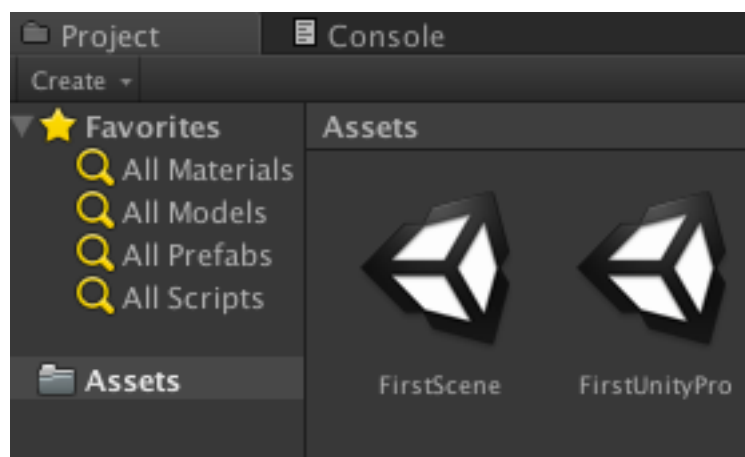
第三章

1 第一个游戏场景

上一章我们介绍了如何去创建一个工程，下面我们来看看新建的Unity工程的工程文件结构如何，打开存放上节我们创建的Unity工程文件，可以看到有4个预先创建好的文件夹，分别是Assets、Library、ProjectSettings、Temp，其中Assets就是存放我们游戏中所要使用的各种资源以及游戏场景文件，Library是我们Unity系统所依赖的库文件，ProjectSettings是Unity默认的工程配置信息存储之地，Temp是Unity中临时存放文件的文件夹。

回到Unity编辑器当中，其实，当我们创建好一个Unity工程的时候，Unity就已经默认的为我们创建好了一个缺省场景，按住键盘cmd+s就会弹出为新场景要选取保存路径以及给新场景取名的窗口。

我们需要自己创建一个场景，创建场景的方式为菜单栏选择File->New Scene，快捷键cmd+N，在我们做开发的时候需要养成一个良好的习惯，新建的场景应及时的去保存，菜单栏选择File->Save Scene，按照要求保存新建的场景，当在场景中修改了一些东西的时候，也要记得保存，快捷键都是cmd+S。这样，我们的第一个游戏场景创建完成，同时保存好的游戏场景会出现在工程视图下的Assets目录中，如下图所示。一款游戏不可能只有一个游戏场景，因此我们需要经常的创建游戏场景。所以场景对我们游戏开发来说必不可少。




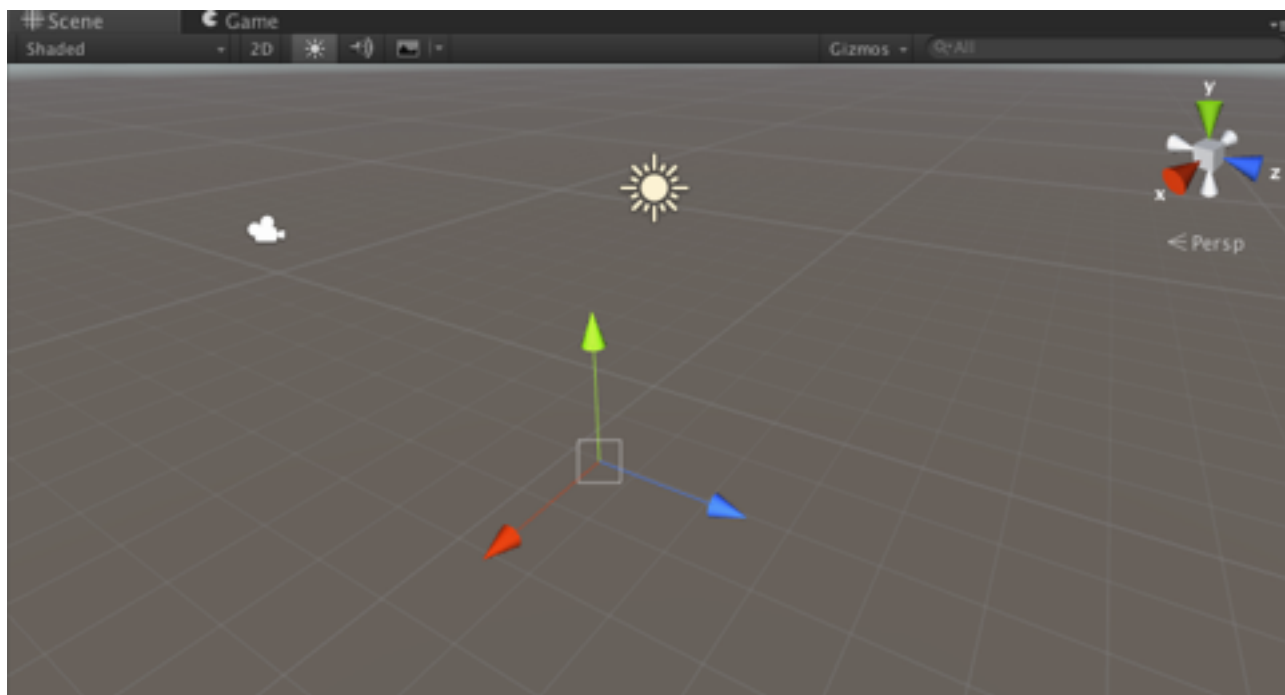
2 第一个游戏对象

Unity中游戏对象都会在层级视图(Hierarchy)中看到。那么打开上节创建的游戏场景“FirstScene”(注意:场景文件的后缀名是.unity)。在层级视图模块中,我们会看到有两个默认的游戏对象Main Camera(相机)、Directional Light(方向光源),选中相机,Scene视图右下角会出现跟没运行时候的Game视图一样的画面,我们在开发的时候可以调节相机的位置和方向来显示不同的游戏画面,具体操作后面会慢慢陈述。方向光源会在光源那节专门讲解,总之方向光源是可以模拟太阳光线的。

除了上面提到的两个游戏对象,我们可以自己创建一个游戏对象。

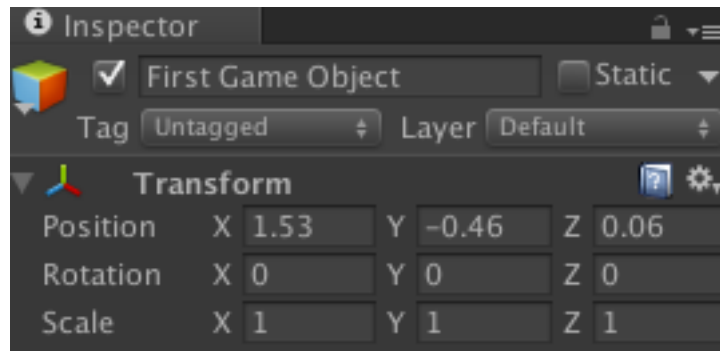
Unity中提供了很多的方式来为游戏场景创建对象,我们就采用其中一种,其他方式都差不多。

在层级视图模块有一个Create下拉列表, , 点击展开,我们创建一个空的游戏对象Create->Create Empty,选中新创建的空的空的游戏对象,改名为“First Game Object”,此时Scene视图如下所示。



在Scene视图出现的3个箭头其实是游戏对象的坐标轴，从Scene视图右上角也能看到类似的箭头，只有当工具栏中控制坐标系系统的选项为Global时，这3个箭头才与右上角的箭头指向完全一样。

再看Inspector(检视)视图，如下。




意味着一个空的游戏对象其实并不空，它至少有一个Transform属性，表示这个对象的位移以及形变信息。在这里出现在Inspector视图中的属性皆可以被看作是挂载在游戏对象上的组件。

我们再回过头来看看相机对象和方向光源对象，其中相机对象会挂载一个相机组件，方向光源对象会挂载一个灯光组件，想一想，如何创建一个普通对象，并且使这个对象成为光源或者相机呢？

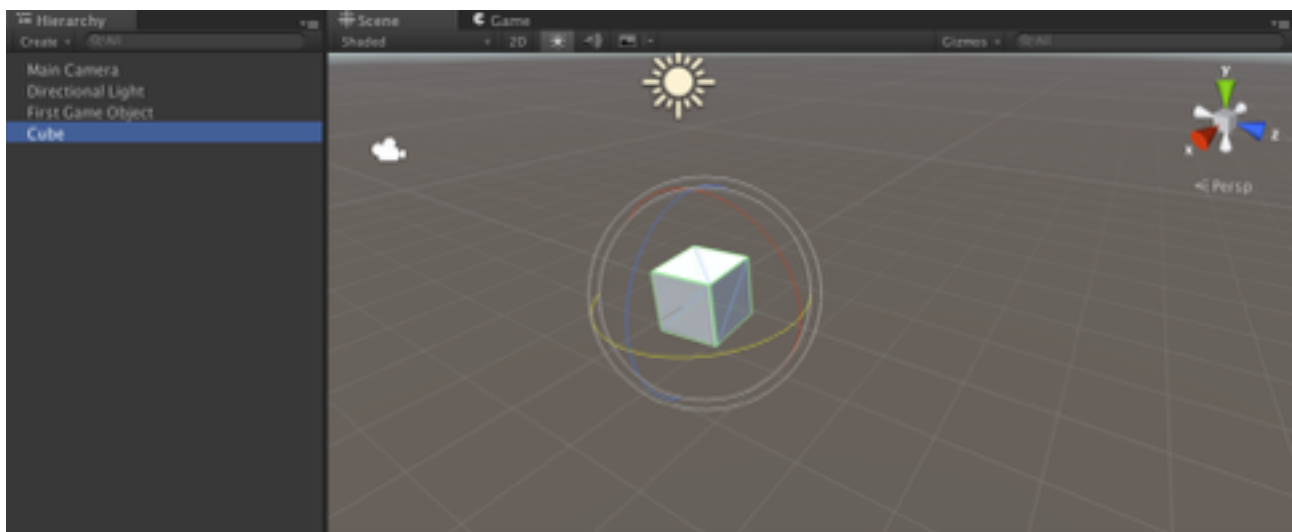
2.1 创建对象案例

(1) 打开我们上节创建的游戏场景，在层级模块，我们创建一个3D对象，选中Create->3D Object->Cube，Cube对象是Unity中的一个立方体模型，是Unity自带的一个模型。

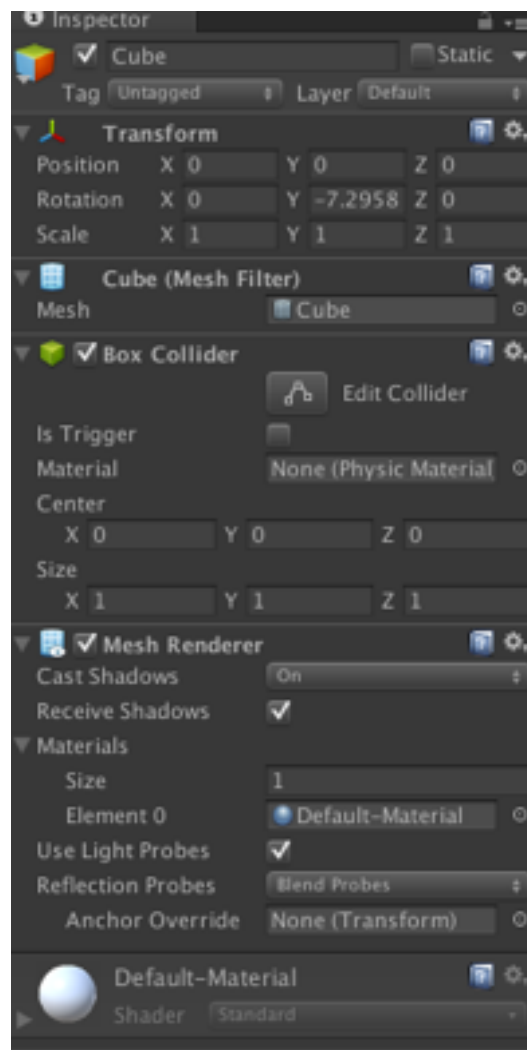
(2) 复习下我们上节所讲的内容，分别按快捷键 Q、W、E、R、T，然后操作Cube对象。


(3) 在2步骤中，按快捷键E，让Cube旋转一定角度，再按快捷键W，然后点击工具栏  Local 图标，观察Cude坐标系的变化情况

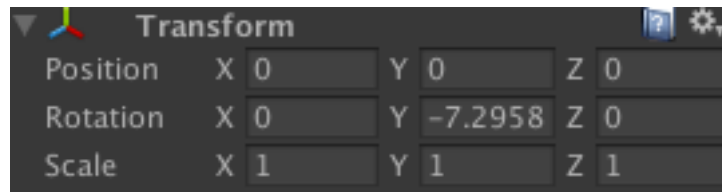
(4) 效果图如下所示。





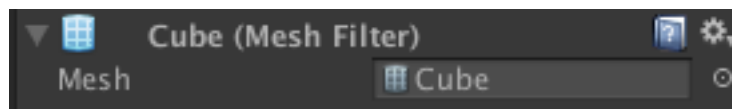
(5) 选中创建的对象，在检视视图中我们可以看到它包含的一些组件，如下图所示。



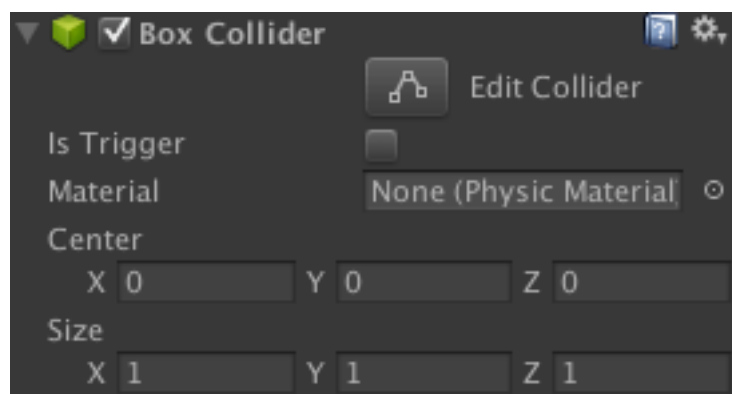
(6) 就上图，我们来简单说下检视视图中各个组件的属性，在检视视图中，Cube对象第一个组件是Transform属性，其图标为



Position可以控制对象的位置，Rotation可以控制对象的角度，Scale可以控制对象的缩放。下图为网格过滤，在组件章节会详细介绍，只要记住网格过滤的图标为。这个组件一般跟组件Mesh Renderer(网格渲染)一起结合着使用。图标为，后面也会有针对网格而设计的课题，在这里了解下就行。



(7) 下图是Cube的另一个组件，叫做盒型碰撞体，用于检测碰撞的，它是物理组件。模仿自然界的各种碰撞。在物理引擎的章节会学习到此内容，在我们开发Unity3d游戏时，物理组件是用的还是很多的。



除了Cube模型外，还有Sphere、Capsule、Cylinder、Plane(地板)、Quad，留下一个问题，Plane如何做到跟Quad一样的效果？有什么区别？

第三章

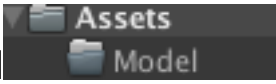
1 资源的导入

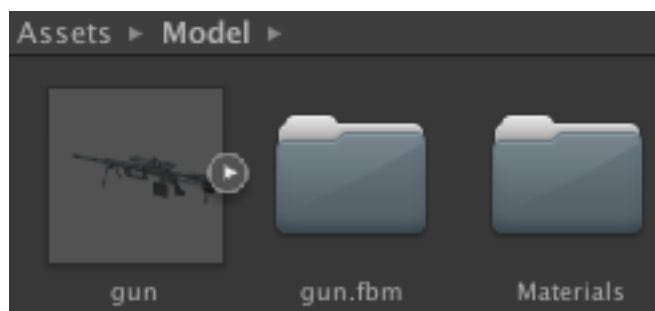
上节我们介绍了如何创建Unity内置的模型对象(有时候我们称之为基本几何体)，这节将学习如何导入外部资源。

Unity中，资源的类型有很多种，有模型资源(指那些3d建模软件导出的3d模型)、图片资源、音频资源、视频资源，以及资源包(包含有Assets文件夹导出的所有资源)。

除了导入资源包以外，其他的资源导入方式大体相同，在Project视图中，右键->Import New Asset，选中需要导入的资源即可，这种导入方式一次只能导入一个资源，也可以直接将要导入的资源一次性拖入到Assets下(注意，不要轻易去Assets工程目录下修改资源，但是可以使用Photoshop等图片制作软件修改图片，Unity会自动更新编辑器中Assets文件下的资源)。

我们来看一组案例。

(1) 打开上节创建的场景，我们在Project视图中，点击Assets，右键创建一个文件夹，Create->Folder，命名为Model ，双击Model文件夹，并在Model文件夹中右键，Import New Asset，将gun模型(导出模型文件以FBX后缀结尾，gun模型名称为gun.FBX)导入到Model文件来。导入后如下图所示。



(2) 我们可以将gun模型直接拖入到层级视图或者场景视图中，如下图所示，可以调节场景显示效果。



(3) 添加完成后，记得保存我们的场景，在层级视图中，我们也会发现有个以gun命名的对象，所以除了上节我们所讲的Unity自带的3d模型对象以外，在资源中也可以直接拖入某个资源成为游戏的对象。


2 创建资源

这节主要学习如何创建一个资源，其实在Unity当中，资源不仅仅可以在外部进行导入来使用，也可以自己创建然后使用，比如创建一个脚本，材质(Material)，动画控制器，太阳耀斑等，例如，创建一个C#脚本，具体操作是Create->C# Script，然后在脚本中进行编写就行。资源可以作为游戏场景中的对象，也可以作为组件挂载在游戏对象上。

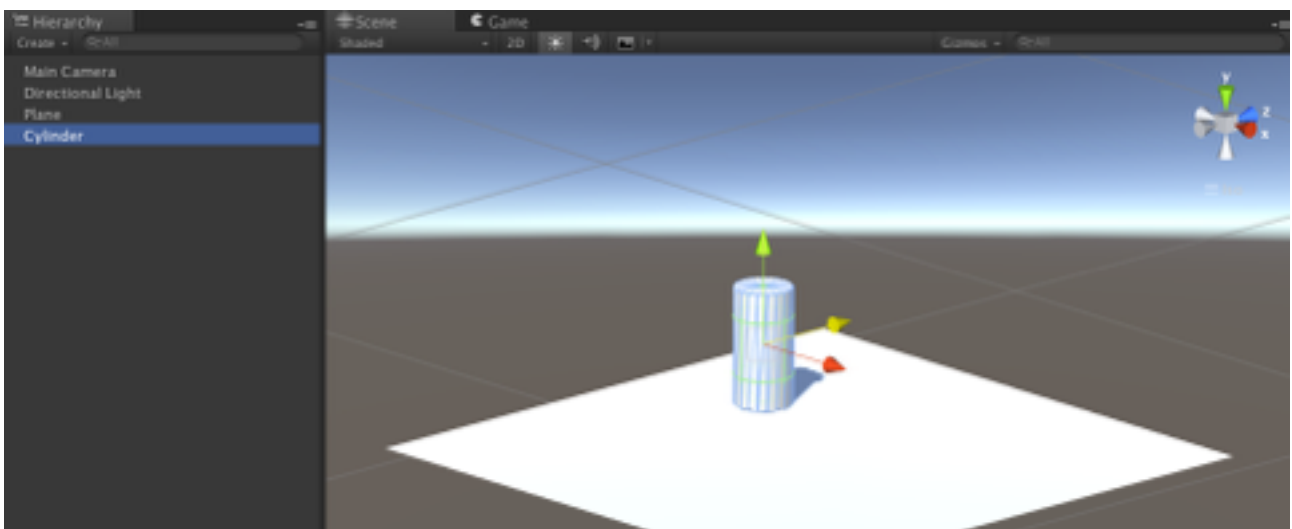
2.1 预设体

何为预设体？Unity3d中的预设体是一种资源，又叫做预制件，当游戏中需要大量的去创建一些具有相同属性的对象，就会将该对象做成预设体，方便使用。比如游戏场景中有一把枪不断的发射子弹，那么子弹对象

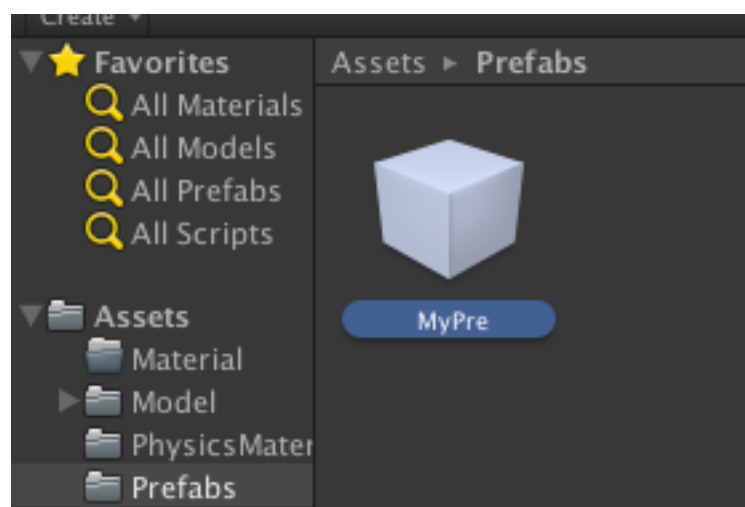
就要一直创建，而且子弹对象也具有一模一样的属性，如果不用预设体，将会造成非常多的操作，影响开发的效率。


创建一个预设体也很方便，在Project视图，右键，Create->Prefab，新创建的预设体图标为，文件后缀名为.prefab。如果图标是这样，那么预设体还没有绑定某个特定的游戏对象，下面通过一个案例讲解预设体的创建、绑定、使用。

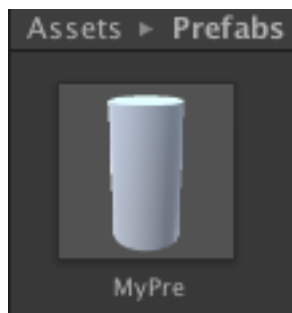
(1) 新建一个场景，创建一个地板(Plane)，和一个圆柱(Cylinder)，如下图所示。



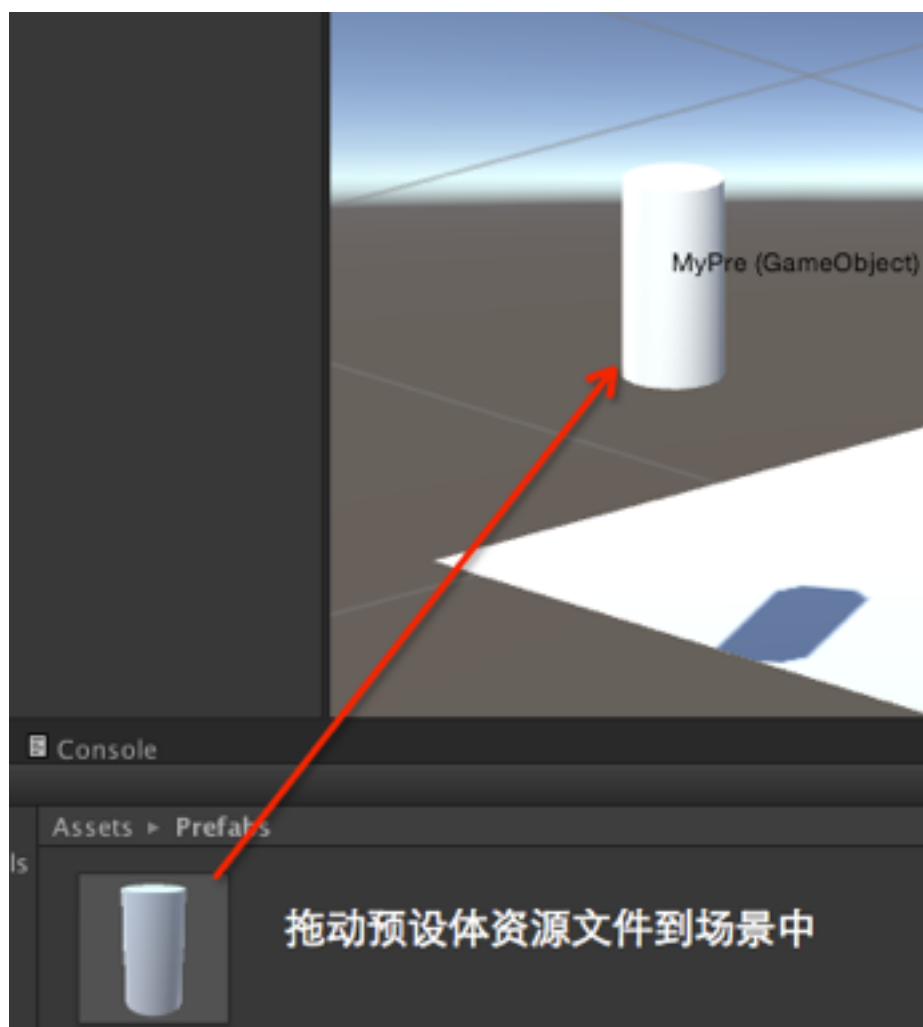
(2) 在Assets目录下，创建一个文件夹Prefabs，用于存放游戏中创建的预设体文件。创建完毕后如下所示。



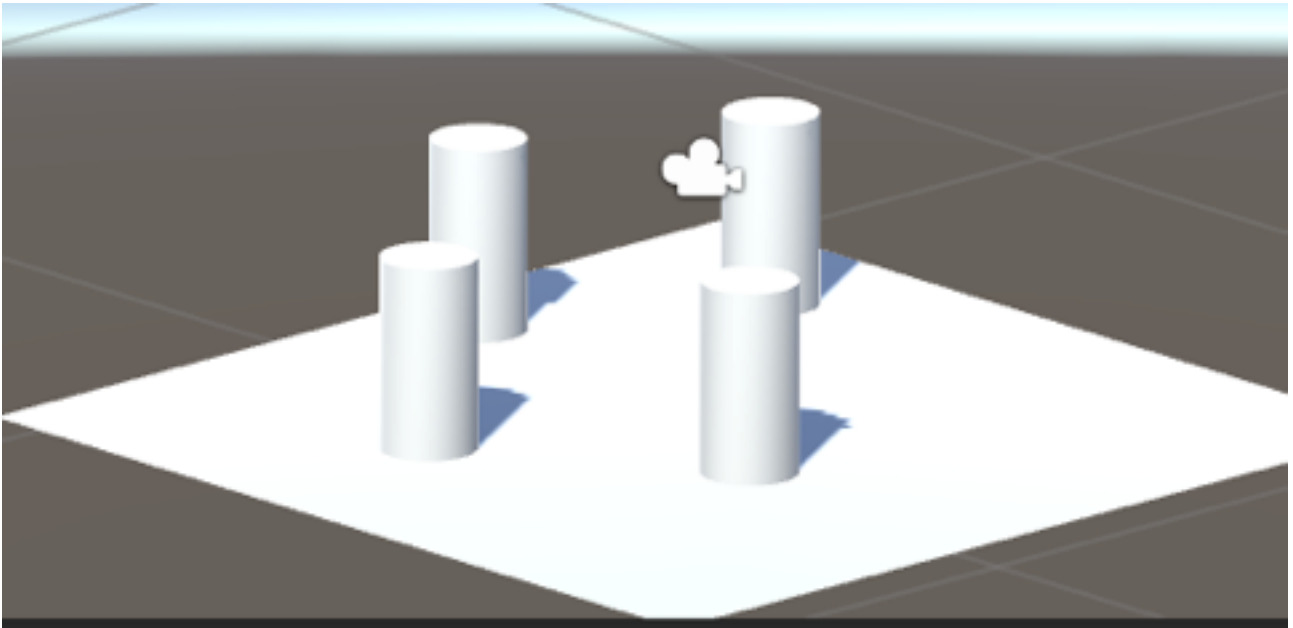
(3) 在层级视图中鼠标选中Cylinder对象不放，直接将其拖入到预设体“MyPre”上，预设体变成如下所示。图标变成。



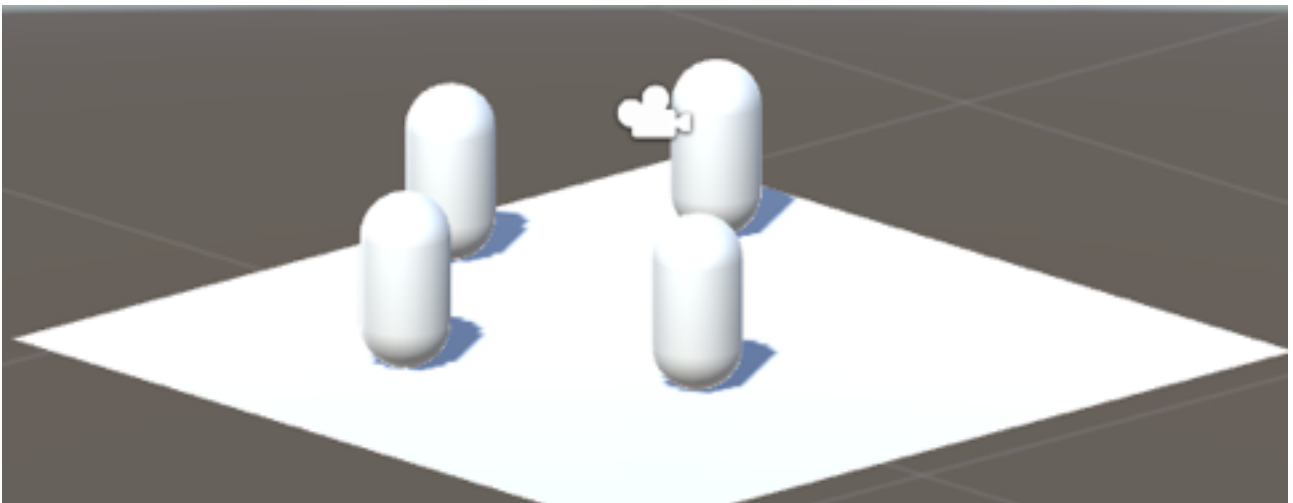
(4) 经过3步骤，预设体与对象的绑定已经成功。同时层级视图中的Cylinder对象的颜色变成蓝色。删除掉层级视图中的Cylinder对象。然后将预设体直接拖入到游戏场景中去。如下所示。



(5) 重复4中操作预设体的步骤，使游戏场景中出现多个MyPre预设体。如下。



(6) 选中预设体，在检视视图中会看到，相对于之前没有绑定好的预设体文件的属性，绑定的预设体多出几个属性来，我们尝试修改网格过滤中的网格(如下)，然后观察到场景中所有的Cylinder对象的形状都变成了胶囊体。



(7) 因此，改变预设体就能改变所有对象。不需要一个一个去修改。

预设体在实际开发当中非常有用，上面的案例是通过直接操作预设体，将其拖入到场景中去，但如果是子弹的话，显然不合理，下节将介绍通过预设体用代码来动态的创建对象。

2.2 创建脚本

在Unity当中，脚本可以作为资源然后挂载在游戏对象上，那么挂载在游戏对象上的脚本文件就是脚本组件。

创建脚本的方式有很多种，在Project视图中可以右键，Create->C# Scripts，默认情况下，脚本命名为NewBehaviourScript，双击脚本，默认会以Mono编辑器打开脚本，然后在Mono中进行编辑(如果换其他的脚本编辑器，需要在偏好设置里去修改)，这里有一点需要注意，脚本的类名必须跟脚本文件名相同，一旦创建成功，就不要轻易的去修改脚本的文件名称。关于Unity API在脚本篇会学习到。

我们还是先来解释下，新创建的脚本中一些函数、类的意义。如下所示，新创建的脚本包含以下内容：

```
1using UnityEngine;
2using System.Collections;
3
4public class NewBehaviourScript : MonoBehaviour {
5
6    // Use this for initialization
7    void Start () {
8
9    }
10
11    // Update is called once per frame
12    void Update () {
13
14    }
15}
```

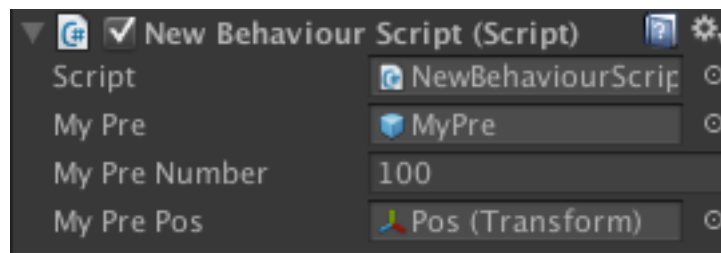
第一行在我们学习了C#之后，也知道这是引入命名空间，同第二行。第四行是一个类，类名是NewBehaviourScript，其继承自MonoBehavior类，第七行和第十二行都是Unity中最常用的回调方法，这些方法不需要手动调用，系统会自动调用这些回调方法，其中Start方法在游戏场景加载时被调用，在该方法内可以写一些游戏场景初始化之类的代码，Update方法会在每一帧渲染之前调用，大部分游戏代码在这里执

行，除了物理部分的代码。类似的回调方法还有很多比如FixedUpdate、Awake，跟物理组件相关的回调方法，比如OnCollisionEnter，OnTriggerStay等。

打开上节所讲的预设体场景，我们通过脚本动态创建对象。代码如下所示。

```
1using UnityEngine;
2using System.Collections;
3
4public class NewBehaviourScript : MonoBehaviour {
5    public GameObject myPre;//在检视视图中可以将一个对象
    (或者预设体)赋值给它
6    public int myPreNumber = 2;//要创建的对象个数
7    public Transform myPrePos;//对象创建后要设置的参考位置
8    // 场景加载时就会调用(它在Update前调用)
9    void Start () {
10        for(int i = 0;i < myPreNumber;i++){
11            float mX = myPrePos.position.x;//参考位置
            的x坐标
12            float mY = myPrePos.position.y;//参考位置
            的y坐标
13            float mZ = myPrePos.position.z;//参考位置
            的z坐标
14
15            Instantiate(myPre,new Vector3(mX,mY,mZ
            +0.8f*i),myPre.transform.rotation);//Instantiate为动态
            初始化对象
16        }
17    }
18
19    //场景运行起来后，每帧调用一次
20    void Update () {
21
22    }
23}
```

在检视视图中，我们可以对上面代码的public变量进行简单赋值，如下。



关于脚本这部分内容，在脚本篇会继续学习。慢慢练习就不会生疏了。