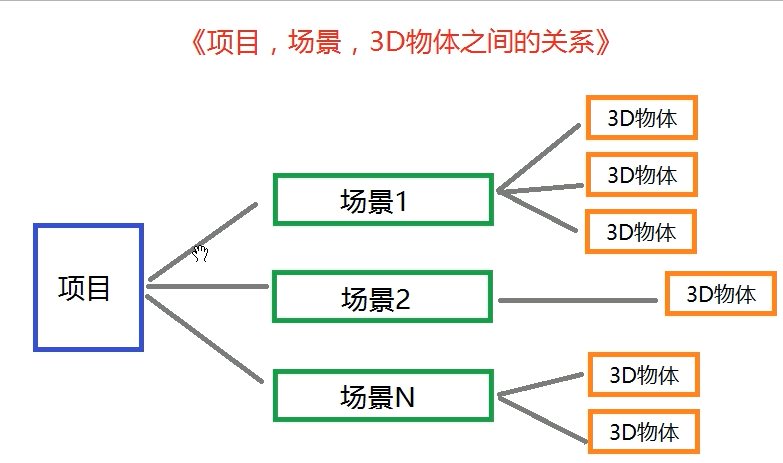
快捷键操作：

Alt+左键：旋转场景

Alt+右键：将场景拉远和拉近

项目、场景、3D物体之间的关系

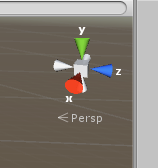
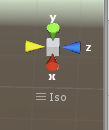


通过工具栏上的“Local”和“Global”可以切换物体自身和世界的坐标系。

世界坐标观察模式

Persp：透视模式，近大远小。

ISO：正交模式，远近一样大。

# 3D模型

Ctrl+D：复制物体

F：快速聚焦某个物体

工具栏对应：Q、W、E、R、T

## 材质球与贴图

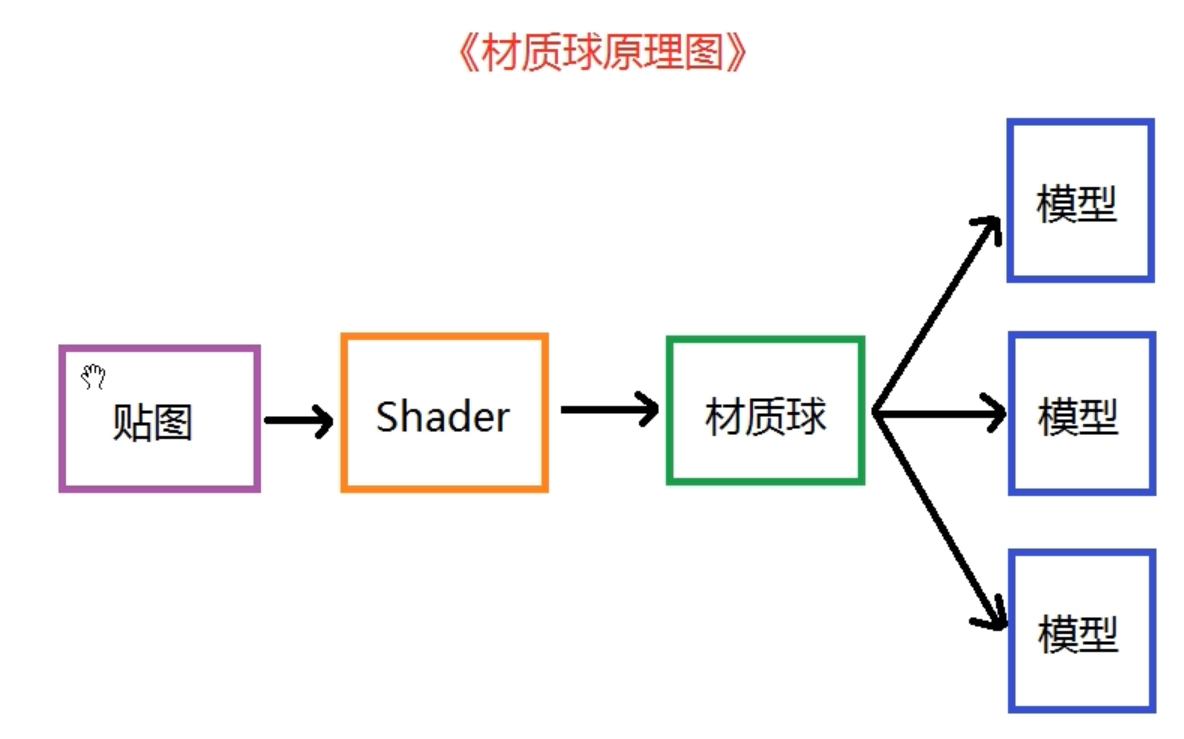
### 材质球与贴图介绍

材质球：Material，材质：物体的质地，物体看起来是什么做的。

贴图：Texture，就是“普通的图片”，用于材质球的显示。

Shader着色器

### 材质球原理图



### 父子关系

有时候一个物体由很多的小物体组成，管理小物体很不方便，这时候就可以在Hierarchy面板中新建一个空物体，然后将其他物体放入到该物体中，就可以将该空物体作为一个所选择的小物体的父类，方便管理。

在该面板上任意两个物体都可以建立父子关系。

### 预制体

Prefab，预先准备好的物体，可以重复利用和使用。比如，重复地面的砖块，栏杆等，重复的物体。

#### 管理预制体

在Assets中，建立Perfabs文件夹。

将层级面板中的物体拖拽到Perject面板的Assets中，即可创建一个预制体。预制体的文件后缀是.perfab

在层级面板中，预制体与非预制体的区别是，预制体是蓝色的，非预制体是白色的。



#### 预制体的好处

编辑其中一个预制体，然后点击Inspector面板中Perfab中的Apply按钮，所有预制体都可以获得相同的效果。

#### 寻找一个预制体在哪个文件中

选择相应的预制体，然后在Inspector面板中点击Perfab中的select按钮，预制体的位置就在Perject面板中显示出来了。

## 6、基本组件之摄像机

**场景美化**

1. 新建plane作为“地板”
2. 给地板附上材质球
3. 附上之后发现，瓷砖太大，需要进行密度的加大，点击材质球，在Inspector面板中，调整Tiling(贴瓷砖的意思)的值。

#### 游戏窗口

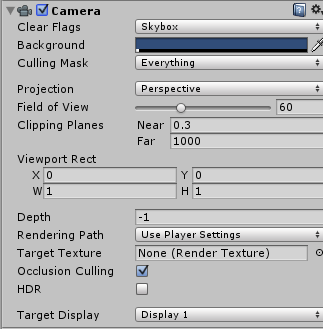
中间上部的游戏运行按钮屏幕剪辑，左边那个，如果选中，则表示处于游戏运行状态，这时切换下面的Scene面板进行编辑，都不会进行保存，重新点击运行按钮会恢复原状。

下面的Game窗口表示游戏预览窗口。

### 摄像机的常用操作

将摄像机快速到达当前的视角，选择GameObject菜单的Align With View，快捷键为Ctrl+Shift+F

#### 摄像机的相关属性



1. Clear Flags

Skybox：天空盒

Solid Color：固定颜色(纯色)，选择下面的background可以选择具体的颜色。

2、Projection(投影)

Perspective：透视模式，3D游戏使用

Orthographic：正交模式，2D游戏使用

1. Clapping planes(切割面)

摄像机有两个平行的平面，近的一个远的一个，Near近的那个的距离，Far远的那个的距离。

# 第7课 基本组件之灯光

## 灯光简介

当我们创建一个新的Scene的时候，默认会带有两个物体，一个摄像机，一个灯光。

隐藏和显示物体：，点击复选框就是显示，不选就是隐藏。

## 方向光

Directional Light，用于模拟太阳，模拟自然光。方向光任何地方都能照射到，就和太阳一样，但是要注意照射方向。

### 属性

Type：用于切换灯光的类型

Intensity：光照强度

Shadow Type：阴影类型

## 点光源

Point Light，用于模拟电灯泡的照射效果。

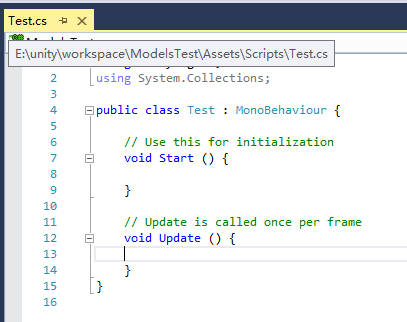
## 聚光灯

Spot Light

# 第8课 组件与脚本

每个物体右侧界面对应的大标题就是一个组件。

## 脚本代码介绍



Start()方法：当游戏运行起来，就会马上执行，且只执行一次。

Update()方法：循环调用，每帧调用一次。一秒钟大概执行60次。

## 使用脚本

1. 直接将脚本拖拽到Hierarchy面板上物体身上。
2. 直接将脚本拖拽到游戏物体的Inspector面板上。

运行游戏，脚本就会执行。

## Console面板

Clear：清除按钮，清除控制台中的信息。

Collapse：折叠功能，将相同的内容合并到一行显示，更新后面的数字。

Clear And Play：运行时清除之前的内容。

## 获取键盘输入

Input.GetKey(); //持续按下某键后，持续返回true

Input.GetKeyDown(); //按下某键的瞬间，返回true

Input.GetKeyUp(); //抬起某键的瞬间，返回true

## 2、获取鼠标的输入

Input.GetMouseButton(); //持续按下某键后，持续返回true

Input.GetMouseButtonDown(); //按下某键的一瞬间，返回true

Input.GetMouseButtonUp(); //抬起某键的一瞬间，返回true

返回值，bool类型

参数，鼠标按键索引值，0->左键，1->右键，2->中键

## 第10课 使用变换组件移动游戏物体

## 1、变换组件移动物体

1. 相关方法

gameObject.GetComponent<T>():获取相应组件的引用。查找当前游戏物体上的某个组件，然后保存它的引用。

Transform.Translate(Vector3, Space)：移动物体的位置。

游戏物体往某个方向移动；以自身坐标系或世界坐标系。

相关参数：Vector是向量，Vector3是三维向量。Vector3[struct]本质上是一个结构体，可以表示一个方向，也可以表示一个位置。

Space[enum]：空间

Space.Self：表示物体自身的坐标系

Space.World：表示物体所在的世界的坐标系

## 2、使用键盘控制移动的方向

使用键盘上的“W，A，S，D”来控制游戏物体的移动。

# 第11课 物理组件之刚体

## 1、变换组件运动特点

普通的移动会穿透物体并且没有重力

## 2、刚体组件简介

### 刚体简介

刚体：Rigidbody，属于物理类组件

作用：添加了刚体组件的游戏物体，就有了重力，就会做自由落体运动，也就意味着可以像现实中的物体一样运动。

### 给物体添加刚体组件

选中游戏物体—》菜单Component—》Physics—》Rigidbody

1、Mass质量的单位是千克

2、Drag(阻力)

空气阻力，0表示无阻力，值很大的时候物体会停止运动。

3、Angular Drag(角阻力)

大体可以理解为惯性，比如两个物体碰撞时会发生旋转，角阻力大的，惯性大，转动的慢，小的转动的快。

4、Use Gravity(是否使用重力)

默认是勾选的

## 3、刚体组件属性

## 4、使用刚体移动物体

1. 相关方法

Rigidbody.MovePosition(Vector3)：使用刚体移动物体的位置。

使用刚体移动物体，物体是根据世界坐标系的方向移动的。

使用刚体移动物体，物体会触发物理相关的事件。

1. 参数

MovePosition中的Vector3要使用“当前位置”+方向的方式。

Transform.Position:属性 当前物体的位置。

# 第12课 物体组件之碰撞体

## 1、碰撞体简介

物体之间的碰撞，其实是目标物体的碰撞体组件与自身物体的碰撞体组件碰撞。碰撞体可以理解为包裹在物体外部的“骨骼”。两个物体都要有碰撞体才能发生碰撞。

模型只要加了刚体就必须加碰撞体。

### 给物体添加碰撞体组件

选中游戏物体，然后选择菜单Component，Physics，XXX Collider

Unity中创建的基本物体(cube，sphere等)都带有碰撞组件，cube就是cube collider碰撞体，等等。

* 一个物体只有碰撞体属性，被其他物体碰撞不会动，因为没有刚体属性，也就没有一些物理的移动属性。这个经常用在游戏中的障碍物中。
* 如果只有刚体属性，没有碰撞体属性，那么模型会直接掉下去

## 2、Box Collider

盒子碰撞体，形状是立方体形，用于包裹类似“立方体的模型，比如：盒子，箱子，门，房子等。

***组建属性***

Center[中心点]

设置Box Collider的中心点

Size[中心点]

设置Box Collider的大小

## 3、Sphere Collider

球形碰撞体，形状是球形，用于包裹类似“球形”的模型。

***组件属性***

Radius[半径]

设置Sphere Collider的大小。

## 4、Capsule Collider

Height，设置Capsule Collider的高度

Direction[方向]，设置Capsule Collider的高度方向(轴向)

## 5、Mesh Collider

网格碰撞体，用于包裹复杂结构的碰撞体。

# 第13课 刚体常用方法介绍

## 1、AddForce()

给刚体添加一个力，让刚体按照“世界坐标系”运动。

Rigidbody.AddForce(Vector3, ForceMode)

Vector3：力的方向和大小；

ForceMode：力的模型[enum类型]

***ForceMode参数***

以什么样的方式添加力给刚体。

Acceleration：加速度；

Force：(力)这种模式通常用于设置真实的物理。

Impulse：(冲击力)这种模式通常用于设置瞬间发生的力

VelocityChange：速度的变化。

## 2、AddRelativeForce()

1、给物体添加一个力，让物体按照“自身坐标系”进行运动。

2、代码

Rigidbody.AddRelativeForce(Vector3, ForceMode)

Vector3：力的方向和大小

ForceMode：力的模式[enum枚举]

### 3、FixedUpdate()

固定更新方法

所有和物理相关的操作，代码都要写在FixedUpdate()方法体内。固定更新的时间间隔是0.02秒，1秒执行50次。

Edit->Project Settings->Time面板中的Fixed Timestep参数设置

Update()方法是每帧执行一次。画面每渲染完一次，就是一帧，每帧的时间是不固定的。

在Update()方法中执行物理操作，会出现卡顿的情况。

## 2、AddRelativeForce()

## 3、FixedUpdate()

# 第14课 刚体碰撞事件检测与处理

## 1、碰撞事件介绍

## 2、碰撞事件检测方法

1. OnCollisionEnter(Collision)

当碰撞开始时调用，只会调用该方法一次。

Collision参数是一个类，用于传递碰撞信息。

Collision.gameObject属性，与当前物体碰撞的物体的引用。

gameObject.name属性，当前物体的名称。

即，被撞物体的信息会传递给Collision参数

1. OnCollisionExit(Collision)

当碰撞结束时调用，只会调用该方法一次。

1. OnCollisionStay(Collision)

当碰撞进行中时，会持续调用该方法。

# 第15课 刚体触发事件监测与处理

## 1、触发事件简介

### 触发器

将碰撞体组件属性面板上的“Is Trigger”选项选中，当前的游戏物体的碰撞体就变成了触发器。

移动的刚体物体会穿透具有触发器的碰撞体。

### 触发事件

当一个用刚体控制的物体进入到另外一个物体的触发器范围内，就是触发事件。

触发用途：不与目标物体发生直接的碰撞，而是只要进入目标物体的“触发范围”，就能执行某些特定操作。

### 触发事件监测方法

1、OnTriggerEnter(Collider)

当进入触发范围时开始时调用，只会调用一次。

2、OnTriggerExit(Collider)

当离开触发范围时开始调用，只会调用该方法一次。

3、OnTriggerStay(Collider)

在触发范围内，会持续调用该方法。

4、Collider参数

碰撞体，一个类。作用：用于传递触发参数。

Collider.gameObject属性，进入触发范围内的目标物体的引用。

gameObject.name属性，当前物体的名字。

# 第16课 网格组件之网格过滤器和渲染器

## 1、网格过滤器组件

网格过滤器：Mesh Filter

该组件只有一个“Mesh”属性，用于设置当前游戏物体使用哪个模型进行展示。

Mesh：网格，也就是模型。

屏幕剪辑

该组件会控制物体的形状，比如选择Cube，物体就是正方体，选择Sphere，物体就是球形等。

## 2、网格渲染器组件

Mesh Renderer

该组件用于“渲染”显示模型。如果没有该组件，模型就不会显示。

### 属性

#### Cast Shadows[投射阴影]

On：开启阴影显示

Off：关闭阴影显示

#### Receive Shadow[接收阴影]

#### Materials材质球

用于选择用哪个材质球来渲染当前的模型(Mesh)

我们拖拽到游戏物体身上的材质球，其实就是赋予给了这个组件的这个属性上。

# 第17课 疯狂教室案例开发

## 1、模型旋转实现开门效果

### 1、模型中心点

模型身上的坐标轴的中心点，也就是我们模型的中心点。

模型的位置，旋转，缩放都是相对于模型的中心点来进行变化的。

### 2、改变模型中心点

创建一个空物体，创建父子关系，通过父物体来实现子物体。也就间接的改变了模型的中心点。

在引擎中有一个按钮可以改变物体的中心点，屏幕剪辑，左侧的按钮会改变物体中心点的位置。

当选择两个模型的时候，如果选中center，中心点就处在两个物体中心的位置；如果选择Pivot，那么中心点就在后面的模型身上。

### 3、使用键盘按键来实现开关门

使用Transform.Rotate(Vector3, float)旋转模型：

Vector3：沿某个轴向旋转。Float：旋转的度数。

# 查找游戏物体

GameObject.Find(name)通过名称查找游戏物体

## 2、触发器实现开关门

private void OnTriggerEnter(Collider other)

{

if (other.gameObject.name == "student")

{

m\_Door.openDoor();

}

}

private void OnTriggerExit(Collider other)

{

if (other.gameObject.name == "student")

{

m\_Door.closeDoor();

}

}

# 第18课 疯狂教室案例(下)

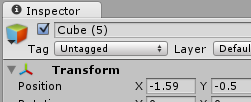
## 1、通过Tag标签查找物体

### 1、简介

同一类的模型，我们可以根据需要给他们设置成统一的标签。

### 2、给模型添加Tag标签

选中一个模型，在模型的Inspector顶部添加标签，



### 3、通过查找标签查找N个物体

GameObject.FindGameObjectWithTag(string):[静态方法]。通过特定的标签，查找到所有“贴有”该标签的游戏物体，返回一个数组。string是标签名

### 4、通过for循环来输出模型信息

通过for循环遍历FindObjectsWithTag()方法返回的数组，输出游戏物体的信息。

### 5、通过键盘按键实现桌椅跳动

按下某键，桌椅全部上移2米；按下某键，桌椅全部下移2米。

# 游戏发布

## 1、Unity发布PC版游戏

### Build Settings【生成设置】

File🡪Build Settings 弹出项目生成设置面板

1. 选择要发布到的平台
2. 添加要发布的场景

### Player Settings【详细设置】

# 第1课 旧版UI组件之GUIText

### 2、常用UI系统简介

＜１＞NGUI

NGUI是一款使用最多的第三方的UI插件。

<2>UGUI

是unity自带的一套UI系统。

<3>OnGUI

现在主要用于unity界面的扩展。

NGUI和UGUI都是“所见即所得”(即拖拽)就可以实现UI的组件，而OnGUI类似Web中的html和css的编写。

### 3、GUIText组件

GUIText主要用于文字的显示。

#### 2、如何创建GUIText

1. 新建一个空游戏物体
2. 给这个空游戏物体添加“GUIText”组件，步骤如下：

Component->Rendering->GUIText

1. 这个组件可以用来显示文字了。

#### 3、GUIText常用的属性

**Text(文字)**

设置GUIText组件显示的文字

**font size**

默认为0

**Pixel Offset**

通过X轴和Y轴两个值，设置文本组件在场景中显示的位置。

注意：GUIText组件只能在Game视图下看到，在Scene视图下看不到

Color颜色

# 第2课 旧版UI组件之GUITexture

## 1、简介

该组件主要用来显示图片。

## 2、创建GUITexture

新建一个空物体，为该物体添加“GUITexture”组件，步骤如下：Component->Rendering->GUITexture

这个游戏物体就可以用来展示游戏图片了。

注意：该组件也是只能在Game视图下才能看到。

## 3、属性

Color

默认状态下，Color属性是不会影响图片显示效果的。

## 4、鼠标事件

之前讲解的Input类下面的鼠标输入是全局的，且只能获取鼠标的按键状态。

今天讲解的鼠标事件是挂在到某一个游戏物体身上的，且只有我们的鼠标操作该游戏物体时，对应的鼠标事件才会生效。

### 1、常用的鼠标事件方法

OnMouseEnter()：鼠标进入

OnMouseExit()：鼠标离开

OnMouseDown()：鼠标按下[单击]

### 2、颜色参数

Color结构体，里面有常用的颜色，Color.red Color.green Color.blue

# 第3课 特效组件之TrailRenderer

## 1、简介

TrailRenderer，拖尾渲染器，作用是用于渲染显示“拖尾特效”。

## 2、拖尾在游戏中的运用

发射出去的子弹或者炮弹等。

## 3、创建拖尾特效

1. 新建一个空游戏物体；
2. 给这个空游戏物体添加TrailRenderer组件，步骤如下：Component—》Effects—》TrailRenderer
3. 在Sence面板移动这个游戏物体的位置，就可以看到最原始的拖尾效果。

## 4、TrailRenderer材质球

### 1、Materials

这个组件是使用哪个材质球来渲染拖尾。

### 2、注意事项

Unity中，以Renderer为后缀进行命名的组件，都需要给他们设置材质球，比如之前讲解的Mesh Renderer。

当这类组件身上没有材质球的时候，默认就会显示“粉红色”。

### 3、制作透明材质球

1. 创建一个材质球；
2. 设置该材质球的Shader为Particles/Additive(粒子/添加物)
3. 给材质球拖拽赋予贴图；

### 4、TrailRenderer常用属性

#### 1、Time

设置拖尾特效的持续时间

#### 2、Start Width

设置拖尾开始时的宽度

#### 3、End Width

设置拖尾结束时的宽度

#### 4、Color

设置拖尾显示的颜色

在实际开发中，美工人员给我们的图往往是“黑白图“，这种图片中黑色是透明区域，白色是不透明区域，可以简单记忆为”黑透白不透“。我们通过设置这里的Color属性，来让白色区域显示成特定的颜色。

可以简单理解，黑色无论设置成什么颜色都是黑色。

在实际运用中，会将空物体作为子弹等的父物体，给空物体添加拖尾特效，然后控制空物体就可以实现拖尾特效。

# 第4课 特效组件之LineRenderer

## 1、LineRenderer简介

## 2、在游戏中的运用

线特效经常用于在游戏中渲染激光效果，或者用于子弹瞄准。

## 3、创建线特效

1. 创建一个空物体
2. 给这个空物体添加LineRenderer组件，步骤如下：Component->Effects->Line Renderer

## 4、LineRenderer材质球

### 1、Material(材质球)

### 2、制作透明材质球

1. 创建一个材质球
2. 设置该材质球的Shader为Particles/Additive;(粒子/添加物)
3. 给材质球拖拽赋予贴图

### 3、LineRenderer常用属性

1、Positions(位置)

设置线的开始点和结束点的位置

2、Start Width

设置线开始时的宽度

3、End Width

设置线结束时的宽度

4、Start Color

5、End Color

# 第5课 声音组件之AudioSource

1、AudioClip声音剪辑

2、AudioSource组件

## 2、AudioClip

音频剪辑(音频片段)

我们导入到Unity中所有的声音文件，他们在unity中的资源类型都是AudioClip类型。

Unity能使用的声音格式：.aif, .wav, mp3，.ogg

## 3、AudioSource组件

### 1、组件简介

用于播放音频资源，可以看作游戏中的“音响”

### 2、创建AudioSource组件

1. 新建一个空物体
2. 给空物体添加AudioSource组件，Component->Audio->AudioSource

## 4、组件属性

### Play On Awake(在唤醒时播放)

勾选之后，在游戏运行起来之后，就会开始播放

### Loop(循环)

勾选之后，声音会进入“单曲循环”状态。

### Mute(静音)

勾选之后，静音，但是音乐还是处于播放状态。

### Volume(音量)

### Spatial Blend（空间混合）

0是2D声音，1是3D声音

## 4、Audio Listener组件

声音侦听器，类似于人在游戏中的“耳朵”，比如，摄像机是人在游戏中的“眼睛”

该组件在main camera身上，右下角

## 5、AudioSource常用函数

### Play函数

### Stop函数

### Pause函数

# 第6课 常用API之实例化与销毁

## 1、实例化游戏物体

实例化生成物体

GameObject.Instantiate(Object, Vector3, Quaternion)

参数说明：

Object：用于实例化的预制体

Vector3：实例化后生成的物体所在的位置

Quaternion：实例化后生成的物体的旋转的状态

Quaternion.identity：无旋转

如果在物体的脚本中定义了GameObject为public类型，那么在Inspector面板的脚本下面会看到该变量，并且通过拖拽可以给该变量赋值。

## 2、销毁游戏物体

## 3、构造随机位置

位置是用Vector3类型进行表示的。

X，Y，Z三个值确定了物体在三位世界中的位置。

Random.Range(min, max):生成随机数