Cha4-Unity3d 和 C#的双剑合璧 10

在上一课的内容中，我们使用意念攻击的方式痛宰了大魔王。然而，这些只是我们的 YY，玩家是看不到的。

因此，在这一课的内容中，我们将实现游戏对象之间的物理碰撞，并使用脚本来处理碰撞事件，让大魔王尝点厉害~

个人微信号：iseedo
微信公众号：vrlife

10 Unity 和 C#的双剑合璧-当精灵遇到巨人

为了让精灵和巨人之间发生碰撞，首先我们需要给精灵公主游戏对象添加一个名为 Rigidbody 的组件。在 Hierarchy 视图中选中 Lady Fairy 游戏对象，在 Inspector 视图中点击 Add Component，选择 Physics-Rigidbody。

有童鞋可能要问，到底什么是 rigidbody？

它的中文名是刚体。什么是刚体，在发生碰撞时不会产生明显变形的物体就叫刚体。比如日常生活中，我们可以把木制的桌椅和台灯等当做刚体，而水杯里的水显然就不是刚体，而是流体。

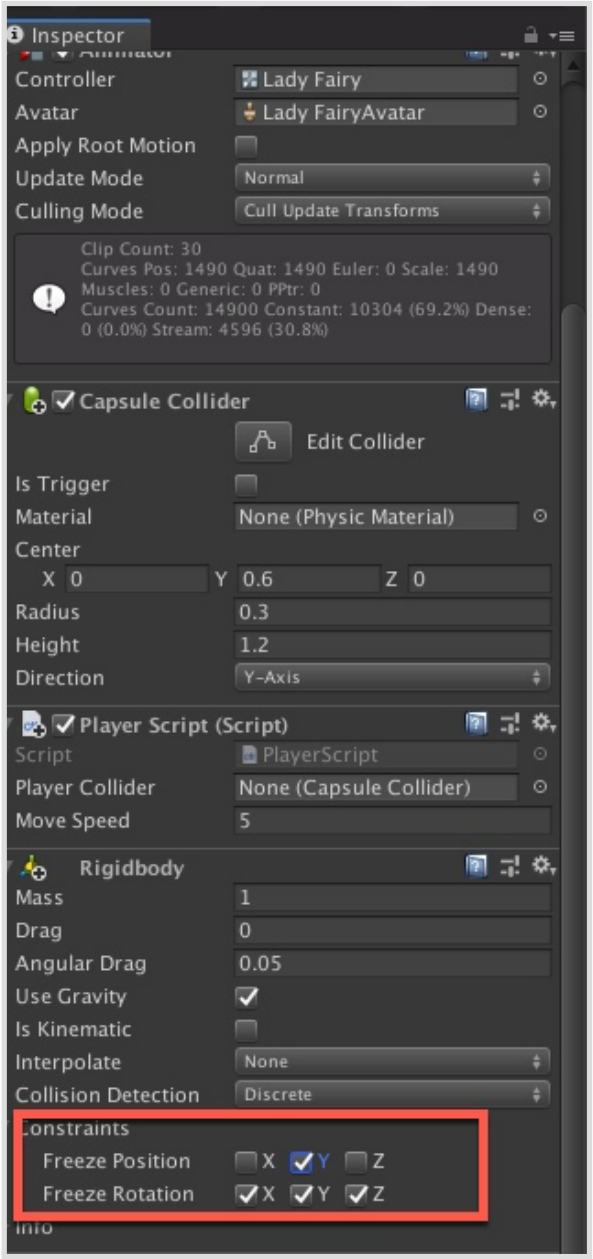
需要注意的是，为了让精灵公主在跟巨人怪物发生碰撞的时候不被装晕过去，我们需要在 Rigidbody 组件的 Constraints（约束）属性处勾选 Freeze Rotation 的 X,Y,Z。此外，还需要勾选 Freeze Position 的 Y，这样精灵公主就不会一撞就飞上天了。

关于 rigidbody 组件的更多知识，可以参考这里：

https://docs.unity3d.com/Manual/class-Rigidbody.html

此外，当给某个游戏对象添加了 Collider 碰撞体组件后，它就会具备跟游戏场景中的其它游戏物体发生碰撞的能力。

幸好之前我们已经给 Lady Fairy 添加了 Capsule Collider（胶囊形状的碰撞体），而 TrollGiant 巨人



的组件中有一个 Mesh Collider（网格碰撞体），这样就可以了吗？

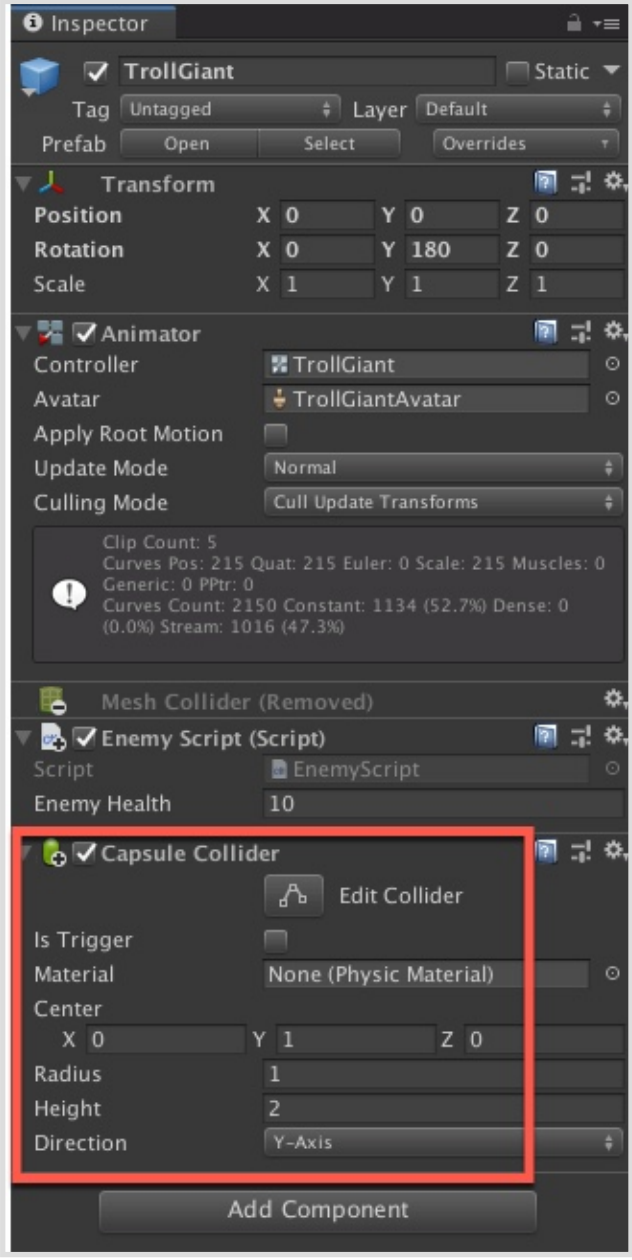
通常来说，我们会选择使用相对简单的碰撞体，因为 Mesh Collider 的使用会受到比较大的限制，特别是当游戏对象的凸面数量过大的时候。

听起来有点复杂，不过，Don’t panic~ 不要恐慌，这些细节知识大家看看就好。

关于 Collider 的详细内容，我们在后续关于物理碰撞机制的章节中会进行更加详细的说明，感兴趣的童鞋可以参考：https://docs.unity3d.com/Manual/CollidersOverview.html

因此这里我们需要手动给 TrollGiant 游戏对象删除 Mesh Collider 组件，然后添加一个 Capsule Collider 对象，并设置中心点、半径和高度等参数。

好了，现在跟碰撞相关的组件设置已经完成。

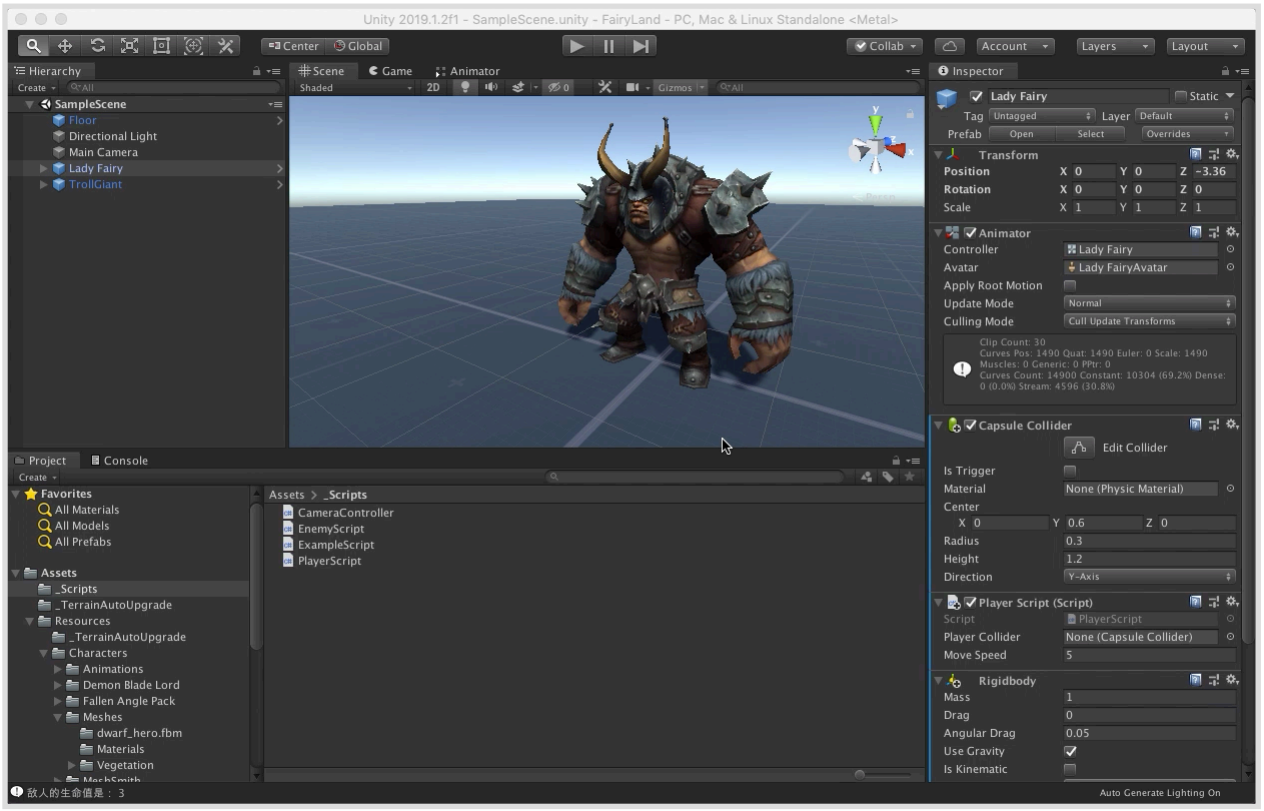


不过还有一个小小的工作要完成，那就是设置游戏对象的 tag。通过设置游戏对象的 tag,我们可以在代码中利用该信息来针对某个具体的游戏对象作出相应的反应~

之前我们在代码中获取某个游戏对象的方式是通过 GameObject 的 Find 方法，而参数是游戏对象的名称。而当我们给游戏对象设置了 tag 之后，就可以使用 tag 来获取一个或多个游戏对象~

在 Hierarchy 视图中选择 Lady Fairy 对象，然后在 Inspector 视图中设置 Tag 属性。

默认情况下，Unity3d 已经提供了一个 Player 选项，我们只需要从下拉列表中选择 Player 就好。



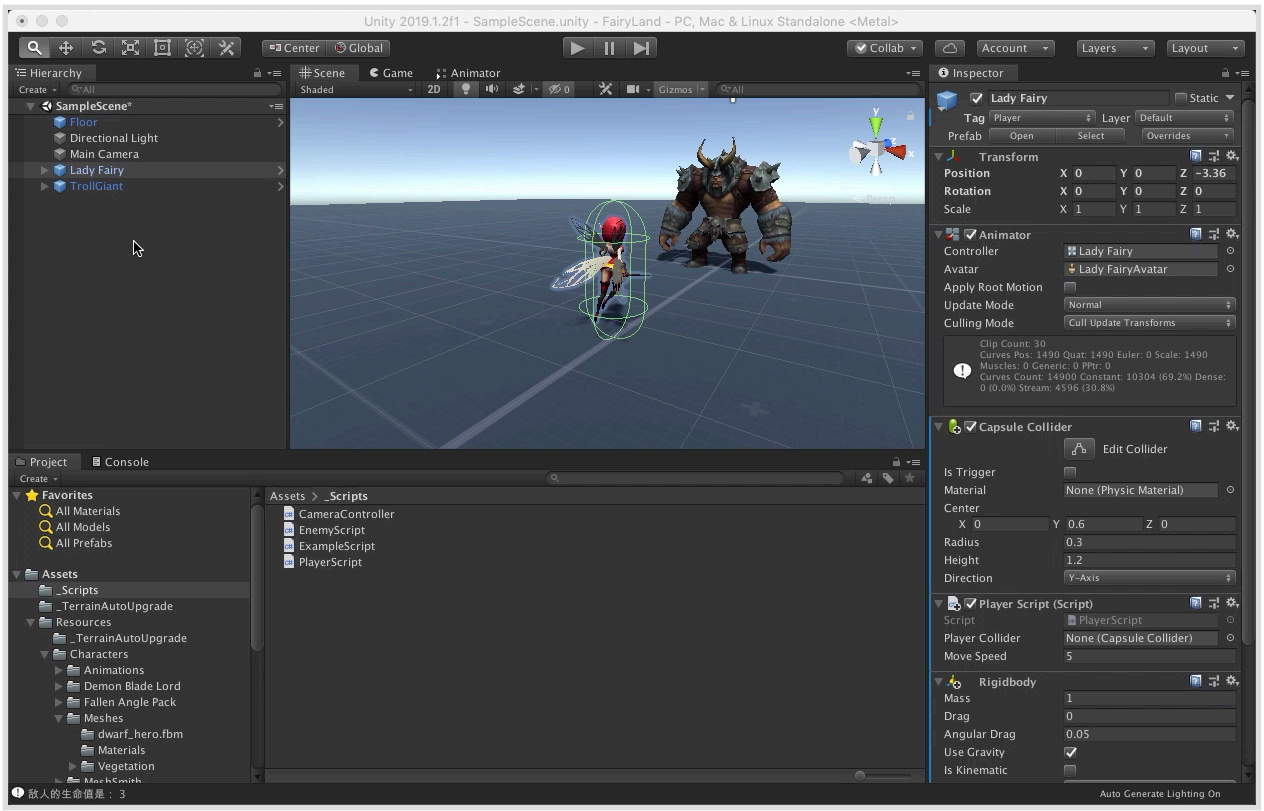
接下来设置巨人怪物的 tag 属性，我们希望给它设置一个名为 enemy 的 tag。

当我们使用刚才的方式设置时，问题来了，默认情况下并没有名为 enemy 的 tag？怎么办呢?

没有天和地，就让我们开天辟地~

具体操作不再赘述，大家参考视频上的操作就好。

在进入代码时间之前，我们还需要完成最后一步操作，那就是多复制几个巨人怪物出来，免得精灵公主嫌虐菜太不过瘾~

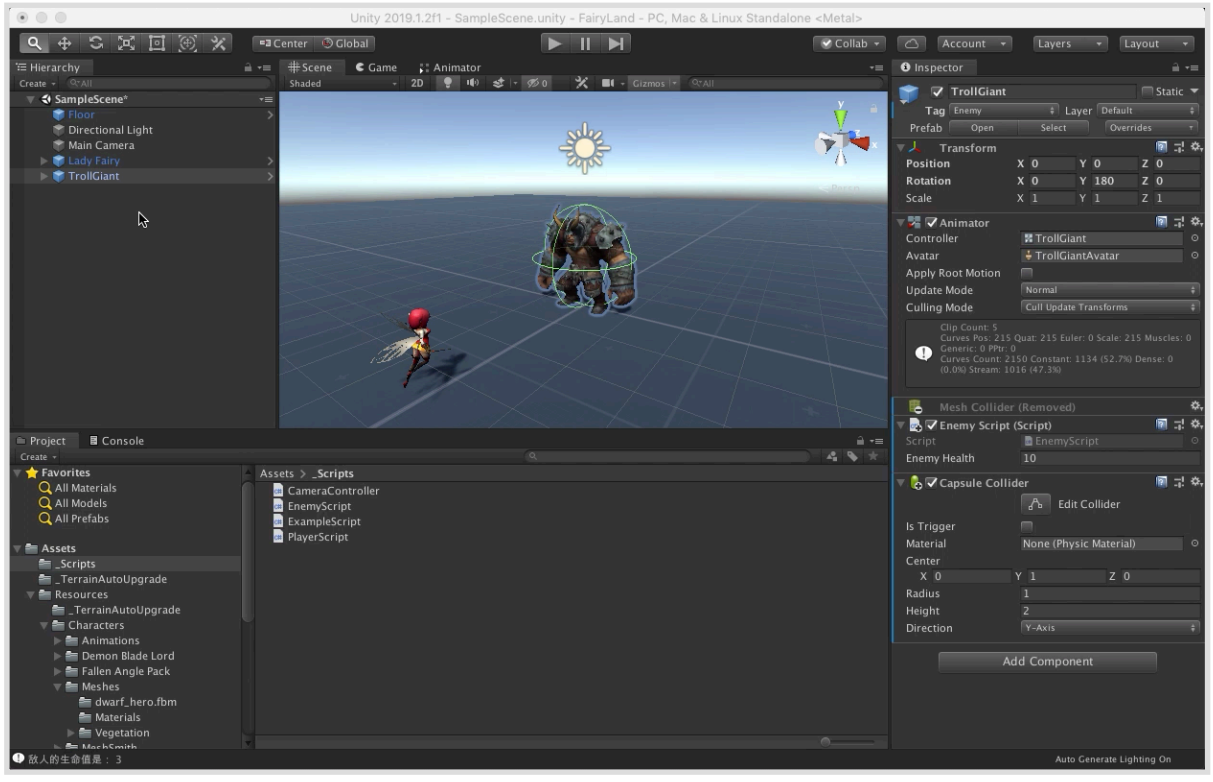


在 Hierarchy 视图中右键单击 TrollGiant，选择 Duplicate，重复操作两次，创建出两个新的怪物。

然后使用工具栏上的移动工具把它们移动到靠谱的位置~

好了，一切就绪，终于可以进入代码时间了~

接下来我们将用脚本的力量来实现物理碰撞机制~



在之前的内容中，我们曾经提到过，Unity3d 提供了大量的系统类和方法，以及事件函数帮我们处理了大量的常规工作。

而关于物理碰撞，Unity3d 也早有准备，我们只需要调用与之相关的事件函数就好了~

那么，怎么才能找到我们所需要的事件函数呢？

首先在浏览器中输入：https://docs.unity3d.com/

此时会自动打开最新版的 Unity User Manual，在上方的搜索栏中输入 event functions（事件函数），回车可以看到搜索结果。

点击 Event Functions 链接，就可以进入对应的页面：https://docs.unity3d.com/Manual/EventFunctions.html

向下滚动鼠标，找到 Physics events，也就是跟物理碰撞相关的事件。

可以看到，Unity 提供了三个事件函数供我们使用，分别是 OnCollisionEnter, OnCollisionStay 和 OnCollisionExit。

如果你 e 文过得去，应该很容易就明白这三个事件函数的用途。

其中 OnCollisionEnter 是当碰撞发生的时候，OnCollisionStay 是当碰撞进行中的时候，而 OnCollisionExit 则是当碰撞结束的时候。

除了这三个事件函数之外，Unity3d 还提供了 OnTriggerEnter, OnTriggerStay 和 OnTriggerExit。

当游戏对象的 collider 碰撞体勾选了 Trigger 属性的时候，就需要改用这三个事件函数。

那么 OnCollisonxxx 和 OnTriggerxxx 这两类事件函数的区别是什么呢？

很简单，用传统的玻璃门和感应门来做对比~

对于传统的玻璃门，我们必须用手或身体推开门，那么 OnCollisionxxx 就比较适合这类场合。而对于感应门，只要门感应到了人类身体的接近，就会自动打开，OnTriggerxxx 显然更适合这类场合。

好了，看到这里，你应该明白我们应该用哪种事件函数了。

双击 PlayerScript.cs，在 Visual Studio 中将其打开。

在 Update()方法之后添加一个新的方法（或者说事件函数）。

```
    //16.当碰撞开始的时候执行该事件函数
    private void OnCollisionEnter(Collision collision)
    {
        //17.如果碰撞的游戏对象的 tag 是 Enemy
        if(collision.gameObject.tag == "Enemy")
        {
            //敌人的生命值减少
            enemyScript.enemyHealth--;
            //在 Console 视图中输出敌人生命值的相关信息
            Debug.Log("敌人的生命值是： " +
enemyScript.enemyHealth);
        }
    }
```

这个新的事件函数的代码不难理解：

1.当检测到碰撞的时候就会执行这个事件函数

2.如果碰撞的游戏对象的 tag 是 Enemy，那么执行下面的操作

（1）让敌人的生命值减少

（2）在 Console 视图中输出相关信息

好了，接下来别忘了删除 Update()方法中注释行 15 之后的一堆代码，因为现在我们不再只是使用意念攻击了~

现在是该精灵公主🧚和巨人怪物们表演的时候了。

回到 Unity 编辑器，点击工具栏上的预览播放按钮，然后。。。

且慢，巨人怪物已经在自动播放动画，还有精灵公主也是。当然，精灵公主的翩翩舞姿我们可以欣赏，但是巨人怪物的动画暂时还不需要。在后续的内容中，我们会详细给大家演示~

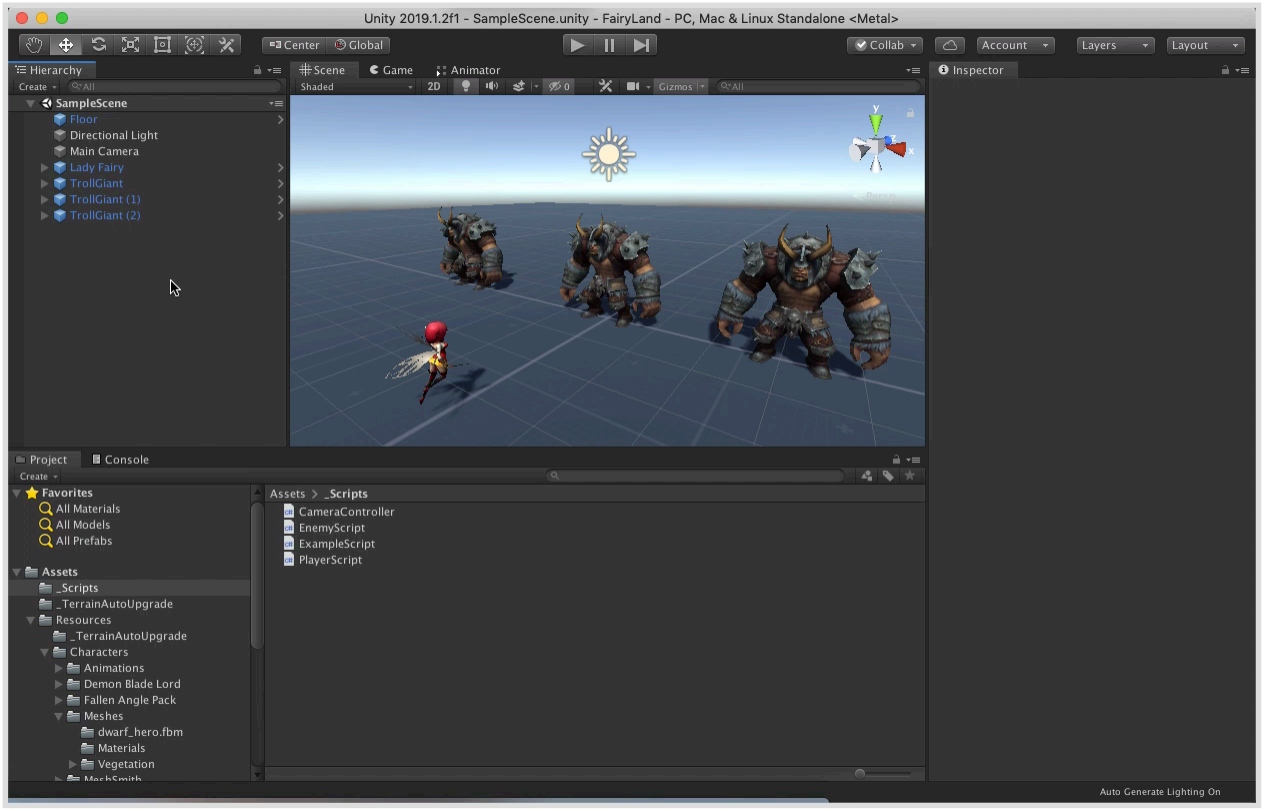
为此，我们先暂时删掉巨人怪物的动画组件。

在 Hierarchy 视图中选中三个巨人怪物，然后在 Inspector 视图中删除 Animator 组件。

好了，看起来应该正常了。

让可爱的精灵公主开始她愉快的表演吧~然鹅。。。

当预览开始后，细心的童鞋会发现，最开始添加的巨人怪物表现正常，该掉血的就掉血。



但是其它两个巨人怪物被精灵公主撞击后，竟然展示出了“无敌”特性，这还了得？！

但是诡异的是，Console 视图中显示怪物的生命值竟然在下降，而 Inspector 视图中的 health 生命值属性并没有发生变化。这一切究竟是怎么回事？？

让我们回到 PlayerScript.cs，看看到底是哪里出了问题？

在 Start()方法体的最后，我们使用下面的两行代码获取了敌人对象。

```
        //13.获取敌人对象
        enemy = GameObject.Find("TrollGiant");

        //14.获取敌人对象上的脚本组件
        enemyScript = enemy.GetComponent<EnemyScript>();
```

但是请注意，这里我们用到了 GameObject.Find 这个系统方法，传递的参数则是游戏对象的名称。

而在 Hierarchy 视图中，只有一个名为 TrollGiant 的游戏对象，另外两个复制出来的对象分别是 TrollGiant(1)和 TrollGiant(2)。虽然理论上说我们可以手动修改它们的名称，但是一个良好的规范则是，游戏场景中的所有游戏对象尽量不要完全同名，否则会出现不必要的麻烦~

所以 Console 视图中的输出没有问题，但是另外两个巨人怪物的实际生命值完全不受影响，这就是为什么 Inspector 视图中的另外两个巨人怪物的 health 不受影响的原因。

为此，我们需要手动注释编号为 13 和 14 的两行代码：

```
        ////13.获取敌人对象
        //enemy = GameObject.Find("TrollGiant");

        ////14.获取敌人对象上的脚本组件
        //enemyScript = enemy.GetComponent<EnemyScript>();
```

然后更改 OnCollisionEnter 事件函数中的代码如下：

```
    //16.当碰撞开始的时候执行该事件函数
    private void OnCollisionEnter(Collision collision)
    {
        //17.如果碰撞的游戏对象的 tag 是 Enemy
        if(collision.gameObject.tag == "Enemy")
        {
            //18.获取碰撞的游戏对象的 EnemyScript 脚本组件，而不管它的游
```

```
戏对象名称是否是 TrollGiant
            enemyScript =
collision.gameObject.GetComponent<EnemyScript>();

            //敌人的生命值减少
            if(enemyScript.enemyHealth >= 1) {
                enemyScript.enemyHealth--;
                //在 Console 视图中输出敌人生命值的相关信息
                Debug.Log("敌人的当前生命值是： " +
enemyScript.enemyHealth);

            }
            else
            {
                //在 Console 视图中输出敌人死亡的信息
                Debug.Log("恭喜你，巨人怪物已被击败~ ");
            }



        }
    }
```

跟刚才的代码有所不同的是，我们做了两处修改：

1.编号 18 的代码行中，

我们直接获取精灵公主所碰撞的游戏对象的 EnemyScript 脚本组件，而不管它的游戏对象名称是否是 TrollGiant

2.增加了一个判断语句

当巨人的生命值大于等于 1 的时候，输出当前生命值。

当巨人的生命值等于 0 的时候，输出信息，恭喜玩家取得的新成就~

当然，细心的童鞋会发现，即便做了这样的优化，这里的逻辑仍然有一个小小的漏洞。

那就是，不管精灵公主撞击的是哪个怪物，所输出的巨人生命值都会下降。

大家可以自己尝试进一步优化以上代码，这里就不再赘述了~

好了，这一课的内容就先到这里了，让我们下一课再见。