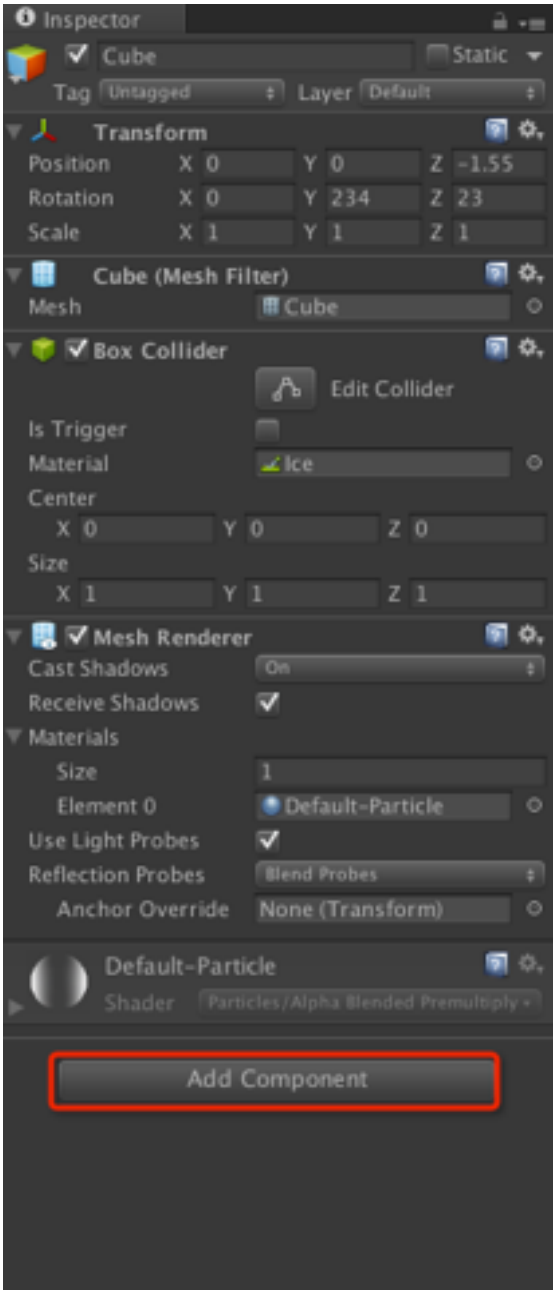
第五章　组件与物理引擎

1　添加组件

这一节我们开始学习组件，Unity中组件必须依赖于游戏对象才能显示在游戏场景中。在Unity编辑器中为游戏添加组件(或者为游戏对象挂载一个组件)，方式有很多种，首先可以选择游戏对象，在检视视图可以看到如下圈出的部分。



选中红色部分，可以给对象添加各种类型的组件。包括我们上节所学习的脚本。另外也可以在菜单栏上，选中Component给对象添加组件。

2　物理组件(引擎)

在介绍物理组件(可以叫物理引擎)的时候，需要先学习下刚体(Rigidbody)。刚体可以理解为一个物理世界的物体，只有这个物体才能被自然界的约束所作用，自然界的约束有重力、扭矩力、冲力，物体与物体之间可以有相互碰撞的作用力、摩擦力等。

刚体只能配合游戏对象才能使用，挂载有刚体组件的游戏对象才能受到物理世界的约束，有了刚体组件，游戏对象的碰撞才更加真实。

试着创建一个对象，并将一个刚体组件挂载到该对象上，如下图所示。

接下来，我们来学习下物理组件中刚体的一些属性。

(1)Mass(质量)：该属性表示刚体的质量，数据类型是float类型，默认情况都是1。一般来说，大部分的物理的Mass属性值应该设置为接近0.1到10.0之间。这样模拟出来的刚体更接近于生活中的感官感受。Mass是没有单位的，开发中，需要控制好物体与物体之间的比例才能提高物理仿真度。

(2)Drag(阻力)：这里的阻力指的是物体移动时的阻力，物体进行任意方向的移动都会感受到Drag的影响。该属性的数值类型也是float，如上

所示，其默认值为0。Drag是阻碍物体做位移运动的，因此它的方向总是与物体移动方向相反，通过设置Drag的值，能达到羽毛和石头掉落的效果。

(3)Angular Drag(旋转阻力)：与Drag类似，旋转阻力也是用来阻碍物体运动的，而旋转阻力是用来阻碍物体旋转的。值类型是float，默认是0.05。

(4)Use Gravity(使用重力)：这个属性的值类型是bool类型，当勾选上后表示使用重力，Use Gravity的值为true。而当不勾选则不受重力，值为false。

(5)Is Kinematic(是否遵循运动学)：该属可用于表示物理遵循牛顿运动学定律，数据类型为bool，初始值为false，如果勾选上，表示该属性的值为true，只受脚本和动画的影响，默认为false，是可以收到物理约束的。

(6)Interpolate(插值方式)：一般该属性默认就是None，由于物理计算和渲染不是同步，利用插值可以近似渲染的时间点，但这样有可能会产生一些轻微的抖动现象，建议在开发中对主要游戏对象设置插值就可以了

(7)Collision Detection(碰撞检测模式)：有时候物体移动速度飞快，而碰撞的厚度又足够小，会产生击穿的现象。为了防止这个现象的产生，Unity提供了三种不同的碰撞检测模式。其中离散模式(Discrete)适用于静止或者速度较慢的物体，对于速度快、体积小的物体建议使用连续模式(Continuous)，被使用了连续检测的物体应该使用动态连续模式(Continuous Dynamic)。

(8)Constrains(约束条件)：一般情况，物体所有方向的位移或者旋转都是会受到物理世界的影响的。但是可以人为的控制各个方向的位移或者旋转的。如下所示。

2.1 刚体变量

为了获取和更改物体的运动状态，Unity3D还预留了多个变量接口，这些接口简化了对物体运动状态的处理，使得开发人员能够轻易地对物体的运动状态进行干预。进入到Rigidbody类中，我们能看到它如下成员。

```
public extern float angularDrag...

public Vector3 angularVelocity...

public Vector3 centerOfMass...
```

注：部分还未截图。

(1)角速度(angularVelocity)：表示刚体的角速度向量，其数据类型为Vector3，该变量的方向即为刚体旋转轴的方向，旋转方向遵循左手定则；该角速度的大小为向量的摸，单位为rad/s。非必要情况下，不建议对此变量进行过多的干预，直接修改该值会造成模拟不真实。具体操作见如下代码。

```
void Start () {
    GetComponent<Rigidbody>().angularVelocity =
Vector3.up;
}
```

(2)位移速度(velocity)：表示物体的位移速度值。一般不建议去修改，同样是为了预防模拟失真情况出现。代码如下。

```
void Start () {
    GetComponent<Rigidbody>().velocity = Vector3.up;
}
```

(3)重心(centerOfMass)：通过调低物体的重心，可以使物体不易因其他物体的碰撞或作用力而倒下。若不对重心进行设置，Unity3D会对重心位置自动进行计算，其计算基础为物体所挂载的碰撞器。

```
void Start () {
    GetComponent<Rigidbody>().centerOfMass = Vector3.up;
}
```

(4)碰撞检测开关(detectCollisions)：用于表示物体是否能够与其他物体产生碰撞效应，默认是true，是可以与其他物体碰撞的。在实际开发中物体并不是时刻都需要进行碰撞检测的，此时可以通过设置该属性值，而不是直接移除掉刚体，因为移除一个刚体的效率远不如直接将碰撞检测开个直接设为false的效率高。如下代码，

```
void Start () {
    GetComponent<Rigidbody>().detectCollisions = false;
}
```

(5)惯性张量(inertiaTensor)：该变量用来描述物体转动惯量，其数据类型为Vector3。如果不对该值进行设置和干预，它将通过挂载在物体对象上的碰撞器组件自动进行计算。如下代码所示。

```
void Start () {
    GetComponent<Rigidbody> ().inertiaTensor = Vector3.up;
}
```

(6)惯性张量旋转值(inertiaTensorRotation)：该变量指物体张量的旋转值，其数据类型为Quaternion，即四元数。代码如下

```
void Start () {
    GetComponent<Rigidbody>().inertiaTensorRotation = Quaternion.identity;  (0f,0f,0f,1f)
}
```

(7)其他变量如最大角速度(maxAngularVelocity)、最大穿透速度(maxDepenetrationVelocity)、坐标(position)、旋转(rotation)、是否使用锥形摩擦(useConeFriction)，具体看如下代码，

```
void Start () {
    GetComponent<Rigidbody>().maxAngularVelocity = 1.9f;//最大角速度，默认为7，不能为负数
    GetComponent<Rigidbody>().maxDepenetrationVelocity = 2.0f;//最大穿透角速度，物体穿透时的临界角速度值
}
void Update () {
    Debug.Log (GetComponent<Rigidbody>().position);//打印物体位置信息
    Debug.Log (GetComponent<Rigidbody>().rotation);//打印物体角度
}
```

2.2 刚体常用方法

我们直接来看下面几行代码。通过注释我们来学习他们。

```
void Start () {
    GetComponent<Rigidbody>().AddForce(Vector3.up*3);//给刚体施加力
GetComponent<Rigidbody>().AddExplosionForce(19.0f,transform.position,10,1.5f,ForceMode.Force);//施加爆炸力
```

```
GetComponent<Rigidbody>().AddRelativeForce(Vector3.up*10,ForceMode.Force);//施加相对力
            GetComponent<Rigidbody>().AddTorque(-Vector3.right*70,ForceMode.Force); //施加力矩
            GetComponent<Rigidbody>().AddRelativeTorque(-Vector3.right*70,ForceMode.Force); //施加相对力矩
        }
        void FixedUpdate () {
GetComponent<Rigidbody>().MovePosition(transform.position + Vector3.right*Time.deltaTime);//移动刚体

GetComponent<Rigidbody>().MoveRotation(transform.rotation*Quaternion.Euler(new Vector3(0,100,0)*Time.deltaTime));//旋转物体

GetComponent<Rigidbody>().AddForceAtPosition(Vector3.up,transform.position,ForceMode.Force);//在指定点施加力
        }
```

以上代码都是刚体中的方法，除了以上方法外还有:

(1)计算相对刚体的最近点(ClosestPointOnBounds)

(2)获取点坐标系的速度(GetPointVelocity)

(3)获取基于相对点坐标系的速度(GetRelativePointVelocity)

(4)确定物体是否处于休眠(IsSleeping)

(5)设置密度(SetDensity)

(6)强制休眠(Sleep)

(7)唤醒(WakeUp)

(8)扫描检测(SweepTest)

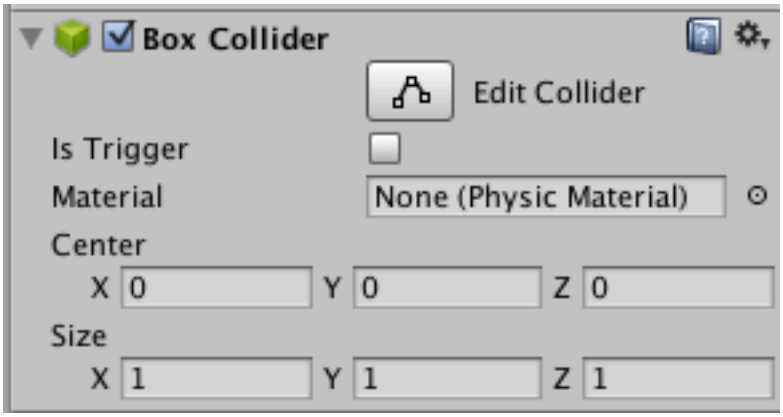
(9)扫描全部检测(SweepTestAll)

值得注意的是8、9两个方法只能检测到简单类型的碰撞器，网格碰撞器无法检测到。关于碰撞器后面会学习到。

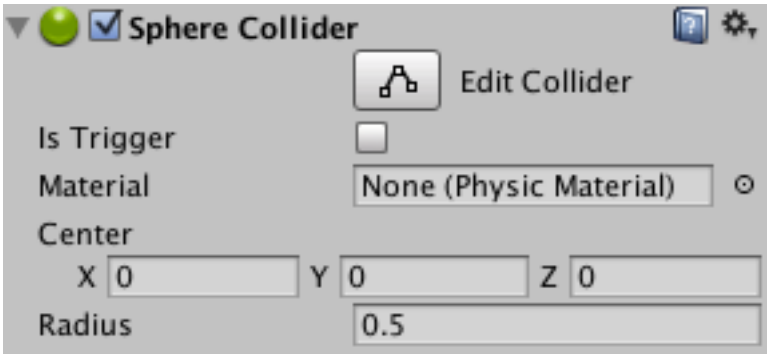
2.3 碰撞器(体)

Unity中碰撞器包括6种，如下所示。

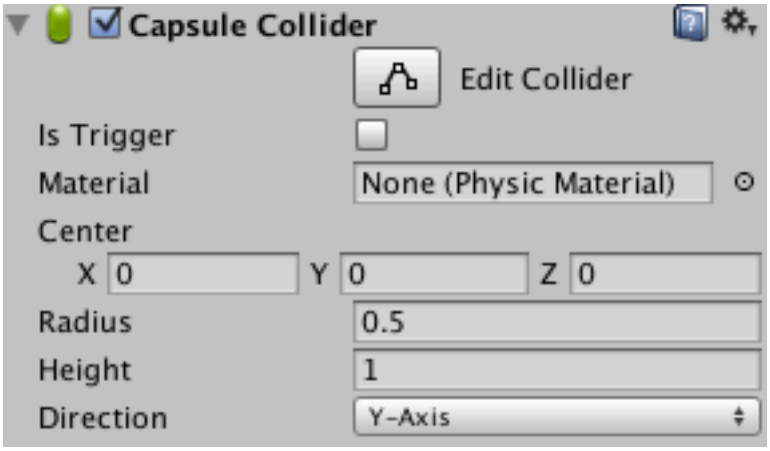
(1)盒子碰撞器:



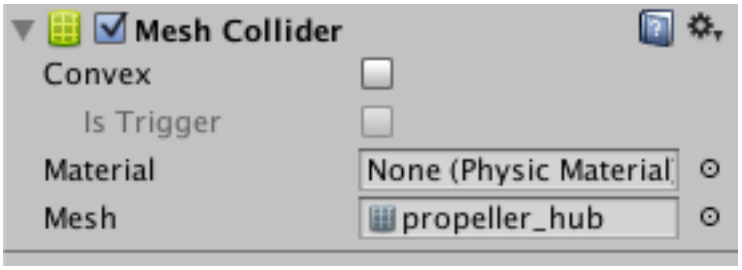
(2)球体碰撞器:



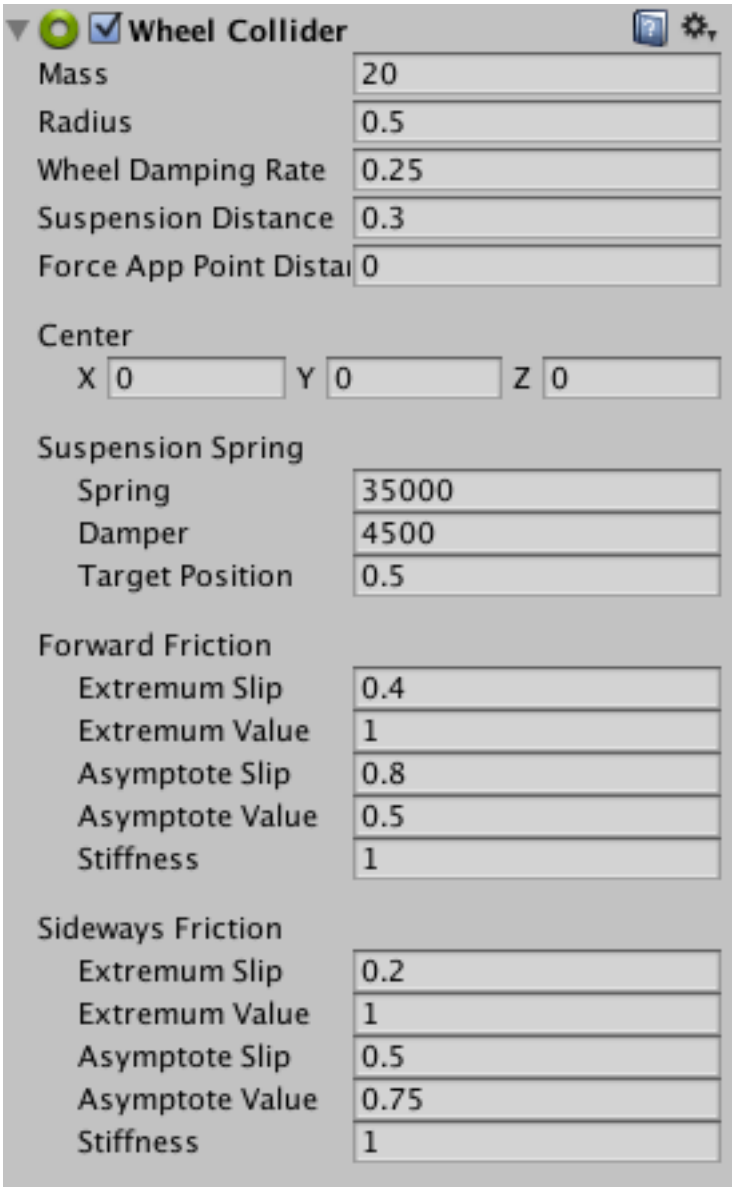
(3)胶囊碰撞器:



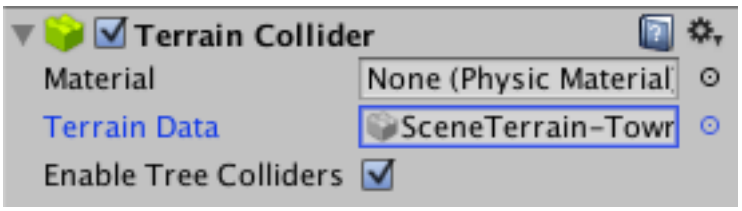
(4)网格碰撞器:



(5)车轮碰撞器:



(6)地形碰撞器:



2.3.1 碰撞器添加案例

见课上安排。

2.3.2 碰撞过滤

在Unity中有时候物体与物体之间是不需要发生碰撞的，比如敌人的子弹碰到了敌人，那么敌人子弹和敌人之间肯定不能发生碰撞。Unity中提供了一种方案，如下代码所示，就能控制物体与物体之间是否发生碰撞。

```
void Start () {
        Physics.IgnoreCollision
(ball1.GetComponent<Collider>(),ball2.GetComponent<Collider>(),true);
    }
```

2.4 物理材质

物理材质相当于组成物体的材料，材料不同，产生的物理效应就不同。

Unity中创建物理材质的方式在项目视图中进行操作，具体操作是Create->Physics Material，然后再对其进行调节，如下图。具体看我们课上内容。

2.5 案例

层与层之间的碰撞。(见课上内容)