1. 解释一下Rigidbody组件中Is Kinematic参数在什么情况下使用？

Is Kinematic 是否为Kinematic刚体，如果启用该参数，则对象不会被物理所控制，只能通过直接设置位置、旋转和缩放来操作它，一般用来实现移动平台，或者带有HingeJoint的动画刚体

1. 编写一个脚本对刚体的几种常用方式进行测试。

using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
public class rigidbodyTest : MonoBehaviour {  
    private Rigidbody \_rigidbody;  
    // Use this for initialization  
    void Start ()   
    { \_rigidbody = transform.GetComponent<Rigidbody>();  
    }  
    // Update is called once per frame  
    void Update ()   
    { if (Input.GetKeyDown(KeyCode.Q))  
        { \_rigidbody.velocity = new Vector3(1,1,1); //给其一个速度矢量  
           // \_rigidbody.position = new Vector3(5,5,5);//给其一个位置}  
        if (Input.GetKeyDown(KeyCode.W))  
        {  
            \_rigidbody.Sleep();//制性使刚体休眠，不动了;休眠是性能优化的一个措施，物理引擎不会处理处于休眠状态的刚体；  
            //刚体在以下情况会被唤醒：1,其他刚体碰撞器作用于休眠刚体。2，被其他刚体通过移动的关节连接  
            //3，修改了刚体的属性。4，添加外力时  
        }  
        if (Input.GetKeyDown(KeyCode.E))  
        {  
            \_rigidbody.WakeUp(); /制唤醒一个刚体  
        }  
        if (Input.GetKeyDown(KeyCode.A))  
        {  
            \_rigidbody.MovePosition(new Vector3(10,100,200));//给其一个位置  
        }  
        if (Input.GetKeyDown(KeyCode.S))  
        {  
            \_rigidbody.freezeRotation = true; //开启則  刚体的XYZ轴全部冻结

        }

        if (Input.GetKeyDown(KeyCode.D))

        {          \_rigidbody.constraints = RigidbodyConstraints.FreezePositionY; //选择性冻结某一轴  
        }        if (Input.GetKeyDown(KeyCode.F))

        {            \_rigidbody.AddExplosionForce(50,new Vector3(0,0,0),20); //添加一个爆炸力  
        }        if (Input.GetKeyDown(KeyCode.R))

        {            \_rigidbody.AddForce(Vector3.forward\*3,ForceMode.Acceleration); //沿着某一方向给刚体添加一个力

        }        if (Input.GetKeyDown(KeyCode.T))

        {            \_rigidbody.AddTorque(transform.forward\*10); //沿着某一方向添加一个扭矩  
        }        //\_rigidbody.transform.Rotate(transform.up, Time.deltaTime); //基于transform的旋转  
        //\_rigidbody.angularVelocity = transform.right \* Time.deltaTime;//基于刚体的旋转  
    }    private void OnTriggerEnter(Collider other)

{        print(11);

    }    private void OnTriggerStay(Collider other)

{        print(22);

    }    private void OnTriggerExit(Collider other)

{ print(33);

    }    private void OnCollisionEnter(Collision other)

    { print(other.relativeVelocity); //两个碰撞物体的相对线性速度    }

}

1. 了解Unity3D游戏引擎自带的规则碰撞器，并导入一个模型为其添加合适的碰撞器。





* Is Trigger（是否为触发器） 这个选项是供脚本使用的，如果勾选了这个则不会有碰撞的物理效果，但是游戏引擎会通知脚本有物体发生了碰撞。
* Material（碰撞器材质） 在这里可以选择一种物理材质，来模拟更真实的碰撞效果，比如金属之间的碰撞与石头之间的碰撞效果肯定是不一样的。
* Center（碰撞器中心点） 可以调整碰撞器离物体中心的距离，也就是移动绿框。
* Size（碰撞器大小） 调整碰撞器的缩放大小，调整XYZ可以让碰撞器变成任意大小的长方体。
* Sphere Collider（球形碰撞器）： 可以调整半径大小。
* Capsule Collider（胶囊碰撞器）： 可以调整半径、高度和方向，即使是圆柱体也可以用这个做近似的碰撞效果。
* Wheel Collider（轮子碰撞器）： 可以调整虚拟质量、半径高度、悬浮距离、弹力、阻力等等参数，主要用于附加在类似于轮胎这种对象上，模拟驱动力的产生，同时提供碰撞范围。
* Mesh Collider（网格碰撞器）： 这是个强力的碰撞器，用它能创造出不规则的碰撞范围，但需要的性能消耗也比较大。所以虽然像前面的树的模型本应该使用网格碰撞器，但是一般场景中的树木数量都比较多，所带来的性能开销也相当可观，所以官方也建议用胶囊作为树木的碰撞器。只有在的确需要不规则碰撞的时候再用此神器。
* Terrain Collider（地形碰撞器）： 为什么我们的第一人称视角控制器不会从“地面”上往下坠？选中游戏中的地形对象，就能发现它的身影啦。 注：在官方文档中有解释地形的Create Tree Collider（创建树木碰撞器）选项，说是勾选这个，就可以自动为用地形画刷刷出来的树木创建碰撞器。