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

答：给虚拟对象使用的，勾选后这个对象只受transform影响不受force影响

is kinematic 是否开启动力学，开启此项，物体不受力的作用

2 编写一个脚本对刚体的几种常用方法进行测试（脚本源码及其注释）

答：

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();//强制性使刚体休眠，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(); //强制唤醒一个刚体

}

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

}

}

3 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（回答有哪些碰撞器）

答：

Box Collider（立方体碰撞器）：点击它，在场景视图中的立方体就会出现一个绿色的轮廓，这就是碰撞器的真身了。在Unity中会将游戏对象的碰撞器以绿色轮廓显示出来，方便我们手动调整碰撞范围。

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。

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。