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

答：Is Kinematic该属性表示对象是否遵循牛顿运动学物理定理，其数据类型是boolean ，初始值为false。如果该属性设置为true表示该物体运动状态不受外力，碰撞和关节的影响，而只受到动画以及附加在物体上的脚本影响，但是该物体仍然能改变其他物体运动状态，例如游戏中倒下的敌人始终不动 ，就是利用这个属性 。这个是给你的虚拟对象使用的，勾选后这个对象只受[transform](https://wenwen.sogou.com/s/?w=transform&ch=ww.xqy.chain)影响不受force影响。

2、编写一个脚本对刚体的几种常用方法进行测试。

答：刚体：Rigidbody；

void Start () {

GetComponent<Rigidbody>().AddForce(Vector3.up\*3);//给刚体施加力

GetComponent<Rigidbody>().AddExplosionForce(19.0f,transform.positi on,10,1.5f,ForceMode.Force);//施加爆炸力

GetComponent<Rigidbody>().AddRelativeForce(Vector3.up\*10,ForceM ode.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 ，参数为移动目的地址

GetComponent<Rigidbody>().MoveRotation(transform.rotation\*Quaterni on.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

}

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（有哪些碰撞器）

答：分类：

Box Collider盒碰撞器--基本形状是个盒子。

Sphere Collider 球碰撞器--基本形状是个球。

Capsule Collider 胶囊碰撞器--基本形状是个胶囊。

Mesh Collider 网格碰撞器--从物体的网格创建一个碰撞器。不能与其他网格碰撞器相碰撞。

Wheel Collider 轮碰撞器--特殊的碰撞器，用于创建车或其他移动交通工具。

添加碰撞器后：

(1)导入资源:Assets--Import Package--Terrain Assets，导入系统的地形贴图等资源。

(2)添加模型:在Assets下建立一个Model文件夹，将自己的3D模型放入Model中。

(3)建立地形:Terrain--Create Terrain。建立一个地形,在地形编辑部分点击Paint theterrain texture图标，选择一个Texture，并调整Tile Size X, Tile Size Y。

(4)将模型拖到Unity3D场景编辑中，并调整位置。随便添加个方向光，看起来会比较顺眼。这个时候，如果你给模型添加RigidBody，虽然它会有重力，但不会与Terrain进行碰撞。

(5)创建碰撞体:GameObject--Create Other--Cube。将Cube放到模型的中心位置(其他位置也可以)，你可以看到Cube有个Box Collider对象。调整Box Collider的Center和Size，也就是碰撞范围(绿色的矩形)，让碰撞范围符合整个模型。建议分别通过X,Y,Z轴朝向进行调整。

(6)在Hierarchy视图上，将模型(我这里是Tank)拖到Cube上。然后点Cube,Component--Physics--RigidBody给Cube添加一个刚体。然后在Cube的属性中，将Mesh Renderer勾去掉，也就是不进行渲染Cube。

(7)运行程序，你会发现模型安稳的落在了Terrain上了。

4、unity3d在场景中新建物理材质，实现小球从高空落下可弹起的功能。

发射台：

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Launcher : MonoBehaviour

{public GameObject ballPrefab; // Use this for initialization

void Start(){}// Update is called once per frame

void Update()

{ if (Input.GetMouseButtonDown(1))

{ Instantiate(this.ballPrefab); //通过预设生成游戏对象实例的方法。 } }}

控制小球代码：

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Ball : MonoBehaviour{

// Use this for initialization

void Start()

{this.GetComponent<Rigidbody>().velocity = new Vector3(10.0f, 9.0f, 0.0f); } // Update is called once per frame

void Update(){}

void OnBecameInvisible()//在游戏对象移动到画面之外不再被绘制时调用的方法。 { Destroy(this.gameObject); //删除跑出画面之外的小球。}}

控制代码：

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class player1 : MonoBehaviour

{protected float jump\_speed = 12.0f;

public bool is\_landing = true; //用来判断发现碰撞时是否为与与地面相撞（也可能是与小球相撞）

// Use this for initialization

void Start() {}// Update is called once per frame

void Update()

{ if (this.is\_landing) //使得落地后才能再次弹起，而不能在空中再次弹起。 {if (Input.GetMouseButtonDown(0))

{this.is\_landing = false;

//this.GetComponent<Rigidbody>().velocity = new Vector3(0.0f, 12.0f, 0.0f);

this.GetComponent<Rigidbody>().velocity = Vector3.up \* this.jump\_speed; }}} void OnCollisionEnter(Collision collision) //和其他游戏对象发生碰撞时调用的方法。

{if (collision.gameObject.tag == "Floor")

{this.is\_landing = true; }}}