**CharacterController**

A CharacterController allows you to easily do movement constrained by collisions without having to deal with a rigidbody.

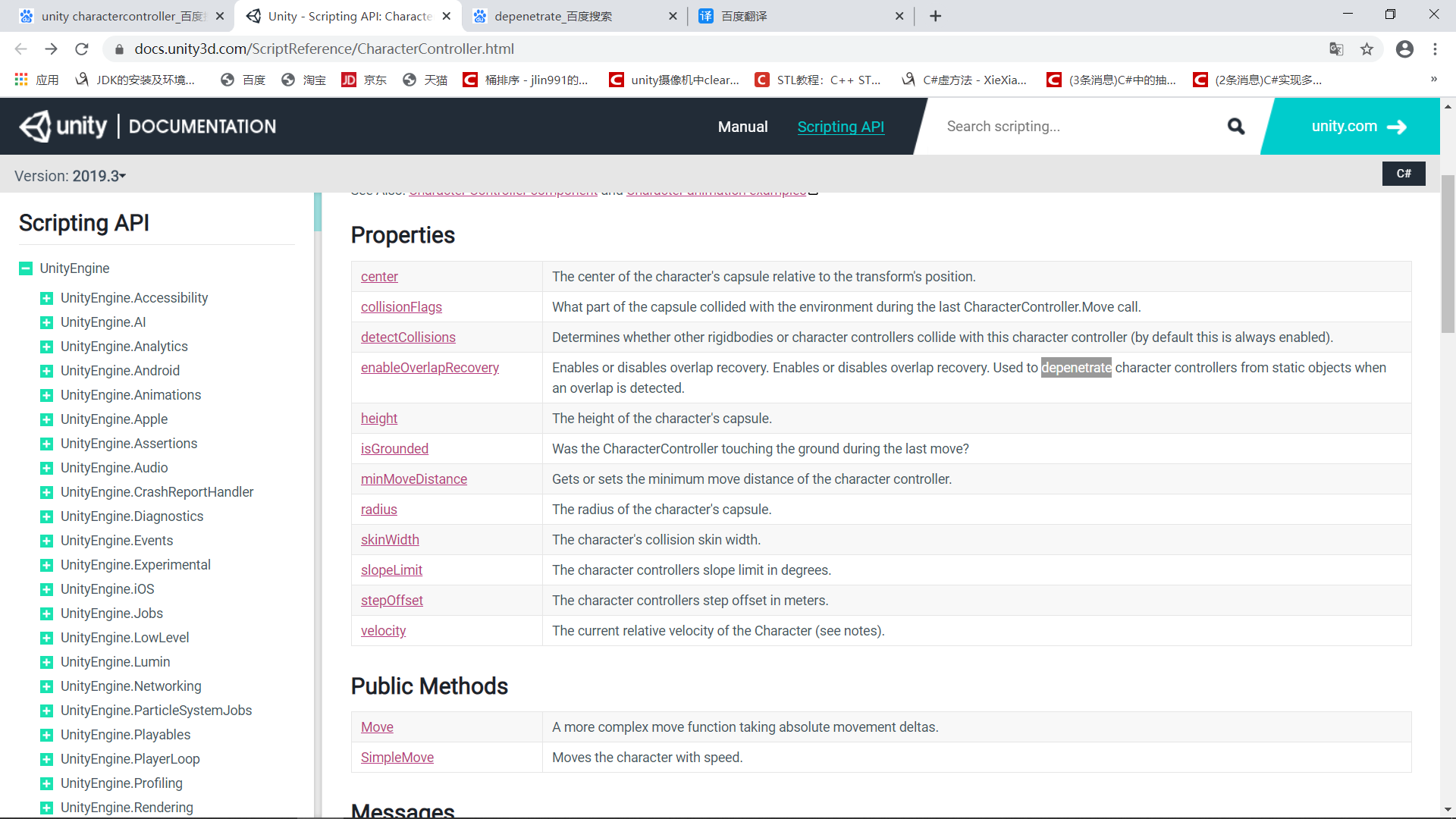
CharacterController允许您轻松地执行受碰撞约束的移动，而不必处理刚体。

A CharacterController is not affected by forces and will only move when you call the Move function. It will then carry out the movement but be constrained by collisions.

CharacterController不受力的影响，只有在调用move函数时才会移动。然后它将执行运动，但受到碰撞的限制。

1. **不用处理刚体，没有碰撞效果**
2. **添加charactercontroller的物体不受力的作用**

**相关属性**



比较复杂的属性

CollisionFlags 物体所带碰撞体与外界碰撞的位置

DetectCollisons 探查是否有其他物体与物体碰撞（默认关闭）

enableOverlapRecovery 启用或禁用重叠恢复用于在检测到重叠时使角色控制器与静态对象分离。（一般用不到）

Slope Limit爬坡限制

Step Offset台阶高度

Skin Width 皮肤宽度

Min Move Distance一次移动最短距离

1. 所能提供的公共方法（最具功能性的地方）
2. **Move**

Attempts to move the controller by motion, the motion will only be constrained by collisions. It will slide along colliders. [CollisionFlags](https://docs.unity3d.com/ScriptReference/CollisionFlags.html) is the summary of collisions that occurred during the Move. This function does not apply any gravity.

官方文档文档实例体验

using UnityEngine;

using System.Collections;

//官方文档实例

// This script moves the character controller forward

// and sideways based on the arrow keys.

// It also jumps when pressing space.

// Make sure to attach a character controller to the same game object.

// It is recommended that you make only one call to Move or SimpleMove per frame.

public class CCmove : MonoBehaviour

{

CharacterController characterController;

public float speed = 6.0f;

public float jumpSpeed = 8.0f;

public float gravity = 20.0f;

private Vector3 moveDirection = Vector3.zero;

void Start()

{

characterController = GetComponent<CharacterController>();

}

void Update()

{

if (characterController.isGrounded)

{

// We are grounded, so recalculate

// move direction directly from axes

**moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0.0f, Input.GetAxis("Vertical"));**

**moveDirection \*= speed;**

if (Input.GetButton("Jump"))

{

moveDirection.y = jumpSpeed;

}

}

// Apply gravity. Gravity is multiplied by deltaTime twice (once here, and once below

// when the moveDirection is multiplied by deltaTime). This is because gravity should be applied

// as an acceleration (ms^-2)

moveDirection.y -= gravity \* Time.deltaTime;

// Move the controller

characterController.Move(moveDirection \* Time.deltaTime);

}

}

moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0.0f, Input.GetAxis("Vertical"));

可以很方便的捕捉键盘方向键

1. SimpleMove

Velocity along the y-axis is ignored. Speed is in units/s. Gravity is automatically applied. Returns if the character is grounded. It is recommended that you make only one call to [Move](https://docs.unity3d.com/ScriptReference/CharacterController.Move.html) or [SimpleMove](https://docs.unity3d.com/ScriptReference/CharacterController.SimpleMove.html) per frame.

同样是官方示例

using UnityEngine;

using System.Collections;

[RequireComponent(typeof(CharacterController))]

public class CCsimplemove : MonoBehaviour

{

CharacterController controller;

public float speed = 3.0F;

public float rotateSpeed = 3.0F;

private void Start()

{

controller = GetComponent<CharacterController>();

}

void Update()

{

// Rotate around y - axis

transform.Rotate(0, Input.GetAxis("Horizontal") \* rotateSpeed, 0);

// Move forward / backward

Vector3 forward = transform.TransformDirection(Vector3.forward);

float curSpeed = speed \* Input.GetAxis("Vertical");

controller.SimpleMove(forward \* curSpeed);

}

}

这里会发现运动的速度变化其实都是相同的，只是Move示例采用 moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0.0f, Input.GetAxis("Vertical"))控制方向，而Simplemove采用transform.Rotate(0, Input.GetAxis("Horizontal") \* rotateSpeed, 0);控制方向

两者的区别其实在于：

# SimpleMove

1、SimpleMovez**自带重力效果，对Y轴运动绝对控制。**

2、SimpleMove**返回值BOOL类型，角色接触地面则返回true，否则返回false。**

# Move

1、Move，**直接定位角色状态为静态或者动态，且没有自带重力效果。除Move以外唯一会影响运动状态的就是各种障碍物的刚体碰撞，会使物体沿着刚体滑动，**

2、Move**返回一系列碰撞物体的信息。**

其他方法

|  |  |
| --- | --- |
| [OnControllerColliderHit](https://docs.unity3d.com/ScriptReference/CharacterController.OnControllerColliderHit.html) | OnControllerColliderHit is called when the controller hits a collider while performing a Move. |

当物体移动时发生碰撞就发触发该函数