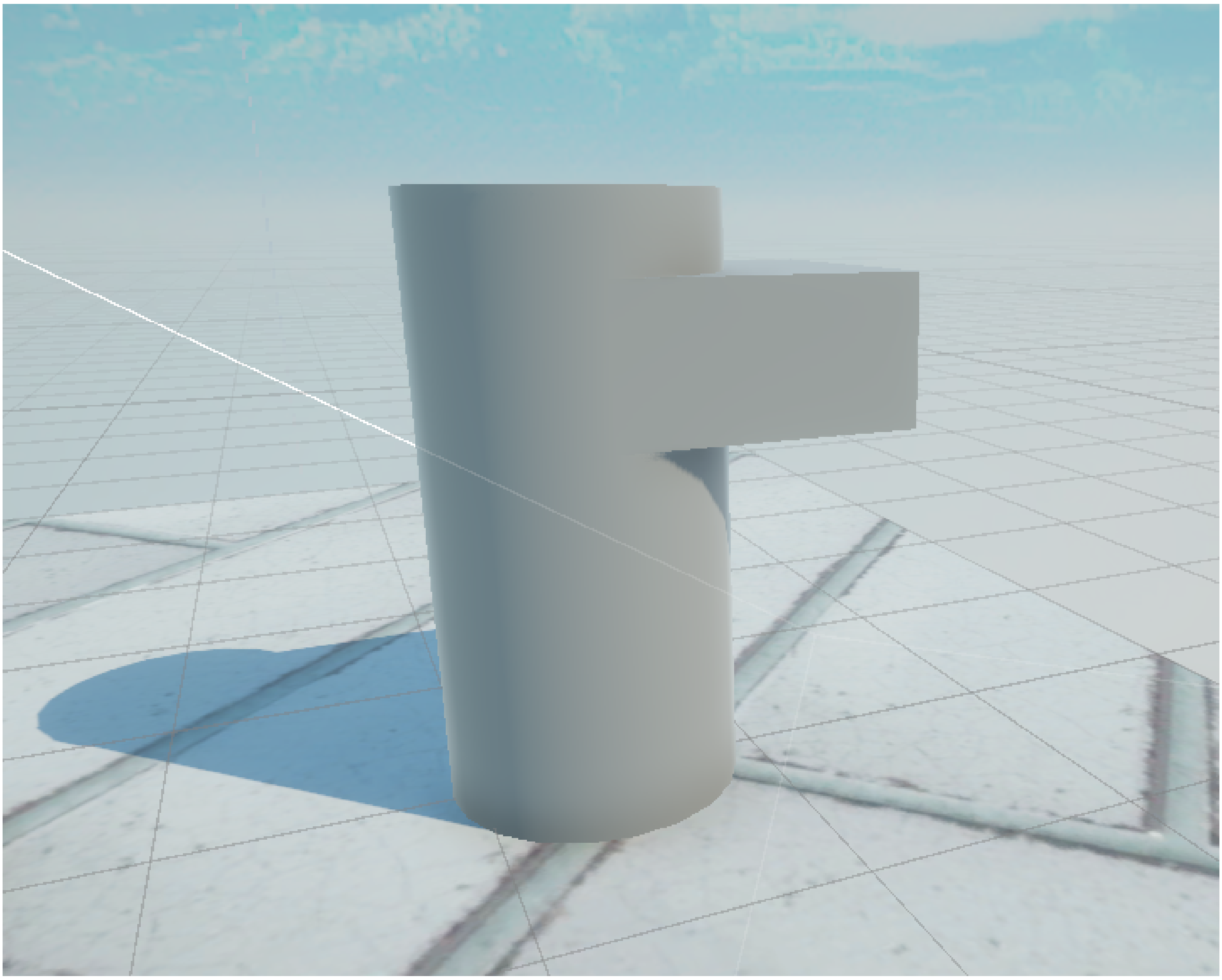
第三人称控制

必要插件（官方内置插件）

- Cinemachine
- Input System

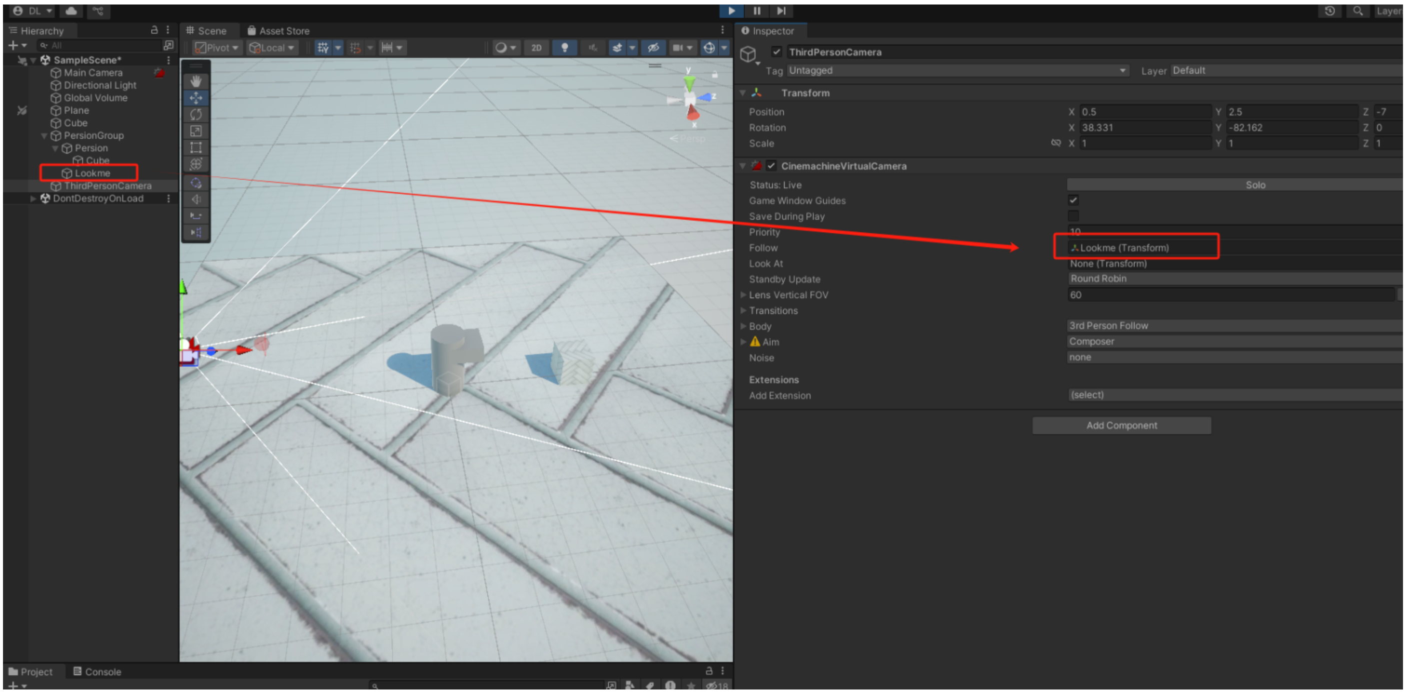
开始使用

- 创建一个空的GameObject,改名为PersionGroup
- 在PersionGroup之下创建一个胶囊(cylinder)，取名为Persion
- 在Persion之下创建一个立方体(cube),调整大小和位置，使其成为这如下图的形状，目的是为了确认目标的方向，如下图的参数为：



此两个步骤可以省略，只是示意一下这是一个任务角色

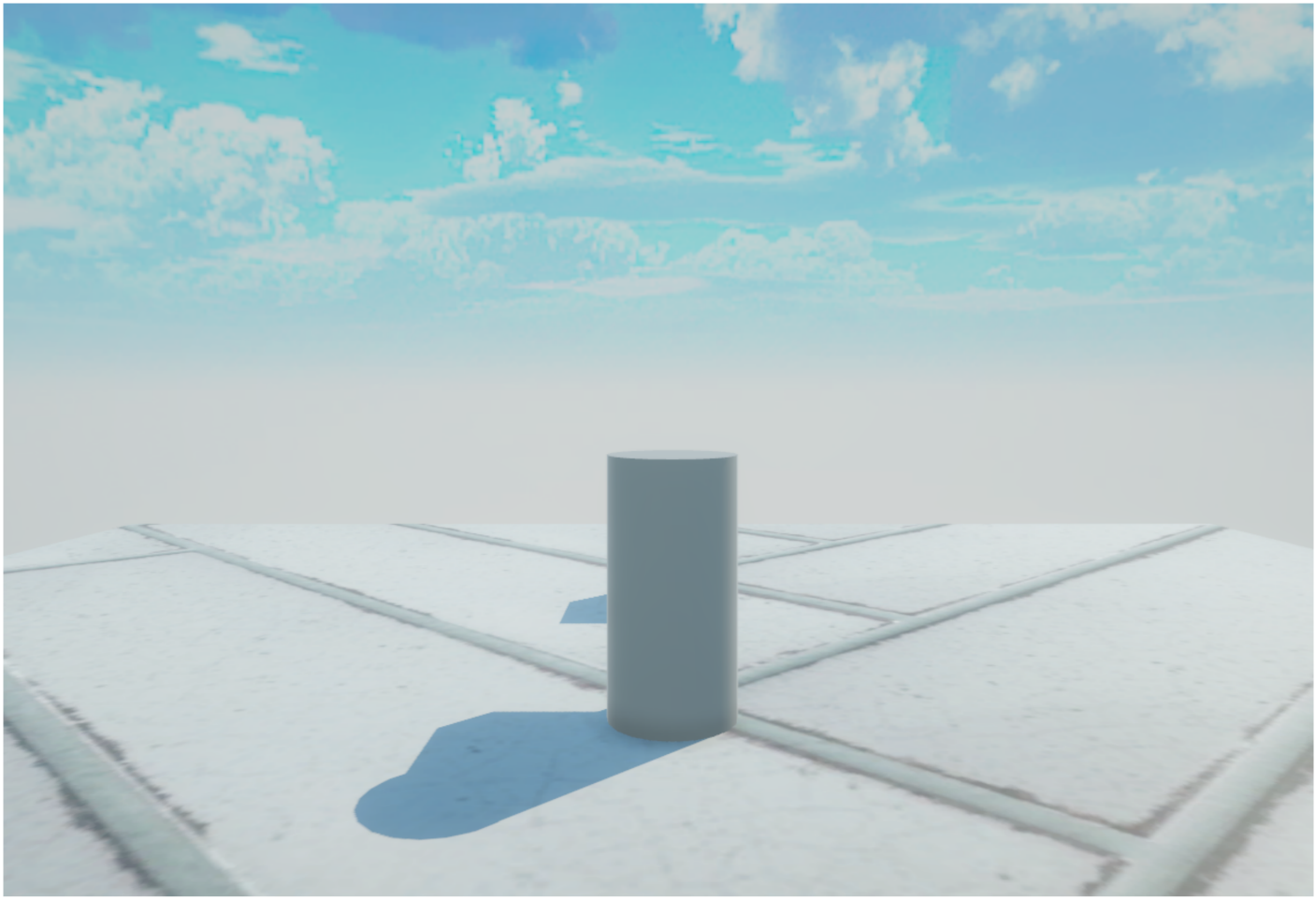
- 在PersionGroup之下创建一个空物体，改名为Lookme，用于摄像机跟随关注目标，设置positon:0,0,0
- 增加一个虚拟摄像机，右键或者GameObject菜单->Cine machine->Virtual Camera，改名为ThirdPersonCamera，选中摄像机后，在B
- 将Lookmet拖到3rd摄像机的follow中



ThirdPersonCamera
Tag Untagged Layer Default
Transform
Position X 0.5 Y 2.5 Z -7
Rotation X 38.331 Y -82.162 Z 0
Scale X 1 Y 1 Z 1
CinemachineVirtualCamera
Status: Live Solo
Game Window Guides ☑
Save During Play ☐
Priority 10
Follow Lookme (Transform)
Look At None (Transform)
Standby Update Round Robin
Lens Vertical FOV 60
Transitions
Body 3rd Person Follow
Aim Composer
Noise none
Extensions
Add Extension (select)

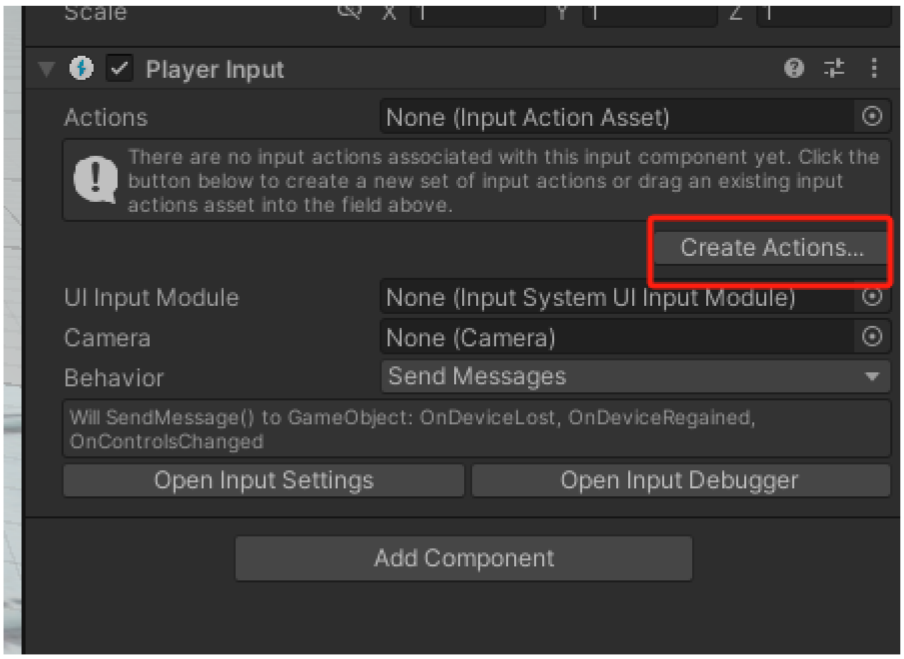
- 调整摄像机的位置和视角
  - Third Person Camera lens面板下的参数:
    - Vertical Fov:视口大小，默认值60，有点类似镜头焦段
    - dutch：镜头水平方向旋转，默认值为0
  - Third Person Camera body面板下的参数
    - Shoulder Offset：直接调整3rd虚拟摄像机的位置
    - Vertical Arm Length：控制摄像机在垂直方向的位置
    - Camera Side：控制摄像机水平方向的位置
    - Camera Distance：控制摄像机垂直方向的位置

虽然调整lookme的pos**Third Person Camera**视整,因为lookme与被跟随的物体不在一个中心点的话，会导致跟随非常奇怪，无法使角色一直在正中间。最终得到的效果如下:



增加鼠标和键盘控制

- 在PersionGroup上增加搜索input,选择player input组件，选择或者create actions



- 增加2个脚本ThirdPersonCameraLook.cs和ThirdPersonCameraMove.cs一个用于旋转，一个用于移动
- **ThirdPersonCameraLook.cs**，将脚本挂载到PersionGroups上

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.InputSystem;

public class ThirdPersonCameraLook : MonoBehaviour
{
    private GameObject mainCamera;

    [Header("Cinemachine")]
    [Tooltip("跟随目标")]
    public GameObject CameraTarget;

    [Tooltip("向上的最大角度")]
    public float TopClamp = 70.0f;

    [Tooltip("向下的最大角度")]
    public float BottomClamp = 0.0f;

    [Tooltip("相机旋转速度")]
    public float RotationSpeed = 1.0f;

    // 输入阈值
    private const float threshold = 0.01f;
    // 相机目标的 Y 轴旋转角度
    private float cinemachineTargetYaw;
    // 相机目标的 X 轴旋转角度
    private float cinemachineTargetPitch;
    // 存储玩家输入的视角变化
    private Vector2 lookup;
    // 标记是否正在旋转
    private bool isRotateing = false;
    void Start()
    {
        if (mainCamera == null)
        {
            // 查找并存储主相机对象
            mainCamera = GameObject.FindGameObjectWithTag("MainCamera");
        }
        // 初始化相机目标的 Y 轴旋转角度
```

```
        cinemachineTargetYaw = CameraTarget.transform.rotation.eulerAngles.y;
    }

    // Update is called once per frame
    void Update()
    {
        // 检查鼠标右键是否按下
        isRotateing = Mouse.current.rightButton.isPressed;
        // 如果正在旋转且输入的平方长度大于等于阈值，则更新相机目标的旋转角度
        if (isRotateing && lookup.sqrMagnitude >= threshold)
        {
            cinemachineTargetYaw += lookup.x;
            cinemachineTargetPitch += lookup.y;
        }
        // 限制相机目标的旋转角度在合理范围内
        cinemachineTargetYaw = ClampAngle(cinemachineTargetYaw, float.MinValue,
float.MaxValue);
        cinemachineTargetPitch = ClampAngle(cinemachineTargetPitch, BottomClamp, TopClamp);

        // 使用插值函数平滑过渡相机的旋转，如果同时加了移动组件，不建议使用平滑旋转
        // Quaternion targetRotation = Quaternion.Euler(cinemachineTargetPitch,
cinemachineTargetYaw, 0.0f);
        // CameraTarget.transform.rotation = Quaternion.Lerp(CameraTarget.transform.rotation,
targetRotation, Time.deltaTime * RotationSpeed);
        // 控制相机的旋转
        CameraTarget.transform.rotation = Quaternion.Euler(cinemachineTargetPitch,
cinemachineTargetYaw, 0.0f);
    }
    // 限制角度在指定范围内
    private static float ClampAngle(float angle, float min, float max)
    {
        if (angle < -360.0f)
        {
            angle += 360.0f;
        }
        if (angle > 360.0f)
        {
            angle -= 360.0f;
        }
        return Mathf.Clamp(angle, min, max);
```

```
    }

    public void OnLook(InputValue value)
    { // 处理玩家输入的视角变化
        lookup = value.Get<Vector2>();
    }
}
```

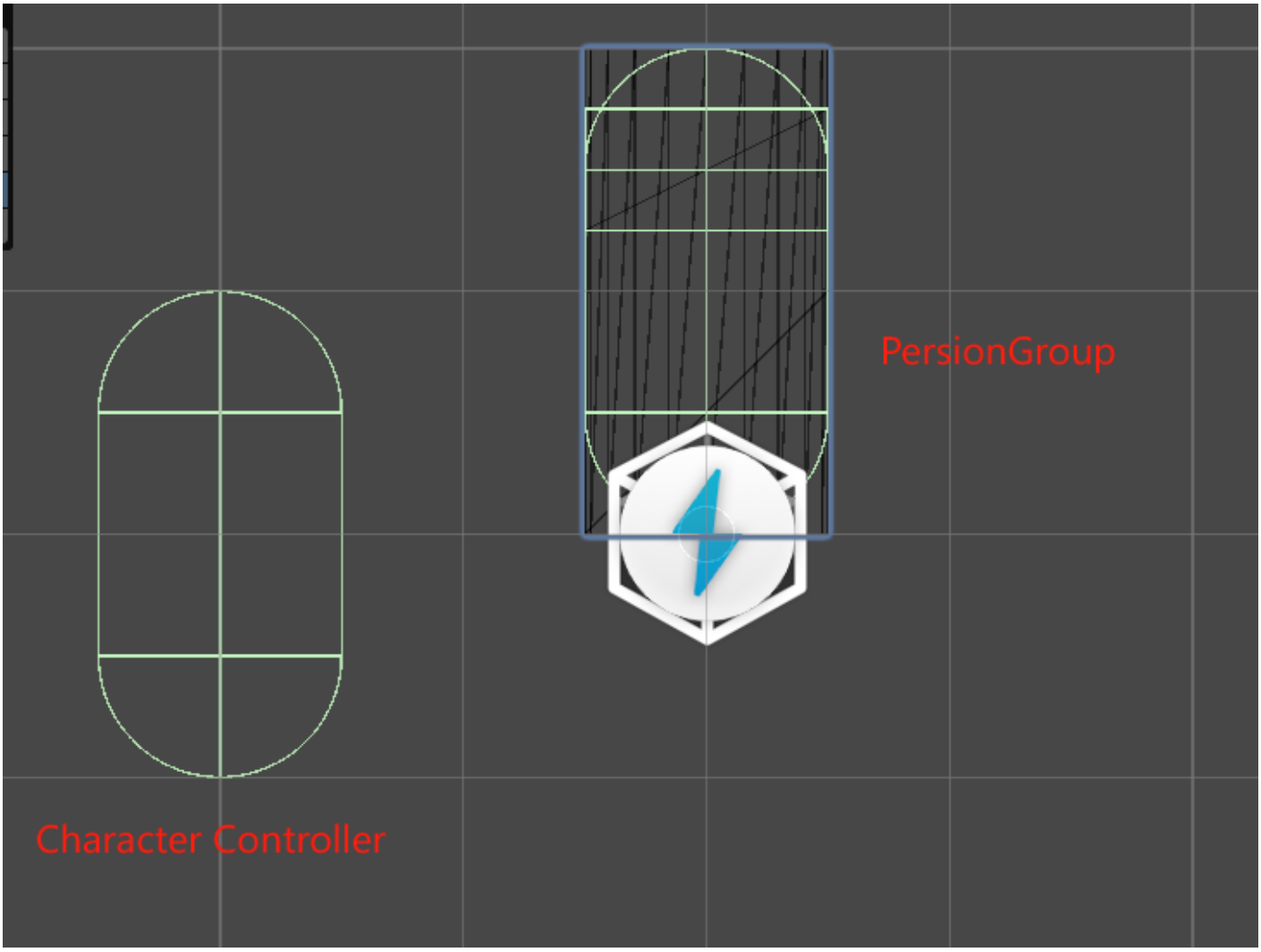
- 用**ThirdPersonCameraLook.cs**，将脚本挂载到PersionGroup上

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.InputSystem;

public class ThirdPersonCameraMove : MonoBehaviour
{

    // 主摄像机对象引用
    private GameObject mainCamera;
    // 角色控制器对象引用
    private CharacterController controller;

    // 移动输入值
    private Vector2 moveValue;

    //    移动速度
    [Tooltip("移动速度")]
    public float speed = 1.0f;
    // 目标旋转角度
    private float targetRotation = 0.0f;

    // 平滑旋转时间
    public float RotationSmoothTime = 0.1f;
    // 旋转速度
    private float rotationVelocity = 0.0f;

    void Start()
```

```
    {
        if (mainCamera == null)
        {
            mainCamera = GameObject.FindGameObjectWithTag("MainCamera");
        }
        // 获取角色控制器对象
        controller = GetComponent<CharacterController>();
    }


    void Update()
    {
        // 记录原始的Y方向数值
        Vector3 velocity = new Vector3(0, -1, 0);
        if (moveValue != Vector2.zero)
        {
            // 计算输入方向，转换为世界坐标
            Vector3 inputDir = new Vector3(moveValue.x, 0.0f, moveValue.y).normalized;
            // 计算目标旋转角度
            targetRotation = Mathf.Atan2(inputDir.x, inputDir.z) * Mathf.Rad2Deg +
mainCamera.transform.eulerAngles.y;

            // 平滑旋转
            float rotation = Mathf.SmoothDampAngle(transform.eulerAngles.y, targetRotation,
ref rotationVelocity, RotationSmoothTime);
            // 旋转角色
            transform.rotation = Quaternion.Euler(0.0f, rotation, 0.0f);
            // 计算移动方向
            Vector3 targetDir = Quaternion.Euler(0.0f, targetRotation, 0.0f) *
Vector3.forward;
             // 记录原始的Y方向数值
            velocity += targetDir.normalized * (speed * Time.deltaTime);
            // 移动角色
            // controller.Move(targetDir.normalized * (speed * Time.deltaTime));
        }
    }

    void OnMove(InputValue inputValue)
    {
        moveValue = inputValue.Get<Vector2>();
```

```
        }


}
```

- 为PersonGroup**Character Controller,**调整centerY**Character Controller**可以看做是为PersonGroup增加了一个网格胶囊 **Character Controlle**增加的网格胶囊有一部分在地板之下，因此需要将Center调整到与PersonGroupw完全一致



版本号 #11
由 董列涛 创建于 6 一月 2025 16:39:10
由 董列涛 更新于 7 一月 2025 16:18:25