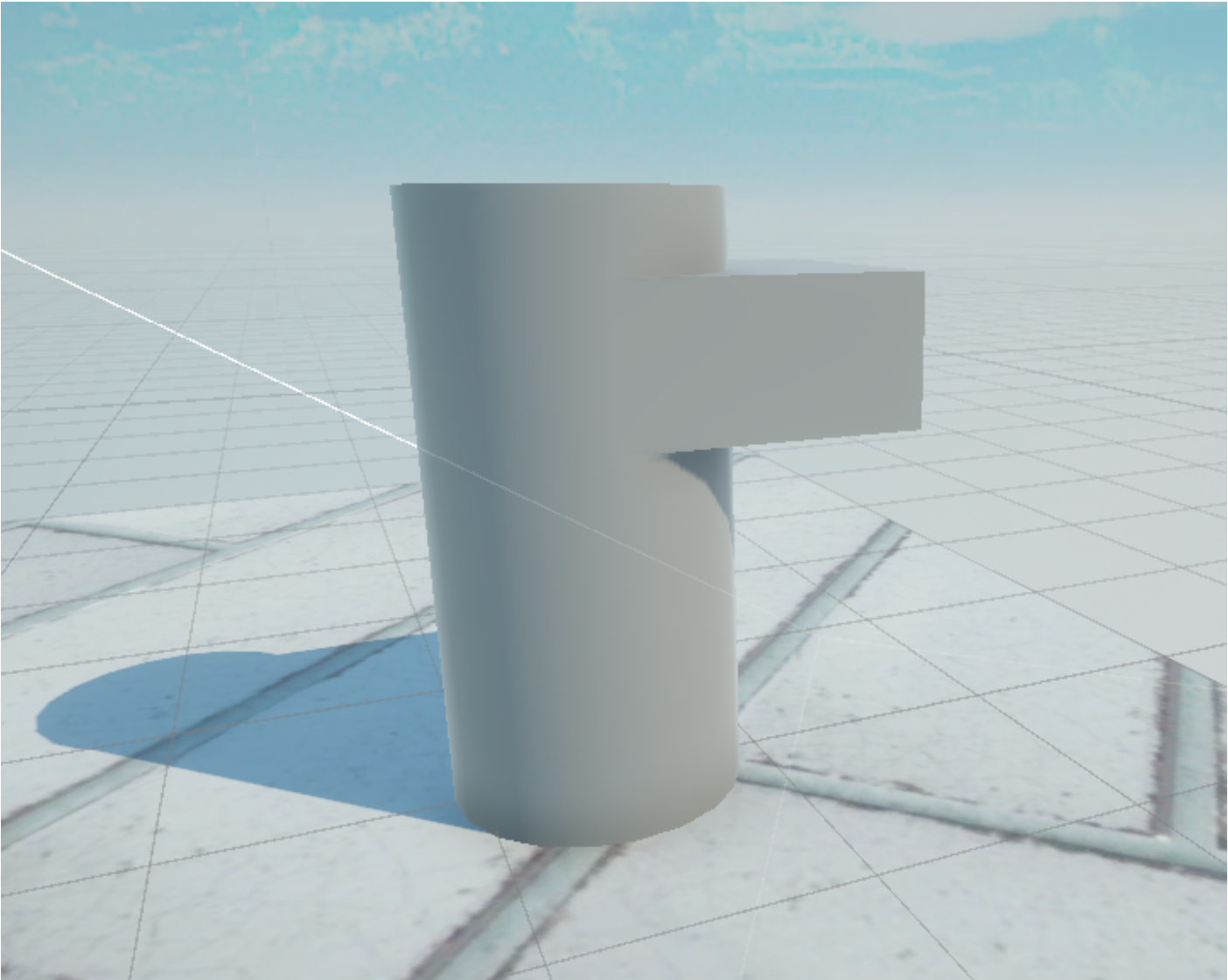




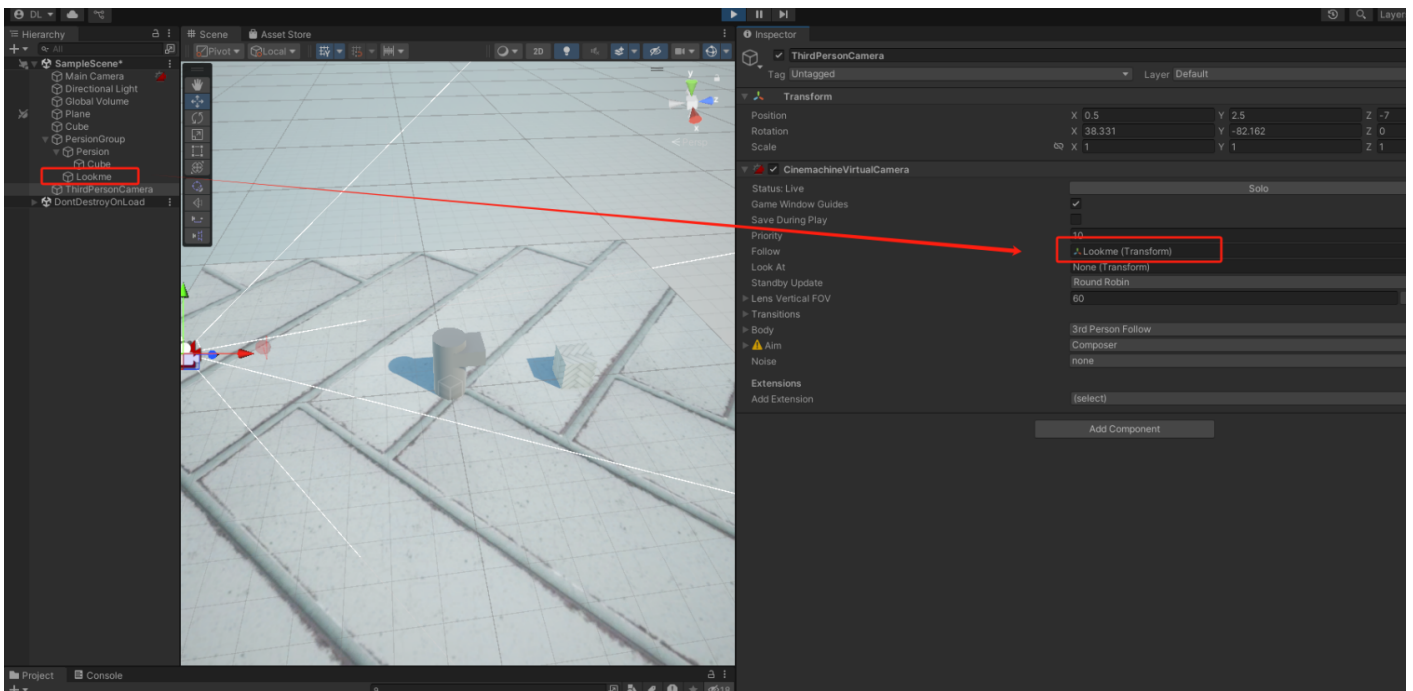
- Cinemachine
- Input System



- `GameObject`, `PersonGroup`
- `PersonGroup` `(cylinder)` `Person`
- `Person` `(cube)`,

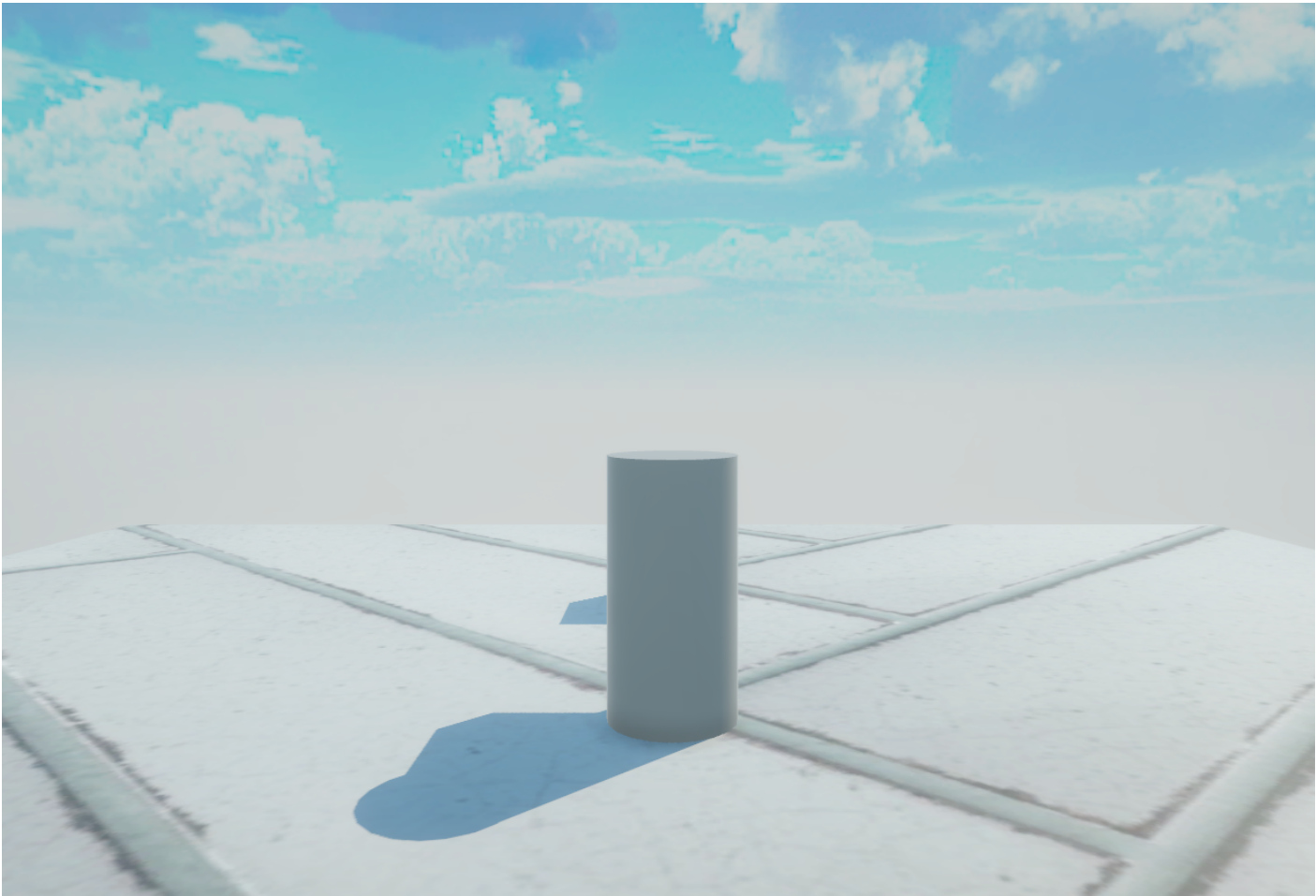


- PersonGroup Lookme position:0,0,0
- GameObject->Cine machine->Virtual Camera ThirdPersonCamera
- Lookmet 3rd follow

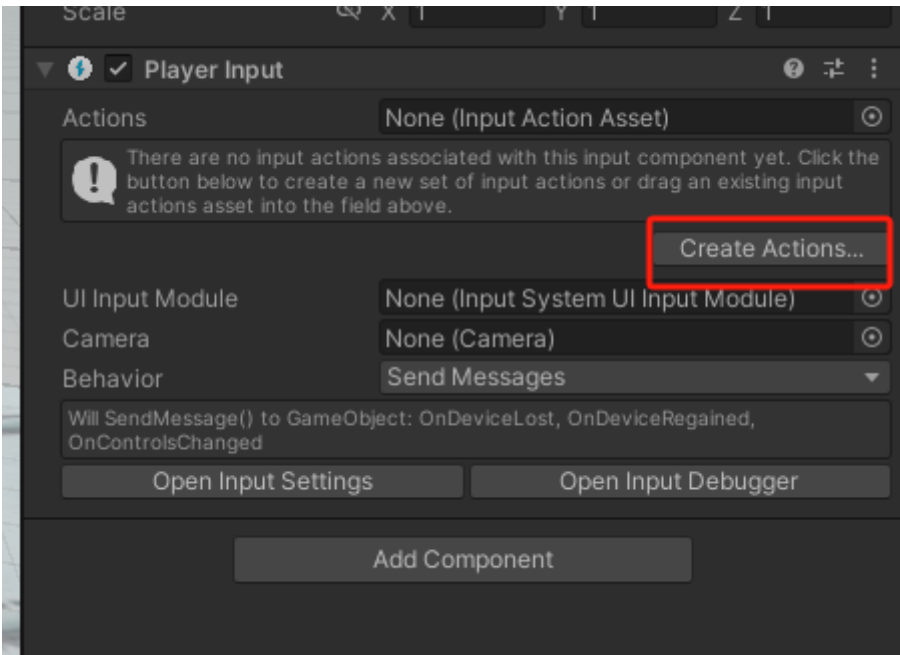


- - Third Person Camera lens
    - Vertical Fov: 60
    - dutch 0
  - Third Person Camera body
    - Shoulder Offset 3rd
    - Vertical Arm Length
    - Camera Side
    - Camera Distance

lookme position Third Person Camera lookme



- PersionGroup input, player input create actions



- 2 ThirdPersonCameraLook.cs ThirdPersonCameraMove.cs
- ThirdPersonCameraLook.cs PersionGroups

```

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

public class ThirdPersonCameraLook : MonoBehaviour
{
    private GameObject mainCamera;

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

    [Tooltip("")]
    public float TopClamp = 70.0f;

    [Tooltip("")]
    public float BottomClamp = 0.0f;

    [Tooltip("")]
    public float RotationSpeed = 1.0f;

    // 
    private const float threshold = 0.01f;
    //  Y 
    private float cinemachineTargetYaw;
    //  X 
    private float cinemachineTargetPitch;
    // 
    private Vector2 lookup;
    // 
    private bool isRotating = false;
    void Start()
    {
        if (mainCamera == null)
        {
            // 
            mainCamera = GameObject.FindGameObjectWithTag("MainCamera");
        }
        //  Y 

```

```

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

    // Update is called once per frame
    void Update()
    {
        // [REDACTED]
        isRotateing = Mouse.current.rightButton.isPressed;
        // [REDACTED]
        if (isRotateing && lookup.sqrMagnitude >= threshold)
        {
            cinemachineTargetYaw += lookup.x;
            cinemachineTargetPitch += lookup.y;
        }
        // [REDACTED]
        cinemachineTargetYaw = ClampAngle(cinemachineTargetYaw, float.MinValue,
float.MaxValue);
        cinemachineTargetPitch = ClampAngle(cinemachineTargetPitch, BottomClamp, TopClamp);

        // [REDACTED]
        // Quaternion targetRotation = Quaternion.Euler(cinemachineTargetPitch,
cinemachineTargetYaw, 0.0f);
        // CameraTarget.transform.rotation = Quaternion.Lerp(CameraTarget.transform.rotation,
targetRotation, Time.deltaTime * RotationSpeed);
        // [REDACTED]
        CameraTarget.transform.rotation = Quaternion.Euler(cinemachineTargetPitch,
cinemachineTargetYaw, 0.0f);
    }
    // [REDACTED]
    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)
    { // 視線を向く
        lookup = value.Get<Vector2>();
    }
}

```

- **ThirdPersonCameraLook.cs** 第三人称カメラの移動と回転

```

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

public class ThirdPersonCameraMove : MonoBehaviour
{
    // 参照するカメラ
    private GameObject mainCamera;
    // 移動速度
    private CharacterController controller;

    // 移動方向
    private Vector2 moveValue;

    // 回転速度
    [Tooltip("回転速度")]
    public float speed = 1.0f;
    // 目標回転
    private float targetRotation = 0.0f;

    // 回転スMOOTHタイム
    public float RotationSmoothTime = 0.1f;
    // 回転速度
    private float rotationVelocity = 0.0f;

    void Start()
    {
        mainCamera = GameObject.FindGameObjectWithTag("MainCamera");
        controller = GetComponent<CharacterController>();
    }

    void Update()
    {
        // 移動
        Move();
        // 回転
        Rotate();
    }

    void Move()
    {
        // 移動方向
        Vector2 moveDir = new Vector2(Input.GetAxisRaw("Horizontal"), Input.GetAxisRaw("Vertical"));
        moveValue = moveDir * speed * Time.deltaTime;
        controller.Move(moveValue + Vector3.up * Time.deltaTime);
    }

    void Rotate()
    {
        // 目標回転
        float targetRotationY = Input.GetAxisRaw("Mouse X") * speed * Time.deltaTime;
        targetRotation += targetRotationY;
        // 回転速度
        float currentRotationY = transform.eulerAngles.y;
        float rotationDifference = targetRotationY - currentRotationY;
        rotationVelocity = Mathf.Lerp(rotationVelocity, rotationDifference, RotationSmoothTime * Time.deltaTime);
        float smoothRotationY = currentRotationY + rotationVelocity;
        transform.eulerAngles = new Vector3(0, smoothRotationY, 0);
    }
}

```

```

{
    if (mainCamera == null)
    {
        mainCamera = GameObject.FindGameObjectWithTag("MainCamera");
    }
    // [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
    controller = GetComponent<CharacterController>();
}

void Update()
{
    // [ ] [ ] [ ] [ ] Y [ ] [ ] [ ] [ ]
    Vector3 velocity = new Vector3(0, -1, 0);
    if (moveValue != Vector2.zero)
    {
        // [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
        Vector3 inputDir = new Vector3(moveValue.x, 0.0f, moveValue.y).normalized;
        // [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
        targetRotation = Mathf.Atan2(inputDir.x, inputDir.z) * Mathf.Rad2Deg +
mainCamera.transform.eulerAngles.y;

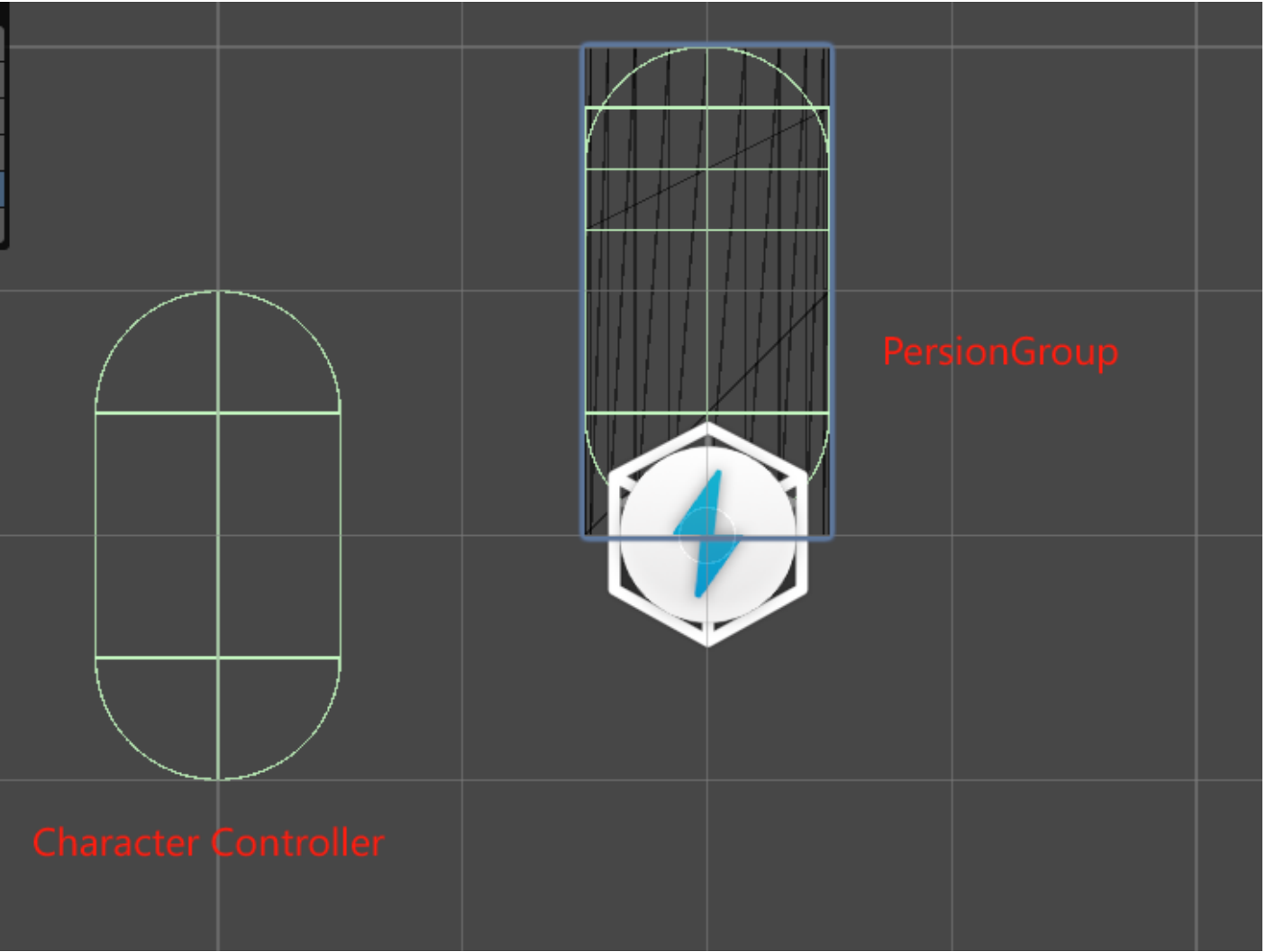
        // [ ] [ ] [ ] [ ]
        float rotation = Mathf.SmoothDampAngle(transform.eulerAngles.y, targetRotation,
ref rotationVelocity, RotationSmoothTime);
        // [ ] [ ] [ ] [ ]
        transform.rotation = Quaternion.Euler(0.0f, rotation, 0.0f);
        // [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
        Vector3 targetDir = Quaternion.Euler(0.0f, targetRotation, 0.0f) *
Vector3.forward;
        // [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
        velocity += targetDir.normalized * (speed * Time.deltaTime);
        // [ ] [ ] [ ] [ ]
        // controller.Move(targetDir.normalized * (speed * Time.deltaTime));
    }
}

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

```

```
    }  
  
}
```

- `PersonGroup` **Character Controller**, `center` **Character Controller** `PersonGroup`  
**Character Controller** `Center` `PersonGroupw`



#11  
6 2025 16:39:10  
7 2025 16:18:25