

```

1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using UnityEngine;
5
6  public class PlayerController : MonoBehaviour
7  {
8      [Header("References")]
9      private CharacterController Controller;
10
11     [SerializeField] private Transform camera;
12
13     [Header("Movement Settings")]
14     [SerializeField] private float walkSpeed = 5f;
15     [SerializeField] private float sprintSpeed = 10f;
16     [SerializeField] private float sprintTransitSpeed = 5f;
17     [SerializeField] private float turningSpeed = 2f;
18     [SerializeField] private float gravity = 9.81f;
19     [SerializeField] private float jumpHeight = 2f;
20     private float verticalVelocity;
21     private float speed;
22
23     [Header("Input")]
24     private float moveInput;
25     private float turninput;
26
27
28     private void Start()
29     {
30
31
32         Controller = GetComponent<CharacterController>();
33     }
34
35
36     private void Update()
37     {
38         InputManagement();
39         Movement();
40
41     }
42
43     private void Movement()
44     {
45         GroundMovement();
46         Turn();
47     }
48
49     private void GroundMovement()
50     {
51         Vector3 move = new Vector3(turninput, 0, moveInput);
52         move = camera.transform.TransformDirection(move);
53
54         if (Input.GetKey(KeyCode.LeftShift))
55         {
56             speed = Mathf.Lerp(speed, sprintSpeed, sprintTransitSpeed * Time.deltaTime);
57         }
58         else
59         {

```

```

59         speed = Mathf.Lerp(speed, walkSpeed, sprintTransitSpeed * Time.deltaTime);
60     }
61
62     move.y = 0;
63     move *= speed;
64
65     move.y = VerticalForceCalculation();
66
67
68     Controller.Move(move * Time.deltaTime);
69 }
70
71 private void Turn()
72 {
73     if (Mathf.Abs(turninput) < 0 || Mathf.Abs(moveInput) > 0)
74     {
75         Vector3 currentLookDirection = Controller.velocity.normalized;
76         currentLookDirection.y = 0;
77
78         currentLookDirection.Normalize();
79
80
81         Quaternion targetRotation = Quaternion.LookRotation(currentLookDirection);
82
83         transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation, Time.deltaTime * turningSpeed);
84     }
85 }
86
87 private float VerticalForceCalculation()
88 {
89     if (Controller.isGrounded)
90     {
91         verticalVelocity = -1f;
92
93         if (Input.GetButtonDown("Jump"))
94         {
95             verticalVelocity = Mathf.Sqrt(jumpHeight * gravity * 2);
96         }
97     }
98     else
99     {
100         verticalVelocity -= gravity * Time.deltaTime;
101     }
102     return verticalVelocity;
103 }
104
105 private void InputManagement()
106 {
107     moveInput = Input.GetAxis("Vertical");
108     turninput = Input.GetAxis("Horizontal");
109 }
110
111 public abstract class Subject: MonoBehaviour
112 {
113     //a collection of all observers
114     private List<IObserver> _observers = new List<IObserver>();

```

```
115
116     public void AddObserver(IObserver observer){
117         _observers.Add(observer);
118     }
119
120     public void RemoveObserver(IObserver observer) {
121         _observers.Remove(observer);
122     }
123
124     protected void NotifyObservers(string action)
125     {
126         _observers.ForEach((_observers) =>
127         {
128             _observers.OnNotify(action);
129         });
130     }
131
132
133
134     }
135
136
137 }
```