HEALTH (HEALTH SCRIPT)

```
using Unity.VisualScripting;
using UnityEngine;

public class Health : MonoBehaviour
{
    public float currHealth;
    public float maxHealth;
    public bool isDead;
    public Animator anim;
    public bool isPlayer;
    public bool bossEnemy;
    private void Start()
    {
        currHealth = maxHealth;
    }
    public void TakeDamage(float damage)
    {
      //currHealth-=damage;
      TakeHealth(damage);
     //PlayerControls.instance.TakeDamage(currHealth);
    }
    public void TakeHealth(float health)
    {
        if (isDead)
            return;
        if (currHealth <= 0 && !isDead)
        {
            isDead = true;
            anim.SetTrigger("isDead");
```

```
            if(isPlayer)
            {
                GameManager.instance.Gameover();
            }
            if (bossEnemy)
            {
                GameManager.instance.Won();
            }
            if(!isPlayer) {
                GameManager.instance.Healing();
            }
            return;
        }

        currHealth -= health;
        if(isPlayer)
        {
            GameManager.instance.UpdatePlayerHealthUI(currHealth);
        }

    }
}
```

MOVEMENT (PLAYER CONTROLS)

```
/*using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerControls : MonoBehaviour
{
    public Animator anim;
    void Start()
    {

    }

    void Update()
    {
        NewControls();
        //  Shooting();
    }
    public float targetRotation;
    public Transform cameraT;
    public float turnSmoothTime = 0.2f;
    float turnSmoothVelocity;
    public Rigidbody rb;
    public float InputVal;
    public float walkSpeed = 2;
    public float runSpeed = 6;
    public float speedSmoothTime = 0.1f;
    float speedSmoothVelocity;
    public float currentSpeed;
    private void NewControls()
```

```
{
    float h = Input.GetAxis("Horizontal");
    float v = Input.GetAxis("Vertical");
    Vector3 input = new Vector3(h, 0, v);//H indicates the player is moving in X axis and V indicates the player is moving in Z axis.
    Vector3 inputDir = input.normalized;//it converts the value or normalizes the value to 1
    Transform t = transform;
    targetRotation = Mathf.Atan2(inputDir.x, inputDir.z) * Mathf.Rad2Deg + cameraT.eulerAngles.y;//it is used for rotating the player on the camera inputs.
    t.eulerAngles = Vector3.up * Mathf.SmoothDampAngle(t.eulerAngles.y, targetRotation, ref turnSmoothVelocity, turnSmoothTime);//this is used for assigning the target rotation value to the player.
    InputVal = input.magnitude;
    bool running = Input.GetKey(KeyCode.LeftShift);
    float targetSpeed = ((running) ? runSpeed : walkSpeed) * input.magnitude;
    currentSpeed = Mathf.SmoothDamp(currentSpeed, targetSpeed, ref speedSmoothVelocity, speedSmoothTime);
    // t.Translate(currentSpeed * Time.deltaTime * t.forward, Space.World);
    Vector3 velocity = transform.forward * currentSpeed; // Calculate the velocity

    rb.velocity = velocity; // Set the Rigidbody's velocity
    if (InputVal <= 0)
    {
        anim.SetBool("isWalking", false);
        anim.SetBool("isRunning", false);
        return;
    }
    if (running)
    {
        Debug.Log("Player is Running");
        anim.SetBool("isRunning", true);
        anim.SetBool("isWalking", false);
```

```
        }
        else
        {
            Debug.Log("Player is walking");
            anim.SetBool("isWalking", true);
            anim.SetBool("isRunning", false);
        }
    }
    public void Shooting()
    {
        if (Input.GetMouseButton(0))
        {
            anim.SetBool("isShooting", true);
        }
        else
        {
            anim.SetBool("isShooting", false);
        }
    }
}*/


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

public class PlayerControls : MonoBehaviour
{
    public static PlayerControls instance;
```

```
public Animator anim;
public float maxHealth;
public float currHealth;
private void Awake()
{
    instance = this;
}
void Start()
{
    currHealth = maxHealth;
}

void Update()
{
    NewControls();
    //  Shooting();
    Attacking();
}
public float targetRotation;
public Transform cameraT;
public float turnSmoothTime = 0.2f;
float turnSmoothVelocity;
public Rigidbody rb;
public float InputVal;
public float walkSpeed = 2;
public float runSpeed = 6;
public float speedSmoothTime = 0.1f;
float speedSmoothVelocity;
public float currentSpeed;
public LayerMask layers;
```

```
public WeaponData weaponData;
public ParticleSystem muzzle;
public bool isAttacking;
public bool isDead;
private void NewControls()
{
    if (health.isDead)
    {
        isDead = true;
        return;
    }
    float h = Input.GetAxis("Horizontal");
    float v = Input.GetAxis("Vertical");
    Vector3 input = new Vector3(h, 0, v);//H indicates the player is moving in X axis and V indicates the player is moving in Z axis.
    Vector3 inputDir = input.normalized;//it converts the value or normalizes the value to 1
    Transform t = transform;
    targetRotation = Mathf.Atan2(inputDir.x, inputDir.z) * Mathf.Rad2Deg + cameraT.eulerAngles.y;//it is used for rotating the player on the camera inputs.
    t.eulerAngles = Vector3.up * Mathf.SmoothDampAngle(t.eulerAngles.y, targetRotation, ref turnSmoothVelocity, turnSmoothTime);//this is used for assigning the target rotation value to the player.
    InputVal = input.magnitude;
    bool running = Input.GetKey(KeyCode.LeftShift) && !isAttacking;
    float targetSpeed = ((running) ? runSpeed : walkSpeed)*input.magnitude;
    currentSpeed = Mathf.SmoothDamp(currentSpeed, targetSpeed, ref speedSmoothVelocity, speedSmoothTime);
    // t.Translate(currentSpeed * Time.deltaTime * t.forward, Space.World);
    Vector3 velocity = transform.forward; // Calculate the velocity

    rb.velocity = velocity * currentSpeed; // Set the Rigidbody's velocity
    if (anim == null)
```

```
            return;
        if (InputVal <= 0)
        {
            anim.SetBool("isWalking", false);
            anim.SetBool("isRunning", false);
            return;
        }
        if (running)
        {
            Debug.Log("Player is Running");
            anim.SetBool("isRunning", true);
            anim.SetBool("isWalking", false);
        }
        else
        {
            Debug.Log("Player is walking");
            anim.SetBool("isWalking", true);
            anim.SetBool("isRunning", false);
        }
    }
    public float fireRate = 15f;
    public float nextTimeToFire;
    public Health health;
    public GameObject trails;
    public void Attacking()
    {
        if (anim == null || health.isDead)
            return;
        if (Input.GetKeyDown(KeyCode.Space))
        {
```

```
        //anim.SetBool("isShooting", true);
        //anim.Play("Attack");
        isAttacking = true;
        anim.SetTrigger("isAttacking");
        trails.SetActive(true);
        Invoke(nameof(disableTrails), 3);
    }
    else
    {
        isAttacking = false;
        // anim.SetBool("isShooting", false);
      //  trails.SetActive(false);
    }
}
void disableTrails()
{
    trails.SetActive(false);
}
public void TakeDamage(float health)
{
    if (isDead)
        return;
    if (currHealth <= 0 && !isDead)
    {
        isDead = true;
        anim.SetTrigger("Dead");
        //anim.SetBool("isShooting", false);
        muzzle.Stop();
        return;
    }
```

```
        currHealth -= health;
        // GameManager.instance.UpdatePlayerHealthUI(currHealth);
    }
}
[Serializable]
public class WeaponData
{
    public float damageValue;
    public float shootRange;
}
```

ENEMY MOVEMENTS (AI NAVIGATION)

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

public class EnemyAI : MonoBehaviour
{
    public NavMeshAgent agent;
    public Transform player;
    public Animator animator;

    // Patrol related variables
    public Transform[] waypoints;
    private int waypointIndex;
    private float patrolSpeed = 3.5f;

    // Chase related variables
    private float chaseSpeed = 5.0f;
    private float chaseRange = 10.0f;

    // Attack related variables
    private bool isAttacking = false;
    private float attackRange = 2.0f;
    private float attackDamage = 10f;

    // Health related variables
    public float health = 100f;
    private bool isDead = false;
    public Health healthscript;
```

```
void Start()
{
    agent = GetComponent<NavMeshAgent>();
    animator = GetComponent<Animator>();
    player = GameObject.FindGameObjectWithTag("Player").transform;
    waypointIndex = 0;
    Patrol();
}

void Update()
{
    if(healthscript.isDead)
    {
        return;
    }
    float distanceToPlayer = Vector3.Distance(transform.position, player.position);

    if (distanceToPlayer <= chaseRange)
    {
        Chase();
    }
    else if (distanceToPlayer <= attackRange)
    {
        Attack();
    }
    else
    {
        if (!isAttacking)
        {
```

```
            Patrol();
            animator.SetBool("isChasing", false);
        }
    }

    if (health <= 0 && !isDead)
    {
        Die();
    }
}

void Patrol()
{
    isAttacking = false;
    agent.speed = patrolSpeed;
    if (Vector3.Distance(transform.position, waypoints[waypointIndex].position) < 1f)
    {
        waypointIndex = (waypointIndex + 1) % waypoints.Length;
    }
    agent.SetDestination(waypoints[waypointIndex].position);

    // Set animation state to patrol
    animator.SetBool("isChasing", false);
    animator.SetBool("isAttacking", true);
}

void Chase()
{
    isAttacking = false;
    agent.speed = chaseSpeed;
```

```
    agent.SetDestination(player.position);
    transform.LookAt(player);
    // Set animation state to chase
    animator.SetBool("isChasing", true);
    animator.SetBool("isAttacking", true);
}

void Attack()
{
    if(PlayerControls.instance.isDead)
    {
        isAttacking = false;
        return;
    }
    isAttacking = true;

    // Stop the enemy from moving
    agent.SetDestination(transform.position);
    transform.LookAt(player);
    // Set animation state to attack
    animator.SetTrigger("Attack");

    // Implement attack logic here (e.g., reduce player health)
    //player.GetComponent<PlayerHealth>().TakeDamage(attackDamage);
}

void TakeDamage(float damage)
{
    health -= damage;
    animator.SetTrigger("Hit");
```

```
}

void Die()
{
    if(isDead)
    {
        return;
    }
    isDead = true;
    animator.SetTrigger("Death");
    GameManager.instance.Healing();
    Destroy(gameObject, animator.GetCurrentAnimatorStateInfo(0).length);
}

void OnTriggerEnter(Collider other)
{

    if (other.CompareTag("Enemy"))
    {

        EnemyAI enemy = other.GetComponent<EnemyAI>();
        if (enemy != null)
        {

            enemy.TakeDamage(20);
        }

        TakeDamage(10);
    }
}}
```

GUI (MAIN MENU)

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

public class MainMenu : MonoBehaviour
{
    public GameObject Info;
    public void PlayGame()
    {
        SceneManager.LoadSceneAsync(1);
    }

    public void QuitGame()
    {
        Application.Quit();
    }
    public void InfoButton()
    {
        Info.SetActive(true);
    }
}
```

GAME SOUND (SOUND MANAGER)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI; // Add this namespace for Image class

public class SoundManager : MonoBehaviour
{
    [SerializeField] public Image soundOnIcon;
    [SerializeField] public Image soundOffIcon;
    private bool muted = false;

    // Start is called before the first frame update
    void Start()
    {
        if (!PlayerPrefs.HasKey("muted"))
        {
            PlayerPrefs.SetInt("muted", 0);
            Load();
        }
        else
        {
            Load();
        }
        UpdateButtonIcon();
        AudioListener.pause = muted;
    }

    public void OnButtonPress()
    {
```

```
    if (muted == false)
    {
        muted = true;
        AudioListener.pause = true;
    }
    else
    {
        muted = false;
        AudioListener.pause = false;
    }
    Save();
    UpdateButtonIcon();
}

private void UpdateButtonIcon()
{
    if (muted == false)
    {
        soundOnIcon.enabled = true;
        soundOffIcon.enabled = false;
    }
    else
    {
        soundOnIcon.enabled = false;
        soundOffIcon.enabled = true;
    }
}

private void Load()
{
```

```
        muted = PlayerPrefs.GetInt("muted") == 1;
    }

    private void Save()
    {
        PlayerPrefs.SetInt("muted", muted ? 1 : 0); // Fix typo: "mute" -> "muted"
    }
}
```

WEAPON ATTACKING (SWORD)

```
using UnityEngine;

public class SwordA : MonoBehaviour
{
    public int damage = 10; // Damage dealt by the sword
    public bool isPlayer;
    public bool isEnemy;
    public GameObject bloodEffectPrefab;
    //public GameObject snowslash;

    public AudioSource swordAudioSource;
    public AudioClip hitEnemySound;
    public AudioClip playerAttackSound;
    public Health mainHealth;

    private void Start()
    {
        mainHealth=GetComponentInParent<Health>();
    }

    void OnTriggerEnter(Collider other)
    {
        if (mainHealth.isDead)
            return;
        if (isPlayer && other.CompareTag("Enemy"))
        {
            TakeDamage(other.GetComponent<Health>());
        }
        else if (isEnemy && other.CompareTag("Player"))
```

```
    {
        TakeDamage(other.GetComponent<Health>());
    }
}

public void TakeDamage(Health health)
{
    if (health != null)
    {
        health.TakeDamage(damage);

        //Instantiate blood effect at the position of the enemy / player
        if (bloodEffectPrefab != null)
        {
            Instantiate(bloodEffectPrefab, health.transform.position, Quaternion.identity);
        }

        // Play sound effects
        if (isPlayer && swordAudioSource != null && playerAttackSound != null)
        {
            swordAudioSource.PlayOneShot(playerAttackSound);
        }
        else if (isEnemy && swordAudioSource != null && hitEnemySound != null)
        {
            swordAudioSource.PlayOneShot(hitEnemySound);
        }

        // Instantiate snowslash effect for player
        //if (isPlayer && snowslash != null)
        //{
```

```
            //    Instantiate(snowslash, transform.position, transform.rotation);
            //}
        }
    }
}
```

SCORE GUI (SCORE MANAGER)

```
using System;
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;
using UnityEngine.UI; // Add this namespace for UI elements

// ScoreManager.cs



public class ScoreManager : MonoBehaviour
{
    public int score = 0;
    public Text scoreText; // Reference to the UI Text displaying the score

    void Start()
    {
        // Initialize UI Text reference (assign it in the Inspector)
        scoreText.text = "Score: " + score;
    }

    public void IncreaseScore(int amount)
    {
        score += amount;
        scoreText.text = "Score: " + score;
    }
}
```

Map(MiniMap)

```
using System;
using System.Collections;
```

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

public class MiniMap : MonoBehaviour
{
   [SerializeField] private Transform player;
   void Update()
   {
      Vector3 newPosition = player.position;
      newPosition.y = transform.position.y;
      transform.position = newPosition;
   }
}
   // Update is called once per frame
```

GUI CONTROL (GAME MANAGER)

```
using System;
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.SocialPlatforms.Impl;
using UnityEngine.UI;
public class GameManager : MonoBehaviour
{
    public static GameManager instance;
    public Slider playerHealthSlider;
    public TextMeshProUGUI healthPercentText;
    public int health;
    public Image HealthBar;
    public Health playerhealth;
    public int score = 0;
    public TextMeshProUGUI scoreText;
    public GameObject gameover;
    public GameObject won;
    //public GameObject WaitingScreen;
    //public TextMeshProUGUI waitingScreenText;
    //public float waitingTime;
    //public float elapsedTime;
    private void Awake()
    {
        instance = this;
    }
    private void Start()
    {

    }



    // Start is called before the first frame update
    public void UpdatePlayerHealthUI(float curHealth)
    {
        float current=curHealth/ 100;

        HealthBar.fillAmount = current;
       // healthPercentText.text = damageValue.ToString();

    }
```

```
    public void UpdatePlayerHealthUI(int healthamount)
    {
        playerHealthSlider.value = healthamount;
        healthPercentText.text = healthamount.ToString();

    }


    internal void EndGame()
    {
        throw new NotImplementedException();
    }
    public void Healing()
    {
        Debug.Log("Healing");
        float current = HealthBar.fillAmount;
        float updatedhealth=0.2f + current;
        HealthBar.fillAmount = updatedhealth;
        playerhealth.currHealth = HealthBar.fillAmount*100;
        IncreaseScore(100);
    }
    public void IncreaseScore(int amount)
    {
        score += amount;
        scoreText.text = "Score: " + score;
    }
    public void Gameover()
    {
        gameover.SetActive(true);

    }
    public void Won()
    {
        won.SetActive(true);
    }
    public void restart()
    {
        SceneManager.LoadScene("MainGame");
    }
}
```