5주차

플레이어 체력 시스템 구현

강의 영상

https://youtu.be/RQrS87FQ2Mc

코드

Weapon.cs

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

public class Weapon : MonoBehaviour
{
    public int rpm = 700;
    private float fireInterval;
    private float fireTimer = 0f;

    public ParticleSystem muzzleFlash;
    public AudioClip fireSound;
    private AudioSource audioSource;

    public LayerMask layerMask;
    public GameObject bulletHolePrefab;

    public float defaultAccuracy = 0.2f;
    private float currentAccuracy;
    public float recoil = 0.1f;

    private Hud hud;

    public int ammoLeft = 30;
    public int maxAmmo = 30;
    private Animator animator;
    private bool isReloading = false;
    public Animator tpsAnimator;

    private void Awake()
    {
        currentAccuracy = defaultAccuracy;
        fireInterval = 60f / rpm;
        audioSource = GetComponent<AudioSource>();
        hud = FindObjectOfType<Hud>();
        animator = GetComponent<Animator>();
    }

    private void Update()
    {
        fireTimer += Time.deltaTime;
        if (fireTimer >= fireInterval)
        {
            if (Input.GetKey(KeyCode.Mouse0) && !isReloading)
            {
                // 총알 발사 처리 로직
                fireTimer = 0f;
                currentAccuracy += recoil;
                ammoLeft--;

                RaycastTarget();
                FireEffect();
            }
        }

        currentAccuracy = Mathf.Lerp(currentAccuracy, defaultAccuracy,
            Time.deltaTime * 10f);

        hud.UpdateCrosshairs(currentAccuracy + 0.05f);
        hud.UpdateAmmoText(ammoLeft, maxAmmo);
        if (ammoLeft <= 0 || (Input.GetKeyDown(KeyCode.R) && ammoLeft < maxAmmo))
        {
            isReloading = true;
            animator.SetBool("isReloading", true);
            tpsAnimator.SetBool("isReloading", true);
        }
    }

    private void FireEffect()
    {
        muzzleFlash.Play();
        audioSource.PlayOneShot(fireSound);
    }

    private void RaycastTarget()
    {
        Vector2 circle = Random.insideUnitCircle * currentAccuracy;
        Vector3 direction = Camera.main.transform.forward
            + Camera.main.transform.up * circle.y
            + Camera.main.transform.right * circle.x;

        Ray ray = new Ray(Camera.main.transform.position, direction);

        RaycastHit hit;
        if (Physics.Raycast(ray, out hit, Mathf.Infinity, layerMask.value))
        {
            HealthControl hc = hit.collider.GetComponentInParent<HealthControl>();
            if (hc != null)
            {
                hc.OnHit(0, hit.point, ray.direction);
            }
            else
            {
                GameObject bh = Instantiate(bulletHolePrefab, hit.point, Quaternion.identity);
                Destroy(bh, 3f);
            }
        }
        else
        {
        }
    }

    public void AnimationEvent(string eventName)
    {
        if (eventName == "Weapon_Reload_Complete")
        {
            isReloading = false;
            ammoLeft = maxAmmo;
            animator.SetBool("isReloading", false);
            tpsAnimator.SetBool("isReloading", false);
        }
    }
```

```
}
```

PlayerControl.cs

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

public class PlayerControl : MonoBehaviour
{
    public enum MoveType { Idle, Walk }
    public Animator tpsAnimator;
    public MoveType moveType;

    public float mouseSensitivity = 100f;
    public Transform headTransform;
    private Vector3 moveDirection;
    private CharacterController characterController;
    private float headX = 0f;

    private void Awake()
    {
        characterController = GetComponent<CharacterController>();
    }

    private void Update()
    {
        MoveControl();
        LookControl();

        tpsAnimator.SetInteger("moveType", (int)moveType);
    }

    private void MoveControl()
    {
        float h = Input.GetAxisRaw("Horizontal");
        float v = Input.GetAxisRaw("Vertical");

        if (h == 0 && v == 0)
        {
            moveType = MoveType.Idle;
        }
        else
        {
            moveType = MoveType.Walk;
        }
        if (characterController.isGrounded)
        {
            moveDirection = new Vector3(h, -1f, v).normalized;
            moveDirection = this.transform.TransformDirection(moveDirection) * 10f;

            if (Input.GetKeyDown(KeyCode.Space))
            {
                moveDirection.y = 5f;
            }


            characterController.Move(moveDirection * Time.deltaTime);
        }
        else
        {
            moveDirection.y -= 10f * Time.deltaTime;
            characterController.Move(moveDirection * Time.deltaTime);
        }
    }

    private void LookControl()
    {
        float mouseX = Input.GetAxisRaw("Mouse X") * mouseSensitivity * Time.deltaTime;
        float mouseY = Input.GetAxisRaw("Mouse Y") * mouseSensitivity * Time.deltaTime;

        Vector3 bodyAngle = this.transform.eulerAngles;
        bodyAngle.y += mouseX;
        this.transform.eulerAngles = bodyAngle;

        headX -= mouseY;
        headX = Mathf.Clamp(headX, -80f, 80f);
        headTransform.localEulerAngles = new Vector3(headX, 0f, 0f);
    }
}
```

Hud.cs

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

public class Hud : MonoBehaviour
```

```
{
    public Transform[] crosshairs;
    public TextMeshProUGUI ammoLeftText;
    public Image bloodScreen;

    public void UpdateBloodScreen()
    {
        StopCoroutine(nameof(BloodScreenRoutine));
        StartCoroutine(nameof(BloodScreenRoutine));
    }

    private IEnumerator BloodScreenRoutine()
    {
        float alpha = 0.3f;
        while(alpha >= 0f)
        {
            Color color = bloodScreen.color;
            color.a = alpha;
            bloodScreen.color = color;

            alpha -= Time.deltaTime;
            yield return null;
        }
    }

    public void UpdateAmmoText(int ammoLeft, int maxAmmo)
    {
        ammoLeftText.text = ammoLeft + "/" + maxAmmo;
    }

    public void UpdateCrosshairs(float dist)
    {
        Vector3 upPosition = Camera.main.transform.position +
            Camera.main.transform.forward + Camera.main.transform.up * dist;
        Vector3 downtPosition = Camera.main.transform.position +
            Camera.main.transform.forward - Camera.main.transform.up * dist;
        Vector3 rightPosition = Camera.main.transform.position +
            Camera.main.transform.forward + Camera.main.transform.right * dist;
        Vector3 leftPosition = Camera.main.transform.position +
            Camera.main.transform.forward - Camera.main.transform.right * dist;

        crosshairs[0].position = Camera.main.WorldToScreenPoint(upPosition);
        crosshairs[1].position = Camera.main.WorldToScreenPoint(downtPosition);
        crosshairs[2].position = Camera.main.WorldToScreenPoint(rightPosition);
        crosshairs[3].position = Camera.main.WorldToScreenPoint(leftPosition);
    }
}
```

HealthControl.cs

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

public class HealthControl : MonoBehaviour
{
    public float health = 100f;
    public bool isDead = false;

    public GameObject bloodPrefab;
    private Rigidbody[] rigidbodies;
    private Hud hud;

    public Animator tpsAnimator;

    private void Awake()
    {
        hud = FindObjectOfType<Hud>();

        rigidbodies = GetComponentsInChildren<Rigidbody>();
        foreach(Rigidbody rb in rigidbodies)
        {
            rb.isKinematic = true;
        }
    }

    public void OnHit(int viewId, Vector3 hitPoint, Vector3 inDir)
    {
        if (!isDead)
        {
            health -= 20f;
            if (health <= 0f)
            {
                health = 0f;

                isDead = true;
                OnDead(viewId, inDir);
            }

            hud.UpdateBloodScreen();
        }

        GameObject blood = Instantiate(bloodPrefab, hitPoint, Quaternion.identity);
        Destroy(blood, 5f);
    }

    public void OnDead(int viewId, Vector3 inDir)
    {
        tpsAnimator.enabled = false;
        foreach(Rigidbody rb in rigidbodies)
        {
            rb.isKinematic = false;
        }

        rigidbodies[0].AddForce(inDir * 300f, ForceMode.Impulse);
    }
}
```