6주차

게임 네트워크 이론 / 멀티플레이 이식 (1)

강의 영상

https://youtu.be/wSqbrzdcfqQ

코드

Weapon.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using Unity.Burst.CompilerServices;
```

```
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;

    public ParticleSystem tpsMuzzleFlash;

    private PhotonView pv;
    private PlayerControl playerControl;

    private void Awake()
    {
        playerControl = GetComponentInParent<PlayerControl>();
        pv = GetComponent<PhotonView>();

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

    private void Start()
    {
        if (!pv.IsMine)
        {
            //this.gameObject.SetActive(false);
            Renderer[] renderers = GetComponentsInChildren<Renderer>();
            foreach(Renderer r in renderers)
            {
                r.enabled = false;
            }
        }
    }

    private void Update()
    {
        if (!pv.IsMine)
        {
            return;
        }

        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);
        }

        playerControl.isReloading = isReloading;
    }

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

        pv.RPC(nameof(RpcFireEffect), RpcTarget.Others);
    }

    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);

                pv.RPC(nameof(RpcOnWorldHit), RpcTarget.Others, hit.point);
            }
        }
        else
        {
        }
    }

    [PunRPC]
    public void RpcFireEffect()
    {
        tpsMuzzleFlash.Play();
        audioSource.PlayOneShot(fireSound);
    }

    [PunRPC]
    public void RpcOnWorldHit(Vector3 hitpoint)
    {
        GameObject bh = Instantiate(bulletHolePrefab, hitpoint, Quaternion.identity);
        Destroy(bh, 3f);
    }

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


}
```

PlayerControl.cs

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

public class PlayerControl : MonoBehaviour, IPunObservable
{
    public bool isReloading = false;
    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 PhotonView pv;

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

    private void Start()
    {
        if (pv.IsMine)
        {
            tpsAnimator.gameObject.SetActive(false);
        }
        else
        {
            GetComponentInChildren<Camera>().gameObject.SetActive(false);
        }
    }

    private void Update()
    {
        if (pv.IsMine)
        {
            MoveControl();
            LookControl();
        }
        tpsAnimator.SetInteger("moveType", (int)moveType);
        tpsAnimator.SetBool("isReloading", isReloading);
    }

    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);
    }
    public void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
    {
        if (stream.IsWriting)
        {
            stream.SendNext(isReloading);
            stream.SendNext(moveType);
        }
        else if (stream.IsReading)
        {
            isReloading = (bool)stream.ReceiveNext();
            moveType = (MoveType)stream.ReceiveNext();
        }
    }
}
```

GameManager.cs

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

public class GameManager : MonoBehaviour
{
    private void Start()
    {
        PhotonNetwork.Instantiate("Player", Vector3.zero, Quaternion.identity);
    }
}
```

Main.cs

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

public class Main : MonoBehaviourPunCallbacks
{
    private void Start()
    {
        PhotonNetwork.ConnectUsingSettings();
    }

    public override void OnConnectedToMaster()
    {
        Debug.Log("On connected to master");
        PhotonNetwork.JoinRandomOrCreateRoom();
    }

    public override void OnJoinedRoom()
    {
        PhotonNetwork.LoadLevel("InGame");
    }
}
```