MiniGameS

작성자 : 김재완

목차

[개발 컨셉]

[게임 구성]

개발 컨셉

1. 개요

항목	설명
제목	MiniGameS (미니게임s)
장르	캐주얼 1 : 1 대전 게임
디바이스 / 플랫폼	- 모바일 (안드로이드)
기획 의도	1. Photon2 사용을 위한 간단한 포트폴리용 프로젝트 2. Photon2 을 이용하여 여러 동기화를 해보기위한 프로젝트
특징	1. Photon2 서버를 이용 2. Photon2에 있는 Photonview, Customproperties를 이용하여 동기화 진행
한 줄 소개	간단한 미니게임을 통해 대결하는 1 : 1 게임
제작 기간	2022.05.23 ~ 2022.07.04

개발 컨셉

2. 구현 목록 – 화면/시스템

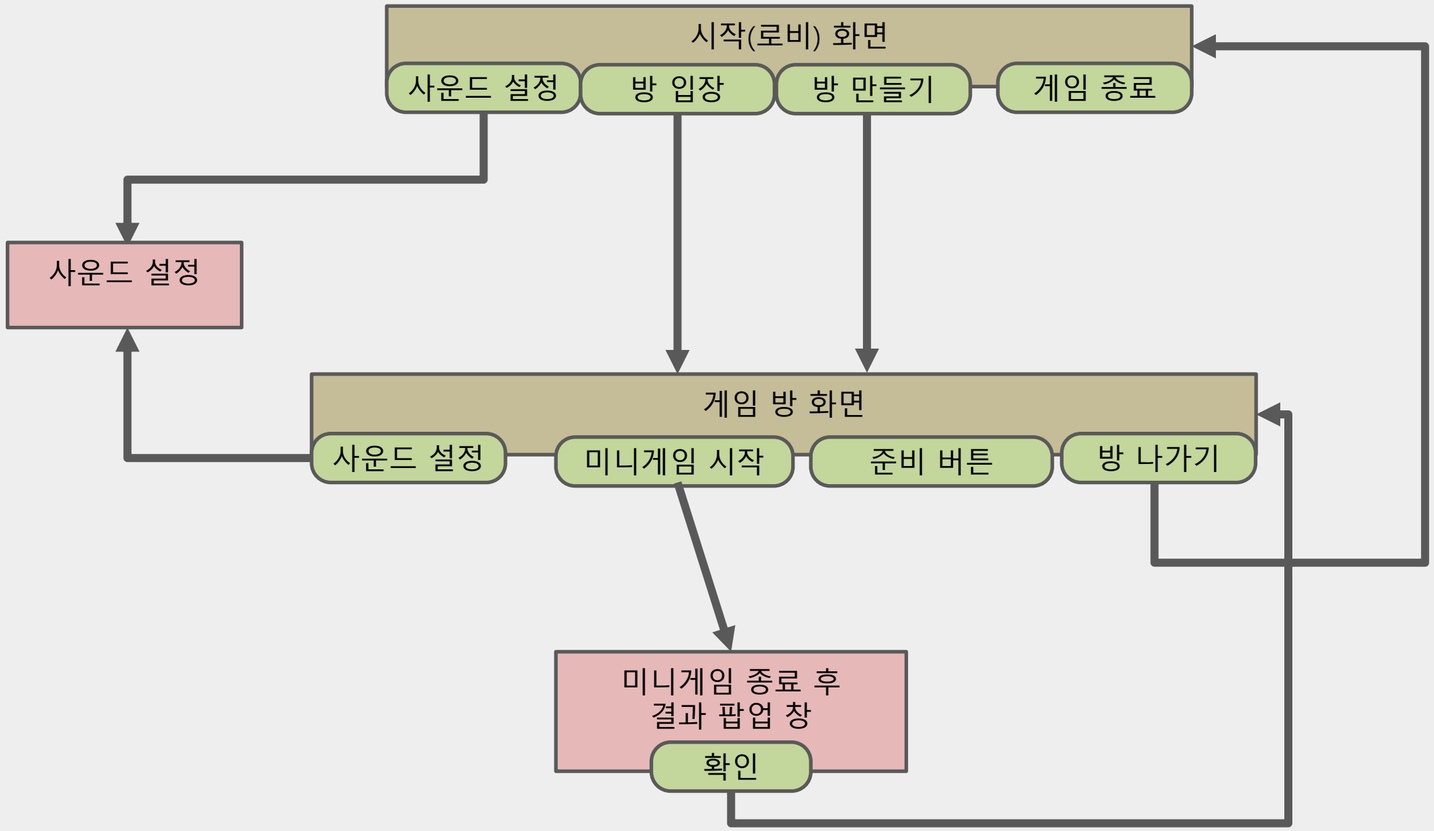
필수 구현 화면	설명
시작 화면	게임 이름, 방 목록, 게임 종료,
게임 로비 메인 화면	방 접속 플레이어 명단, 플레이어 캐릭터, 게임시작 버튼
각 게임 화면	점수, 게임 관련 이미지
사운드 설정창	사운드 설정

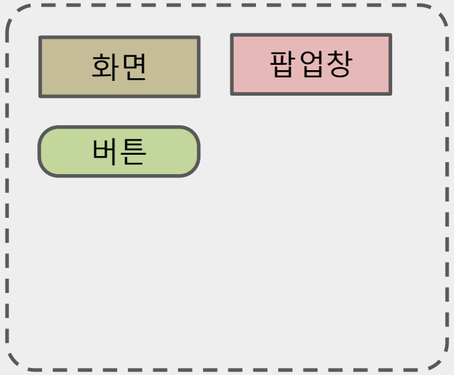
필수 구현 시스템	설명
방 생성 및 입장	Photon를 이용한 방 생성 및 입장
게임승리점수 동기화	Photon를 이용한 점수 동기화
각 미니게임 시스템	1. 과일먹는 게임
	2. OX 문제
	3. 계단 오르기 게임
	4. 기억력 게임

게임 구성

1. 게임 흐름도

(씬/화면을 기준으로 게임이 진행되는 흐름도)

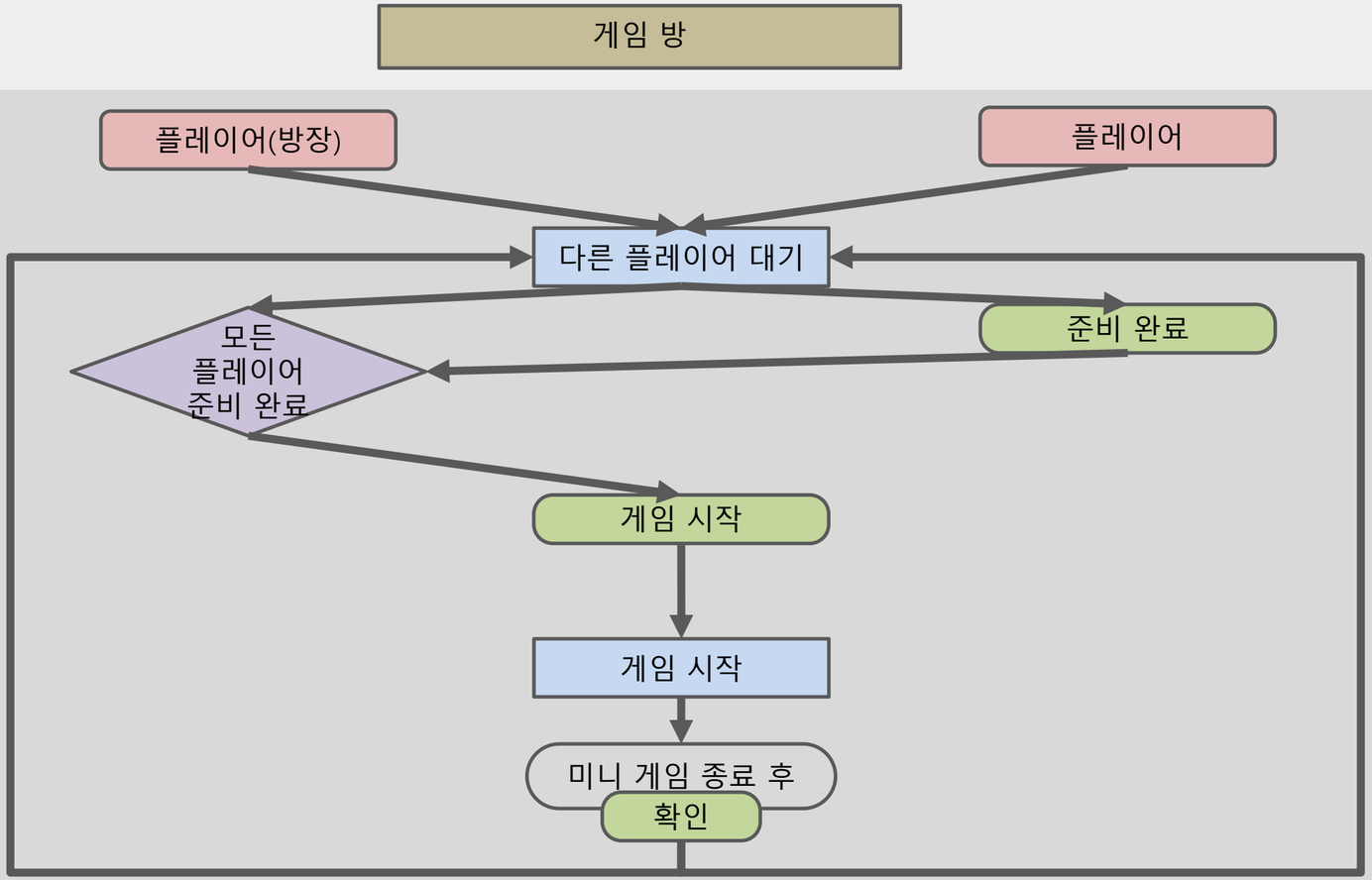


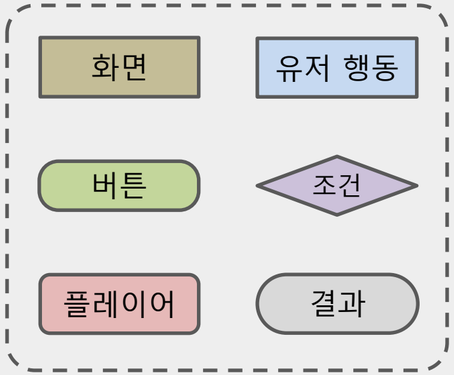


게임 구성

1. 게임 흐름도

(플레이하는 유저가 어떤 순서로 플레이하게 될 지)





게임 구성

2. Photon이용 하여 방 만들기

1) 닉네임 : 게임에서 사용할 이름
    1) 닉네임이 없으면 접속 불가
2) 방 이름 : 방 생성시 방이름
    1) 없으면 랜덤값을 이름이 정해짐
3) 캐릭터 선택 : 게임에서 사용할 캐릭터
4) 랜덤 방 입장
    1) 만들어진 방 중 입장 가능한 방에 들어간다.
    2) 만들어진 방이 없다면 방을 생성해서 들어간다.
5) 방 생성
    1) 새로운 방을 생성 한다.
6) 만들어진 방에 들어감
    1) 입장 가능한 방에 들어갈 수 있다.
    2) 1 : 1게임이기에 2명이 들어가면 풀방
7) 사운드 설정 : 소리 설정 가능
8) 게임 종료

게임 구성

2. Photon이용 하여 방 만들기

1) 방 생성 코드

```
참조 0개
public void ClickCreateRoom()
{
    //버튼 사운드 효과
    SoundMgr.Inst.PlayEffect(buttonSound);

    //포톤네트워크가 로비에 있는지 확인
    if (!PhotonNetwork.InLobby)
        return;

    if (!CheckNickName()) //닉네임 확인
        return;

    string _roomName = roomName.text;
    //룸 이름이 없거나 Null일 경우 룸 이름 지정
    if (string.IsNullOrEmpty(roomName.text))
    {
        _roomName = "ROOM_" + Random.Range(0, 999).ToString("000");
    }

    //로컬 플레이어의 이름을 설정
    PhotonNetwork.LocalPlayer.NickName = userNick.text;
    //플레이어 이름을 저장
    PlayerPrefs.SetString("USER_ID", userNick.text);

    //생성할 룸의 조건 설정
    RoomOptions roomOptions = new RoomOptions();
    roomOptions.IsOpen = true;      //입장 가능 여부
    roomOptions.IsVisible = true;  //로비에서 룸의 노출 여부
    roomOptions.MaxPlayers = 2;    //룸에 입장할 수 있는 최대 접속자 수

    //지정한 조건에 맞는 룸 생성 함수
    PhotonNetwork.CreateRoom(_roomName, roomOptions, TypedLobby.Default);
}
```

2) 랜덤 방 입장 코드

```
//랜덤 방 버튼 클릭 시 호출되는 함수
참조 0개
public void ClickJoinRandomRoom()         //3번 방 입장 요청 버튼 누름
{
    //버튼 사운드 효과
    SoundMgr.Inst.PlayEffect(buttonSound);

    if (!PhotonNetwork.InLobby)
        return;
    if (!CheckNickName()) //닉네임 확인
        return;


    //로컬 플레이어의 이름을 설정
    PhotonNetwork.LocalPlayer.NickName = userNick.text;
    //플레이어 이름을 저장
    PlayerPrefs.SetString("USER_ID", userNick.text);

    //무작위로 추출된 방으로 입장
    PhotonNetwork.JoinRandomRoom();
}
```

3) 방 입장 성공 시 코드

InGame씬으로 씬 전환코드 호출

```
//방입장 성공시
참조 3개
public override void OnJoinedRoom()
{
    //룸 씬으로 이동하는 코루틴 실행
    LoadMgr.Inst.LoadScene("InGame");
}
```

2. Photon이용 하여 방 만들기

4) Photon 방 업데이트 시 코드

OnRoomListUpdate는 Photon에 있는 함수 이다

```
참조 0개
public override void OnRoomListUpdate(List<RoomInfo> roomList)
{
    //내가 가진 룸정보와 비교하여 삭제 및 추가
    int roomCount = roomList.Count;
    for (int i = 0; i < roomCount; i++)
    {
        if (!roomList[i].RemovedFromList)
        {
            if (!myList.Contains(roomList[i]))
                myList.Add(roomList[i]);
            else
                myList[myList.IndexOf(roomList[i])] = roomList[i];
        }
        else if (myList.IndexOf(roomList[i]) != -1)
            myList.RemoveAt(myList.IndexOf(roomList[i]));
    }

    //룸 목록을 다시 받았을 때 갱신하기 위해 기존에 생성된 RoomItem을 삭제
    foreach (var obj in GameObject.FindObjectsOfType<RoomItem>())
        Destroy(obj.gameObject);

    //스크롤 영역 초기화
    scrollContents.GetComponent<RectTransform>().sizeDelta = Vector2.zero;

    for (int i = 0; i < myList.Count; i++)
    {
        if (!myList[i].IsVisible)
            continue;

        GameObject room = Instantiate(roomItem) as GameObject;
        room.transform.SetParent(scrollContents.transform, false);

        //생성한 RoomItem에 표시하기 위한 텍스트 정보 전달
        RoomItem roomData = room.GetComponent<RoomItem>();
        roomData.roomName = myList[i].Name;
        roomData.connectPlayer = myList[i].PlayerCount;
        roomData.maxPlayer = myList[i].MaxPlayers;

        //텍스트 정보를 표시
        roomData.DispRoomData(myList[i].IsOpen);
    }
}
```

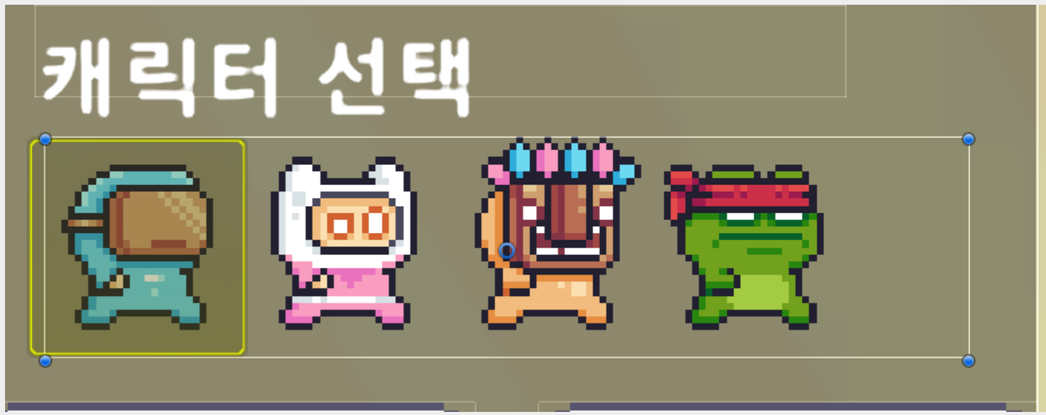
RoomItem Class 코드

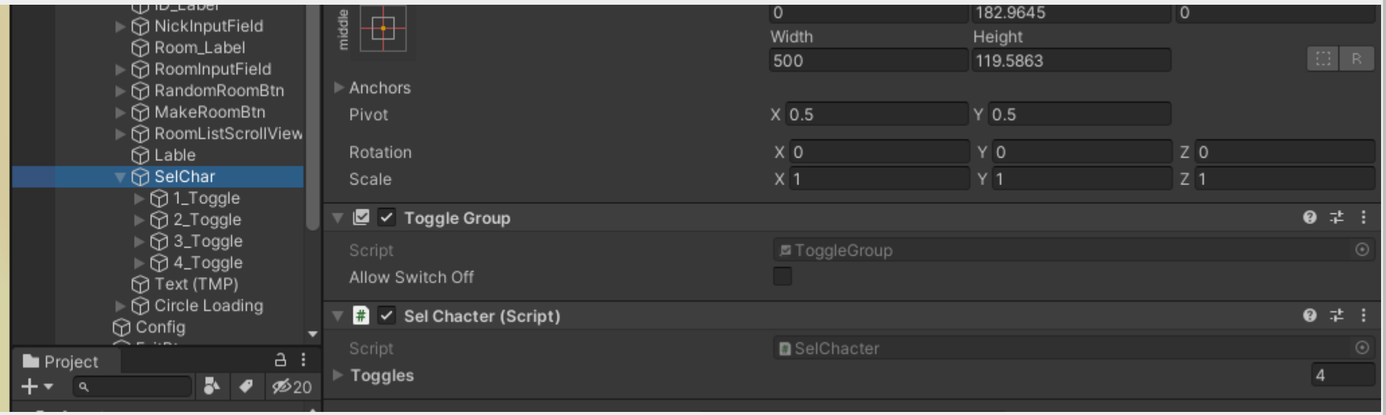
방 생성시 로비에 있는 방목록에 보여주는 클래스 입니다.

```
public class RoomItem : MonoBehaviour
{
    //외부 접근을 위해 public으로 선언했지만 Inspector에 노출하지 않음
    [HideInInspector] public string roomName = "";
    [HideInInspector] public int connectPlayer = 0;
    [HideInInspector] public int maxPlayer = 0;

    //룸 이름 표실할 Text UI 항목
    public Text textRoomName;
    //룸 접속자 수와 최대 접속자 수를 표시할 Text UI 항목
    public Text textConnectInfo;

    [HideInInspector] public string ReadyState = "";  //레디 상태 표시

    // Start is called before the first frame update
    참조 0개 Unity 메시지 | 참조 0개
    void Start()
    {
        this.GetComponent<Button>().onClick.AddListener(() =>
        {
            PhotonMgr RefPtInit = FindObjectOfType<PhotonMgr>();
            if (RefPtInit != null)
                RefPtInit.OnClickRoomItem(roomName);
        });
    }
    참조 1개
    public void DispRoomData(bool a_IsOpen)
    {
        if (a_IsOpen == true)
        {
            textRoomName.color = new Color32(0, 0, 0, 255);
            textConnectInfo.color = new Color32(0, 0, 0, 255);
        }
        else
        {
            textRoomName.color = new Color32(0, 0, 255, 255);
            textConnectInfo.color = new Color32(0, 0, 255, 255);
        }

        textRoomName.text = roomName;

        if (connectPlayer == maxPlayer)
            textConnectInfo.text = "풀방";
        else
            textConnectInfo.text = "(" + connectPlayer.ToString() + "/"
                                       + maxPlayer.ToString() + ")";
    }
}
```

2. Photon이용 하여 방 만들기

3) 캐릭터 선택





유니티 토글그룹을 이용하여 하나만 선택 가능하게 하였고 각 토글 마다 이미지에 맞는 캐릭터를 선택할 수 있도록 함수를 연결하였습니다.

```
Unity 스크립트(자산 참조 1개)|참조 0개
public class SelChacter : MonoBehaviour
{
    public Toggle[] toggles;
    Unity 메시지|참조 0개
    private void Start()
    {
        toggles[(int)UserData.CharName].isOn = true;
    }
    참조 0개
    public void SelChar1(bool isOn) {  if (isOn) UserData.CharName = CharName.Virtual_Guy ; }
    참조 0개
    public void SelChar2(bool isOn) {  if (isOn) UserData.CharName = CharName.Pink_Man; }
    참조 0개
    public void SelChar3(bool isOn) {  if (isOn) UserData.CharName = CharName.Mask_Dude; }
    참조 0개
    public void SelChar4(bool isOn) { if (isOn) UserData.CharName = CharName.Ninja_Frog; }

}
```

```
참조 6개
public enum CharName
{
    Virtual_Guy,
    Pink_Man,
    Mask_Dude,
    Ninja_Frog,
}

참조 6개
public class UserData
{
    static public CharName CharName = CharName.Virtual_Guy;
}
```

게임 구성

3. 게임 방 로비 화면

1) 게임 방 정보 : 접속한 플레이어들의 닉네임, 승점
    1) 자신의 이름이 위에 있고 방장은 앞에 별이 붙는다
2) 게임 시작 버튼 및 준비 버튼
    1) 방장이 아닌 플레이어는 준비버튼이 활성화 되고
    2) 방장은 모든 플레이어가 준비완료가 되면 게임시작 버튼이 활성화 된다.
3) 채팅 창
4) 팔레트 : 닉네임의 색깔을 바꿀 수 있다.
5) 플레이어 캐릭터
    1) 방장은 닉네임 옆에 별이 있다
    2) 준비 완료시 머리위에 표시
6) 캐릭터 조작 버튼

3. 게임 방 로비 화면

1) 방 입장 시 설정 코드

InGame.cs

```
Unity 스크립트 | 참조 23개
public class InGame : MonoBehaviourPunCallbacks
{
    public static InGame Inst; //싱글턴을 위한

    PhotonView pv;  //포톤 동기화를 위한 포톤뷰

    //캐릭터 스폰 위치
    public GameObject spawnPos; //게임입장시 중심 스폰위치

    //동기화를 위한 변수 선언
    ExitGames.Client.Photon.Hashtable playerHash = new ExitGames.Client.Photon.Hashtable();

    [Header("UI")]
    public GameObject roomCanavas;  //게임 방 정보 창
    public GameObject pallet;       //닉네임 색깔 창
    public GameObject configAndLobbyBtn;  //설정 버튼 나가기버튼
    public Button readyBtn;               //레디 버튼    ... 반장은 없는 버튼
    public Text readyTxt;                 //레디 버튼 (준비완료, 준비) 를 나타낼 텍스트
    public Button StartBtn;               //시작 버튼 .... 반장만 나올 버튼


    public TextMeshProUGUI myNickName;          // 내닉네임
    public TextMeshProUGUI otherNickName;       //상대 닉네임
    public TextMeshProUGUI myWinCountTxt;       //내 승점
    public TextMeshProUGUI otherWinCountTxt;  //상대 승점
    public Button soundBtn; //사운드 설정버튼

    [Header("ResultUI")]
    public ResultUI resultUI;

    [Header("Game")]
    public GameObject roomMark; //방장마크
    public GameRollController GameRoll; //게임 선택을 위한 룰러
    public Game[] MiniGame;       //미니게임이 담겨있는

    TalkBox talkBox;

    public  MyColor myNickNameColor = MyColor.black;
    public  MyColor otherNickNameColor = MyColor.black;

    //캐릭터
    public PlayerCharacter[] playerCharacters = new PlayerCharacter[2];
    bool isReady = false;   //레디 상태
```

```
Unity 메시지 | 참조 0개
private void Awake()
{
    Inst = this;

    //PhotonView 컴포넌트 할당
    pv = GetComponent<PhotonView>();

    talkBox = GetComponent<TalkBox>();

    //이미 다른 플레이어가 있으면 그캐릭터 닉네임 색깔 적용
    if (PhotonNetwork.PlayerListOthers.Length > 0)
        otherNickNameColor = (MyColor)((int)PhotonNetwork.PlayerListOthers[0].CustomProperties["NickColor"]);

    //게임 종류갯수 적용
    GameRoll.GameCount = (int)GameType.Last;
}

// Start is called before the first frame update
Unity 메시지 | 참조 0개
void Start()
{
    SoundMgr.Inst.PlayBGM("InGame");

    //방에 입장에 성공적이면 통신 시작
    PhotonNetwork.IsMessageQueueRunning = true;
    //패널 셋팅
    InitPanel();

    //플레이어들 만들기
    CreatePlayer();

    //플레이어 정보 SetCustomProperties 시키기
    playerHash.Add("winCount", 0);
    playerHash.Add("ready", false);

    PhotonNetwork.LocalPlayer.SetCustomProperties(playerHash);

    //방 설정 끝나면 방 활성화
    if (PhotonNetwork.IsMasterClient)
        PhotonNetwork.CurrentRoom.IsVisible = true;
}
```

```
참조 1개
void InitPanel() //입장시 패널 설정
{
    //사운드 설정하는 버튼 함수 연결
    soundBtn.onClick.AddListener(SoundMgr.Inst.OnSoundCtrlBox);
    //패널 셋팅,버튼 UI 셋팅하기
    StartBtn.gameObject.SetActive(false);
    //내닉네임 적용
    myNickName.text = "<color=" + myNickNameColor.ToString() + ">" + PhotonNetwork.LocalPlayer.NickName + "</color>";

    if (PhotonNetwork.IsMasterClient) //방장일경우
    {
        readyBtn.gameObject.SetActive(false);
        roomMark.transform.SetParent(myNickName.transform, false);
    }
    else
    {
        readyBtn.gameObject.SetActive(true);
        roomMark.transform.SetParent(otherNickName.transform, false);
    }

    SetWinCount();
}

참조 1개
void CreatePlayer() //캐릭터 만들기
{
    //랜덤한 위치에 만들어주기
    Vector3 a_HPos = Vector3.zero;
    Vector3 a_AddPos = Vector3.zero;
    GameObject a_HPosObj = GameObject.Find("SpawnPos");

    if (a_HPosObj != null)
    {
        a_AddPos.x = Random.Range(-2.0f, 2.0f);
        a_HPos = spawnPos.transform.position + a_AddPos;
    }
    //포톤으로 캐릭터 스폰하기
    PhotonNetwork.Instantiate("Player", a_HPos, Quaternion.identity, 0);
}
```

3. 게임 방 로비 화면

2) 게임 시작 – 준비/시작 하기

OnPlayerPropertiesUpdate으로

변화를 감지할수 있다.

PunRPC으로

모든 플레이어 들에게 호출

```
#region GameController 게임 진행 관련 함수
//OnPlayerProperties 변화를 감지해서 레디를 확인해서 게임 시작할수 있도록
참조 2개
public override void OnPlayerPropertiesUpdate(Player targetPlayer
        , ExitGames.Client.Photon.Hashtable changedProps)
{
    if (!PhotonNetwork.LocalPlayer.IsMasterClient)
        return;

    if (targetPlayer != PhotonNetwork.LocalPlayer)
    {
        if (changedProps.ContainsKey("ready"))
        {
            StartBtn.gameObject.SetActive((bool)changedProps["ready"]);
        }
    }
}

참조 0개
public void ReadyBtn()  //레디 버튼
{
    SoundMgr.Inst.PlayEffect("Button");

    isReady = !isReady;

    if (isReady)
        readyTxt.text = "준비완료";
    else
        readyTxt.text = "준비하기";

    //레디 정보를 저장해서 방장쪽에서 확인할수 있게 한다.
    playerHash = PhotonNetwork.LocalPlayer.CustomProperties;
    playerHash["ready"] = isReady;
    PhotonNetwork.LocalPlayer.SetCustomProperties(playerHash);

    //캐릭터 레디 표시
    playerCharacters[0].Ready(isReady);
}
```

```
참조 0개
public void GameStartBtn() //방장만 누를수 있는버튼
{
    SoundMgr.Inst.PlayEffect("Button");
    //모든 플레이어에게 전달해 게임을진행한다.
    pv.RPC("GameSelStart", RpcTarget.AllViaServer);
}

[PunRPC]
참조 0개
public void GameSelStart()
{
    //채팅창 클리어
    talkBox.ClearText();
    //준비완료 한거 준비하기로 초기화/방장이든 아니든 초기화
    readyTxt.text = "준비하기";
    isReady = false;
    playerHash = PhotonNetwork.LocalPlayer.CustomProperties;
    playerHash["ready"] = isReady;
    PhotonNetwork.LocalPlayer.SetCustomProperties(playerHash);
    //UI 꺼주기
    roomCanavas.SetActive(false);
    pallet.SetActive(false);
    configAndLobbyBtn.SetActive(false);

    GameRoll.roller.SetActive(true);

    //플레이어 캐릭터 등록하기
    playerCharacters[1] = GameObject.FindGameObjectWithTag("OtherPlayer").GetComponent<PlayerCharacter>();
    playerCharacters[1].Ready(false);

    //주사위 돌리는거 혹시 모를 딜레이를 위한
    playerHash = PhotonNetwork.LocalPlayer.CustomProperties;
    if (playerHash.ContainsKey("DiceEnd"))
        playerHash["DiceEnd"] = false;
    else
        playerHash.Add("DiceEnd", false);
    PhotonNetwork.LocalPlayer.SetCustomProperties(playerHash);


    if (PhotonNetwork.LocalPlayer.IsMasterClient)
        StartCoroutine(StartSelGame());
}
```

```
참조 1개
IEnumerator StartSelGame()  //게임 돌림판 돌리기
{
    yield return null;
    int curGame = GameRoll.Roll(); //돌림판을 돌려나온 다음 게임 번호
    yield return null;

    while (!GameRoll.EndRoll()) //돌림판이 멈추면
        yield return null;
    // 정해진 미니게임 시작하기
    pv.RPC("StartMiniGame", RpcTarget.AllViaServer, curGame);
}

[PunRPC]
참조 0개
void StartMiniGame(int idx)//정해진 미니게임 활성화 하기
{
    GameRoll.waitText.SetActive(false);
    GameRoll.roller.SetActive(false);
    MiniGame[idx].StartGame();
}
```

3. 게임 방 로비 화면

2) 캐릭터 동기화 - 1

PlayerCharacter.cs

```
Unity 스크립트|참조 10개
public class PlayerCharacter : MonoBehaviourPunCallbacks, IPunObservable
{
    Rigidbody2D rigidbody;
    Animator animator;
    SpriteRenderer spriteRenderer;
    [HideInInspector]public PhotonView pv;

    //플레이어 닉네임
    public TextMeshPro nickNameTxt;
    public GameObject starImg;
    public GameObject ready;

    //방향 값과 이동속도 값
    [SerializeField] int h = 0;
    [SerializeField]  Vector2 velocity = Vector2.zero;

    //원격 조종용 변수 (동기화를 위한)
    public Vector3 currPos = Vector3.zero; //위치
    public bool isMove = true;

    Unity 메시지|참조 0개
    private void Awake()
    {
        rigidbody = GetComponent<Rigidbody2D>();
        animator = GetComponent<Animator>();
        spriteRenderer = GetComponent<SpriteRenderer>();
        pv = GetComponent<PhotonView>();
    }
```

```
void Start()
{
    InGame gameMgr = FindObjectOfType<InGame>();

    if (pv.IsMine)//내 캐릭터 등록 시키기
    {
        animator.SetInteger("Char", (int)UserData.CharName);

        FindObjectOfType<CharController>().player = this;
        //내캐릭이 먼저 보이게하게
        transform.position -= Vector3.forward;
        this.tag = "Player";
        //닉네임 색깔
        nickNameTxt.text = "<color=" + InGame.Inst.myNickNameColor.ToString() + ">" + pv.Owner.NickName + "</color>";
        InGame.Inst.playerCharacters[0] = this;

    }
    else    //원격 동기화 캐릭터일 경우
    {
        rigidbody.gravityScale = 0.0f;     //중력 끄기
        this.tag = "OtherPlayer";   //태그 설정
        //닉네임 색깔
        nickNameTxt.text = "<color=" + InGame.Inst.otherNickNameColor.ToString() + ">" + pv.Owner.NickName + "</color>";
        InGame.Inst.playerCharacters[1] = this;
    }


    //방장이 아니면 방장표시 끄기
    if (!pv.Owner.IsMasterClient)
        starImg.SetActive(false);
}
```

3. 게임 방 로비 화면

2) 캐릭터 동기화 - 2

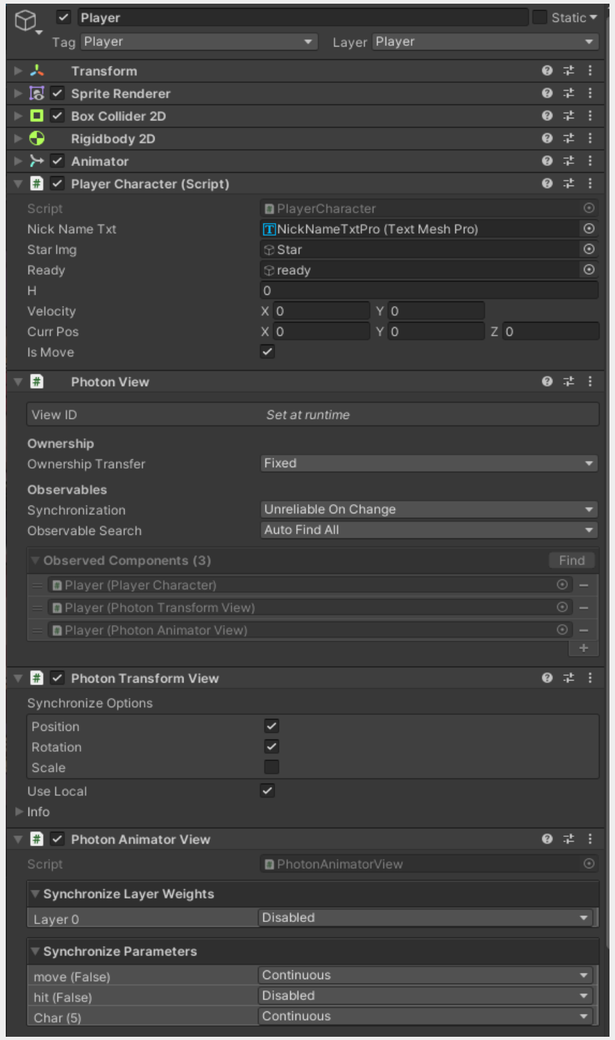
PlayerCharacter.cs

```
//원격동기화
참조 13개
public void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
{
    //로컬 플레이어의 위치 정보 송신
    if (stream.IsWriting)
    {
        stream.SendNext(spriteRenderer.flipX ? 1 : 0);
        stream.SendNext(animator.GetInteger("Char"));
        stream.SendNext(nickNameTxt.text);
    }
    else //원격 플레이어의 위치 정보 수신
    {
        spriteRenderer.flipX = ((int)stream.ReceiveNext()) == 1 ? true : false;
        animator.SetInteger("Char", (int)stream.ReceiveNext());
        nickNameTxt.text = (string)stream.ReceiveNext();
    }
}

참조 2개
public void SetHit()
{
    pv.RPC("RPCHit", RpcTarget.Others);
    animator.SetTrigger("hit");
}

[PunRPC] //트리거 동기화 다른 플레이어도 보여지게
참조 0개
void RPCHit() { animator.SetTrigger("hit"); }

참조 3개
public void Ready(bool bReady) //머리위 레디 표시
{
    ready.SetActive(bReady);
    pv.RPC("RPCReady", RpcTarget.Others, bReady);
}

[PunRPC]// 동기화를 위한
참조 0개
void RPCReady(bool bReady) { ready.SetActive(bReady); }
```

OnPhotonSerializeView를
이용해서 추가 동기화 진행

[PunRPC]를 이용해서
필요시에만 호출해준다.

PhotonView를 이용하여 캐릭터의
애니메이션과 위치를 동기화 시킬수 있음



4. 미니게임 구현

메인 게임 Class

모든 미니게임은 Game. Class를 상속 받아 사용한다.

```
public class Game : MonoBehaviourPunCallbacks
{
    //Photon
    protected PhotonView pv;
    //동기화를 위한 변수 선언
    protected  ExitGames.Client.Photon.Hashtable playerHash;
    //점수
    protected int score = 0;


    Unity 메시지 | 참조 0개
    private void Awake()
    {
        Init();
    }

    참조 7개
    protected virtual void Init()
    {
        pv = GetComponent<PhotonView>();
    }

    참조 9개
    public virtual void StartGame()
    {
        //등록된 점수 초기화
        playerHash = PhotonNetwork.LocalPlayer.CustomProperties;
        if (playerHash.ContainsKey("score"))
            playerHash["score"] = 0;
        else
            playerHash.Add("score", 0);

        PhotonNetwork.LocalPlayer.SetCustomProperties(playerHash);

        score = 0;

    }
```

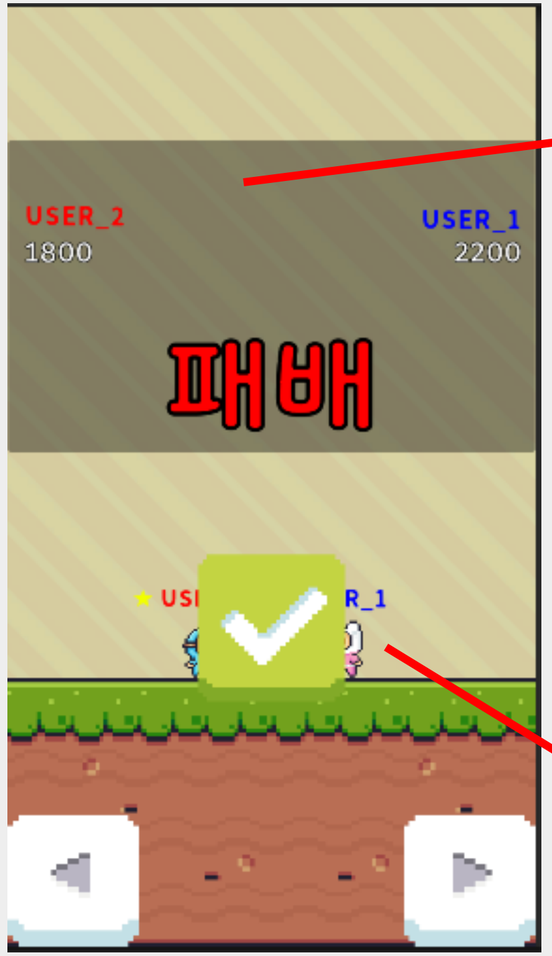
PhotonView를 이용하여 클라이언트끼리 상황을 동기화 시킨다.

초기 셋팅

게임을 시작하는 함수

미니게임 종료 시 나오는 결과 창

ResultUI.cs




```
public class ResultUI : MonoBehaviour
{
    static public ResultUI Inst;
    [Header("ResultPanel")]
    public GameObject ResultPanel;          //결과창
    public TextMeshProUGUI myNickTxt;       //내 닉네임
    public TextMeshProUGUI otherNickTxt;    //상대 닉네임
    public TextMeshProUGUI myScoreTxt;      // 내 점수
    public TextMeshProUGUI otherScoreTxt;   //상대 점수
    public TextMeshProUGUI winOrLose;       //승리판정
    public GameObject ok_Btn;               //확인 버튼

    참조 1개
    public void SetResult()//결과창 셋팅
    {
        ResultPanel.SetActive(true);
        SoundMgr.Inst.PlayEffect("ResultOpenSound");
        SoundMgr.Inst.PlayBGM("InGame");

        myNickTxt.text = "<color=" + InGame.Inst.myNickNameColor.ToString() + ">" + PhotonNetwork.LocalPlayer.NickName + "</color>";
        otherNickTxt.text = "<color=" + InGame.Inst.otherNickNameColor.ToString() + ">" + PhotonNetwork.PlayerListOthers[0].NickName + "</color>";
        //각 점수 가져오기
        int myscore = PhotonNetwork.LocalPlayer.CustomProperties.ContainsKey("score") ? (int)PhotonNetwork.LocalPlayer.CustomProperties["score"] : 0;
        int otherscore = PhotonNetwork.PlayerListOthers[0].CustomProperties.ContainsKey("score") ? (int)PhotonNetwork.PlayerListOthers[0].CustomProperties["score"] : 0;
        //점수 표시
        myScoreTxt.text = myscore.ToString();
        otherScoreTxt.text = otherscore.ToString();
        //승리 판정하기
        if (myscore == otherscore)
        {
            winOrLose.text = "무승부";
            winOrLose.color = Color.green;
        }
        else if (myscore > otherscore)
        {
            winOrLose.text = "승리";
            winOrLose.color = Color.blue;
            InGame.Inst.WinGame(); //본 게임메니저에서 승리 카운트 해주기
        }
        else if (myscore < otherscore)
        {
            winOrLose.text = "패배";
            winOrLose.color = Color.red;
        }
        ok_Btn.SetActive(true);
    }
```

```
참조 0개
public void OnOkBtn()  //게임 종료후 확인버튼 누르면
{
    //미니게임 종료
    ok_Btn.SetActive(false);
    ResultPanel.SetActive(false);
    //화면 갱신해주기
    InGame.Inst.SetLobby();
}
```

게임 구성

4. 미니게임 구현 - 과일먹기

일정시간 동안 떨어지는 과일을 많이 먹는 게임

1. 일정시간 동안 과일 스폰
2. 과일 하나당 100점
3. 점수 높은 사람이 승점 + 1

FallingFruitGame.CS - 1

```
0 Unity 스크립트 | 참조 5개
public class FallingFruitGame : Game
{
    참조 1개
    enum FruitsType //과일 오브젝트를 불러오기 위한 ...
    public static FallingFruitGame Inst;    //싱글턴 패턴을 위한
    public PlayerCharacter[] playerObj;   //과일들의 중돌캐릭터들의 거리 계산을 위한 플레이어들 캐릭터변수
    public GameObject fruitsSpanwPos; //과일의 중심스폰위치

    //과일 먹고 나올 이펙트
    public GameObject collectObj;   //과일을 먹고 나올 이펙트 프리팹
    Transform collectObjTr; //이펙트를 넣을 부모오브젝트
    Queue<GameObject> collectQu = new Queue<GameObject>();  //오브젝트 풀로 사용할 큐

    [Header("UI")]
    public GameObject GamePanel;        //점수, 타이머 캔버스
    public TextMeshProUGUI scoreTxt;    //점수
    public TextMeshProUGUI CountTxt;    //남은시간

    int count = 30; //타이머

    float spawnTimer = 0.0f;       //게임시간
    float nextSpawnTime = 1.0f; //과일 스폰 주기
    bool gameStart = false;  //게임 상태 bool 변수

    참조 7개
    protected override void Init()
    {
        base.Init();
        Inst = this;

        collectObjTr = new GameObject("CollectObjPool").transform;
        collectObjTr.SetParent(transform);

        //미리 이펙트 만들어놓기
        for (int i = 0; i < 10; i++)
        {
            GameObject obj = Instantiate(collectObj, collectObjTr);
            obj.SetActive(false);
            collectQu.Enqueue(obj);
        }

    }
```

```
참조 9개
public override void StartGame()
{
    base.StartGame();
    SoundMgr.Inst.PlayBGM("FallingFruitGame");

    GamePanel.SetActive(true);
    //씬에 있는 플레이어 오브젝트 불러오기
    playerObj = InGame.Inst.playerCharacters;
    //점수 셋팅
    scoreTxt.text = "0";
    //게임 로직 스타트
    StartCoroutine(Game_Co());

    if (PhotonNetwork.IsMasterClient)   //과일 스폰
        StartCoroutine(SpawnFruits_Update());
}
```

게임 구성

4. 미니게임 구현 - 과일먹기

FallingFruitGame.CS - 2

```
참조 1개
IEnumerator Game_Co()
{
    gameStart = true;
    count = 30; //30초

    CountTxt.gameObject.SetActive(true);
    CountTxt.text = count.ToString();
    while (count >= 0)
    {
        yield return new WaitForSeconds(1.0f);
        count--;
        CountTxt.text = count.ToString();
    }

    //게임 종료
    gameStart = false;
    //과일오브젝트 삭제
    if (PhotonNetwork.LocalPlayer.IsMasterClient)
    {
        Fruits[] fruits = FindObjectsOfType<Fruits>();
        for (int i = 0; i < fruits.Length; i++)
        {
            PhotonNetwork.Destroy(fruits[i].gameObject);
        }
    }

    CountTxt.text = "종료!!";
    yield return new WaitForSeconds(1.5f);
    CountTxt.text = "결과발표";
    yield return new WaitForSeconds(1.0f);
    CountTxt.text = "";



    InGame.Inst.ShowResult();
    GamePanel.SetActive(false);

    StopAllCoroutines();
}
```

카운트 해주기

타이머 종료후

스폰된 과일 제거

결과 보여주기

InGame에 ShowResutl 호출해서

게임 종료

```
참조 1개
IEnumerator SpawnFruits_Update()
{
    while(gameStart)
    {
        yield return null;

        //과일 스폰
        spawnTimer += Time.deltaTime;
        if (spawnTimer >= nextSpawnTime)
        {
            int randCount = Random.Range(3, 8);
            for (int i = 0; i < randCount; i++)
            {
                SpawnFruits(); //과일 스폰
            }

            spawnTimer = 0.0f;
            nextSpawnTime = Random.Range(0.7f, 1.2f);
        }

    }
}
```

```
참조 1개
public void SpawnFruits()  //과일 스폰하기
{
    //랜덤과일 및 스폰위치 잡기
    int rand = Random.Range(0, (int)FruitsType.Max);
    float randx = Random.Range(-4.0f, 4.0f);
    float randy = Random.Range(-0.5f, 0.5f);

    //과일 리소스폴더에서 가져와 스폰하기.. 포톤으로 소환하여 모든 클라에게 전달
    GameObject fruitObj = PhotonNetwork.InstantiateRoomObject("Fruits", fruitsSpanwPos.transform.position + Vector3.right * randx + Vector3.up * randy, Quaternion.identity);
    fruitObj.GetComponent<Fruits>().type = rand;
}
```

과일 스폰 하기

일정 주기마다 랜덤갯수 스폰하기

게임 구성

4. 미니게임 구현 - 과일먹기

FallingFruitGame.CS - 3

```
참조 1개
public void GetFruit(PlayerCharacter player , Vector3 pos)//과일 충돌시 점수와 이펙트 소환 //호출되는 곳은 마스터 클라이언트에서만 호출된다.
{
    //PunRPC 점수 증가 함수 먹은 유저에게 결과 보내기
    pv.RPC("GetFruit", player.pv.Owner, pos);
}

[PunRPC]
참조 0개
void GetFruit(Vector3 pos)//점수증가 와 먹은 위치에 효과보여주시
{
    //플레이어 에게 점수 적용시켜주기
    playerHash = PhotonNetwork.LocalPlayer.CustomProperties;
    //포톤 플레이어 SetCustomProperties을 이용하여 동기화
    if (playerHash.ContainsKey("score"))
    {
        playerHash["score"] = (int)playerHash["score"] + 100;
        PhotonNetwork.LocalPlayer.SetCustomProperties(playerHash);
    }

    SpawnCollect(pos);  //획득 이펙트 보여주기
    SoundMgr.Inst.PlayEffect("GetFruits");
}

//PlayerProperties들이 업데이트 된다면 점수 갱신
참조 20개
public override void OnPlayerPropertiesUpdate(Player targetPlayer
            , ExitGames.Client.Photon.Hashtable changedProps)
{
    //자기자신일때
    if (targetPlayer.Equals(PhotonNetwork.LocalPlayer))
    {
        if (changedProps.ContainsKey("score"))
        {
            score = (int)changedProps["score"];
            scoreTxt.text = score.ToString();
        }
    }
}
```

과일 획득시 점수 올려주기

```
참조 1개
void SpawnCollect(Vector3 pos)//과일 먹을시 이펙트 나오게 하기  //추후 오브젝트 풀로 바꾸기
{
    GameObject collect;
    if (collectQu.Count <= 0)
        collect = Instantiate(collectObj, collectObjTr);
    else
        collect = collectQu.Dequeue();

    collect.SetActive(true);
    collect.transform.position = pos;

    StartCoroutine(CollectActiveOff(collect));
}

WaitForSeconds collectOffTime = new WaitForSeconds(0.3f);
참조 1개
IEnumerator CollectActiveOff(GameObject obj)
{
    yield return collectOffTime;

    obj.SetActive(false);
    collectQu.Enqueue(obj);
}
```

이펙트 큐에서 이펙트를 꺼내 사용후 **0.3**초 뒤에 다시 이펙트를 끄고 큐에 넣어준다.

4. 미니게임 구현 - 과일먹기

과일 오브젝트가 가지고 있는

Fruits.CS

```
public class Fruits : MonoBehaviourPunCallbacks
{
    PhotonView pv;
    Animator animator;

    PlayerCharacter player;
    float dis = 10000.0f;
    float temp = 0;

    public int type = 0;

    0 Unity 메시지 | 참조 13개
    public override void OnEnable()
    {
        pv = GetComponent<PhotonView>();
        animator = GetComponent<Animator>();
    }

    0 Unity 메시지 | 참조 0개
    private void FixedUpdate()
    {
        //PhotonView로 인해 위치 동기화 됨
        if (PhotonNetwork.LocalPlayer.IsMasterClient)
            transform.position += (Vector3.down * Time.fixedDeltaTime * 2);
    }
```

마스터클라이언트 에서만

이동시키기

```
// Update is called once per frame
0 Unity 메시지 | 참조 0개
void Update()
{
    if (PhotonNetwork.LocalPlayer.IsMasterClient)
    {
        animator.SetInteger("Fruits", type);
        bool get = false;
        for (int i = 0; i < FallingFruitGame.Inst.playerObj.Length; i++)
        {
            temp = Vector2.Distance(FallingFruitGame.Inst.playerObj[i].transform.position, transform.position);

            if (temp <= 0.7f)
            {
                get = true;
                if(temp < dis)  //가장 가까운 유저
                {
                    player = FallingFruitGame.Inst.playerObj[i];
                    dis = temp;
                }
            }
        }
        if(get)
        {
            FallingFruitGame.Inst.GetFruit(player, this.transform.position);
            PhotonNetwork.Destroy(this.pv);
        }
        if(transform.position.y < -7)
            PhotonNetwork.Destroy(this.pv);
    }
}
```

두명의 플레이어의 거리를

비교하여

범위안에 들어오면

점수 올리기

만약 범위안에 동시에

들어오면 가장 가까운

플레이어게 적용

게임 구성

4. 미니게임 구현 – OX문제

1) 타이머 : 시간이 끝나면 정답 공개
2) OX 문제 : OX문제 테이블에서 가져옴
3) OX선택 버튼
4) OX출력 : 유저들이 선택한것을 보여줌
5) 시간 종료시 결과

총 5문제 진행

많이 맞춘 사람이 승점 + 1

게임 구성

4. 미니게임 구현 - OX문제

OXGame.CS

```
public class OXGame : Game
{
    참조 36개
    enum OX
    {
        O, X, None,
    }
    //문제가 나오는 Text
    public TextMeshProUGUI questionTxt;
    //문제 테이블
    List<KeyValuePair<string, OX>> questionList = new List<KeyValuePair<string, OX>>();
    //문제 번호가 저장되어있는
    List<int> questionNum = new List<int>();
    int currQuestion = 0;   //현재 문제 인덱스
    int step = 0;           //현재 문제 순서
    bool bIng = false;      //타이머 종료 구하기
    //플레이어가 선택한것
    OX myChoose = OX.None;
    OX otherChoose = OX.None;
    bool choose = false;
    //OX 선택 버튼
    public GameObject O_Btn;
    public GameObject X_Btn;
    [Header("UI")]
    public GameObject GamePanel;
    public TextMeshProUGUI myNickName;
    public TextMeshProUGUI otherNickName;
    public TextMeshProUGUI myscore;
    //플레이어들이 선택한 것
    public GameObject myChoose_Img;
    public TextMeshProUGUI myChoose_Txt;
    public GameObject otherChoose_Img;
    public TextMeshProUGUI ohterChoose_Txt;
    //정답이펙트
    public TextMeshProUGUI myAnswerEffect;
    public TextMeshProUGUI ohterAnswerEffect;
    //타이머
    [Header("Timer")]
    public Image gageBar;
    float timer = 0.0f;
```

```
참조 7개
protected override void Init()
{
    base.Init();

    //문제 테이블
    QuestionTableSet();
}

참조 1개
void QuestionTableSet() //문제 테이블 셋팅하기
{
    questionList.Add(new KeyValuePair<string, OX>("문어다리는 10개이다", OX.X));
    questionList.Add(new KeyValuePair<string, OX>("달팽이도 이빨이 있다.", OX.O));
    questionList.Add(new KeyValuePair<string, OX>("고래는 5M 이하의 물속에서 잠을 잔다.", OX.X));
    questionList.Add(new KeyValuePair<string, OX>("원숭이에게도 지문이 있다", OX.O));
    questionList.Add(new KeyValuePair<string, OX>("남극에도 우편번호가 있다", OX.X));
    questionList.Add(new KeyValuePair<string, OX>("BUS라는 단어는 미국에서 처음 사용하였다", OX.X));
    questionList.Add(new KeyValuePair<string, OX>("닭도 왼발잡이 , 오른발잡이가 있다.", OX.O));
    questionList.Add(new KeyValuePair<string, OX>("새는 뒤로도 날 수 있다.", OX.O));
}
```

문제 테이블(리스트)

문제의 내용과 답을 가질수 있게 저장

```
참조 9개
public override void StartGame()
{
    base.StartGame();

    SoundMgr.Inst.PlayBGM("OXGame");
    //게임 UI On
    GamePanel.SetActive(true);

    //방장만
    if (PhotonNetwork.IsMasterClient)
    {
        //문제들 셋팅하기 ..무슨문제를 낼지
        QuestionSet();
    }

    //초기화
    currQuestion = 0;
    step = 0;
    myChoose = OX.None;
    otherChoose = OX.None;

    //원격동기화를 위해 CustomProperties에 선택한것 올리기
    playerHash = PhotonNetwork.LocalPlayer.CustomProperties;

    if (playerHash.ContainsKey("ChooseOX"))
        playerHash["ChooseOX"] = myChoose;
    else
        playerHash.Add("ChooseOX", myChoose);

    PhotonNetwork.LocalPlayer.SetCustomProperties(playerHash);


    //게임 로직 시작
    StartCoroutine(Game_Co());
}
```

```
void QuestionSet()//방장만 무슨문제 출제할지
{
    List<int> temp = new List<int>();
    while (questionNum.Count < 5)
    {
        int rand = Random.Range(0, questionList.Count);
        if (!questionNum.Contains(rand))
        {
            questionNum.Add(rand);
        }
    }

}
```

중복 없이 문제 등록

시작전 초기화

게임 구성

4. 미니게임 구현 - OX문제

OXGame.CS

```
참조 1개
IEnumerator Game_Co()
{
    questionTxt.text = "OX 문제입니다.";
    myscore.text = "내 점수 : 0";
    yield return new WaitForSeconds(1.0f);

    while (step < 5)
    {

        yield return new WaitForSeconds(1.0f);
        //문제 내기 방장만
        if (PhotonNetwork.IsMasterClient)
        {
            pv.RPC("SetTextQuestion", RpcTarget.AllViaServer, (int)questionNum[step]);
        }
        yield return new WaitForSeconds(0.5f);

        //모든 플레이어가 결정할때까지 대기 //또는 시간이 다지나면
        choose = false;
        bIng = true;
        while (bIng) //타이머 돌때까지 루프
        {
            yield return null;
            if (choose) //만약 내 선택을했으면 상대 선택한것도 체크하기
                OtherChooseCheck();
        }

        Choose_TimeOver();

        questionTxt.gameObject.SetActive(true);
        questionTxt.text = "타임오버";
        yield return new WaitForSeconds(1.0f);
        questionTxt.text = "정답은";
        //시간 종료후
        yield return new WaitForSeconds(1.0f);
        //결과 보여주기
        OnCheckOX(); //OX 맞춘거 카운터
        yield return new WaitForSeconds(2.0f);

        if (step.Equals(4))
            questionTxt.text = "게임 종료";
        else if (step.Equals(3))
            questionTxt.text = "마지막 문제";
        else
            questionTxt.text = "다음 문제";

        //UI 및 값 초기화
        yield return new WaitForSeconds(1.5f);
        ResetUI();
        step++; //다음 문제
    }

    //모든 문제 완료시 결과 보여주고
    //최종결과 창
    InGame.Inst.ShowResult();
    //게임 UI Off
    GamePanel.SetActive(false);

}
```

```
[PunRPC]
참조 0개
void SetTextQuestion(int num)  //방장쪽에서 정해진 문제 재출
{
    questionTxt.text = (step + 1) +"번 문제\n"+ questionList[num].Key;
    currQuestion = num;
    O_Btn.SetActive(true);
    X_Btn.SetActive(true);

    SoundMgr.Inst.PlayEffect("NextQuestion");

    StartCoroutine(Timer_Update());

}
참조 1개
IEnumerator Timer_Update()
{
    timer = 5.0f;
    while (timer >= 0)
    {
        yield return null;
        timer -= Time.deltaTime;
        gageBar.fillAmount = timer / 5.0f;
    }
    bIng = false;
}
```

```
참조 0개
public void Choose_OBtn () //유저 선택 O
{
    SoundMgr.Inst.PlayEffect("Button");
    myChoose = OX.O;
    choose = true;
    OnUserChoose();
}

참조 0개
public void Choose_XBtn()//유저 선택 X
{
    SoundMgr.Inst.PlayEffect("Button");
    myChoose = OX.X;
    choose = true;
    OnUserChoose();
}
```

```
void OnUserChoose()
{
    myNickName.text = PhotonNetwork.LocalPlayer.NickName;
    myNickName.gameObject.SetActive(true);
    otherNickName.text = PhotonNetwork.PlayerListOthers[0].NickName;
    otherNickName.gameObject.SetActive(true);

    //내가 선택한 OX 보여주기
    myChoose_Img.SetActive(true);
    questionTxt.gameObject.SetActive(false);

    //버튼 비활성화
    O_Btn.SetActive(false);
    X_Btn.SetActive(false);

    //선택에 따른 문양 보여주기
    if (myChoose.Equals(OX.O))
    {
        myChoose_Txt.text = "O";
        myChoose_Txt.color = Color.blue;
    }
    else if(myChoose.Equals(OX.X))
    {
        myChoose_Txt.text = "X";
        myChoose_Txt.color = Color.red;
    }
    else
        myChoose_Txt.text = "";

    if(!myChoose.Equals(OX.None))
    {
        //원격동기화를 위해 CustomProperties에 선택한것 올리기
        playerHash = PhotonNetwork.LocalPlayer.CustomProperties;

        if (playerHash.ContainsKey("ChooseOX"))
            playerHash["ChooseOX"] = myChoose;
        else
            playerHash.Add("ChooseOX", myChoose);

        PhotonNetwork.LocalPlayer.SetCustomProperties(playerHash);
    }
```

```
float temp = 0;
참조 2개
void OtherChooseCheck()//다른 유저 선택완료되면
{
    otherChoose_Img.SetActive(true);

    //선택에 따른 문양 보여주기
    if (otherChoose.Equals(OX.O))
    {
        ohterChoose_Txt.text = "O";
        ohterChoose_Txt.color = Color.blue;
    }
    else if (otherChoose.Equals(OX.X))
    {
        ohterChoose_Txt.text = "X";
        ohterChoose_Txt.color = Color.red;
    }
    else
    {
        temp += Time.deltaTime;

        ohterChoose_Txt.color = Color.black;
        ohterChoose_Txt.text = "";

        if (temp > 0 && temp <= 1.0f)
            ohterChoose_Txt.text = ".";
        else if (temp > 1.0f && temp <= 2.0f)
            ohterChoose_Txt.text = "..";
        else if (temp > 2.0f && temp <= 3.0f)
            ohterChoose_Txt.text = "...";

        if (temp > 3.0f)
            temp = 0.0f;
    }

    if(timer <= 0 && otherChoose.Equals(OX.None))
        ohterChoose_Txt.text = "";
}
```

```
//PlayerProperties들이 업데이트 된다면 //여기서 상대방 선택 체크
참조 20개
public override void OnPlayerPropertiesUpdate(Player targetPlayer
                , ExitGames.Client.Photon.Hashtable changedProps)
{
    if (targetPlayer != PhotonNetwork.LocalPlayer)
    {
        if (changedProps.ContainsKey("ChooseOX"))
        {
            otherChoose = (OX)changedProps["ChooseOX"];
        }
    }
}
```

유저의 선택으로

PlayerProperties가 변화가

생기면 캐치하여 가져옴

게임 구성

4. 미니게임 구현 - OX문제

OXGame.CS

```
void OnCheckOX()//정답 확인
{
    SoundMgr.Inst.PlayEffect("CheckOX");

    myAnswerEffect.gameObject.SetActive(true);
    questionTxt.text = "정답은 : " + questionList[currQuestion].Value;


    if (myChoose == questionList[currQuestion].Value)
    {
        myAnswerEffect.text = "O";
        myAnswerEffect.color = Color.blue;

        //각자 정답인 사람이 각자클라이언트에서 업데이트 해준다.
        playerHash = PhotonNetwork.LocalPlayer.CustomProperties;

        if (playerHash.ContainsKey("score"))
            playerHash["score"] = (int)playerHash["score"] + 1;
        else
            playerHash.Add("score", 1);

        PhotonNetwork.LocalPlayer.SetCustomProperties(playerHash);
        score++;
        myscore.text = "내 점수 : " + score.ToString();

    }
    else
    {
        myAnswerEffect.text = "X";
        myAnswerEffect.color = Color.red;
    }

    ohterAnswerEffect.gameObject.SetActive(true);
    //효과 만
    if (otherChoose == questionList[currQuestion].Value)
    {
        ohterAnswerEffect.text = "O";
        ohterAnswerEffect.color = Color.blue;
    }
    else
    {
        ohterAnswerEffect.text = "X";
        ohterAnswerEffect.color = Color.red;
    }
}
```

```
참조 1개
void ResetUI()//한문제 끝나면 다시 원상복구
{
    //UI 리셋
    ohterAnswerEffect.gameObject.SetActive(false);
    myAnswerEffect.gameObject.SetActive(false);

    myNickName.gameObject.SetActive(false);
    otherNickName.gameObject.SetActive(false);

    myChoose = OX.None;
    otherChoose = OX.None;

    myChoose_Img.SetActive(false);
    otherChoose_Img.SetActive(false);
    questionTxt.gameObject.SetActive(true);
}
```

AnswerEffect.CS

```
유니티 스크립트 | 참조 0개
public class AnswerEffect : MonoBehaviour
{
    TextMeshProUGUI text;

    private int count = 4;
    private float timer = 0.0f;

    private Color originColor;

    유니티 메시지 | 참조 0개
    private void Awake()
    {
        text = GetComponent<TextMeshProUGUI>();
    }
    유니티 메시지 | 참조 0개
    private void OnEnable()
    {
        count = 4;
        timer = 0.0f;
    }

    유니티 메시지 | 참조 0개
    private void Update()
    {
        if (count <= 0)
            return;

        timer += Time.deltaTime;

        if (text.text == "O")
            originColor = Color.blue;
        else
            originColor = Color.red;

        if(timer >= 0.2f)
        {
            timer = 0.0f;
            text.color = text.color.a == 0 ? originColor : Color.clear;
            count--;
        }

    }

}
```

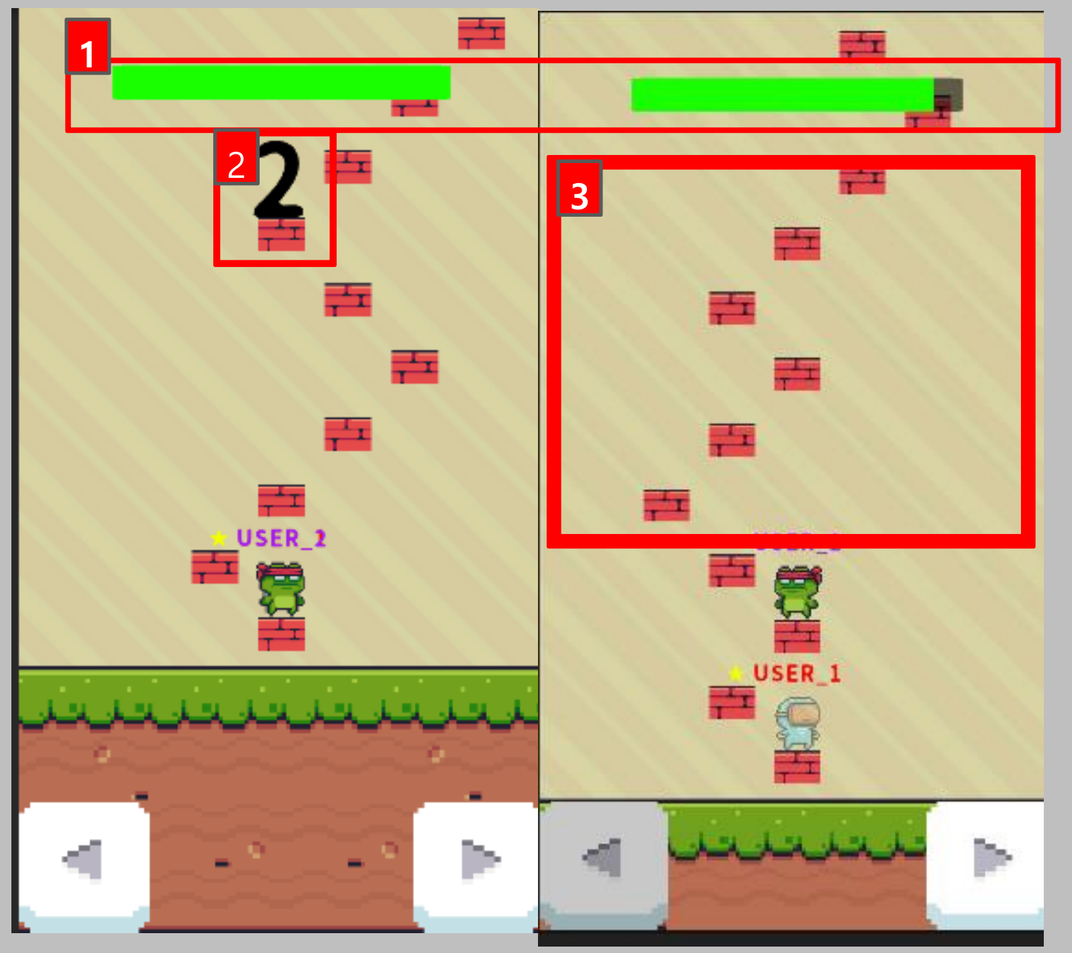
정답 체크시

나오는 이펙트

깜빡임 효과

게임 구성

4. 미니게임 구현 – 계단 게임



1) 타이머
   1) 타이머 다 되기전에 하나 올라가야함
   2) 타이머 다되면 게임 오버
2) 점수 : 계단 오른 수
3) 계단 오브젝트

방향키로 이동

누구한명이라도 떨어지면

게임 종료

그 순간 점수 비교

4. 미니게임 구현 – 계단 게임

StairGame.CS

```
public class StairGame : Game
{
    public PlayerCharacter myPlayer;    //내 플레이어
    PlayerCharacter[] players;          //캐릭터들
    public GameObject GamePanel;        //게임관련 UI
    public GameObject stairPrefab;      //계단 프리팹
    public GameObject spawnPosObj;      //계단 처음 스폰 위치
    Vector3 spawnPos = Vector3.zero;
    //소환된 계단들을 담아둘 큐
    Queue<Stairs> stairs = new Queue<Stairs>();
    //마지막 계단을 확인하기 위해
    Stairs lastStair;
    //소환된 계단을 담아둘 게임오브젝트
    public GameObject stairsGroup;
    //카메라 //캐릭터가 올라가기 때문에 카메라와 배경화면도 같이 올리기 위해서
    GameObject camera;
    //배경
    GameObject BG;
    //점수 텍스트
    public TextMeshProUGUI scroe_Txt;
    //제한시간
    public Image gageBar;
    float nextTimer = 2.0f;
    float timer = 2.0f;
    //게임 진행 중
    bool game = false;
    //상대 Sprite 투명도를 주기위해
    SpriteRenderer otherSprite;
    //시작 카운트
    public TextMeshProUGUI countText;
```

```
참조 0개
protected override void Init()
{
    base.Init();

    camera = Camera.main.gameObject;
    BG = GameObject.Find("BG");
}

참조 0개
public override void StartGame()
{
    base.StartGame();
    SoundMgr.Inst.PlayBGM("StairGame");
    //캐릭터 설정
    players = InGame.Inst.playerCharacters;
    myPlayer = players[0];
    //상대캐릭터는 반투명으로
    otherSprite = players[1].GetComponent<SpriteRenderer>();
    otherSprite.color = new Color(1, 1, 1, 0.5f);

    //올라가는 버튼 적용
    Button LBT = GameObject.Find("LeftButton").GetComponent<Button>();
    LBT.onClick.RemoveAllListeners();
    LBT.onClick.AddListener(LeftMove);

    Button RBT = GameObject.Find("RightButton").GetComponent<Button>();
    RBT.onClick.RemoveAllListeners();
    RBT.onClick.AddListener(RightMove);

    //계단 생성하기
    if (PhotonNetwork.IsMasterClient)
        for (int i = 0; i < 20; i++)
            SpawnStair();
    //좌우 이동을 막는다
    myPlayer.isMove = false;
    //캐릭터 이동
    spawnPos = spawnPosObj.transform.position + Vector3.up;
    spawnPos.z = -1;
    myPlayer.transform.position = spawnPos;
    gageBar.fillAmount = 1.0f;

    GamePanel.SetActive(true);
    StartCoroutine(Game_Co());
}
```

초기 카메라, 배경 오브젝트 캐싱

좌우 이동 버튼 함수 추가하기

4. 미니게임 구현 – 계단 게임

StairGame.CS

```
public void RightMove()
{
    if (!game)
        return;

    SoundMgr.Inst.PlayEffect("UpStair");

    myPlayer.transform.position += Vector3.up + Vector3.right;

    camera.transform.position += Vector3.up;
    BG.transform.position += Vector3.up;

    CheckStair();
}
참조 2개
public void LeftMove()
{
    if (!game)
        return;

    SoundMgr.Inst.PlayEffect("UpStair");

    myPlayer.transform.position += Vector3.up + Vector3.left;

    camera.transform.position += Vector3.up;
    BG.transform.position += Vector3.up;

    CheckStair();
}
```

좌우 이동

배경 및 카메라도 같이 이동

Stairs.CS

```
Unity 스크립트 | 참조 7개
public class Stairs : MonoBehaviourPunCallbacks
{
    public int num =0;
}
```

```
참조 1개
IEnumerator Game_Co()
{
    countText.text = "3";
    yield return new WaitForSeconds(1.0f);
    countText.text = "2";
    yield return new WaitForSeconds(1.0f);
    countText.text = "1";
    yield return new WaitForSeconds(1.0f);
    countText.text = "Start!!";

    game = true;
    gageBar.fillAmount = 1.0f;
    timer = 2.0f;

    yield return new WaitForSeconds(1.0f);
    countText.text = "";


    while (true)
    {
        yield return null;

        if (PhotonNetwork.IsMasterClient && game)
            SpawnCheckStair();


        if (!game)
            continue;

        if (timer > 0)
        {
            timer -= Time.deltaTime;
            gageBar.fillAmount = timer / nextTimer;
            if (timer <= 0)
            {
                GameOver();
            }
        }
    }
}
```

```
참조 1개
void SpawnCheckStair()
{
    bool destory = true;

    //맨 마지막 블럭 체크
    for (int i = 0; i < players.Length; i++)
    {
        if (stairs.Peek().transform.position.y + 1.2f > players[i].transform.position.y)
            destory = false;
    }

    if (destory)
    {
        var d = stairs.Dequeue();
        PhotonNetwork.Destroy(d.gameObject);
    }

    //맨 위 블럭에서 일정 밑에 있는 유저를 보고 새로운 계단 만들기
    bool spawn = false;
    for (int i = 0; i < players.Length; i++)
    {
        if (lastStair.transform.position.y - 10 < players[i].transform.position.y)
            spawn = true;
    }

    if (spawn)
        SpawnStair();

}
```

4. 미니게임 구현 – 계단 게임

StairGame.CS

```
참조 2개
void SpawnStair()
{
    //새로운 다음 계단 만들기
    if(stairs.Count.Equals(0))
    {
        Stairs newStairs = PhotonNetwork.InstantiateRoomObject("Stairs", spawnPosObj.transform.position, Quaternion.identity).GetComponent<Stairs>();
        newStairs.num = 0;
        stairs.Enqueue(newStairs);
        lastStair = newStairs;
        newStairs.transform.SetParent(stairsGroup.transform);
        return;
    }

    int nextX = 0;
    if (lastStair.num == 3)
        nextX = -1;
    else if (lastStair.num == -3)
        nextX = 1;
    else
        nextX = Random.Range(0, 2) == 0 ? -1 : 1;

    Vector3 nextPos = new Vector3(nextX , 1);//이전 칸 보다 좌우는 랜덤 위로 1칸 증가
    nextPos = lastStair.transform.position + nextPos;

    GameObject newStairObj = PhotonNetwork.InstantiateRoomObject("Stairs", nextPos, Quaternion.identity);
    Stairs newStair = newStairObj.GetComponent<Stairs>();
    newStair.transform.SetParent(stairsGroup.transform);
    newStair.num = lastStair.num + nextX;
    stairs.Enqueue(newStair);
    lastStair = newStair;
}
```

```
참조 2개
void CheckStair()
{
    RaycastHit2D hit = Physics2D.Raycast(myPlayer.transform.position, Vector3.down, 1.0f);
    if(hit)
    {
        nextTimer -= 0.01f;
        if (nextTimer <= 0.5f)
            nextTimer = 0.5f;

        timer = nextTimer;
        gageBar.fillAmount = timer / nextTimer;

        score += 1;
        scroe_Txt.text = score.ToString();

        playerHash = PhotonNetwork.LocalPlayer.CustomProperties;
        if (playerHash.ContainsKey("score"))
            playerHash["score"] = (int)playerHash["score"] + 1;
        else
            playerHash.Add("score", 1);

        PhotonNetwork.LocalPlayer.SetCustomProperties(playerHash);
    }
    else
        NoStair();
}

참조 1개
void NoStair()
{
    SoundMgr.Inst.PlayEffect("HitStair");
    myPlayer.SetHit();
    GameOver();
}
```

좌우이동시

바닥을 체크 하여 계단 유무 확인

4. 미니게임 구현 – 계단 게임

StairGame.CS

```
참조 2개
void GameOver()
{
    pv.RPC("GameEnd", RpcTarget.AllViaServer);
}

[PunRPC]
참조 0개
void GameEnd()
{
    game = false;
    StartCoroutine(GameEnd_Co());
}

참조 1개
IEnumerator GameEnd_Co()
{
    yield return new WaitForSeconds(0.5f);
    otherSprite.color = Color.white;
    players[0].transform.position = Vector3.zero;
    players[1].transform.position = Vector3.zero;

    camera.transform.position = new Vector3(0, 0, camera.transform.position.z);
    BG.transform.position = new Vector3(0, 0, BG.transform.position.z);

    while (stairs.Count > 0)
        PhotonNetwork.Destroy(stairs.Dequeue().gameObject);

    stairs.Clear();

    countText.text = "종료";
    yield return new WaitForSeconds(1.0f);
    countText.text = "결과는";
    yield return new WaitForSeconds(1.0f);
    myPlayer.isMove = true;
    countText.text = "";

    GamePanel.SetActive(false);

    InGame.Inst.ShowResult();
}
```

게임 구성

4. 미니게임 구현 – 기억력 게임

1) 맞춰야하는 화살표 순서
   1) 난이도 3 ~ 6 개: 성공할수록 점점 늘어남
   2) 일정시간 이지나면 사라짐
2) 점수 : 맞춘갯수
3) 내가 입력한 커맨드
4) 결과 판정

일정시간동안

가장 많이 성공한 사람이

승점 + 1

게임 구성

4. 미니게임 구현 – 기억력 게임

RememberGame.CS

```
Unity 스크립트 | 참조 0개
public class RememberGame : Game, IPunObservable
{
    //화살표 이미지   0 : left , 1 : right
    public Sprite[] arrowSprite;
    //레벨
    int level = 0;
    //레벨별 이미지 갯수
    int[] levelCount = { 3, 4, 5, 6 };
    float[] leveltimer = { 1.0f, 1.2f, 1.2f,1.4f };
    int count = 0;
    //레벨별 그룹
    public GameObject[] levelGroup;
    public List<Image[]> imgs = new List<Image[]>();
    //내가 입력한 순서를 저장할    //레벨에 따라 최대 6개 까지입력
    int mySelNum = 0; //내가 입력한 순서;
    public Image[] mySetImg;
    //정답을 저장할
    int[] answer = new int[6];
    public TextMeshProUGUI OX;
    //맞힌갯수 UI
    public TextMeshProUGUI answerCountTxt;
    //맞힌갯수 저장변수
    int answerCount = 0;
    //체크하는 큐//입력한
    Queue<int> check = new Queue<int>();
    public GameObject timerObj;
    float timer = 40.0f;
    public Image timerbar;
    bool wait = false;

    //화살표를 숨기게 하는 함수를 담은 코루틴
    Coroutine currCo;
    public TextMeshProUGUI infoText;
    public GameObject GamePanel;        //게임 패널
    //캐릭터
    PlayerCharacter myPlayer;
```

```
참조 9개
protected override void Init()
{
    base.Init();

    for (int i = 0; i < levelGroup.Length; i++)
        imgs.Add(levelGroup[i].GetComponentsInChildren<Image>());
}

참조 9개
public override void StartGame()
{
    base.StartGame();

    SoundMgr.Inst.PlayBGM("RememberGame");
    //좌우 이동을 막는다
    myPlayer = InGame.Inst.playerCharacters[0];
    myPlayer.isMove = false;

    //버튼 적용
    Button LBT = GameObject.Find("LeftButton").GetComponent<Button>();
    LBT.onClick.RemoveAllListeners();
    LBT.onClick.AddListener(() => { SelMyArrow(0); });

    Button RBT = GameObject.Find("RightButton").GetComponent<Button>();
    RBT.onClick.RemoveAllListeners();
    RBT.onClick.AddListener(() => { SelMyArrow(1); });

    //게임 UI On
    GamePanel.SetActive(true);

    //게임 로직 시작
    StartCoroutine(Game_Co());
}
```

게임 구성

4. 미니게임 구현 – 기억력 게임

RememberGame.CS

```
참조 1개
IEnumerator Game_Co()
{
    //초기화
    level = 0;
    answerCount = 0;

    timerObj.SetActive(false);
    infoText.gameObject.SetActive(true);
    infoText.text = "나오는 화살표 방향에 맞게\n순서대로 입력해주세요\n시간이 지나면 사라집니다.";
    yield return new WaitForSeconds(1.5f);
    infoText.gameObject.SetActive(false);

    //화살표 셋팅
    ArrowSetting();

    answerCountTxt.gameObject.SetActive(true);
    answerCountTxt.text = "정답 갯수 : 0";


    timerObj.SetActive(true);
    timer = 40.0f;
    timerbar.fillAmount = timer / 40.0f;
    //타이머
    StartCoroutine(Time_Update());
```

```
    //게임 로직
    while (timer > 0)
    {
        yield return null;
        //체크하기 //입력받은 큐에들어있는 것을 확인하면서
        if (check.Count > 0)
        {
            int RL = check.Dequeue(); //앞에 입력하거 가져와서
            mySetImg[mySelNum].gameObject.SetActive(true); //내가 선택한 화살표 보여주기
            mySetImg[mySelNum].sprite = arrowSprite[RL];    //이미지 적용

            if (RL != answer[mySelNum]) //정답이 아니면
            {
                OX.gameObject.SetActive(true);
                OX.text = "X";
                OX.color = Color.red;
                //판정시 잠시 대기
                wait = true;
                yield return new WaitForSeconds(0.5f);
                wait = false;
                OX.gameObject.SetActive(false);
                FailRemember();
            }
            else
            {
                mySelNum++; //다음 화살표로
                if (mySelNum > levelCount[level] - 1)
                {
                    OX.gameObject.SetActive(true);
                    OX.text = "O";
                    OX.color = Color.blue;
                    //판정시 잠시 대기
                    wait = true;
                    yield return new WaitForSeconds(0.5f);
                    wait = false;
                    OX.gameObject.SetActive(false);
                    SuccessRemember();
                }
            }
        }

        //타임 종료후
        levelGroup[level].SetActive(false);
        MySetClear();

        answerCountTxt.text = "타임오버!!";
        yield return new WaitForSeconds(1.0f);

        answerCountTxt.gameObject.SetActive(false);
        GamePanel.SetActive(false);

        myPlayer.isMove = true;
        InGame.Inst.ShowResult();

    }
```

입력된 커맨드를 확인하는 작업

시간을 따로 다루는 코루틴

```
참조 1개
IEnumerator Time_Update()
{
    while (timer >= 0)
    {
        yield return null;
        timer -= Time.deltaTime;
        timerbar.fillAmount = timer / 40.0f;
    }
}
```

타이머 동기화

```
//타이머 동기화를 위한
참조 13개
public void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
{
    if (stream.IsWriting)
    {
        stream.SendNext(timer);
    }
    else
    {
        timer = (float)stream.ReceiveNext();
    }

}
```

게임 구성

4. 미니게임 구현 – 기억력 게임

RememberGame.CS

```
void ArrowSetting() //레벨에 맞게 무작위 화살표 셋팅
{
    check.Clear();
    wait = false;

    levelGroup[level].SetActive(true);
    //무작위 선택을 위해
    int rand = 0;
    //레벨에 따라 화살표 배치
    for (int i = 0; i < imgs[level].Length; i++)
    {
        rand = Random.Range(0, 2);
        imgs[level][i].sprite = arrowSprite[rand];
        answer[i] = rand;
    }


    if (currCo != null) //만약 함수가 발동되기전에 이미 돌아가고 있으면 기존 함수멈춤
        StopCoroutine(currCo);

    // 화살표 일정시간 뒤에 사라지게 하는 코루틴 함수
    currCo = StartCoroutine(OffArrow(leveltimer[level]));
}

참조 1개
IEnumerator OffArrow(float time)//일정시간 뒤에 화살표 안보이게
{
    yield return new WaitForSeconds(time);
    levelGroup[level].SetActive(false);

    currCo = null;
}
```

레벨에 맞게
이미지 적용

랜덤값으로
이미지 설정

```
참조 1개
void FailRemember()
{
    SoundMgr.Inst.PlayEffect("Fail");

    //다음
    ArrowSetting();
    MySetClear();

    //내 케릭터 틀린거 //뭐에 맞은것 처럼
    InGame.Inst.playerCharacters[0].SetHit();
    mySelNum = 0;
}
```

실패 시

```
참조 4개
void SelMyArrow(int RL)
{
    //만약 선택한 횟수가
    if (mySelNum == levelCount[level])
        return;

    if (wait)
        return;

    if (RL.Equals(0))
        SoundMgr.Inst.PlayEffect("BtnL");
    else
        SoundMgr.Inst.PlayEffect("BtnR");

    check.Enqueue(RL);
}

참조 3개
void MySetClear()
{
    for (int i = 0; i < mySetImg.Length; i++)
    {
        mySetImg[i].gameObject.SetActive(false);
    }
}
```

버튼에 등록되는 함수

내가 입력한 커맨드 이미지 초기화 함수

```
void SuccessRemember()
{
    SoundMgr.Inst.PlayEffect("Success");

    levelGroup[level].SetActive(false);

    //다음 단계로 갈수 있으면 카운트up
    if (level < levelCount.Length - 1)
        count++;

    if (count == 5)//5개씩 맞추면
    {
        if (level < levelCount.Length)
            level++;

        count = 0;
    }

    answerCount++;
    answerCountTxt.text = "정답 갯수 : " + answerCount;

    //점수 동기화를 위해 CustomProperties에 저장
    playerHash = PhotonNetwork.LocalPlayer.CustomProperties;
    //점수
    if (playerHash.ContainsKey("score"))
        playerHash["score"] = answerCount;
    else
        playerHash.Add("score", answerCount);

    PhotonNetwork.LocalPlayer.SetCustomProperties(playerHash);
    //다음
    ArrowSetting();
    MySetClear();

    mySelNum = 0;
}
```

성공 시

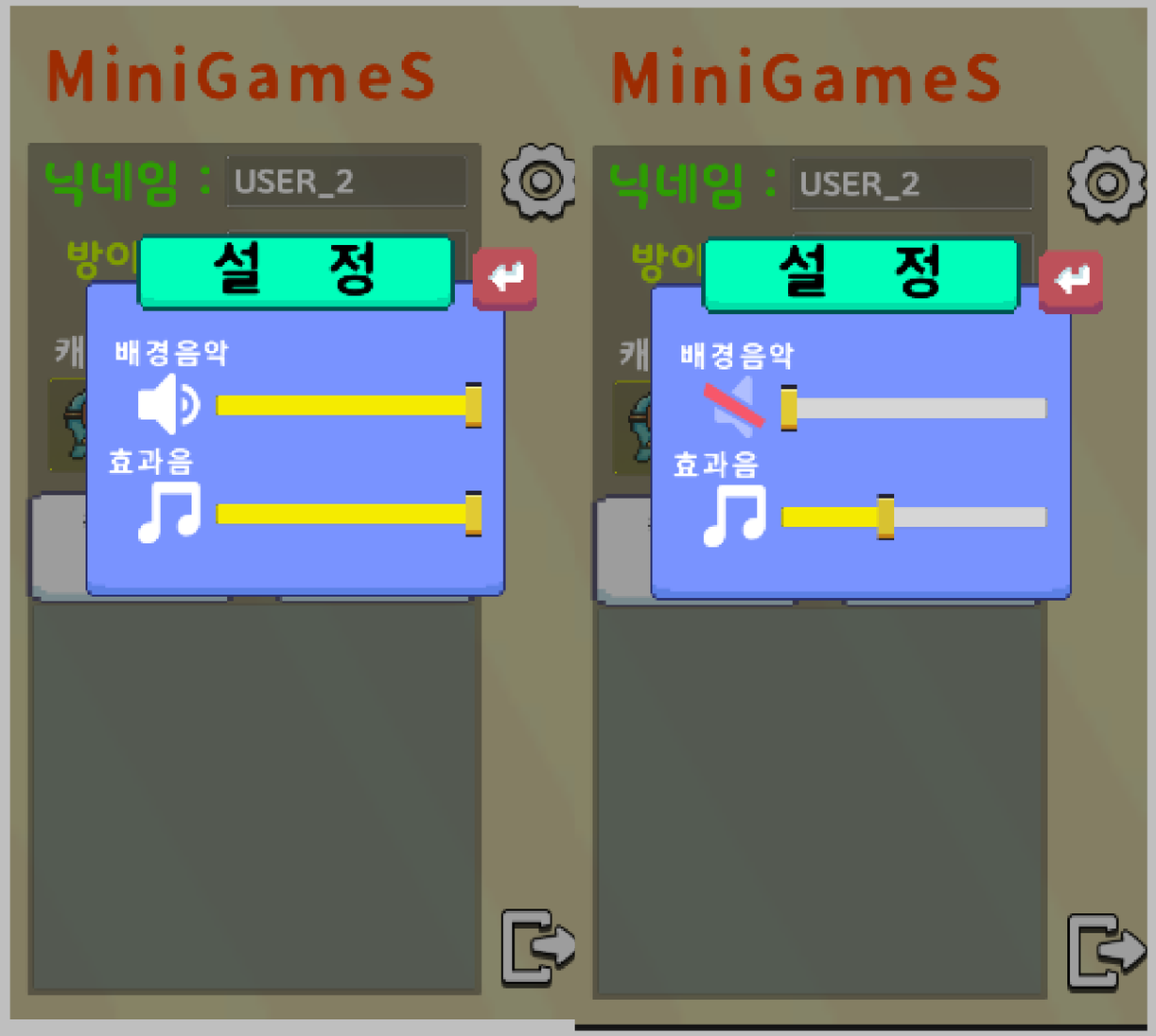
5개 성공시

난이도 상승

다음 난이도 설정

게임 구성

5. 사운드 시스템



슬라이더 바 조절 하여 음량 조절 가능

팝업창을 프리팹으로 사용하여

다른 씬에서도 오픈가능하게 구현

게임 구성

5. 사운드 시스템

SoundMgr.CS

```
public class SoundMgr : MonoBehaviour
{
    //싱글턴 패턴
    static public SoundMgr Inst;
    //오디오클립
    public AudioClip[] audioClips;
    //bgm을 실행 시키는
    AudioSource bgmSource;
    //Effect sound 를 실행시켜 주는
    Queue<AudioSource> effectSource = new Queue<AudioSource>();
    //사운드 클립을 담아 놓는 곳
    Dictionary<string, AudioClip> sounds = new Dictionary<string, AudioClip>();
    //이펙트 사운드
    float effectVolum = 1;
    참조 3개
    public float EffectVolum { get { return effectVolum; } set { effectVolum = value; } }
    참조 3개
    public float BGMVolum { get { return bgmSource.volume; } set { bgmSource.volume = value; } }
    public GameObject SoundCtrlBox; //사운드 설정 하는 팝업창
    유Unity 메시지 | 참조 0개
    private void Awake()
    {
        if (Inst == null)
        {
            Inst = this;
            DontDestroyOnLoad(this.gameObject);
        }
        else
            Destroy(this.gameObject);

        //BGM 전용 셋팅
        bgmSource = this.gameObject.AddComponent<AudioSource>();
        bgmSource.loop = true;

        //effect 셋팅  5개정도 생성
        for (int i = 0; i < 5; i++)
        {
            AudioSource newAudioSource = this.gameObject.AddComponent<AudioSource>();
            effectSource.Enqueue(newAudioSource);
        }
        SoundInit();
    }
```

```
참조 1개
void SoundInit()
{
    for (int i = 0; i < audioClips.Length;i++)
        sounds.Add(audioClips[i].name, audioClips[i]);
}
```

등록된 오디오 클릭을

이름으로 접근가능하게 딕셔너리에 넣어줌

다른 씬에서도 사용가능하고

BGM, Effect사운드 따로 사용을 위한

게임 구성

5. 사운드 시스템

SoundMgr.CS

```
참조 7개
public void PlayBGM(string BgmName, float speed = 1)  //BGM 실행
{
    //사운드가 있는지 체크
    if(sounds.ContainsKey(BgmName))
        bgmSource.clip = sounds[BgmName];
    else //없으면
    {
        return;
    }

    bgmSource.pitch = speed;
    bgmSource.Play();
}

참조 25개
public void PlayEffect(string EffectName)
{
    AudioSource nowAudio = effectSource.Dequeue();

    //사운드가 있는지 체크
    if (sounds.ContainsKey(EffectName))
        nowAudio.clip = sounds[EffectName];
    else //없으면
    {
        return;
    }

    nowAudio.volume = effectVolum;
    // nowAudio.PlayOneShot(nowAudio.clip);
    nowAudio.Play();

    effectSource.Enqueue(nowAudio);
}
```

사운드 호출을 위한 함수

팝업창을 띄우는

```
public void OnSoundCtrlBox()
{
    Instantiate(SoundCtrlBox);
}
```

```
Unity 스크립트 | 참조 0개
public class SoundCtrlBox : MonoBehaviour
{
    public Image BGM;
    public Image Effect;
    public Sprite[] imgs;  //BGM : ON OFF/Effect : ON OFF/ 그림 순서
    public Slider bgmSlider;
    public Slider EffectSlider;
    public Button backBtn;

    Unity 메시지 | 참조 0개
    private void Start()
    {
        SoundMgr.Inst.PlayEffect("OpenSonudBox");
        bgmSlider.value = SoundMgr.Inst.BGMVolum;
        ChangeBGMSoundVolume(SoundMgr.Inst.BGMVolum);
        EffectSlider.value = SoundMgr.Inst.EffectVolum;
        ChangeEffectSoundVolume(SoundMgr.Inst.EffectVolum);

        bgmSlider.onValueChanged.AddListener(ChangeBGMSoundVolume);
        EffectSlider.onValueChanged.AddListener(ChangeEffectSoundVolume);
        backBtn.onClick.AddListener(BackBtn);
    }

    참조 2개
    void ChangeBGMSoundVolume(float v)
    {
        SoundMgr.Inst.BGMVolum = v;
        if (v == 0)
            BGM.sprite = imgs[1];
        else
            BGM.sprite = imgs[0];
    }

    참조 2개
    void ChangeEffectSoundVolume(float v)
    {
        SoundMgr.Inst.EffectVolum = v;
        if (v == 0)
            Effect.sprite = imgs[3];
        else
            Effect.sprite = imgs[2];
    }

    참조 1개
    void BackBtn()
    {
        SoundMgr.Inst.PlayEffect("Button");
        Destroy(this.gameObject);
    }
}
```

사운드 팝업창 (SoundCtrlBox)

시작시 저장된 값으로 설정

값이 변화가 생기면

SoundMgr의 값을 변경 시켜준다.